AN INVESTIGATION OF WEB USE DURING PROGRAMMING

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2023

Omar Alghamdi

Department of Computer Science

Contents

Word Count: 43773

List of Tables

List of Figures

Abstract

Websites are a key resource consulted by programmers during their coding tasks, providing essential information, including code snippets. However, the implications of website use are poorly understood with respect to both programmers cognition and their code outputs. Programmers' (human) memory has also been shown to be an important resource in coding tasks, and there is some evidence from psychology that website use may inhibit memory. Studies of online code repositories also suggest that problematic code propagates through the Web. To date there has been little research on programmers' memory implications from using the Web, nor on programmers' experiences of encountering problematic online code, and whether coding with the Web leads to the adoption of problematic online code in programmers' own code.

This thesis sets out to contribute to understandings of the role of websites in programmers' coding activities, and the possible implications of their usage. Three studies provide qualitative and quantitative data describing participants' use of the Web when coding, including its role, follow-on activities and consequences (perceived and actual).

The results confirm that the Web and human memory are essential resources used by programmers when coding, and that they make frequent use of search engines and online code when using the Web. Programmers perceived little impact of this web use on their memory, but recognised the prevalence of problematic online code. Through an observed coding task and analysis of resulting source code, we find evidence that encounters with problematic online code can have negative consequences for programmers code outputs. The results advance the current understanding of Web usage for coding and how it affects programmers' memory and code.

Declaration

Copyright

Acknowledgements

In the name of Allah, The Most Gracious, The Most Merciful.

All praise and thanks to Allah, the lord of the worlds for giving me the ability and patience to finish My Ph.D. thesis. I am grateful for all your blessings.

I would like to express my sincere gratitude to my supervisor Dr Sarah Clinch for her guidance and assistance throughout my PhD journey. Thank you for supporting me, for your enlightenment and instructions that helped my research, for your patience whenever I am stressed and for lifting me when I was feeling down. I also thank my co-supervisor, Dr Caroline Jay, for all the helpful long discussions and suggestions to improve my thesis. Thank you for sparing me time whenever I needed it and having patience. I am grateful to have both of you as my supervisors, your unique personalities have allowed me to become a better researcher.

I would like to thank members of the Interaction Analysis and Modelling Laboratory (IAM) for welcoming me. My thanks reach all the friends who support me throughout my journey. A special thanks to all the participants who participate in studies within this thesis. Without your time and commitment, this work would not have been possible. Moreover, I gratefully acknowledge the funding source provided by the Government of Saudi Arabia that made my PhD work possible. Many thanks to my sponsor, Saudi Electronic University, for giving the opportunity to pursue my dream.

I would like to devote this work to my parents, who made numerous sacrifices to get me to this point in my life. Although you're no longer with us, my father Salim, I want to express my heartfelt gratitude for all that you did for me. Your love and wisdom will live on in my memories and continue to guide me in life. Thank you, my mother (Fauziah), for all your love, help, patience and prayers. A special thanks to my wife (Arwa). I am grateful for your support through my journey, endless love, listening to my complaining and encouraging me to overcome all the hurdles. Such a journey would be much more challenging without you beside me. A heartfelt thanks to my son (Salem) for your smile and joyful playing while I am busy with my thesis. Words cannot describe all my wife and son’s sacrifices to help me get through this journey. My thanks extend to my sister (Maha), her kids (Ruba, Rama, and Meshari), and my brothers (Walid, Zuhair, Majid, and Baraa).

Chapter 1

Introduction

Programmers augment their knowledge by seeking information from a variety of resources, including their own memory and external sources. Whilst industry-programmers make use of colleagues [Maalej et al., 2014], libraries and codebases [Sadowski et al., 2015], the availability of websites may make them preferable for coding-related information-seeking [Brandt et al., 2009, 2008, Gallardo-Valencia & Sim, 2011]. Targeted websites for programmers include Question and Answer (Q&A) websites, tutorials, documentation, and technical blogs. In particular, the Stack Overflow Q&A website is one of the most used websites by programmers [Acar et al., 2016, Yang et al., 2017]. Knowledge and code snippets from the site (amounting to millions of lines of code [*StackExchange Data Explorer*, 2023]) are used in knowledge acquisition, debugging, refactoring and redesign; and many of its code snippets are directly reused [Abdalkareem et al., 2017b, Hora, 2021]. Programmers' activities while coding the Web are under-explored. Furthermore, study of programmers resource use and its implications have been limited to software engineering concerns (e.g. code outputs), but recent results from psychology suggest that cognitive implications for programmers may also need to be considered.

In the general population, use of the Web has led to concerns about so-called "Google effects" [Sparrow et al., 2011, Schooler & Storm, 2021], a result of individuals offloading responsibility for retaining information to the Web [Risko & Gilbert, 2016], affecting memory. This behavior reduced encoding and recall of information known to be stored online/digitally [Macias et al., 2015, Fisher et al., 2022, Schooler & Storm, 2021, Sparrow et al., 2011], and a false sense that information sourced online was known [Loh & Kanai, 2016, Fisher et al., 2015] or had been learned [Fisher et al., 2022]. However, the extent of these "Google effects" on programmers' memory is unclear and has not been explored.

Programmers use both their memory [Ichinco & Kelleher, 2017, LaToza et al., 2006] and the Web [Acar et al., 2016] when coding. Memory plays an important role in code understanding [Fritz et al., 2007, Krüger et al., 2018]. Furthermore, experiments with memory have shown that it can be used to distinguish between experienced and non-experienced programmers [Bateson et al., 1987]. Use of the Web is also very prevalent amongst programmers, with in-situ studies suggesting that interactions with memory (e.g. learning, remembering) play an important role [Brandt et al., 2009]. Studies of programmers' Web use have raised concerns about potential negative impacts on code (using code that is not understood and/or is problematic) [Fischer et al., 2017, Maalej et al., 2014, Schröter et al., 2017].

Websites contain a plethora of code available to use. Since coding is a complex task, programmers are motivated to save time by using the code presented online [Brandt et al., 2008, 2009, Abdalkareem et al., 2017a], constituting a common coding activity [Brandt et al., 2009, Umarji et al., 2008, Xia et al., 2017]. Programmers look for code snippets that add new functionality [Kim et al., 2004, Brandt et al., 2009], provide improvements and resolve issues [Abdalkareem et al., 2017a]. Copying code, also known as code cloning [Roy et al., 2009], is a matter of considerable debate. It has been argued that cloning code can impact software stability and maintenance both positively and negatively [Kapser & Godfrey, 2008, Krinke, 2008, Mondal et al., 2018]. With such matter being unresolved, the validity of the cloned code itself is one factor that impacts code cloning. Programmers, who use the websites, could encounter code with issues they are unaware of. Previous research has investigated online code and shown that code snippets on Stack Overflow contain issues, including outdated code [Ragkhitwetsagul et al., 2019], security issues [Acar et al., 2016, Fischer et al., 2017, Meng et al., 2018, Licorish & Nishatharan, 2021] and bugs [Abdalkareem et al., 2017a]. Code on Stack Overflow could also propagate into other venues and disseminate problematic code, this includes GitHub [T. Zhang et al., 2019], Android applications [An et al., 2017], and open-source systems (OSS) [Ragkhitwetsagul et al., 2019]. Furthermore, it could propagate to the programmers' source code, causing issues affecting its functionality [Acar et al., 2016]. One possible reason for problematic online code is the nature of questions and answers websites (Q&A). Code posted online, such as Stack Overflow code, is not always credible since users could unknowingly provide code with issues. Another one is the role of the programmers. When adopting code, assessing the suitability of the code snippets is challenging and requires reading and understanding of the code [Schröter et al., 2017]; nevertheless, programmers often do not fully engage in understanding online reused code because of the time and effort involved [Maalej et al., 2014], a lack of knowledge [Escobar-Avila et al., 2019], or low readability of the snippet itself [Meldrum et al., 2020].

Prior research on the effects of website usage on memory shows that the relationship is far from simple. Nonetheless, it could have implications for programmers' memory when coding. Programmers could become more dependent on the Web for even basic known information and unable to code without using the websites. With such usage, their experience could lack advancement, affecting their coding. To date, there are no explorations regarding the notion of memory impacts on programmers caused by using the websites. Programming is a challenging task [Détienne, 2001] that could be more complex than basic searches for regular users. Thus, given that the programmers still use their memory, it is vital to understand the usage of memory and websites within programming, and investigate the perceived implications. Investigating the possible implications of using the Web on programmers' memory will provide a better understanding of the role of memory in coding and better support programmers in using Web resources effectively. It could also help minimise the efforts of searching online, increase programmers' knowledge and their experiences, enhance their abilities of learning and increase their code familiarisation.

Besides the websites, programmers' expertise and background could also factor in facing online code issues. While previous discoveries bring insights into reusing online code, most studies have focused on extracting possible issues that reside on the websites but have not dealt with the programmers' point of view, including their experience of reusing online code and usage of the code that contains issues. In addition, websites prompt an easy way to reuse the available contents, and programmers could miss checking the contents and adopt problematic code. Thus, it is likely that online code with the associated issues propagates to programmers' code. Prior research attempted exploring online activities and linked it with programmers' source code and found that the complexity of the source code coupled with more compiling and looking for online help [Astromskis et al., 2017]. However, no single study examines in-depth online usage and the activities that shape the outcome. In addition, the association of online activities with the source code provide an understanding of problematic online code that could propagate into the programmers' source code.

1.1 Aims and research questions

This thesis concerns coding with websites and explores the perceived impacts on programmers' memory while coding. In addition, work in this thesis investigates the implications of Web usage on the code by either encountering or adopting problematic code online. Thus, this thesis seeks to understand programmers' coding with websites, help them identify their usage of memory to reflect on it and increase awareness of the possible risks to the code from coding with the Web. The thesis addresses the following research questions:

RQ1 *What resources do programmers use to support the coding process? How and why do they use them?* Programmers resort to various resources while coding, and their primary one is attributed to the websites [Brandt et al., 2009]. However, their choices of resources while coding will likely differ based on their programming experience level. Also, programmers' use of websites lacks in-depth exploration of coding activities. Therefore, this thesis explores the various coding resources that programmers with different experiences choose, along with the reasons for such choices, and examines the in-practice strategies of seeking and utilising online knowledge while coding. Studies in this thesis interview students and professional programmers about their selections of resources they used for coding, then survey wider responses of students and professional programmers (see Chapter 3 and Chapter 4). It also examines in-depth the programmers' activities when coding with the websites, providing in-practice observations that increase the understanding of coding with the Web (see Chapter 5).

RQ2 *What is the perceived impact of using websites on programmers' memory?* With the concerns of 'Google effects' on the users [Sparrow et al., 2011, Schooler & Storm, 2021], similar implications could also occur for the programmers. Thus, this thesis explores the perceived implications of the programmers' memory when coding using websites. It first interviews students and professional programmers about their perception of memory and the perceived impacts when coding using websites (see Chapter 3). Then, it tests the generalisability of interview outcomes among students and professional programmers using a survey (see Chapter 4). A sequential mixed methods approach provides an in-depth understanding and breadth perspective of memory usage and perceived implications from the Web.

RQ3 *How does use of websites affect programmers' code?* Websites contain code available for programmers to use, but such code could suffer from multiple issues. While previous research addressed some of these issues using data mining approaches, this thesis focuses on the perspectives of programmers with various experiences regarding encountering problematic code online. It also takes a further step by examining programmers resulting code associated with their online coding activities for any issues. Two studies in this thesis (in Chapters 3 and 4) examine the usage of online code and the encounter of problematic code for students and professional programmers. A further study (in Chapter 5) observes programmers' in-practice coding to associate their use of online resources with the resulting code for possible implications and propagation of problematic online code.

1.2 Thesis structure

- **Chapter 2: Related Work.** This chapter outlines the interdisciplinary related work in psychology and software engineering. At first, it provides an overview of the previous work studying the implications of coding with websites on users' memory and provides a brief overview of the usage of programmers' memory for coding. Then, it gives an overview of the resources programmers use for coding and the involved searching activities. At last, it reviews prior work focusing on the code, including code reusing and problematic code.

- **Chapter 3: An interview study of websites' usage and the implications.** This chapter introduces a qualitative study by interviewing students and professional programmers to learn more about Web usage during coding and the implications. The results of the interviews were analysed using thematic analysis with a focus on the programmers' memory and code. These data report that students are more dependent on the websites, and coding with websites involved implications for memory and code. Students reported little perceived implications on their memory when coding with the websites. In terms of code, students and professionals adopted online code and encountered multiple problematic code.

- **Chapter 4: A survey study of websites' usage and the implications.** This chapter designs a survey study that tests the generalisability of the interview outcomes to students and professional programmers. The survey outcomes stress the critical role of memory for programmers and show that programmers use the websites with little perceived implications on their memory. In addition, the outcomes reveal that programmers engage in using online code and face most of the mentioned problematic online code.

- **Chapter 5: Strategies for using websites to support programming and their impact on source code.** The qualitative study examines programmers' coding activities in-depth when using the Web to produce the outputs. Programmers' coding with websites was recorded to provide insights into the programmers' resources and their in-practice coding activities. The results confirm that programmers use websites to seek and utilise syntax. When analysing their source code, the results reveal issues with programmers' code, including non-executable and incorrect code; some issues matched the problematic online code mentioned earlier.

- **Chapter 6: Discussion, conclusion and future work.** Understanding the usage of the Web for coding and the possible implications should help to understand programmer coding better and mitigate the possible effects. The concluding chapter considers the extent to which the studies described in this thesis further the understanding of the implications of coding with websites on coding and how the results could inform programmers, organisations, and educators. A final section highlights the directions for future research that would maximise the understanding of memory's role in programming and the problematic online code.

1.3 Overview of studies

Figure 1.1 gives an overview of the studies conducted in this thesis. *Study one* interviewed and collected data from 18 novice and professional programmers to perform semi-structured interviews. *Study two* included 276 novice and professional programmers with the interview results to test and confirm interview findings using a survey. The final study, *Study 3*, performed an online experiment that collected observations through video recordings of programming sessions with 10 participants, collected their source code for further analysis, and concluded with an interview.

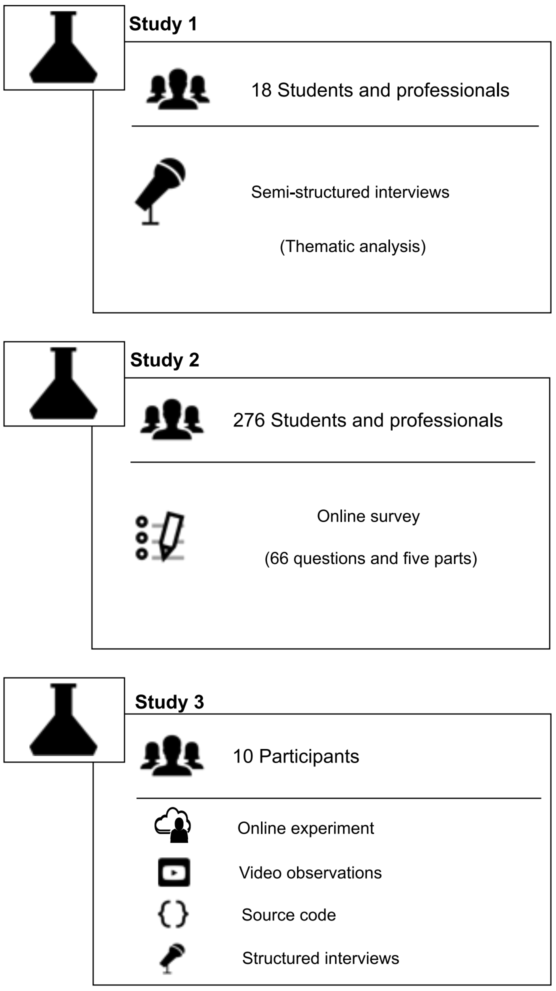


Figure 1.1: Overview of studies carried out in this thesis

1.4 Contributions

This thesis makes contributions to software engineering by providing new knowledge concerning programmers' use of the Web when coding, of their perceptions of the Web and of their Web use, and the impacts on code. These contributions are made through a combination of qualitative and quantitative methods – a literature review and three empirical studies. The main contributions of this thesis are:

- *Literature review*: A review collates and summarises previous work studying how programmers' memory may be affected by the use of coding resources, particularly websites; their online search strategies; their online code reuse and associated implications for code outputs. To the best of the author's knowledge, this is the most comprehensive review of work in this area.

- *Investigation into the resources programmers use while coding*: The exploration of coding resources across studies in this thesis show that programmers have different preference for resources based on their level of programming experience. The results also stress that students and professionals use the websites significantly during coding. The preference for coding with websites is more evident for students than professionals who, along with the Web, use existing code-based resources or their colleagues. Searching the Web is common for acquiring syntax. Websites usage usually starts with the Google search engine; then programmers choose different website types based on their experience or the programming tasks. This thesis establishes the prevalence of websites for coding and stresses that programmers use websites to find and reuse code. It is hoped that this research will contribute to a deeper understanding of the role of the Web in coding.

- *Investigation into the perceived impacts of websites on programmers' memory*: The interview and the survey studies (see Chapters 3 and 4) investigated the role of memory when coding and the perceived impacts of using the websites on the programmers' memory. Memory was found to play a role in coding as programmers express using their memory to recall the needed information when coding. Along with the memory, programmers also use websites to offset the missing information and consider them a supporting companion. They report little perceived impacts on their memory when coding with the Web. This thesis shows that websites serve as supporting coding resources along with memory. The study of programmers memory emphasize the benefits of memory for coding. This is the first work to focus on the programmers' memory and the possible implications of coding with websites.

- *Investigation into the implications on the code*: The thesis explored the use of online code and investigated its implications by examining programmers' encounters with code issues found online and their resulting code for possible implications. Programmers reuse online code and report encountering multiple problematic code while coding with the websites. The advance in programmers' experience allows more confidence in reusing online code, but with a high possibility of encountering problematic online code. In addition, programmers' observations and analysis of their resulting code reveal that coding with websites produces issues with their code and propagates problematic code into the resulting code. While programmers search websites and reuse code, accessing websites does not necessarily produce correct code. This thesis presents eleven problematic code that programmers could encounter while using websites, some of these problematic code were used by programmers, and shows that coding with the websites code produce incorrect or non-executable code. The list of problematic code provide awareness for stakeholders across software engineering, especially programmers

1.5 Publications

The work undertaken in this thesis is also reported in a workshop paper, journal submission and conference submission. These items are presented below in chronological order.

1. Clinch, S., Alghamdi, O., & Steeds, M. (2019). Technology-induced human memory degradation. *Proceedings of Creative Speculation on the Negative Effects of HCI Research. Workshop at The ACM CHI Conference on Human Factors in Computing Systems (CHI 2019).*

2. Alghamdi, O., Clinch, S., Skeva, R., & Jay, C. How are Websites Used During Development and What are the Implications for the Coding Process?. *Journal of System and Software*. Available at SSRN 4206818.

3. Alghamdi, O., Clinch, S., Alhamadi, M., & Jay, C. Novice programmers strategies for online resource use and their impact on source code. *16th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE 2023).*

1.6 Terminology

In this thesis, specific terms were used when describing the resources.

- **Documentation** will represent the documentation websites such as Oracle documentation.

- **Memory** is used to refer to human memory, typically semantic memory (long-term memory for facts/knowledge).

- **OSS** Open Source Software.
- **Q&A** questions and answer websites such as Stack Overflow, Reddit, and Quora.
- **SO** will represent the Stack Overflow website.
- **tutorial** will represent the programming tutorial websites such as “Tutorial-Point”, “W3school”.

Chapter 2

Related Work

This chapter summarises current knowledge with respect to implications of website use on human memory (Section 2.1), use of resources during coding activities (with particular focus on website use and information seeking, Section 2.2), and on code-reuse practices, benefits and risks/costs (Section 2.3).

2.1 Impacts of technology on human memory

The Web, and other technologies, allow readily-accessible storage and retrieval of information. As a result, researchers have noted that they are used as a transactive memory partner[1] [Wegner, 1995, Ward, 2013], a practice that is also referred to as a form of "cognitive offloading" [Risko & Gilbert, 2016, Wegner, 1995, Ward, 2013]. While offloading information storage from the mind to the Web has obvious benefits (availability, capacity) [Sparrow et al., 2011, Wegner, 1995, Ward, 2013], concerns have also been raised about possible negative consequences for human memory. Empirical studies indicated that offloading reduces information recall for the offloaded information and increases false recall [Kelly & Risko, 2022, Grinschgl et al., 2021, Lu et al., 2020], with less information kept internally [Ward, 2013]. Furthermore, individuals who source information from the Web engaged in source attribution errors, i.e. they confuse information from online stores with their knowledge [Ward, 2013], giving them false confidence in their abilities [Fisher et al., 2015, 2022]. Moreover, the extent of such implications for programmers' memory, a resource used for coding, was not

[1]In a transactive memory system, members share responsibility for storing and retrieving information, with each holding unique information and all maintaining an awareness of the information held by others [Wegner, 1987]

clearly defined.

2.1.1 Positive impacts

Use of the Web, and particularly search, is a valuable tool for information seeking [Rose & Levinson, 2004]. Constant availability and timely response of online information retrieval have led psychologists to dub the Web a "supernormal" transactive memory partner, better able to remember information than any human colleague, friend or family member [Ward, 2013]. Thus, use of the Web allows individuals' access to information without the need to store and recall it, freeing up capacity to process further information [Sparrow et al., 2011, Wegner, 1995, Ward, 2013]. Furthermore, as Internet browsing gets faster (e.g. with improvements in users' home and mobile connectivity) online information retrieval becomes even more responsive, and improved responsiveness itself appears to increase cognitive offloading [Grinschgl et al., 2020].

There have been limited studies of the benefits of Web use on memory specifically. However, studies of digital offloading (saving information to file) indicate offloading improved memory for future information [Storm & Stone, 2015]. Recent narrative reviews [Cecutti et al., 2021, Heersmink, 2016] argue that although technology may change cognition, there is insufficient evidence to conclude that these changes are negative. Similarly, studies of pretesting (known to strengthen subsequent learning) have shown no negative effect when participants use Google to help answer the pretest questions [Storm et al., 2021]. Other studies suggest that negative memory effects observed in studies of digital technologies may be the result of other factors (e.g. cognitive load) [Kahn & Martinez, 2020].

2.1.2 Negative impacts

Anecdotal accounts have expressed concern about the deterioration of human abilities as a result of website use, such as the diminishing ability to concentrate [Carr, 2008]. Early empirical evidence from Sparrow et al. [2011] led to the coining of the term "Google Effect" to describe these effects, but the term refers to the Web more widely (i.e. not just Google) and is often used to refer to multiple different effects on aspects of cognition. Sparrow et al. [2011] conducted four experiments on the effects of technology on information processing (Experiment 1) and recall (Experiments 2 and 3). The first experiment asked 46 undergraduate students 16 easy and 16 hard trivia questions, followed by a Stroop task[2] with stimuli that was either computer-related or unrelated. Participants took longer to detect the colour of computer-related words than unrelated ones, the response time increasing with the difficulty of the questions. Sparrow et al. [2011] suggests that not knowing the answer to trivia questions primed participants to think of the computer and the Web, interfering with their processing of the stimuli. The second and third experiments presented undergraduate students with trivia statements to remember. Half the participants were told that the statements would be saved on the computer and half that they would be deleted. Results suggested that the participants who thought the information would be deleted remembered it more consistently than the participants who thought it would be saved. Sparrow et al. [2011] have concluded that participants offloaded the need to remember information to the computer rather than their using their own memory. Furthermore, participants in the third study remembered the location of the information better than the actual information.

Schooler & Storm [2021] replicated [Sparrow et al., 2011] study and conducted two experiments involved 162 undergraduate students assigned either to a condition in which they had access to the information later or to another without access to information. They found the "Google Effect" present in the former condition (access to the information). In addition, Fisher et al. [2022] investigated the implications for websites by conducting five experiments sharing a similar structure. Using incentives to induce participants to learn about specific topics, they assigned participants to one of the two conditions that govern the ways of solving questions about the topics: Web search or no-Web search. In the first experiment, participants first solved the questions, then took a quiz based on the questions. The second and third experiments shared a similar structure, but with changes, such as a distraction for the no-Web condition, time constraints for topic learning and pauses for thinking. The last experiment included changes in the second condition, to search online using a ready link instead of searching the Web. The findings from these experiments confirmed that using the Web impacted participants' memory by reducing their ability to store new information. Another experiment of 150 participants using Amazon Mechanical Turk revealed that the information of the kind easily retrieved by search engines was more poorly remembered in a recall test, particularly by those with better searching skills [Macias et al., 2015]. The "Google Effect" has also been studied in a different context, using mobile applications. In their experiment, Kahn & Martinez [2020] showed their participants

[2]In a Stroop task participants are presented with word stimuli written in coloured text. Some variants of this task involve asking participants to respond with the word, some with the colour of the text, and some vary the request with the stimuli.

pictures with text for a specified time, then performed a memory test. They argued that no implications were found similar to the “Google Effect”. However, one limitation of such study is the focus on mobile applications. While factors, such as participants selection, could limit findings from Sparrow et al. [2011], other evidence [Schooler & Storm, 2021, Fisher et al., 2022, Macias et al., 2015] support and extend implications from using the Web.

In another aspect, the Web provides access to the information in a way that limits processing it and could cause learning to deteriorate [Loh & Kanai, 2016]. The Web could also affect users’ judgement of their memory. Using nine experiments, Fisher et al. [2015] empirically investigated how online information inflates user knowledge and concluded that searching the Web increases the users’ estimation of their knowledge. A recent paper investigated the use of the Web as an external resource and found implications for API learning [Gao et al., 2020].

One of the suggestions to overcome or limit Web usage implications is engaging the memory. Giebl et al. [2021] performed a controlled experiment with 240 non-CS undergraduates, some of whom had prior programming experience and some who did not. After a basic primer on core concepts, participants were presented with a programming task that went beyond those initial concepts. Half the participants were required to attempt the task before consulting the websites, whilst the other half were immediately permitted to search for the additional information needed. Following the task, participants were tested on both the initial concepts and those additional concepts required to solve the task. Results indicated that students who attempted the task before consulting the websites were better able to retain concepts, and that this effect was strongest in those with prior programming experience. Such investigation continued in a recent paper [Giebl et al., 2022] as they implemented three experiments, each with multiple conditions. They found that answering questions first and then searching Google revealed more learning outcomes than using Google first to look for the answer. This stresses the importance of reflecting on the recall process and recognising the memory before searching, which could enhance memory in general as well as for the materials to be learned.

2.1.3 The usage of programmers’ memory

Multiple investigations have explored the role of programmers’ memory when coding. In their model, Shneiderman & Mayer [1979] stated that as part of long-term memory were semantic memory containing concepts unrelated to any programming language, and syntactic memory containing precise information. Programmers recall and retain information from both semantic and syntactic memories during programming [Bateson et al., 1987, Shneiderman, 1976]. They may not directly recall syntax but linked it with the semantic information to ease code comprehension in future encounters [Parnin & Rugaber, 2012]. Semantic memory could be more vital in programming skills and differentiating between different experiences of programmers [Bateson et al., 1987].

Memory is used to recall information during coding [LaToza et al., 2006, Ichinco & Kelleher, 2017, Ormerod, 1990]. In an empirical investigation, code snippets were given to programmers with different experiences, and they recalled the code later. They found that programmers who program frequently recalled code from their memory more than those who either program less frequently or were not programmers [Ichinco & Kelleher, 2017]. In an interview with developers, they considered conceptual knowledge necessary for memorising and recalling [Krüger & Hebig, 2020]. In another experiment, Kelleher & Ichinco [2019] developed a model for API learning, and suggested that programmers use long-term memory to retrieve search keywords and evaluate results. Factors that influenced the programmers' ability to recall the code included understanding the code and the familiarity with the code [Fritz et al., 2007, Krüger et al., 2018]. In addition, advancing expertise is supported by using the memory [Ericsson & Kintsch, 1995], requiring more time and practice [Simon, 1980] especially in a complex matter such as programming.

2.1.4 Implications for this thesis

Current evidence indicates that (a) individuals offload responsibility for storing information to the Web, (b) this offloading may impact individuals' memory (and metamemory that referred for information about the memory), and (c) memory is used by programmers during development. The Web is also one of the resources programmers use for coding as external memory to avoid remembering syntax [Brandt et al., 2009]. The previous implications of using the websites, along with the complexity of coding [Détienne, 2001], suggest that programmers' memory is not isolated from the impacts of coding with the websites. While the possible implications of using websites on the programmers' memory remain undiscovered, this thesis aims to discover whether programmers perceived any implications of coding with the websites.

2.2 Programming resources

Complex coding and multiple languages [Détienne, 2001] make it inevitable that programmers will search information through various resources. One example of coding resources is the traditional textbooks that students use. The Web has supplemented using such resources, as a study showed by asking 338 undergraduate students at public and private universities in the Philippines about their use and perceptions of educational materials when programming [Lausa et al., 2021]. The study found low rates of book or e-book use ($< 30\%$) and highest rates of use for YouTube (67-86%) and friends/peers (53-72%). The participants attributed the low-level use of books to the challenges of finding code compared to online resources or peers. In a similar paper, a survey of 190 students indicated that students took advantage of the websites more than textbooks during coding projects, caused by the former's quick response [Bringula, 2017].

Face-to-face communication is another way to seek and share information, especially for industry programmers, given human nature [LaToza et al., 2006, Maalej et al., 2014, Hucka & Graham, 2018, Ko et al., 2006]. In a survey study of $1,477$ professionals [Maalej et al., 2014], participants reported seeking knowledge about software from other people and sharing their own knowledge by the same means. However, they also reported that websites were more suitable for accessing the information than other resources, such as colleagues. Students also seek help from their peers and teachers when coding [Bringula, 2017, Lausa et al., 2021] but report difficulties in understanding the code, preferring such websites as YouTube [Lausa et al., 2021]. Due to limitations of face-to-face communication with professionals (e.g. the programmers' availability and experience level), they consider accessing the established codebases. A study observed developers at Google making an average of 12 queries per day for established codebases [Sadowski et al., 2015]. In a study of their behaviours while solving an introductory exam, students also considered their prior solutions [Nygren et al., 2017]. Still, the code-based content could be limited, creating the need to explore other avenues. In a survey study, developers chose the Web first to find code on such websites as Stack Overflow and GitHub [Hucka & Graham, 2018]. In an observational study, students (both CS and non-CS) preferred using online code [Koenzen et al., 2020]. In this regard, resorting to resources other than websites was challenging for programmers who preferred to refer to the websites when coding.

The convenience of seeking information, including ease of use and time, determines the choices of resources [Connaway et al., 2011], and students prefer immediate answers that match their expectations [Wong-Aitken et al., 2022]. Thus, websites offer a convenient resource for programmers, and students and professionals make substantial use of them when coding [Acar et al., 2016, Brandt et al., 2009, Xia et al., 2017]. Brandt et al. [2009] empirically performed two studies, observing 20 students during coding, and stressed that programmers employ the websites as an aid in finding the required information. A recent survey of 103 developers reported that search engines, Question and Answers (Q&A), and online code repositories were the most visited resources for architectural information [de Dieu et al., 2022]. Search engines specifically were used for such activities as program comprehension and code reuse [Hora, 2021, Maalej et al., 2014, Xia et al., 2017], debugging [Hora, 2021, Xia et al., 2017] and background knowledge acquisition and reference (e.g. syntax) [Hora, 2021, Xia et al., 2017].

In addition, several studies have begun to examine programmers' activities to infer the role of websites in coding. Astromskis et al. [2017] quantitatively collected behaviours of six industry programmers for six months, then categorised their coding activities, including working on and compiling the code and searching websites. They affirmed that online visits were complex and caused coding interruption to search. Similarly, G. R. Bai et al. [2019] focused on the "Regular expressions" programming tasks and used video records of 29 students to explore their programming activities. They identified programmer events as visiting website, searching and reuse code. These investigations emphasis that the Web is integrated into programmers' coding and considered a part of the process.

The Web encompasses many specialised programming websites, such as Q&A websites, designed for user posts and the exchange of information. Programmers' coding activities were qualitatively observed to explore the resources used, and technical blogs claimed to be the most used [Li et al., 2013]. Given the rise of new websites tailored to programmers' needs, such as Stack Overflow, these results seemed obsolete. Stack Overflow is a popular social website to which programmers, especially professionals, can resort for help and information while coding [Nasehi et al., 2012]. While interviews with 20 students revealed low-level use of Stack Overflow for help [Bhasin et al., 2021], the website continues to attract programmers' attention. In 2022, the website received 23 million questions, 34 million answers and 88 million comments [*StackExchange Data Explorer*, 2023]. In a lab study, Acar et al. [2016] assigned 54 developers to four conditions in which to solve four security-related tasks: documentation, Stack Overflow, specified books or free online search. The findings highlighted that the developers preferred Stack Overflow for resolving problems over other resources because of the quick answers. Fuchs et al. [2014] also supported this view, capturing students' interactions and finding that Stack Overflow was the most visited website. Students, who were engaged in data-related courses, further consider using Google and Stack Overflow to strengthen their coding while using established codebases, as Han et al. [2020] found in a lab-based experiment. Similar findings were observed by Abtahi & Dietz [2020], who empirically found that professional programmers learning new programming languages used Stack Overflow. Therefore, the rise of Q&A websites, especially Stack Overflow, has become one of the main venues to which developers resort when coding.

The contents on Stack Overflow benefited programmers, particularly novices, by providing knowledge and clarifying contents [Treude et al., 2011]. Likewise, Abdalkareem et al. [2017b] extracted projects from the GitHub website that refer to Stack Overflow and investigated the source code commits. They presumed that Stack Overflow was used for providing knowledge that aids programmers in fixing bugs, enhancing their code and adopting code. Inside Stack Overflow, programmers can post questions and answers, comment and vote on others' questions, receive answers and browse the answered questions. In addition, the website applies a reputation system for the registered user, with user activity and other users' votes on his or her posts determining points and badges. Furthermore, users can use upvote or downvote features. The upvoting helps place the answers first on the Web page, giving both the owner and the voter more points; conversely, the downvoting contents will appear at the end of the Web page and remove points from both the post owner and the user who downvoted [*Stack Overflow privileges*, 2023]. This reward system provides more reputation points to the user who contributes to the community with knowledge implying their expertise.

Programmers resort to websites for information during coding and prefer Stack Overflow. However, the current research around using the resources during coding needs more detailed exploration. Programmers with various experiences use different resources, and investigating their resource usage during coding is not well established. In addition, a detailed examination of programmers' in-practice coding with the websites remains unaddressed. Therefore, this thesis aims to explore the types of resources and websites students and professional programmers use when coding (see Chapters 3 and 4) and investigates in depth the activities used while coding with the websites (see Chapter 5).

2.2.1 Information searching activities

Information searching is a common and fundamental process conducted along with coding, even when the information is familiar, and augments the missing information [LaToza et al., 2006, Singer et al., 2010, Li et al., 2013, Sadowski et al., 2015]. Programmers devote their time to searching since it is a complex task comprising analyzing and choosing from the search results [LaToza et al., 2006, Li et al., 2013]. Previous research focused on programmers' searching activities. Starke et al. [2009] conducted a lab-based experiment, with 10 industry professionals and graduate students working with familiar change tasks, and video recorded the session and audio recorded discussion between participants and the researcher. Their observations related to searches within the Eclipse platform, and revealed that participants' search results were unrelated, and participants infrequently examined the results, encountering challenges in reaching information. Another study surveyed 396 professional programmers and used a log analysis of 27 for their search activities [Sadowski et al., 2015]. The data suggested that searching occurs within codebases, not online, for code examples and functionality. Such results failed to address the role of online resources and were based on programmers preferring to access existing established codebases. Searching could be investigated based on the information foraging theory, similar (as multiple studies described it) to animals tracking their prey (the reader can read more about such research in [Ko et al., 2006, Chattopadhyay et al., 2018]).

Currently, searching while coding is coupled with websites considered vital resources for coding. Brandt et al. [2009] investigated the usage of websites for development by observing 20 students coding and manually analysing 300 search-query sessions. They found that programmers used websites to support learning and clarification. Similarly, using direct observations of 20 developers and self-reported observations from 25 others, it was believed that programmers search online for information on ways of doing things, using the online search as a reminder when they could not recall information and look for information to resolve issues [Gallardo-Valencia & Sim, 2011]. In exploring programming activities, Wang [2017] recorded video and audio data, collected source code and conducted a questionnaire with seven participants coding, using their choice of resources. They found behaviours related to the online search, including visiting and searching the websites and examining results. Using a similar methodology, Gao et al. [2020] affirmed that programmers devoted enormous time to searching, visited web pages, even the previously visited ones. In addition, Koenzen et al. [2020] performed an observational study with eight graduate students from computer science (CS) and chemistry, solving three tasks with different level of difficulty. They audio-recorded participants' discussions and recorded their screens, and found that searching consumed 18% of their participants' time, involving repeating search even if it was searched recently. In an experiment regarding the outcomes of using the Web, G. R. Bai et al. [2019] screen-recorded and collected logs for 29 students while solving "Regular expressions" tasks. They found that participants searching using documentation and tutorial websites produced fruitful searches compared to Q&A websites.

Programmers commonly search websites to reuse code, as a survey of 69 developers revealed [Umarji et al., 2008]. A similar study with 69 participants, comprising students and academic members, affirmed searching the Web for code [Hucka & Graham, 2018]. In a study that observed 60 developers, interviewed 12 and surveyed 235, one of the search objectives was reused code snippets [Xia et al., 2017]. Code reuse activities were also supported when extracting projects from GitHub that referred to Stack Overflow [Abdalkareem et al., 2017b]. In a more detailed investigation, Fuchs et al. [2014] studied students' activities while coding, using browser logs and screen recording, and emphasised that most searches were for code. Another recent study extracted and analysed over 1.3 million queries from programming websites, such as Stack Overflow, and found that nearly half of these queries concerned code reuse [Hora, 2021].

Searching activity is not always pleasant and successful, and could result in many challenges. In a survey, 205 participants expressed the difficulty of searching and browsing online where searching required assistance [Escobar-Avila et al., 2019]. In a similar survey, developers exhibited problems retrieving relevant information along with required time to assess high volume of information available online [de Dieu et al., 2022]. The process of searching online consumed time, including searching for easy tasks with unproductive searches [Wang, 2017]. In addition, online code with poor quality induces more time to find the required results, as [Xia et al., 2017] found empirically. Students, in particular, stressed more over the difficulties of searching online, as they have a shortage of vocabulary aiding the search, resulting in unsuccessful searches [Wong-Aitken et al., 2022].

Collectively, these studies outline the fundamental role of searching activity on websites considered important for programmers to search to seek knowledge, especially code, and the difficulties around searching activities. However, the previous investigations of searching activity lack finely-grained details regarding activities of searching the websites while coding. Searching is often coupled with websites and conducted throughout coding, where exploring searching activities is beneficial. This thesis uses the observation method to examine the actual searching activities of programmers during coding, inferring the role of websites in such a process and exploring the possible challenges (see Chapter 5).

2.3 Code reuse

A computer program contains syntax representing a text written by humans, based on specific principles, representing specific meanings and purposes, using explicit language [Détienne, 2001]. The notion of code reuse has received attention in software engineering since the early 1950s [Détienne, 2001]. Initially, code reuse was motivated by commercial interests, but as software complexity increased, it became more common [Détienne, 2001, Krueger, 1992] to minimise time and reduce mental efforts [Krueger, 1992, Krinke, 2008]. Programmers reuse code for various reasons, such as the need for templates used for structural purposes, as stated by Kim et al. [2004] on observing and examining editing activities for nine industry programmers using logs.

Websites were used to search, assess and reuse suitable code [Brandt et al., 2008, 2009, Abdalkareem et al., 2017a]. A study by Abdalkareem et al. [2017b] extracted projects from GitHub with references to Stack Overflow, and suggested that copying code was one of Stack Overflow's uses. Other research further established that code reuse was commonplace when using and searching websites [Brandt et al., 2009, Umarji et al., 2008, Hucka & Graham, 2018, Xia et al., 2017, Han et al., 2020]. Exploiting online code can provide improvements and resolve issues [Abdalkareem et al., 2017a]. Another study found from programmers' behaviours that copying online code resolves errors while debugging [Ciborowska et al., 2018].

Online code reuse activities were further explored using a ready data set of developers' behaviours [Ciborowska et al., 2018]. They found that participants edited the online code copied, but delayed checking it. These explorations did not involve concrete code-reuse activities; rather, their copy explorations were based on periods of inactivity, assuming paste activities occurred after the period of copy activities, which could easily interfere with other copying from other resources. In a similar study, Jacques & Kristensson [2021] used Amazon Mechanical Turk and recruited 200 participants to solve two tasks. They found, from analysing a sample of participants, that their activities involved typing or copying code, without specifying the source, and their code familiarity increased when solving the second task. Another study by Han et al. [2020] conducted a lab-based experiment using a platform and an eye-tracking feature with 39 participants solving five tasks, to identify quality issues on a given dataset. They found that participants considered using code from previous tasks. The reader can refer to other research that expressed interest in a specific programming activity, like debugging, to inform an educational field [Jadud, 2005, 2006, Watson et al., 2013, Carter et al., 2015, 2017, Becker, 2016].

Reusing code, especially from websites, is a common coding activity. However, the explorations of reusing online code are superficial, ignoring programmers' underpinning activities. There is a paucity of research exploring code reuse from the perspective of various programmers using different resources. In addition, exploring in depth the activities of code reuse from websites, along with the challenges, receive little attention. This thesis explores programmers' use of resources, including websites, to reuse code (see Chapters 3 and 4) and the code reusing activities from the websites and the possible challenges (see Chapter 5).

2.3.1 Code reuse: benefits and drawbacks

Code copying-and-pasting activities help in coding [Kim et al., 2004] but involve many challenges and efforts [LaToza et al., 2006]. The notion of code reusing—code cloning using code detection techniques [Krinke, 2008, Roy et al., 2009]—involves considerable debate about benefits and consequences. Prior research has considered code-cloning activities with online open-source systems (OSS), containing code governed under a licence enabling programmers to use it, such as the Apache Web server, Mozilla Firefox and LibreOffice. Two core issues drive discussions regarding reuse of OSS code: stability and maintainability. Krinke [2008] studied OSS stability by observing the evolution of cloned and non-cloned code in five OSS for over three years. They concluded that the change activities (such as adding and deleting code) occurred more for the non-cloned code than the cloned code, making a claim that it was more stable than the non-cloned code. Kapser & Godfrey [2008] performed two case studies using the Apache Web server and Gnumeric spreadsheet, and believed that most of their observed code cloning had positive software maintainability and stability outcomes.

Although multiple papers have argued that cloned code with OSS does not pose the threat of any serious negative outcomes, others revealed that OSS-cloned code poses consequences relating to code changeability and stability. Lozano & Wermelinger [2010] tracked the cloned code in five OSS for over 2 years to observe its stability and found that the cloned code had less stability with the possibility of remaining cloned. In a similar study, Mondal et al. [2018] applied seven methodologies from previous papers that investigated the stability of cloned code and used two clone-detection to check three types of clones across different procedural and Object-Oriented programming languages. They affirmed the previous findings that cloned code is changeable and unstable more than non-cloned code.

The research on the risks of reusing OSS code is still inconclusive, and the possible risks are unavoidable. However, social websites create another challenge for code reusing. Websites like Stack Overflow allow programmers to communicate, ask and answer questions and comment on each other's answers. The posted contents contain code snippets written by programmers with various experiences. These presumably made the use of online code snippets different from OSS, due to the scale of using websites and the difference in contents, as online code snippets tend to be smaller [Misu & Satter, 2022]. In addition, the source of OSS code tends to be known and likely trusted to be written by experienced programmers, unlike code snippets posted by various users. Thus, the status of online code snippets, especially code posted on social websites, remains unknown and an aspect of the main focus of this thesis, namely, the consequences of reusing online code.

2.3.2 Risks of reusing online code

Stack Overflow is one of the most common social Q&A websites for programmers [Acar et al., 2016, Fuchs et al., 2014]. Code in Stack Overflow is under a Creative Commons Attribution (CCA) licence that allows programmers to share and adapt contents with source attribution [*Creative Commons: Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)*, n.d., *What is the license for the content I post?*, 2023]. With the predominant use of the code posted on Stack Overflow, users acknowledged their concerns about reusing it [*Do I have to worry about copyright issues for code posted on Stack Overflow?*, 2008]. In response, multiple investigations studied Stack Overflow code. Meldrum et al. [2020] extracted Stack Overflow code and explored code quality. While the extracted code contains issues, they claimed that the quality of Stack Overflow code is not worrying. Other investigations countered such a conclusion and stressed that reusing the code from Stack Overflow can produce issues that potentially affect code quality. A study by T. Zhang et al. [2018] analysed over 200K

Java and Android Stack Overflow posts, and found that 31% of the Stack Overflow posts could produce unpredictable behaviours that violate API usage. In a similar study, Abdalkareem et al. [2017a] collected code from Stack Overflow, compared it to the Android applications, and concluded that code reuse from Stack Overflow could potentially produce quality issues in the Android applications. In an empirical analysis of OSS projects with links to Stack Overflow, and 453 survey entries from OSS developers, the findings revealed that the presence of low-quality code on Stack Overflow inhibited reusing the code [Wu et al., 2019]. These findings along withe empirical study that extracted Stack Overflow code [Baltes & Treude, 2020], corroborate the need for more attention, especially when reusing Stack Overflow code.

Stack Overflow implemented specific metrics that detected issues and facilitated reusing creditable code, which prior research assessed. Meldrum et al. [2020] believed that the code with issues would be given fewer votes on Stack Overflow using downvotes as they argued that such a feature encourages using the answers with high scores and fewer violations. A similar paper [Fischer et al., 2017] empirically extracted code from Stack Overflow and found that high numbers of votes were associated with secure code. However, others reached opposite conclusions from investigating Stack Overflow metrics. In analysing Stack Overflow code, T. Zhang et al. [2018] argued that the number of votes did not necessarily predict the quality of the code on the site. Programmers could still reuse answers from Stack Overflow with lower votes or were not accepted (Stack Overflow metrics were not applied), as found empirically by [Wu et al., 2019]. The inadequacy of such metrics could lead programmers to use the code with issues from high-voted answers, as Meng et al. [2018] empirically found from extracting and analysing Stack Overflow code. This includes answers with contents contrary to the question [Wu et al., 2019] or insecure code, as found empirically extracting code from Stack Overflow [Chen et al., 2019]. Overall, code with Stack Overflow still contains issues, even with the metrics applied.

Code could move within Stack Overflow, as Baltes & Treude [2020] exhibited by finding code duplicated between different Stack Overflow threads. Code could also move from Stack Overflow to other websites and projects. Two similar experiments [T. Zhang et al., 2019, Yang et al., 2017] used clone-detection tools to check code similarities between GitHub and Stack Overflow, and found that code clones propagate between the two websites. In a practical example, code snippets presented in Stack Overflow helped produce the Android applications. Abdalkareem et al. [2017a] used clone detection to examine the code similarity between Stack Overflow posts and

Android applications, and concluded that contents are used between Stack Overflow posts and the applications. Furthermore, code from Stack Overflow appears in the OSS. A study by [Ragkhitwetsagul et al., 2019] conducted clone detection between 72,365 code from Stack Overflow and 111 OSS projects from the Qualitas, and found that code cloned from OSS to Stack Overflow. Moreover, code can be reused from tutorial to Stack Overflow. A study of duplicate code between five Android tutorial websites and 3 million posts Stack Overflow with Android tags found more than 2000 duplicate instances [Nishi et al., 2019].

The risk of adopting issues from online is not particularly associated with the use of Stack Overflow. W. Bai et al. [2019] extracted security vulnerabilities from Stack Overflow, found GitHub projects that contain similar code clones, and contacted project authors to survey 133 and interview 15 participants. They found that most participants acknowledged that adopting online code introduces issues, and 63% of respondents to their survey reported using online forums to write vulnerable projects. Additionally, programmers were unaware of their responsibility when copying code from the Web [Sojer & Henkel, 2011], and tended to postpone testing the code copied [Brandt et al., 2009, Ciborowska et al., 2018].

Overall, the difference between the code on OSS and Stack Overflow is clear, given the nature of the contents and the imposed licence. The risk involved in copying online code is far more than that from the code within OSS. Online code on Stack Overflow will likely contain issues undetected by the implemented metrics, and such issues are also likely to propagate to different venues. Thus, it is essential to address the issues and consequences around copying online code, certainly from Stack Overflow.

2.3.2.1 Online problematic code

Online code is prone to multiple issues that could deteriorate code quality. In a recent investigation of online code quality using two systematic reviews, issues with the code, such as bugs and security vulnerability, were the topics discussed most [Ndukwe et al., 2023]. Literature on online code has highlighted several issues primarily related to Stack Overflow. Reusing and posting online code on Stack Overflow without proper attribution could introduce licence violation issues. An et al. [2017] checked licence violations between Stack Overflow posts and Android applications, and discovered instances of reusing code between Stack Overflow and Android applications with licence violations. Another paper surveyed 453 developers, and found 80% were unaware of the imposed licence when taking code from Stack Overflow [Wu et al., 2019]. In addition, the outdated problematic code concerns the versioning of the programming language, where the new version could dismiss previous functions as deprecated or outdated. The survey with Stack Overflow answerers and visitors suggested that outdated code is presents on Stack Overflow, with answerers rarely correcting such code [Ragkhitwetsagul et al., 2019]. Similarly, H. Zhang et al. [2021] extracted over 52K answer threads from Stack Overflow with user comments that mentioned outdated code. They found that 58.4% of the answers were outdated when they were written, and most of these answers were not updated.

On another issue, online code could contain bugs that may propagate to programmers' code. Ragkhitwetsagul et al. [2019] confirmed that code with bugs is one of the issues Stack Overflow visitors faced, as mentioned in their survey. The buggy code in Stack Overflow could be introduced to different venues. Abdalkareem et al. [2017a] empirically investigated reusing Stack Overflow code in Android applications, revealing that such reuse increases the chance of introducing bugs. Online code could further suffer from incompleteness content that does not deliver the full purpose of the code. Such incomplete code could be regarded as a code fragment with a segment of code lines [Roy et al., 2009]. To study the presence of code fragments online, Treude & Robillard [2017] examined 120 code fragments from Stack Overflow, surveyed 321 developers about these code fragments, and indicated that over half of these fragments require further information. Similarly, Misu & Satter [2022] extracted 276,547 code snippets from Stack Overflow, and found that most of the code within Stack Overflow was of limited reusability due to code fragment. Code fragments seemed to be an issue for students, through analysis for their discussion within a forum for their course [Piwek & Savage, 2020].

The literature discussed security vulnerabilities in the Stack Overflow code. W. Bai et al. [2019] used specific security vulnerabilities issues to search Stack Overflow for the code snippets with these issues and GitHub to survey and interview the author of these code snippets. The code authors acknowledged that the activity of reusing online code caused the security vulnerabilities, and they did not have the required knowledge to deal with it but delegated such responsibilities to the user. In another study mentioned earlier, Acar et al. [2016] performed a lab study assigning 54 developers to four resource conditions while solving security-related problem, then analysed their final source code. They concluded that participants introduced more vulnerabilities to their code when using Stack Overflow. That said, the existence of security issues within Stack Overflow may not be significant. In an investigation, Licorish & Nishatharan

[2021] used a FindBugs tool to check for security vulnerabilities, and analysed 8,010 Stack Overflow code snippets. They found security vulnerabilities were only present in a very small number of posts. Still, security vulnerabilities of online code pose threats to Stack Overflow contents that could propagate to other venues. Fischer et al. [2017] empirically identified over 4,000 code snippets from Android posts on Stack Overflow, then matched the extracted code in 1.3 million Android applications. They assumed that 97.9% of Android applications that contain security code snippets from Stack Overflow also contained at least one security-related problem.

The extent of problematic code in the programmers' source code has received attention. In an experiment with six programmers, Astromskis et al. [2017] assessed the contents of their participants' source code using a tool that measures source code metrics, including size and complexity, then links it with participants' online usage. They suggested that complicated source code necessitates further Web research. In addition, Acar et al. [2016] analysed participants' written source code to check security implications and found that participants coding with Stack Overflow produced working code but with security vulnerabilities, unlike participants who use documentation.

Code understanding functions as an aspect of code quality [Kirk et al., 2020, Börstler et al., 2017] that could impact the propagation of problematic code. Programmers engage in understanding code [Schröter et al., 2017, Kim et al., 2004], and code reuse would be challenging without it [Wu et al., 2019]. Understanding the reused code is not a principal activity for programmers. In an empirical study of code understanding, Maalej et al. [2014] interviewed 28 and surveyed $1,477$ professional programmers, and discovered that programmers ignored understanding copied code for a variety of reasons including a focus on task completion. In addition, programmers' experience level is also another factor affecting code understanding. Novice programmers struggle more when comprehending the code; their limited knowledge of the programming language makes such a process harder [Escobar-Avila et al., 2019]. In an lab study of novices and professionals, they found that while experts tend to address the causes of a software problem, novices are more likely to focus on the symptoms. Experts were also better than novices at determining the relevance of methods and explained the code at a higher level of abstraction [LaToza et al., 2007]

There are attempts to explore the possible online problematic code based on the programmers' assumptions. Escobar-Avila et al. [2019] mentioned a few quality issues in their survey, such as outdated code. Such study did not include an extensive investigation of the possible issues that arose from coding with the website. Overall, the current investigations of online problematic code focused solely on the Stack Overflow website and lacked the exploration of other online venues where issues could also exist. In addition, most empirical studies are code-based investigations, collecting and analysing code retrieved from the website, while programmers' experiences play little role in such investigations. Furthermore, relatively little is known about the extent of problematic code in programmers' code. Thus, there is a lack of a conceptualised idea about what issues different programmers might face when using websites; current investigations do not involve programmers' perspectives and whether such issues propagate to programmers' code. This thesis focuses on the programmers' experience encountering different problematic code when using the websites(see Chapters 3 and 4). It also focuses on the possibility of propagation of problematic online code to programmers resulting code (see Chapter 5).

2.4 Summary

The Web provides many advantageous features for users. Still, at the same time, users are susceptible to consequences for their memory that Web usage introduces, affecting their recall and increasing the false sense of information familiarity. Much previous research around Web implications for memory has focused on the users, without focusing on the possibility of extending such implications to the programmers. During coding, programmers use their memory and external resources, mainly websites. In addition, coding by referring to websites is far more complex than regular usage; it involves other activities like code reuse. Thus, it is not feasible to generalise the previously investigated implications to the programmers' memory without further investigation. These would require engaging the programmers' inputs on their memory usage while coding, and possible implications of Web usage.

Programmers require information throughout their coding and engage various resources. According to previous research, programmers exploited the information presented on websites using such activities as searching and code reusing, to fulfil their needs and produce outcomes. Addressing programmers' activities provides an understanding of the usage of the Web and helps to investigate the possible implications for the produced code. Therefore, addressing the activities underpinning the coding in depth, particularly with the websites, is essential.

When reusing the online code, the code source and status are unknown, especially when users posted the code. More recent research empirically explored problematic code that could arise from using online code. These investigations provide an ideal way of directly knowing the websites' contents that could pose issues to the programmers' code. However, the research either focuses on a single online venue or misses the programmers' role in such investigations, as programmers could use multiple websites and face various issues. Thus, addressing programmers' perspectives and experiences will enrich the understanding of online code reuse. In addition, a paucity of previous investigations addresses the propagation of online problematic code to programmers' code.

This thesis set out to understand the resources, particularly the websites, that programmers consider during their coding and activities while using the Web. Approaching programmers is suitable for investigating memory usage when coding, and different programmers' expertise provides different perspectives to the investigation. This understanding allows examining the possible implications of coding with websites for programmers' memory. In addition, this thesis investigated programmers' encounters with problematic code while coding using the websites, providing aspects of programmer experience and perspectives. It further explored the propagation of online problematic code to programmers' resulting code while coding with the websites, to examine the implications of Web usage for the code.

Chapter 3

An Interview Study of Websites' Usage and the Implications

The first part of this thesis explores the human aspect of coding, investigating the Web effect on the programmer's memory and code. Individuals offloading information to websites encounter implications on their memory. Using websites requires less encoding and recalling information [Macias et al., 2015, Fisher et al., 2022, Schooler & Storm, 2021, Sparrow et al., 2011], creating a false impression that information is obtained from the Web [Loh & Kanai, 2016, Fisher et al., 2015]. The extent of these implications on programmers' memory is still not explored. Websites are commonplace resources that programmers frequently use for coding tasks. Programmers could offload their information to the websites. Additionally, programmers consider using their while coding, helping them assess and comprehend the code and increasing code familiarity. Website impact on memory has mostly focused on regular users without taking into consideration programmers. Addressing memory impacts makes programmers more aware of the potential implications of memory and makes them reflect on their memory. Programmers' experiences and perceptions could help examine potential memory consequences from coding with websites.

Programmers exploit websites to reuse code during their coding, especially from Stack Overflow. There is a potential for online posted code to have problematic code, such as nonworking code. As a result, using websites and reusing code could mean that programmers will likely adopt problematic code. Novice programmers might be unaware of encountering problematic code due to their lack of experience [Piwek & Savage, 2020]. Previous research has investigated current issues in online code by implementing data-mining approaches, focusing mostly on limited issues pertaining to

Stack Overflow [Acar et al., 2016, Fischer et al., 2017, Ragkhitwetsagul et al., 2019]. No detailed investigation has been conducted into the possible problematic code programmers face when coding using websites. In addition, no research to date has examined the experiences and perceptions of programmers with different levels of expertise regarding problematic code online. Websites continue to be the most popular source of coding by programmers; encountering problematic code is an unfortunate occurrence that needs to be avoided.

This chapter carries out a semi-structured interview study with 18 programmers: 13 undergraduate computer science (CS) students and 5 professional programmers working in the industry. The interview method provides in-depth information that helps construct a greater understanding of the topic [Braun & Clarke, 2013]. The interview explores coding resources, ensuring a comprehensive understanding of development resources. This exploration helps emphasize the importance of the Web and other resources for use in coding. The interview also investigates the perceived impact of websites on programmers' memory when coding and their experience with online code. By analysing the interviews using reflexive thematic analysis [Braun & Clarke, 2006], the findings revealed that programmers regarded websites as a significant resource and mainly preferred Stack Overflow. In addition, students showed little perceived impact on their memory when coding, affecting their experience. Furthermore, programmers—especially students—copy online code without understanding it and encounter multiple problematic code when using websites.

The research questions regarding this chapter are:

RQ1 What resources do programmers use to support the coding process? How and why do they use them?

RQ2 What is the perceived impact of using websites on programmers' memory?

RQ3 How does use of websites affect programmers' code?

3.1 Methodology

Semi-structured, one-on-one interviews were conducted to understand the programmer's use of the resources, the perceived memory impacts of online resources, the reuse of online code and possible problematic code while coding the websites. Next, recruiting participants, interview procedures and questions and analysis were discussed.

3.1.1 Recruiting participants

The interview recruited 18 participants, including undergraduate students ($n = 13$) and professionals ($n = 5$) (see Table 3.1 for more demographic information). The students were enrolled in a computer science course at the University of Manchester: seven from first-year and six from second-year undergraduate courses.

Interviews with students took place during the first semester of the academic year. Thus, many of the first-year students were extreme novices or hobbyist programmers with no formal training. Second-year students had completed at least one year of undergraduate computer-science tuition. Professional programmers had to have been in the UK and in paid employment, in a job role involving programming, for at least a year. All participants had to self-identify as fluent English-language speakers who were competent in at least one programming language. Participants reported their programming experiences (see Table 3.2 for more information about participants experience), similar to previous research that suggested using self-estimation to measure programming experience [Siegmund et al., 2014].

Recruitment of student participants occurred via email. All students enrolled in a core first-year programming course unit (COMP16321 Programming 1) or in a core second-year software engineering course unit (COMP23311 Software Engineering) received an invitation to participate, in the form of an email to the student mailing list (˜500 students in total). Twelve students responded, with another recruited through snowball sampling. Thus, a total of 13 students (referred to hereafter as P1-P13) agreed to take part in an interview. Choosing students as participants is a convenient way to recruit participants, with little difference between students and professionals [Höst et al., 2000]; there is no choice better than another, but selecting students does not affect the relevance and benefit of the outcomes [Falessi et al., 2018].

Professional programmers were recruited through personal social networks and snowball sampling (email and word of mouth). Five professionals (referred to hereafter as P14-P18) agreed to participate in an interview.

All participants received a a £10 GBP Amazon gift certificate. [1]. The full text used for recruiting students and professionals are presented in Appendix A.1).

[1]Approximately $14 USD.

ID	Gender	Age	Formal Learning	Professional role/experience	Other experience
P1	Male	18	Current: Year 1 undergraduate Prior: High School	None	None
P2	Male	18	Current: Year 1 undergraduate Prior: High School (GCSE, A-Level)	Internship	Cybersecurity outside the school
P3	Male	18	Current: Year 1 undergraduate Prior: High School (GCSE, A-Level)	None	Independent learning.
P4	Male	18	Current: Year 1 undergraduate Prior: Middle School	None	None
P5	Male	18	Current: Year 1 undergraduate	None	Independent learning
P6	Female	18	Current: Year 1 undergraduate	None	Independent learning
P7	Female	18	Current: Year 1 undergraduate	None	None
P8	Female	18	Current: Year 2 undergraduate Prior: School project	None	None
P9	Male	20	Current: Year 2 undergraduate	Internship	Independent learning
P10	Female	20	Current: Year 2 undergraduate Prior: High School (GCSE, A-Level)	No	Hackathons, research placement
P11	Male	20	Current: Year 2 undergraduate Prior: High School (A-Level)	No	Hackathons
P12	Male	18	Current: Year 2 undergraduate Prior: Middle School	No	Independent learning
P13	Male	20	Current: Year 2 undergraduate Prior: High School	No	Hackathons
P14	Male	˜50	Prior: Postgraduate research (PhD)	Current: Developer Prior: 17+ years	Independent learning
P15	Female	34	None	Current: Senior developer Prior: ˜10 years	None
P16	Female	45	Prior: Undergraduate and postgraduate (MSc.) degree	Current: Programmer Prior: ˜30 years	Independent study (online)
P17	Male	48	Prior: Undergraduate degree (electrical engineering)	Current: Developer Prior: ˜25 years	Microsoft courses.
P18	Male	32	Prior: Undergraduate degree	Current: Principal developer Prior: 10+ years	None

Table 3.1: Participant demographics. Note that the *Formal Learning* column includes only qualifications/study related to computer science or software engineering. GCSEs and A-Levels are formal high school assessments typically completed at ages 16 and 18 respectively.

ID	**Preferred programming language(s)**	**Use (years)**	**Other languages used/known**	**Self-description of programming competency**
P1	Python	<1	Assembly language, C	Novice
P2	Python	2	C, Haskell, Java, Rust	Intermediate in preferred language
P3	Python	2	C#, Java	Intermediate in preferred language
P4	Python	<1	Lego, Visual Basic	Novice$\leftrightarrow$Intermediate in preferred language
P5	Python	2	C++, Dart	Intermediate in preferred language.
P6	Python	1	CSS, HTML, JavaScript	Novice in preferred language
P7	Python	<1	-	Between novice and intermediate
P8	Python	2	Java, JavaScript, SQL	Intermediate
P9	Python, Java, C#	n.d.	C, C++, CSS, HTML	Intermediate, Established in C#
P10	Java	1.5	C, PHP, Python, Visual Basic	Intermediate, Intermediate$\leftrightarrow$Established in preferred language
P11	Java	1.5	C, CSS, HTML, Javascript, PHP, Python	Intermediate
P12	Java	n.d.	C++, HTML, LaTeX, PHP, Python, SQL	Intermediate in preferred language and C++
P13	Python, Java	1	C, C++	Intermediate
P14	C#, Python	n.d.	-	Expert
P15	Objective-C	10+	Hack, Python	Expert
P16	Xamarin (C#)	<1	Java, LIMS Basic	Expert
P17	C#	8-10	HTML, Java, SQL, Visual Basic	Expert
P18	C#	10+	JavaScript, SQL, TypeScript	Expert in preferred language

Table 3.2: Participants' experiences with programming languages, including preferred programming language and number of years that the participant had used that language ("n.d." indicates that the usage period was not disclosed). Note that some language descriptions are vague or slightly inaccurate, but reflect participant's own descriptions. E.g. P4 was unable to recall the name of the language they had used to program Lego construction toys.

3.1.2 Interview procedures

Before the start of the interview, participants received a briefing sheet and provided informed consent (A complete list of both documents available in the Appendix A.2). The interviews took place in English, either face-to-face or remotely, between November 2019 and March 2020. Face-to-face interviews ($n = 14$) occurred in a quiet place at the University of Manchester. Distance constraints required interviewing most of the professional participants ($n = 4$) remotely via audio or video calls, using *Skype* video-conferencing software. Interviews lasted between 25 and 45 minutes, and an approved recorder audio-recorded them. The collected data were anonymised at the time of collection to ensure against revealing any participant's identity. Then, a commercial third party transcribed the audio files verbatim, and transcriptions were then verified against the original audio prior to analysis. In addition to the audio files, in-situ notes were taken during the interviews to ensure complete information. All of these procedures were performed following the relevant guidelines and regulations from the University of Manchester, and the Department of Computer Science Ethics Committee approved them [2].

3.1.3 Interview questions

All participants were asked to provide basic demographic information, such as age, gender, degree and speciality. They also asked to describe their programming experiences to date, including programming languages they had used, number of years programming, and their self-described programming competency (see Table 3.1 for full demographic information and Table 3.2 for full information about the participants experience).

Subsequent interview questions were designed to address the research questions mentioned in this chapter as follow:

- **First, explore the programming resources by focusing on the websites and examine how and why programmers use resource (RQ1).** Participants were asked to walk through the steps they took to approach a programming task or problem, which helped examine the resources involved in such processes. Then, participants were explicitly asked about the websites they usually employed during programming, followed by specific questions, such as the reason for their choice and its purpose, features and issues. In addition, questions particularly

[2]Reference: 2019-6829-12032.

addressed the Q&A and Stack Overflow websites, asking such questions as how helpful they are in finding content.

- **Second, investigate the involvement of the programmers' memory in the coding activities and the perceived impacts of websites on memory (RQ2).** Participants were asked about the use of memory during coding and the experience they gained from using the websites. Questions followed regarding using the websites for coding and the perceived impact on memory

- **Third, understand the adoption of online code and its impacts on the programmers and their code (RQ3).** Participants were asked about their practices while adopting online code and possible issues regarding the code and its quality.

The complete list of interview questions appears in the Table 3.3. Although the overall structure and line of questioning were similar, some adaptations were made for the two different populations. Student participants were asked specific questions regarding their courses and available resources; professional programmers were asked about their professional settings and approaches. Before recruitment, the interview was piloted ($n = 4$) to ensure the clarity and validity of the questions and to assess the duration.

3.1.4 Analysis

The interview scripts were analysed using reflexive thematic analysis, widely used to analyse qualitative data in different fields because it offers a flexible approach unconstrained by theory [Braun & Clarke, 2006]. Interview transcripts (available as supplementary material [3]) and in-situ notes were imported into the qualitative data analysis software *NVivo 12*. Then, the analysis followed the six steps of the reflexive thematic analysis method, using inductive thematic analysis focused on broad data-based thematic patterning (see Table 3.4 for more information about the followed steps).

To develop initial familiarity, each excerpt was read multiple times without a predetermined focus, to ensure greater emphasis on data familiarisation. Then, inductive coding followed, ensuring data-driven coding linked to the research questions, without a prior understanding of the subject matter or a ready-made coding manual and with a greater focus on the participants' data. Subsequent inductive coding of the excerpts led

[3] `https://figshare.com/s/3bbb37f34c1bf8dff530`(Note that for review purposes this DOI is not yet active, but this text will link to a private preview of its content).

Topic	**Question/Prompt**
Demographics	Please describe your age and gender?
Programming experience and expertise	What is your current year of study? *(students only)* How many years have you been employed in a programming-related role? *(professionals only)* How would you describe your current level of programming expertise? How long have you been programming? With how many programming languages are you familiar? Have you received any (prior) formal training/tuition? Have you completed any paid/unpaid employment in which programming was core to the role? *(students only)* Would you consider yourself to be a novice, intermediate or expert programmer? What does this mean that you can do / cannot do?
Resource use	What kinds of resources do you use when engaging in programming-related activity? Do you find these resources more helpful than other materials? Which resources are most useful? Least useful? What do you use the most? What do you prefer to use?
Preferred programming language	Which programming language would you say you were most experienced in / most likely to use? Can you describe your current level of programming expertise in this language? How long have you been using this language? Have you received any formal training/tuition in this language? Have you completed any paid/unpaid employment using this language? Would you consider yourself to be a novice, intermediate or expert programmer in the language? What does this mean that you can do / cannot do?
Approaching programming tasks and problems	Could you describe to me how you typically approach a task (or problem) in your preferred language? Tell me about the entire process from beginning to end. Suppose you encounter a problem when trying to complete a programming task. For example: an error that you cannot immediately fix by yourself, a piece of functionality for which you are unsure of the syntax, or a subtask that you cannot translate into the relevant algorithm. Could you describe for me the techniques you would usually use trying to resolve such problems when programming in your preferred language?
Website use	What are the websites you were using while coding? Why do you use these websites? Are you typically trying to learn about new topics, looking for solutions (code)...? Why did you choose these websites? Are there particular features that draw you to these sites? What's special about these resources? Do you encounter any particular problems when using these websites? Describe your experiences of using these websites.

Table continued on next page

Table 3.3: Prompts used in semi-structured interviews. Note that exact wording and order varied by participant (based on context), and the groupings used in this table were not made explicit during the interviews.

Q&A use	Have you ever used Internet Q&A websites (e.g. Stack Overflow) to help resolve problems when programming in your preferred language? Can you describe your approach? Which specific Q&A site(s) would you use? How frequently would you say you use these websites? How do you find relevant content? Does use of these websites usually help to resolve your problem? How long does this take on average?
Human memory	After solving a programming problem using websites, do you find yourself more experienced with the problem? Can you then solve a similar problem without using any resources? Do you struggle with the websites when programming? Do you find yourself searching repeatedly for the same code? How many times might these repeated searches occur? Do you find yourself searching for concepts that you would think of as trivial? Things that you would think of as easy? Things that you think you should be able to remember? How many times would you use a web search to confirm something you think you already know (i.e. searching for programming concepts/solutions to make sure that they are correct)? Do you consider your memory to be capable/sufficient of storing and retrieving, basic programming-related information? When problem solving and/or sourcing programming-related information, would you be more likely to draw on your memory or the web?
Code quality	Do you have any concerns about the quality of code you find on the web? Have you faced any specific problems when using or adopting code from websites? Have you experienced any code-security related issues? Bugs? Code-redundancy?

Analysis phase	**Description**
Familiarisation with the data	1. Transcribe the data by listening the audio files. 2. Reading the excerpts multiple times and record initial codes.
Generate an initial set of codes	1. Allocate each data to the relevant code using systematic way. 2. Inductive coding: data-driven approach without pre-knowledge of the data.
Searching for themes	1. Themes is a pattern that accrued constantly over the results. 2. Grouping similar codes into potential related themes.
Reviewing themes	1. Ensuring the theme is consistent with the collected extracts. 2. Ensuring the theme is consistent with the entire data.
Defining and naming themes	1. An iterative process of refine each theme and the story behind it. 2. Clear definition and name for each theme.
Producing the report	1. Choosing vivid and compelling extracts then analysing them. 2. Check the final analysis with the research questions.

Table 3.4: Steps followed during the analysis of the raw interview data, based on the reflexive thematic analysis [Braun & Clarke, 2006].

to a set of expansive codes, including (but not limited to) those articulated by research questions. This full set of codes was reviewed to ensure it carried representational data related to the study objectives, and another researcher confirmed the consistency and clarity of the full set of codes. Grouping similar codes helped locate themes across the data.

A theme is a pattern recurring through the data that strongly related to the research questions. The grouped codes formed a coherent and comprehensive theme, and a data extract was coded to as many relevant themes as identified. Each theme's contents were refined and revised iteratively. The final set of themes was reviewed to ensure that no theme constituted a sub-theme to another theme or was not a theme in itself. Then, a thematic map was drawn to illustrate the final themes. With another researcher, revisiting the resulting candidate themes in the context of the original data ensured good levels of compatibility with the original transcripts. As a result, after refinements, the final set comprised two themes and six subthemes. Inter-rater reliability with another researcher was calculated for a subset of the dataset (30%, i.e. six interview transcripts from six participants), with nearly full agreement (Cohen's $k = 0.84$). The tables used for the reflexive thematic analysis are included in the Appendix A.3.

3.2 Results

Interview thematic analysis revealed two main themes and six subthemes concerning website usage and its impact on memory and code (see Figure 3.1). Before presenting these themes, the resources people reported using was described.

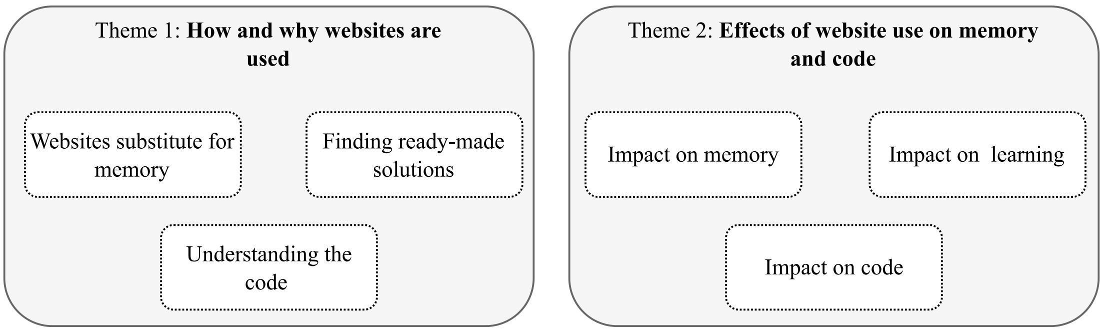

Theme 1: **How and why websites are used**
Websites substitute for memory
Finding ready-made solutions
Understanding the code
Theme 2: **Effects of website use on memory and code**
Impact on memory
Impact on learning
Impact on code

Figure 3.1: Themes identified during analysis of interview data. The two primary themes are subdivided into a further three subthemes in Themes 1 and 2.

3.2.1 Overview of resources used

Participants reported using a variety of resources. Whilst the primary resource was the Web, participants also described use of books/e-books, other people, existing code-bases, course materials and the development environment (e.g., the auto-complete functionality of an IDE).

Book use varied considerably among interview participants, but was discussed by both students and professionals. Some advocated strongly for books, particularly when encountering a new language or concept:

> *Sometimes if I've got a big task ahead of me and I don't know any of it or if I'm completely unfamiliar with the language, I buy a book on it, just to try and follow that through. (P6-Student)*

Others said they rarely or never used books, often because they felt that the convenient and searchable online resources supersede the books. Both students and professionals also discussed other people as a potentially valuable resource. For students, this included their friends or peers, teaching assistants and course tutors, whilst for professionals, this was typically colleagues and superiors.

> *I think it is quite useful to get help from an actual human because you can explain your question even better, and they can obviously interpret it a bit better than someone on Stack Overflow. And they also will get to kind of speak you through it and show you the kind of step-by-step. (P10-Student)*

However, some professionals also noted that frequently consulting others might cause inconvenience.

Students mentioned use of course materials, and the code written as part of exercises or assignments.

> *I've got lots of iterations of it, I always rely on what I have saved above what I actually remember. (P6-Student)*

Professionals were more likely to have access to larger bodies of code written by themselves and others. However, professional codebases were often considered challenging to use, especially in cases where the code was written by others.

One professional participant also noted that the coding environment itself could act as a resource, with auto-complete features prompting them on language syntax.

Irrespective of their use of books, peers, codebases etc., all participants reported that websites were their primary resource for help with programming tasks, and students were particularly inclined to use them. This preference was influenced by multiple factors including the ease and speed of access, use of search engines to identify relevant content and the volume of information and exemplars. Many participants reported use of a search engine (specifically Google) as their initial entry point on the Web. Google search results tended to point to Stack Overflow website, and all participants reported significant use of this as a resource:

> *[I] usually just Google and often the first thing that comes up is Stack Overflow but I don't generally search Stack Overflow specifically. It's just what comes up on Google. (P8-Student)*

Participants were positive about their Stack Overflow experiences, singling it out as the most helpful, useful and efficient website that provided the information they needed.

> *Stack Overflow is very, very good [. . .] I would say Stack Overflow is the best resource for getting help in programming in general. [. . .] I think Stack Overflow is definitely the best in my opinion. Well, it's what I use the most. I think it is very popular with a lot of people as well. (P10-Student)*

> *It's quite nice on Stack Overflow how you'll get lots of different answers to the same question. (P11-Student)*

By comparison, relatively few mentions were made of other Q&A websites (Reddit, Quora). Use of other online content (tutorials, documentation) was mentioned by some participants, particularly for obtaining information about basic concepts or in cases where more detail was needed.

> *If I want a bit more detail, I guess documentation. (P8-Student)*

Some participants also used online code repository websites (GitLab and GitHub), both for uploading their own code and exploring similar code uploaded by others.

> *I often [. . .] use GitHub and look at projects from other people, so example projects that maybe go towards the same direction, and I look at how they solve that issue and if that might be applicable for my problem as well. (P5-Student)*

Participants reported using websites for various reasons. When working on something novel or in a new area, websites were used to provide support.

> *When you've actually got something a bit new [. . .] then you can keep dipping into it, even on a daily basis. (P17-Professional)*

Similarly, the relatively novice students used Stack Overflow to look for answers posted by experienced programmers.

> *In Stack Overflow there'll be people who have used those libraries for ages, and they will have the best way of doing something in a library. (P2-Student)*

Both professionals and students used tutorial websites to clarify understanding and identify best practices.

> *Stack Overflow is good [. . .] if you just want to see the code [. . .] whereas GeeksforGeeks it will give you the code but also actually it kind of explained what they've done, why they've done it, time complexities of the code, so it is quite nice having that much more detail. (P11-Student)*

Finally, another common trigger for website use was the need to resolve bugs and errors.

> *If I have any problem that I can't solve, I'm using the Stack Overflow. (P9-Student)*

Most participants, including those with extensive industry experience, relied on websites and believed that they would be unable to program without them:

> *Stack Overflow has its problems, but we would be lost without it, you have to have it, there's no two ways about it. (P14-Professional)*

3.2.2 Theme 1: How and why websites are used

Websites were reported as the resource most extensively used to support development; this theme examines in detail the reasons for this and how they are used.

3.2.2.1 Websites as a substitute for memory

Websites' availability and ease of access meant that most participants reported using them to repeatedly access relevant information.

> *The websites are just there, on tap, it is too easy to go there. (P17-Professional)*

Many students attributed their reliance on websites to inexperience.

> *I think I just don't have enough experience like to remember stuff yet. I mean, I don't look for every single thing but most of the times I have to look. (P7-Student)*

> *Because I'm not very experienced in it at all [. . .] I will always look it up. (P10-Student)*

Websites helped students to build on their experience, resolving unknowns and problems that emerged when students reached the limits of their current knowledge.

> *When I start a task I base it on my previous knowledge, on my previous experience [. . . when I] make the code very messy or very ambiguous or there's a major issue that I can't resolve, that's the point where I go to websites and try to look at solutions. (P5-Student)*

As students developed experience, and for professionals, there was a growing tendency to rely on their memory in the first instance. In these cases, participants reported trying to program without external help, with professionals approaching tasks more confidently. Websites were used when this did not yield the required results.

> *I try to remember it first just because it's tedious to have to go over and actually Google it so I might try it from memory, see if it works. If it doesn't, I'll Google it. (P8-Student)*

One student thought that programming does not require a good memory as long as the concepts are understood.

> *I would say my memory isn't that great but when it comes to coding you don't need, I would say, a fantastic memory [. . .] because as long as you can understand the concept, you can always implement it in code. (P11-Student)*

3.2.2.2 Using online code

Most participants mentioned copying and pasting code found online, with some students reporting they continuously copied basic coding information like syntax, structures and function format. These activities tended to involve small pieces of code, which were easy to appropriate for their own work.

> *Usually it won't be for like an entire block of code. It will just be for a specific function [. . .] from there I can see if that specific function works. (P11-Student)*

One of the main purposes students reported for using websites was to copy a ready-made solution to specific course requirements. In these cases, they are aware that the code is unlikely to work, but want to submit some form of solution.

> *I've probably been guilty of it in the past when I'm just trying to rush to get an assignment done, and [. . .] I don't care, even if it not works, but I just want to send it off and I don't want [to be] late. (P10-Student)*

Nevertheless, both students and professionals were generally cautious when copying online code; they either avoided copying and pasting code directly or screening the copied code to ensure its validity, especially when using Stack Overflow. Reasons for this included: online code does not have meaningful variable names; getting code to work required multiple alterations; it was difficult to find code that exactly met requirements.

> *I don't think there's much value in just taking someone else's code and putting it in your system. (P10-Student)*

Thus, both students and professionals reported behaviours such as editing the code, removing irrelevant code and changing parameters and variables.

> *I changed out the bits I didn't need, swapped things around. [...] So, I just ended up stripping away all the code that was not relevant to mine and then changing the parameters in the function. (P6-Student)*

3.2.2.3 Understanding the code

Online code was sometimes copied into a program so the developer could try to understand how it was working.

> *I try to copy their snippets of code, try it in my program, see how the results vary and see what exactly is happening in their code. (P13-Student)*

> *Stack Overflow is great, but you've got to understand the answer as best you can. (P17)*

However, some students admitted that they did not always understand the code they used.

> *I feel that if I copy and paste it, I don't necessarily always understand it. (P11-Student)*

This occurred for two reasons: online code is hard to understand, and/or the purpose of searching for code online is to solve a problem quickly.

> *Sometimes it gets a bit annoying because some of the things that they suggest can be really difficult to understand and then I feel really overwhelmed about it. (P7-Student)*

> *It helps if you like try and understand, but if you need to get something done and the code's there that does that, then take it. (P2-Student)*

3.2.3 Theme 2: Effects of website use on memory and code

This theme describes the perceived consequences of using online resources on the participants' memory and code.

3.2.3.1 Impact on memory

Students felt that the accessibility of online information led to a reliance on it, even for frequently used syntax.

> *I just forget how to add things to a list, and then it's really embarrassing when I look it up, and it is like, oh, append. (P6-Student)*

Instead of remembering the code construct itself, they would remember where to find it posted online, and/or would continually look up the same concept.

> *I'll often remember a Stack Overflow post if I've used it before [. . .] saving the location to memory, and I always remember that. (P2-Student)*

> *Moving average in NumPy, I've searched for that at least five times [. . .] I can never remember the correct way of doing it so I pretty much always just look up the blog posts for that or the Stack Overflow post. (P2-Student)*

Some students felt using online resources exacerbated their reluctance to rely on their memory.

> *[Q: Do you think your memory is capable to depend on when programming?]*
>
> *Not really, because I think now the situation has become a bit worse, because anything you go to find, you find out online. So you're more inclined to go online and get the stuff out. (P13-Student)*

> *If you use it [the web] excessively you will not improve your memory, you will keep using it a lot all the time. (P4-Student)*

> *In some cases, I would be able to fix the problem again on my own but I'm not sure I always would. (P3-Student)*

Others actively tried to remember concepts that they had looked up online to improve their programming skills.

> *I'm trying to actually understand and remember things because I think that this is just for me to get better in programming and better in technologies that I am using. [. . .] Googling things could be really beneficial for me in the way that I can learn something, I can reinforce my knowledge or I can revise it. (P9-Student)*

For both students and professionals, 'outsourcing' information storage helped to address cognitive limitations caused by things like age or working with multiple languages.

> *I am nearly 50 years old. Now when I was a young boy, then my memory was fantastic, I could remember things, no problem. Now [. . .] I don't tend to remember things anymore. (P14-Professional)*

> *I was working on more than one language, it's not necessary that you will remember syntax always, because you get confused between, what is the syntax in Python and in C++ [. . .] you can just use sites like W3school. (P12-Student)*

Some professionals did not believe there were any negative effects of website usage, and that it did not inhibit the use of memory if websites were visited only once to solve a specific problem.

> *There's a lot in the memory that you need. I wouldn't say that the internet takes that away from you because [. . .] personally, you only use it that once to get it and get it working and then it's yours, you claim it. (P16-Professional)*

3.2.3.2 Impact on learning

Although many participants used online resources to improve their understanding, in some circumstances it was thought to have the opposite effect especially for students. Some students sought out and used code they did not understand, although this happened less as they gained experience.

> *When I started out with Python I have sometimes found myself to copy or adopt a solution that I didn't really necessarily understand so it worked and I knew to a certain degree why it worked but I couldn't fiddle with the code too much to adapt it to my issues. I just had to go with what it*

> *was but once I got better at using Python, I also learned to understand code snippets more properly so when I adopt code snippets, I can actually mould them so that they suit my issues more properly.* (*P5-Student*)

Students noted a number of potential learning impacts from utilising online code, particularly code that was not understood.

> *I do think that it [copying and pasting online code] kind of hinders people's understanding a little bit [. . .] in certain situations it can be bad because it can stop the learning aspect of it and just force the more I just want to get a grade aspect.* (*P10-Student*)

Specifically, a solution was thought less likely to be remembered if it was used without understanding it.

> *If I have a problem in a specific thing that I am using, then a specific solution won't necessarily implant in my mind because I don't understand necessarily why it works.* (*P3-Student*)

> *I do always try to like avoid just directly copy and pasting.* (*P11-Student*)

Understanding code was also important for effectively editing it.

> *Unless you understand your code, then if someone tells you that there is an error in this part, then you have to be dependent on the other person, and find their error code, it's completely illogical to do that.* (*P12-Student*)

Students were additionally motivated to understand code sourced online, in order to gain marks in their assignments.

> *Are you learning the stuff? Because it's not copy paste at the end, you need to know what you did at the end, because that's what the marker ask you.* (*P13-Student*)

Indeed, some participants reported that using code found online helped with learning new concepts.

> *I think it is quite important because it changes completely how you code in a way but it's more of a long term thing. [. . .] I might read that and then I'll understand it and then I might permanently change the way that I program.* (*P3-Student*)

For some students, being unable to understand the code they found online was not considered to be problematic.

> *I do not think it is a negative feeling when you do not really understand it. (P2-Student)*

3.2.3.3 Impact on code

Students frequently copied code with little regard for the code's functionality and correctness.

> *What I look for is which one fits to my problem, and I try to implement that one. I don't really go through which code quality issues. (P13-Student)*

Editing code to address its weaknesses (see also Section 3.2.2.2) could be a relatively complex process, with no guarantee that the code would ultimately function as required.

> *To use their code I had to do string dot `valueOf`, and then put it in a char array and blah, blah, blah, and do all this rubbish just to be able to shoehorn that code into my system, where in fact it probably really wasn't efficient because I'd just gone to all the trouble to have to change the types and mess around with all that. (P10-Student)*

One of the challenges of sourcing code online, lies in differentiating between good quality code and poor quality code, code that meets the problem specification and code that does not. For professionals, contributions by novice users were considered to be problematic:

> *On Stack Overflow, you have some developers, inexperienced developers, or junior developers, who will put answers on there which come with a weak data structure. (P18-Professional)*

Using Stack Overflow's accepted, upvoted and downvoted answers was considered to be insufficient by both students and professionals.

> *Sometimes the best or most voted answer isn't necessarily the one I'm looking for [. . .] sometimes you do have to look slightly harder for what you are actually looking for. (P11-Student)*

Having to choose between different, sometimes conflicting solutions caused issues for students.

> *Sometimes it's contradictory; someone would write something, someone would write something else. (P1-Student)*

Problems with code snippets, particularly in Stack Overflow solutions, were reported by all participants.

> *There is a lot of buggy code snippets out there that are presented as answers. (P5-Student)*

These problems varied from fundamental compilation/execution errors, through to more concerns including code quality, licensing and versioning. Problems with code snippets that **did not compile or run**, or that did not work as they expected, led some students to source their code from tutorials rather than Stack Overflow:

> *Sometimes the code wouldn't work and then they wouldn't run. (P6-Student)*

> *I like the full tutorial ones because I know they're going to work at the end. Whereas on Stack Exchange, you're not sure it's always going to work. (P6-Student)*

Trying to address errors in online code was time consuming for both students and professionals, leading them to spend longer fixing issues than it would have taken to write code from scratch.

> *You assume it's right but maybe it isn't and it would probably take you longer to try and work out why that isn't right than to just write your own code in the first place. (P10-Student)*

Furthermore, not all problematic code was immediately evident. In some cases code executed correctly but produced **unexpected or erroneous output**.

> *The output of it wasn't what I wanted to do. (P4-Student)*

Insufficient or incomplete code snippets often looked plausible at first glance, with students finding it challenging to identify and add the missing elements.

> *Sometimes the code fragment will look like it works and you'll try and dry run it and maybe you'll skip over the little thing that makes it not work [. . .] put in my code and then it doesn't work. (P10-Student)*

In other cases, such issues might not have impacted overall functionality, but were observed to have negative impacts on code quality.

> *A lot of the time I feel like people on there [the Web] have really, really bad code practices. I don't know if it's just because they maybe haven't been formally taught or maybe if that's just their style [. . .] I just don't necessarily think that's the clearest way to write things. (P10-Student)*

Examples included snippets with **redundant code**, **extraneous (unwanted) output**, or **inefficient and overly complex** structures.

> *I'm sure I take a lot online that's code redundancy, but I'm sure a lot in my own code was that too (P6-Student)*

> *So I know that the code that I copied [. . .] especially when I look back at it [. . .] it was like really, really bad. Like I had like repeated if statements in a row. (P2-Student)*

Students also noted the presence of **code that was undocumented, or poorly documented**, making it difficult to understand.

> *I found this code fragment and it literally made no sense. I can't even stress. All the variable names were a letter, it had no comments. (P10-Student)*

> *They are not commented well. (P13-Student)*

Other problems with online code centred on changes in languages and libraries, leading to solutions that were **deprecated, outdated and no longer worked when used in the context of a current version**.

> *Sometimes they post answers for Python two and we usually use Python three. (P4-Student)*

These issues were difficult to identify and resolve.

> *What's on the internet is outdated but you don't know because it still sort of works. (P16-Professional)*

Sometimes code from Stack Overflow included **security vulnerabilities**.

> *They just had like inputting a get parameter and putting it on the webpage. And I was like, wait, you can put a script tag in the get parameter and it works. (P2-Student)*

> *There were problems that the actual variables had quite weird scopes, there were problems that using the MVVM design pattern that the developers were putting the logic into the view where it shouldn't be, which is quite a huge mistake. (P9-Student)*

> *You can't just take out some stuff on random websites [. . .] you don't even know if it's trusted source, what source they come from. (P13-Student)*

However, the majority of students had never embarked on projects that they felt could be impacted by security issues.

> *The stakes are low, yes, I could test it and nothing would happen. (P1-Student)*

> *My programs never were too much security based. (P5-Student)*

One students and two professionals noted the potential impact of **licence restrictions** on code reuse.

> *Creative Commons is a fairly open licence, isn't it, so I'm pretty sure you're allowed to just copy it. But there's presumably restrictions on where you can use the code. So none of the code I write is for commercial purposes so I'm presumably exempt from that. (P2-Student)*

Some participants tried to check for quality issues as they went along; this was easier for professionals as they had more experience.

> *I try to look out for, like, more than code quality issues, I try and find out if it's in the required programming language [. . .] check if it's best way. (P13-Student)*

Having done this myself for many years, I can spot weak answers and strong answers. (P18-Professional)

Students observed that over-reliance on online code snippets could result in "piece-meal" software (P3-Student) where individually valid pieces of code were bolted together into an inelegant whole.

If you are just dealing with every issue as you come to it, it can make code that's a bitstream of consciousness [. . .] just random snippets [. . .] it can make the code a lot less elegant. (P3-Student)

This approach, combined with many of the previously reported problems, meant that students frequently found code snippets **difficult to incorporate/integrate** into their projects.

3.3 Discussion

This section discusses the interview results and summarises the findings for each research question.

3.3.1 RQ1: What resources do programmers use to support the coding process? How and why do they use them?

The findings revealed that all interviewed participants preferred online resources when coding. Professionals used the websites to augment their understanding or add missing elements. They also used the codebases on which they were working or to which they had access as another primary resource. On the other hand, students' inexperience and time-bound assignments motivated them to value websites for their ease and speed of access. These constraints and website features explain the low-level use of course materials, such as books and course tutors (when completing work outside of teaching sessions); this results match those observed in [Lausa et al., 2021]. A further resource is the use of help from other programmers. Professionals reported working closely with colleagues as a suitable way to explain programming matters more conveniently. This result supports [Maalej et al., 2014], who found that industry programmers preferred communicating with colleagues over accessing resources. However, they noted that such means could be disruptive to others and not always possible. In addition, students reported that friends were particularly helpful when they could not find relevant content online or had difficulty understanding and integrating online materials. Those findings emphasize the importance of the Web for students and professionals.

Participants commented that online resources provided them with a quick response and an effortless and fitting way of accessing the information, similar to the finding of [Brandt et al., 2009, Maalej et al., 2014, Hucka & Graham, 2018] but unlike [Astromskis et al., 2017], who suggested limited usage of online resources. The results of this study show that participants do not prefer certain websites but instead search and observe the results. All participants, except for (P15-Professional), noted their access to search engines, particularly Google—e.g. *"I just Google it"*, (P12-Student); *"Google is your best friend"*, (P18-Professional). The usage of the Google search engine is consistent with prior studies [Maalej et al., 2014, Xia et al., 2017], and interviewees sometimes described query types similar to those observed in prior research [Brandt et al., 2009, Hora, 2021, Xia et al., 2017], particularly general search, debugging and bug fixing and language-specific syntax.

Through their initial search engine, participants reported high-level use of Stack Overflow to meet their needs, similar to prior studies that showed Stack Overflow's high return rates from technical searches [Hora, 2021, Xia et al., 2017], and extensive use of the website in development [Acar et al., 2016]. Other studies [Acar et al., 2016, Treude et al., 2011] also reported the importance of Stack Overflow, but the findings contrast with those of [Bhasin et al., 2021], who found that Stack Overflow was not the preference of their student participants. Participants reported using Stack Overflow to clear understanding, solve problems and acquire novel information, even when the usage consequences are known; students also mentioned the consult of experienced programmers' answers within Stack Overflow. These uses extend the work of [Abdalkareem et al., 2017b], who highlighted website usage for knowledge and solving bugs. Students and professionals noted the importance of Stack Overflow features, such as multiple solutions, comments and up/down votes, but they also mentioned instances of conflict between answers, where an answer to solve their problem was not the one upvoted or accepted. Studies of source code [Wu et al., 2019] have also observed the use of not accepted answers in Stack Overflow as an information source. In addition, participants rarely mentioned other Q&A competitor websites, such as Reddit and Quora. They further mentioned other websites, including documentation and tutorial websites, but not as frequently as Stack Overflow. While programmers did not mention the extensive use of tutorial websites, they surpass Stack Overflow in fetching functional code—e.g.

> *I like the full tutorial ones because I know they're going to work at the end. Whereas on Stack Exchange, you're not sure it's always going to work. (P6-Student).*

The mention of code hosting websites was scarce among the interviewers, as only one professional made any mention of Git, and none discussed GitHub. In the end, Stack Overflow appears to be a preferred coding website for programmers.

3.3.2 RQ2: What is the perceived impact of using websites on programmers' memory?

The findings suggest that participants' usage of online content was motivated to some extent by their inability to recall programming concepts and syntax, or the perceived effort associated with retrieving that information. The Web is likely perceived as more accessible than memory to students with relatively little experience and programmers working in multiple programming languages. In this respect, the Web supplemented their use of memory for coding, mainly to remember locations of information. One vivid example is that little effort is required to memorise syntax because it will always be available online:

> *It's not necessary that you will remember syntax always (P12-Student).*

Such results comply with those of [Brandt et al., 2009], who stated that programmers leave some programming aspects to the websites and choose not to store and remember the complicated syntax. With the concern that cognitive offloading may negatively impact the independent ability to recall [Fisher et al., 2022, Sparrow et al., 2011], participants reported using the Web more than their memory when coding with little perceived implications. Participants consideration of the Web to recall information locations comply with those of [Sparrow et al., 2011], who empirically found that the users of the online resources recall the information location more easily than the information itself. In addition, students reported more reliance on the Web than professionals. Their usage of the websites triggered the intention to fetch the information online. Such usage could reduce their expertise advancement, similar to findings on [Fisher et al., 2022]. While other reasons could factor in memory usage, including the age of the programmers and using multiple programming languages, programmers would benefit further from reflecting on and using their information before resorting to online resources. One approach mentioned by one participant was the attempt to solve a problem first and then utilise online resources in case of difficulties. Empirical evidence suggests that such an approach has a positive effect, showing that pretesting (i.e. not referring to the online resources but reflecting on the required task) is more beneficial in forming information than directly referencing the online resources [Giebl et al., 2021].

Some evidence shows that programmers' perceptions may not be wholly accurate. A source attribution error may cause programmers to subsequently believe that the information originated from their own semantic memory, giving them false confidence in their abilities [Fisher et al., 2015, 2022]. This attribution error could explain the conflict in participants' reporting, as students reported online content both augmenting and impeding memory and learning at the same time. In addition, data from participants suggest a differentiation between memory uses for syntax and for understanding, with prior work also suggesting the importance of conceptual knowledge to remember by programmers [Krüger & Hebig, 2020]. The way programmers perceive memory might shed light on why participants gave contradictory statements about its importance for coding.

3.3.3 RQ3: How does use of websites affect programmers' code?

Programmers reported that they copied and pasted online code and spent efforts avoiding or carefully performing such activities. Their mention of copying pieces of online code is consistent with the description by [Yang et al., 2017], who found that copy from Stack Overflow is prevalent among programmers. Whilst programmers commented that they preferred using Stack Overflow when coding, one participant reported that code from the tutorials is more reliable than Stack Overflow. In addition, students did not explicitly state their copy-and-paste activities throughout the interview, but such behaviours became more evident in later discussions. One explanation is that students are reluctant to reveal their copy behaviours. Another possible explanation is that students copy code in a habitual, unnoticed way, as they were motivated to reuse code to fulfil their course requirements, such as assignments. While copying online code provides valuable input for programmers, it could also introduce many consequences.

Participants mentioned good reasons to try and understand online code. However, understanding the online code was not always possible, as students reported ignoring that aspect. Some online code was beyond students' current competency, as they lacked adequate necessary information, a problem noted by survey respondents in [Escobar-Avila et al., 2019]. Another factor is the time pressure of specific deadlines means students lack the time they would need for understanding and usually use code to fulfil the requirements. Students lacking understanding lose their ability to control the copied code (as they do not have the necessary knowledge to edit it) as well as any benefits from reused code that would enrich their experience, impacting their learning. The low level of understanding of the code prompted programmers to avoid reusing the code [Wu et al., 2019]. The present study raises the possibility that using the Web does not help students understand the copied code or increase their learning. These concerns do not necessarily apply to professionals, who reported that they consider understanding the online copied code; such results disagree with [Maalej et al., 2014], who stated that professional programmers do not understand copied code.

One possible risk of copying online code snippets is encountering problematic code [Abdalkareem et al., 2017a, Acar et al., 2016, Fischer et al., 2017]. The results of this study show that programmers are vulnerable to encountering problematic code online. Students and professionals reported that during their use of the websites, they encountered various problematic code that could deteriorate code quality. One interesting observation is that participants who expressed their awareness and avoidance of online issues still faced some. Students might not always be able to identify problematic code from online snippets—e.g.

> *I don't know what code quality means. (P4-Student)*

—unlike professionals who were more confident assessing the likelihood of code from Stack Overflow to be good or poor.

Students and professionals reported eleven distinct problematic code online, listed in Table 3.5. Some of the reported problematic code had previously been noted in the literature, such as security vulnerabilities [Acar et al., 2016, Fischer et al., 2017, Licorish & Nishatharan, 2021, Meng et al., 2018], licence issues [Ragkhitwetsagul et al., 2019] and poor code quality [Ragkhitwetsagul et al., 2019, Treude & Robillard, 2017]. While some of the reported problematic code corroborate the findings of previous work, considerable differences exist. The discovered problematic code online was based on data-driven approaches that reported findings regardless of the programmers' experience, focused on one specific issue and did not compare the issue in terms of programmers' expertise. The findings in this study acknowledged programmers' experience by recognising problematic code from both students and professional programmers, emphasizing the possible issues that may have arisen during coding with websites. Therefore, the study provided a holistic list of the possible problematic code that students and professionals could face while coding with the Web.

Issue	Occurrences	
	Students	Professionals
Inefficient or overly complex code	✓	✓
Undocumented code	✓	-
Code that is difficult or impossible to incorporate/integrate into an existing project	✓	-
Insufficient or incomplete code	✓	-
Code that does not compile or does not run	✓	-
Outdated – code that does not work with the current version of a language or library	✓	✓
Redundant code	✓	-
Code with extraneous output	✓	✓
Code with incorrect output	✓	-
Security issues	✓	✓
License violations	✓	✓

Table 3.5: Comparison of issues observed in online code snippets by participants. Denoted occurrence means that one or more participants reported directly encountering this issue.

Notably, the mentioned problematic code related to the Stack Overflow website, consistent with that of [Ragkhitwetsagul et al., 2019, Treude & Robillard, 2017, Acar et al., 2016] who investigated the existence of issues on the site. While Stack Overflow applied various quality indicators to help inform users when assessing answers and code snippets, participants profited from the "non-top" and unattributed answers. This suggests that the programmers do not follow the proposed quality indicators, consistent with [Wu et al., 2019, T. Zhang et al., 2018], who discovered that programmers retrieve answers from the not accepted and "not-high-score" answers on Stack Overflow.

Ultimately, copying and pasting online code produces different problematic code, extending the work of [Wu et al., 2019, W. Bai et al., 2019], who discovered that Stack Overflow has low-quality code, and adopting it causes issues. The findings are unlike what [Escobar-Avila et al., 2019] mentioned, namely, that the programmers were not concerned but, rather, pleased with the quality of the online code. In addition, the findings stressed students' vulnerability to online problematic code and lacking skills and knowledge to assess such encounters. This agrees with what [W. Bai et al., 2019] addressed that lacking the required experience means having less ability to assess the code, where the current courses do not teach students about code quality [Kirk et al., 2020].

3.3.4 Limitations and Threats to Validity

Discussing the study limitations utilises the four threats to validity proposed by Runeson & Höst [2009]: construct validity, internal validity, external validity and reliability.

The nature of questions in the semi-structured interview may introduce bias in the follow-up questions, the order of the questions and the wording. The initial interview questions were piloted before recruiting participants to validate and refine the questions and minimise threats to construct validity. Another threat to construct validity may have emerged as a property of interviewing student participants from the University of Manchester. Whilst it was clearly indicated to participants that their responses would be treated anonymously, with no impact on their study or outcomes, students may still have been reluctant to divulge behaviours that they thought were negatively perceived by the academe. Students at UK universities are regularly advised against activities that might constitute plagiarism, such as copying and pasting from external sources, and this may have led students to minimise disclosure of these behaviours during their interviews. Additionally, another researcher reviewed the process of coding the interview transcriptions to ensure consistency and remove bias.

Although some differences exist between students' and professionals' perceptions and behaviours, this research does not set out to examine causal relationships; thus, internal validity is of limited concern. However, the potential influence of external factors, particularly on student responses, does arise. For many students, course requirements (e.g. deadlines) drive time spent on programming activities, and their current workload could have influenced the interviewers' responses.

The interviews used sampling methods that required participants to self-select (i.e. respond to advertisements), with potential implications for external validity. In particular, the recruitment materials explicitly referred to the use of resources during programming activities, which may have led to particular patterns of self-selection/non-response.

The participants were recruited from two populations in the UK: undergraduate students in a computer science or software engineering programme and software developers who had been in a professional role for at least one year. Student participants came exclusively from the University of Manchester. Understanding the degree to which the perceptions and behaviours articulated in the interviews reflect larger populations requires further study.

Interview samples are dominated by students, a reflection of the relative ease of recruiting students when compared to professionals. This imbalance may have led to bias in the results; however, there is a considerable alignment in the results from students and professionals.

To maximise research reliability, the research team took steps to involve multiple members of the research team at every step. Two researchers participated in planning the interviews and multiple researchers coded the qualitative data iteratively (inter-rater reliability $k = 0.84$).

At last, the findings did not report counts or numbers as this study is intended to be qualitative in nature.

3.4 Conclusion

This chapter presented an interview study investigating the use of online resources during programming tasks and the perceived effects of that use on programmers' memory and code. The study analysed interviews with 18 students and professional programmers. The thematic analysis results showed that the Web is a predominant coding resource, especially for students, dominated by the use of Stack Overflow. Both students and professionals reach Stack Overflow through search engines, specifically Google. Professionals also incorporate codebases with equal frequency. The study also found that students' Web usage produced little perceived impact on their memory, with a tendency to recall the location of the information rather than the information itself.

Participants encountered problematic code when using websites, despite their awareness and caution, and reported eleven specific problematic code. In addition, participating programmers reported reusing online code but not necessarily understanding the reused code. Unlike professionals, students reported encountering all the problematic code. These findings suggest that students generally face more consequences for using the websites, perhaps attributable to their lack of experience. Interviewed participants reported Stack Overflow' metrics that support assessing online content to be of limited value.

Overall, the results highlight that student programmers must adopt an independent problem-solving approach and increase their awareness of memory usage. Students should also treat the Web as a venue for learning by refactoring and understanding unfamiliar code and developing their skills, to mitigate potential problems when using online code.

This chapter used a qualitative method that ensured in-depth investigations. However, one limitation is the limited number of programmers involved. Thus, the findings require incorporation into a quantitative approach with a larger programmer population. The next Chapter 4 adopts the interview findings and builds a survey instrument for distribution to students and professionals.

Chapter 4

A Survey Study of Websites' Usage and the Implications

This chapter presents a survey study that tests and extends the interview findings from the previous chapter (Chapter 3). The interviews revealed that programmers utilised various resources for coding, with websites being the most prevalent. Student programmers found little perceived implications on their memory resulting from coding with websites. In addition, participating programmers also reported not understanding online copied code and encountering problematic code. While the interview study investigated in-depth the usage of websites and possible implications for memory and code, the nature of that qualitative study limits engaging a broader population of programmers. On the other hand, the survey method systematically collects information describing a population sample from a greater population [Braun & Clarke, 2013, Creswell & Creswell, 2017, Pfleeger & Kitchenham, 2001].

This chapter presents a survey that used the interview results (from Chapter 3) to generalise the outcomes to programmers. In particular, the survey set out to confirm the resource choices and explore the usage of websites and Stack Overflow. Additionally, the survey continues investigating the role of the programmers' memory in coding and the possible implications of using the websites. It also reflects more about adopting online code and tests the extent of encountering problematic code. The survey was deployed to two distinct sets of programmers: undergraduate students and professional industry programmers. The findings from the survey confirmed that the websites continue to be the resource students and professionals use most for coding, and Stack Overflow is predominant. In addition, the survey respondents perceived little implications for their memory from using the websites. However, they prefer to know where to find information on the Web and continue to rely on it when coding. Furthermore, professionals revealed greater willingness to reuse online code that they did not fully understand. They also reported more direct encounters with most of the reported problematic code than students.

While the survey continues the investigation from the interview study, this study address similar research questions:

RQ1 What resources do programmers use to support the coding process? How and why do they use them?

RQ2 What is the perceived impact of using websites on programmers' memory?

RQ3 How does use of websites affect programmers' code?

4.1 Methodology

The design of the survey questions followed the interview findings (see Figure 4.1 for more information about the survey methodology). This section describes the survey design, which followed the principles specified by [Pfleeger & Kitchenham, 2001], including establishing specific and measurable goals, planning and designing the survey, preparing and validating the survey instrument, selecting and recruiting participants, analysing the collected data and presenting the findings. The following sections discuss each step in detail.

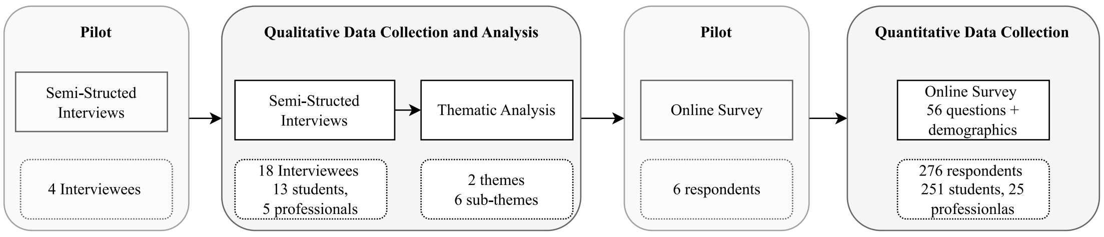

Pilot
Semi-Structed Interviews
4 Interviewees
Qualitative Data Collection and Analysis
Semi-Structed Interviews
Thematic Analysis
18 Interviewees 13 students, 5 professionals
2 themes 6 sub-themes
Pilot
Online Survey
6 respondents
Quantitative Data Collection
Online Survey 56 questions + demographics
276 respondents 251 students, 25 professionlas

Figure 4.1: Overview of research methodology. A sequential mixed methods approach combines semi-structured interviews whose qualitative data is thematically analysed (see Chapter 3); these themes form the basis for a larger quantitative data collection through online survey.

4.1.1 Survey objectives

The survey investigated three areas based on the interview findings to answer this chapter's research questions:

1. For the first research question (**RQ1**), the interview revealed that websites were the resource participants used most, and Stack Overflow was one of the frequently mentioned websites. The survey confirmed such choices by asking student and professional respondents about the resources and websites they used for coding. It then asked specific questions about the features and traits of Stack Overflow, such as the applied metrics, since the website received much attention in the interviews.

2. For the second research question (**RQ2**), the interview findings indicated that students perceived little impact on their memory when coding with websites. The survey assesses the role of memory in coding, and how students and professionals use their memory during coding with websites. It also investigates the possible implications on the memory from coding with websites.

3. Relative to the third research question (**RQ3**), the interview findings mentioned issues with understanding online reused code and problematic code that programmers encountered when using the websites. Thus, the survey continues this investigation by seeking to understand the reuse of online code and testing the problematic code amongst student and professional programmers.

4.1.2 Survey planning and design

This study carries an unsupervised online survey with a combination of undergraduate students enrolled on UK computer science or software engineering programmes, and professional programmers employed in the UK. Prior to recruitment, a pilot ($n = 6$) was used to ensure that the questions were clear and understandable, any ambiguity was eliminated and fluent but non-native English speakers could complete the survey within the expected time (10–15 minutes). A sample of the population read the survey questions and provided feedback to enable refining and updating the survey before recruiting participants. All participants received a briefing sheet prior to participating and were required to return a checkbox consent. All data were anonymised at the time of collection. The Department of Computer Science Ethics Committee at the

University of Manchester reviewed and approved procedures for survey study [1]. The following sections describe the survey questions and the instrument.

4.1.2.1 Survey questions

The survey design followed the interview study's outcomes as presented in the previous chapter (see Chapter 3). The interview findings addressed two themes: **How and why websites are used** and **Effects of website use on memory and code**. The interview results supported designing a collection of survey questions with respect to the main research questions. A few survey questions were designed based on participant quotations that were not included in the interview study to limit the focus and the number of quotes (see Appendix A.3 for the tables used in coding the interview study). The survey questions were analysed, reviewed and refined to ensure each question was clear, stated one meaningful idea that was understandable, direct, related to research objectives and easy for the target population to answer [Sue & Ritter, 2012, Kitchenham & Pfleeger, 2002]. The design of the questions included both open-ended and closed-ended questions [Sue & Ritter, 2012]. The survey comprised two variants, one for students and one for professionals. The two variants differed only in the initial questions that collected basic demographic information and programming experiences to date (see Table 4.1 for demographic questions with difference between students and professionals, and the Table B.1.1 in the Appendix for the full version of the demographic questions presented to respondents). The remaining 56 Likert-style questions were identical across both surveys (see Table 4.2 for the survey questions and Table B.1.2 in the Appendix for the full version of the demographic questions presented to respondents). The following sections address both editions and the used internment.

4.1.2.1.1 Students survey The student survey variant started with 10 demographic questions about participants and their programming experience, including year of birth, gender, the year of study, the current university, the programming experience before the degree, the start of coding, the proficient programming languages, the start of coding using proficient programming language, the self-description of programming competence using the proficient programming language and email for entry in the prize drawing. Age and email questions were optional, but the remaining demographic questions were all mandatory. Then, 56 Likert-style questions were presented and were all mandatory. To increase participant focus and minimise cognitive load [Lazar et al.,

[1] Reference: 2019-6829-12032

Question	**Choices**	**Sample**
Q1: In which year were you born?	From 1920 till 2003	All
Q2: What is your gender?	Male, female, Non-binary/third gender, Prefer not to disclose, Prefer not to say.	All
Q3: What is your current year of study?	Year 1 - UG, Year 2 - UG, Year 3 - UG, Year 4 - Integrated MSc.	Students.
Q4: Which university are you studying at?	List of UK universities.	Students.
Q5: Which of the following best describes your programming experience before starting your degree program?	Hobby programmer. Completed programming internship. Casual employment in a programming-related role (including voluntary or charity work). Full time employment in a programming-related role. Prior study of computer science for a qualification (e.g. GCSE / A-Level). No prior programming experience.	All
Q6: When did you write your first line of code (in any programming language)?	Less than 1 year ago. At least 1, but less than 2 years ago. At least 2, but less than 3 years ago. At least 3, but less than 5 years ago. 5-9 years ago. 10-14 years ago. 15-19 years ago. 20+ years ago.	All
Q7: In which programming language would you consider yourself to be most proficient?	Open text	All
Q8: When did you write your first line of code in the language with which you are most proficient?	Less than 1 year ago At least 1, but less than 2 years ago. At least 2, but less than 3 years ago. At least 3, but less than 5 years ago. 5-9 years ago. 10-14 years ago. 15-19 years ago. 20+ years ago.	All
Q9: How would you describe your competency in the language with which you are most proficient?	Beginner. Between Beginner and Intermediate. Intermediate. Between Intermediate and Expert. Expert.	All
Q10: Enter your email	Free text.	All

Table 4.1: Demographics questions asked in the survey.

2017], questions that discussed a similar topic were grouped to form four parts: programming resources, experiences of Stack Overflow, use of code snippets and memory. Responses under Likert-scale were a mixture of three- (12 questions) four- (14 questions) and five-points (30 questions) scales. The survey included a skip logic that helped present the most relevant question to the respective participants. For example, a participant who mentioned not using the Stack Overflow website in the programming resources part was not directed to the experiences of Stack Overflow part.

4.1.2.1.2 Professionals survey The professional's survey contained questions identical to the student survey, but differ on the demographic section. The demographic questions were similar to those asked of the students but excluded the year of study and the university questions. In addition, the wording of the programming experience questions was changed to address professionals. For example, instead of using "degree" in the questions, professionals were asked about programming experience before "their current role". The remaining parts of the survey were identical to the student survey variant.

4.1.2.2 Survey instrument

The study was conducted using a University of Manchester supervised and approved tool called Qualtrics [2]: an online software designed to perform a survey, with flexible question-creation and a user-friendly interface. This survey used Qualtrics to create and administer both survey variants. One main advantage of the tool is deploying the survey questions on pages to facilitate deploying survey parts. An introductory page showed the inclusion and exclusion criteria, participant's information sheet, survey instructions, incentive information and researcher contact information (see Figure B.2.1 in the Appendix for the introductory page presented to both students and professionals). At the start of the survey questions, the demographic questions appeared, to ensure capturing participants' information in case of dropouts. Then, participants were directed to the following pages: programming resources on the second page, experience with Stack Overflow on the third page, the use of online code snippets on the fourth page and memory on the last page. Programmers saw questions in order, with similar questions grouped and arranged from easy to difficult on each page [Sheatsley, 1983, Lazar et al., 2017]. The tool supported the skip logic and allowed participants to leave at any time; they were not compelled to complete the survey.

[2]https://www.qualtrics.com

Section	**Question**
Resource use	*Q: Which of the following resources do you use when programming?* (4-item scale: Frequently→Never) R1. Books. R2. Websites. R3. Friends / Colleagues. R4. Tutors. R5. Existing codebases (code you have written or worked with). *Q: Which of the following websites do you use when programming?*(4-item scale: Frequently→Never) R6. GitHub. R7. Reddit. R8. Quora. R9. Stack Overflow R10. Language documentation. R11. Tutorial websites. *Q: How do you access information from these websites when programming?*(4-item scale: Frequently→Never) R12. I visit the site directly. R13. I use a search engine with the intention of finding content from a specific website. R14. I use a search engine and click whichever results look most relevant.
Use and perceptions of Stack Overflow	S1. Comparing to three years ago, I used Stack Overflow: (3-item scale: More than I used to→Less than I used to). *Q: To what extent do you agree with the following statements about the Stack Overflow website?* (5-item scale: Strongly Agree→Strongly Disagree) S2. When searching for programming-related concepts on the Web, the Stack Overflow website is the most dominant result S3. I cannot program without the Stack Overflow website. S4. I can find what I am looking for on Stack Overflow. S5. I find it hard to tell if the question and/or answers on Stack Overflow are relevant to my programming tasks. S6. I prefer to use the most upvoted solutions on the Stack Overflow website. S7. I take the author's reputation into account when deciding how likely the answer will help. S8. I can identify poor quality solutions on Stack Overflow because they will have been down voted. S9. Having multiple different solutions, and others' comments on those solutions, is very helpful to me. S10. I find that different answers and/or comments conflict with each other. S11. I am wary when reading unaccepted answers on Stack Overflow website.

Table continued on next page

Table 4.2: Questions asked in online survey (excluding demographics). All questions were established based on findings from the interview study

Use and perceptions of online code snippets	C1. How often do you copy and paste a source code snippet from the web? (5-item scale: Most days$\rightarrow$Rarely). *Q: To what extent do you agree with the following statements about the code snippets you find on the web?* (5-item scale: Strongly Agree$\rightarrow$Strongly Disagree) C2. I trust code snippets found on the Web. C3. I copy code snippets to make up for gaps in my experience / knowledge. C4. I copy code snippets only if I fully understand their contents. C5. I copy code snippets only if they are consistent with my own code quality standards. C6. Copying and pasting code hinders programmers' understanding and learning. C7. Copying and pasting code from websites reduces code quality. C8. The majority of online code snippets are of good quality. *Q: To what extent have you found the following to be present in code snippets on the web?* (3-item scale: I have encountered this problem myself$\rightarrow$I am unaware of or don't think this is a problem) C9. Code that does not work with the current version of a language or library. C10. Security issues. C11. Licence violation issues. C12. Code that does not compile or does not run. C13. Code with extraneous output (e.g. unwanted prints). C14. Code with incorrect output (e.g. 5 + 1 = 7). C15. Code that is difficult or impossible to incorporate into an existing project. C16. Undocumented code. C17. Redundant code. C18. Insufficient or incomplete code. C19. Inefficient or overly complex code (the problem could be solved much more simply another way).

Table continued on next page

Programming and human memory	*Q: To what extent do you agree with the following statements about human memory when programming?* (5-item scale: Strongly Agree→Strongly Disagree) M1. Having a good memory is critical to successful programming. M2. I have a good memory for programming concepts and syntax. M3. When solving a new programming problem, I am able to remember similar problems I have solved in the past. M4. It is faster to remember programming-related information than it is to look it up. M5. I can program non-trivial applications using my memory alone. M6. Being unable to remember programming concepts bothers me. *Q: To what extent do you agree with the following statements about remembering content you find on the web?* (5-item scale: Strongly Agree→Strongly Disagree) M7. There is no need to try and remember programming concepts because websites are always available. M8. If I have previously solved a problem using the Web, I will be able to solve the same problem in the future without looking up the information again. M9. If I have previously solved a problem using the Web, I will remember where to find the information needed to solve the problem next time. M10. Looking at programming content on the Web confirms what I already know or reminds me of something I had forgotten. M11. The more I use programming contents on the Web, the less I remember. M12. Programming content on the Web is for reference not learning.

4.1.3 Recruiting participants

The survey recruited 276 participants, comprising 251 undergraduate students and 25 professionals (see Tables B.1 and B.2 for full demographic information). All participants were required to self-identify as fluent English-language speakers and competent in at least one programming language. Participants could choose to be entered in a prize drawing for a £100 GBP Amazon gift certificate[3]

4.1.3.1 Students

Undergraduate participants were students enrolled in a computer science or software engineering programme at a UK university. Students were recruited predominantly online, through emails sent from their department's director of undergraduate studies (or their delegate) and social media. The initial recruiting step was reaching UK universities by emailing the directors of undergraduate teaching and asking their permission to recruit their students. A list of UK universities was used for recruiting, used in the demographics, to search manually for the respective directors of undergraduate studies within each university. To fetch their contacts, the websites of 168 UK-based universities were visited, and 65 prospective contacts were saved to reach with the information about the survey and request permission to recruit their students. Some universities did not have a computer science department, did not clearly state the director of undergraduate students, did not show the roles of their academic staff or had an inaccessible directory. Prospective contacts were made in two rounds. The first included sending emails with information about the researcher and the research, the survey contents and time, the permission to recruit students and any possible implications, along with the ethical approval. The responses were as follows: three universities refused to recruit their students, six universities asked to delay the recruitment due to exam periods and one university accepted and delivered survey information to their students. The second round of contacts followed up by sending reminder emails.

Ultimately, nine universities approved recruiting their students, and the universities were Swansea University, University of Lancaster, Queen Mary University of London, University of Nottingham, Newcastle University, University of Warwick, University of St. Andrews, University of Surrey and Manchester Metropolitan University. A survey poster and written emails were sent to the accepting universities, to circulate to their undergraduate students. Each university had its own way of communicating with

[3]Approximately $140 USD.

students, e.g. mailing lists and newsletters. In addition, two recruiting methods were employed to gain more students for the survey. The Students Room [4] online forum was used to post a poster, along with the inclusion and exclusion criteria. Students at the University of Manchester were recruited using offline and online methods. Multiple printed flyers were posted around places that undergraduate CS students frequented. Since courses at that time were presented online due to the COVID pandemic, more online efforts were devoted to posting the survey. Specifically, the survey poster and text were used for advertising the survey on Twitter, using the Interaction Analysis and Modelling Laboratory (IAM) account and retweeted by the CS department account. Also, the survey was advertised using "MondayMail", a weekly round-up (newsletter) for the undergraduate students in the department (see B.4.2 in the Appendix for the material used to recruit students).

4.1.3.2 Professionals programmers

Professional participants were required to have been in paid employment in a UK-based job role involving programming for at least a year. Targeting professional programmers is not as direct as the students because there are no specific venues to target programmers. Thus, professional programmers were recruited through a combination of personal social networks and snowball sampling. Participants in the previous interview study (Chapter 3) were sent the link and invited to share it with others. Acquaintances programmers who fit the survey requirements were also contacted to participate in the survey. In addition, online forums were used to reach a wider range of UK-based programmers. The survey poster and information were sent to the the Society of Research Software Engineering[5] using their Slack channel, and were posted on the related pages on the Reddit website [6] (see B.4.3 in the Appendix for the material used to recruit professionals).

4.1.4 Survey analysis

Surveys were completed between January and June 2021, i.e. in the second half of the academic year. The analysis started after closing both surveys at the end of June 2021. The surveys received 311 responses from 282 students and 29 developers. From

[4]`https://www.thestudentroom.co.uk`

[5]A UK-based organisation for software developers in academia and other research institutions (`https://society-rse.org/`).

[6]`https://www.reddit.com`

the raw data, 31 entries that contained only consent and/or demographic information. One further developer response was discarded due to not meeting the inclusion criteria (programming experience of less than one year), and three student responses were discarded for providing inappropriate responses to the free text question (suggesting that they may not have given due attention/understanding to the survey as a whole). The final dataset consisted of 276 responses (251 student; 25 developer).

The survey versions were downloaded from Qualtrics software in a ".csv" format. Using Excel software, a manual analysis of the results was performed for each survey to calculate demographic questions including age and programming languages. In analysing the remaining results, Likert-style questions were treated as ordinal data. The analysis was undertaken in Python, using the `pandas` and `matplotlib` libraries (the raw data and the used analysis scripts are available as supplementary material [7])

4.2 Results

The survey took around 11 minutes to complete (students: mean 10.94, median 8.02, std. dev. 14.10; professionals: mean 13.07, median 6.63, std. dev. 19.79). There were 251 student respondents and 25 professionals. Both samples were predominantly male (students: 73.71%; professionals: 80.00%). Student sample was tightly distributed in age (mean 21-22 years, median 20-21 years), and most of them were students on third year of their undergraduate study. Students' programming experience came from either their previous study or programming practice, and their programming activities started within five to nine years. They mainly program using *"Java"* programming language where they started their first line of code more than three but less than five years with intermediate level based on their self-assessments (A sample of demographics for students sample is shown in the Figures 4.2 and the full demographics is given in Appendix in Table B.1). Professionals sample, on the other hand, was bimodal – one mode just slightly older than the students (23-24 years, likely reflecting those in a graduate job or first professional role) and one approximately ten years later (32-33 years). Their previous experiences before starting the current role came from full-time employment. Most of the professionals started their programming activities over twenty years ago, and their most used programming language was *"Python"* where they started their first line of code five to nine years ago with intermediate to expert

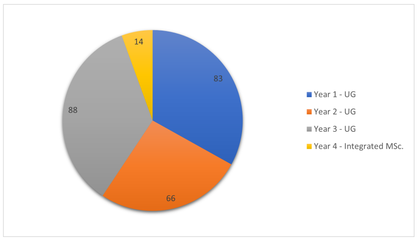
[7]`https://figshare.com/s/3bbb37f34c1bf8dff530`(Note that for review purposes this DOI is not yet active, but this text will link to a private preview of its content).

level based on their self-assessments (A sample of demographics for professionals sample is shown in the Figures 4.3, and the full demographics is given in Appendix in Table B.2). At the following, results presented based on the survey's sections: programming resources, use and perceptions of Stack Overflow, use and perceptions of code online code snippets and programming and human memory. It is worth noting that reporting the results of the five-point Likert-type questions include grouping the answer for readability reasons. For example, the 5-Likert scale consists of: strongly agree, agree, neutral, disagree and strongly disagree, and the survey's results often group the "strongly agree" reported number with the "agree" choice and the "strongly disagree" with "disagree".

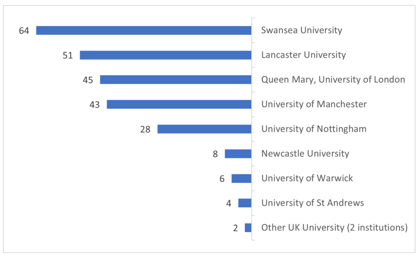
4.2.1 Resource use

The section asked three questions about the resources used by students and professionals during coding, and used a Likert scale of four points: frequently, occasionally, rarely and never. All students and professionals answered this section's questions. The first question was, *"Which of the following resources do you use when programming?"*, with five options: books, websites, colleagues, course tutors and existing codebases. All respondents used at least one of five resource types either frequently or occasionally. Students report use of a wider set of resources, with fourteen students (5.58%), and no professionals, using all five resources frequently or occasionally, and one of those reporting frequent use of all five. Figure 4.4 (top) indicates that website use was universal, and patterns of book use were somewhat similar across the two samples (3% of students and 4% of professionals report frequent use). However, other resources are more heavily used by professionals (frequent use of existing codebases 80% professionals vs. 37% students; friends/colleagues 44% professionals vs. 20% students) or by students (tutors 6% students vs. 0% professionals).

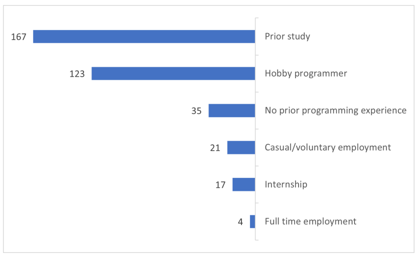
The second question was *"Which of the following websites do you use when programming?"* and offered the following options: GitHub, Quora, Reddit, Stack Overflow, programming language documentation and tutorial websites. Looking closer at website use (summarised in Figure 4.4, center), Stack Overflow found to be the most frequently used by all respondents (85% students; 80% professionals). However, professionals report equally frequent use of GitHub (80%, compared to just 40% of students). Just over half (57%) the student respondents reported frequent use of tutorials (compared to just 40% of professionals), and half of both samples report frequent use of documentation (students: 51%; professionals: 52%). Quora and Reddit use is low



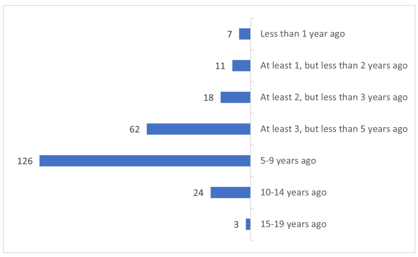
(a) Year of study



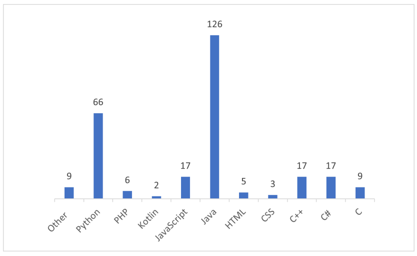
(b) Place of study



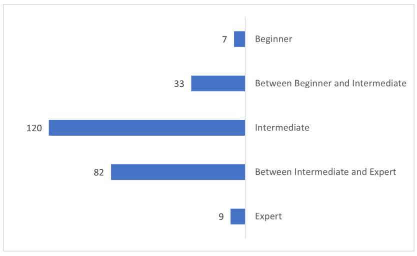
(c) Previous experience



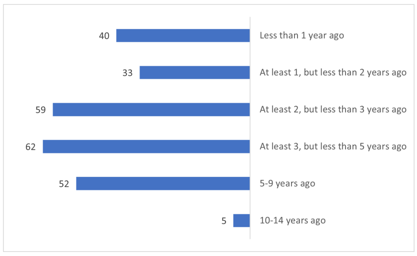
(d) First line of coding



(e) Most proficient programming language

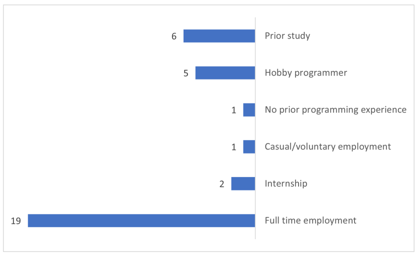


(f) Competency

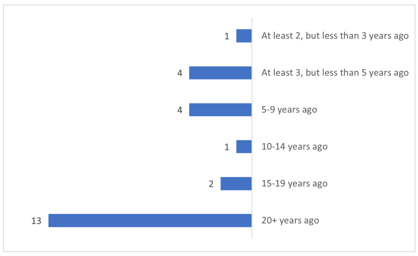


(g) First line of coding with their proficient programming language

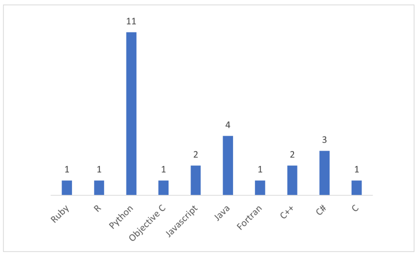
Figure 4.2: Illustration of Students demographic



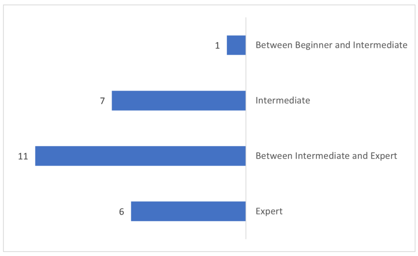
(a) Previous experience



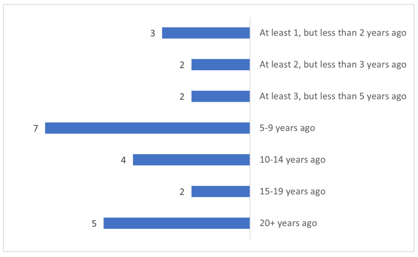
(b) First line of coding



(c) Most proficient programming language



(d) Competency



(e) First line of coding with their proficient programming language

Figure 4.3: Illustration of Professionals demographic

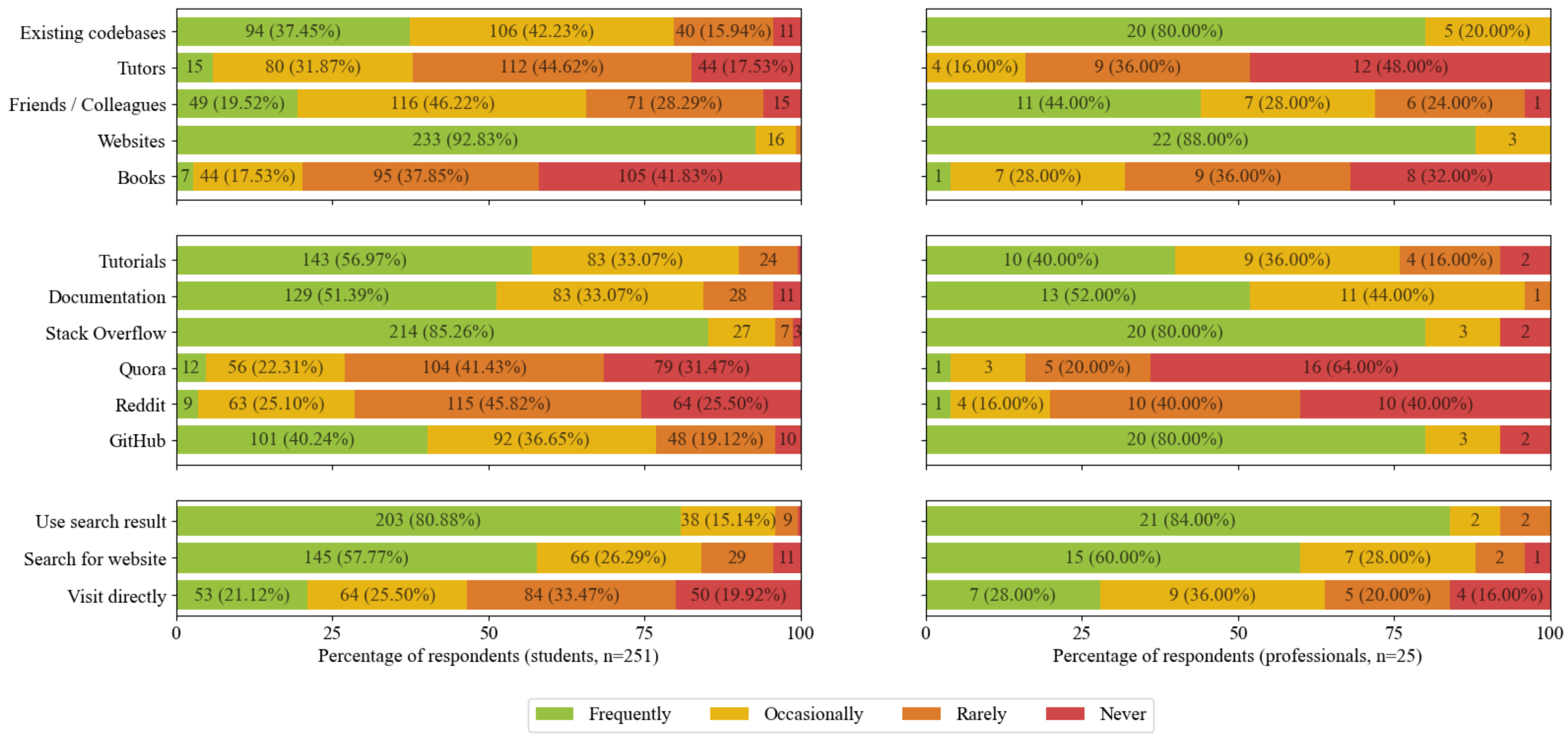


Figure 4.4: Resource use (top, R1-5), website use (centre, R6-11) and website access mode (bottom, R12-14) by students (left) and professionals (right); see Table 4.2 for full question text. For questions R13 and R14, the bar for 'Use search result' corresponds to the option 'I use a search engine and click whichever results look most relevant' and 'Search for website' to 'I use a search engine with the intention of finding content from a specific website'. For legibility reasons, very low numbers of responses (> 3) are unlabelled on the rightmost plot, likewise some percentages are omitted from value labels.

Resource	**Description**
Books	Textbooks and programming books.
Colleagues	Friends and work associate.
Course tutors	Help during programming course.
Codebases	A collection of code.
Websites	Online resources.
GitHub	Website for hosting code.
Quora	Social Q&A website.
Reddit	Social Q&A website.
Stack Overflow	Q&A website designed for programming.
Documentations websites	Official programming websites designed to provide guidance, code samples and tutorials.
Tutorial	Websites that provide examples, explanation, and code sample help to learn programming concepts.

Table 4.3: A list of programming resources with descriptions

in both samples (frequent or occasional use by 27% and 29% of students and 16% and 20% of professionals for Quora and Reddit, respectively). For the list and description of the used programming resources and websites, see Table 4.3.

The last question in this section was *"How do you access information from these websites when programming?"* with the following options: I visit the site directly, I use a search engine with the intention of finding content from a specific website, I use a search engine and click whichever results look most relevant. Figure 4.4, bottom indicates that both students and professionals were most likely to report finding their Web content by clicking the most relevant looking results returned by a Web search (81% of students and 84% of professionals report doing this frequently). The majority also frequently used a search engine to find content from a specific website (58% of students, 60% of professionals), rather than visiting the site directly (21% of students; 28% of professionals).

4.2.2 Use and perceptions of Stack Overflow

This section carries eleven questions regarding the Stack Overflow specifications. Two professional respondents, and ten student respondents, were not asked about their use of Stack Overflow, due to their previous indication that they rarely or never used the website (Figure 4.4, center). A further seven frequent and one occasional student Stack Overflow users chose not to answer the Stack Overflow questions; thus the results in this section are based on a sample of 23 professionals and 233 students. The initial question was *"Comparing to three years ago, I used Stack Overflow"* with three options: less than I used to, more less than I used to and about the same. Figure 4.5 shows that the majority of students (62%) had increased their use of Stack Overflow over the last three years, compared to approximately one third (35%) of professional respondents. However, students with prior programming employment (including casual employment, full-time employment and internships; $n = 31$) reported increased use at a rate that was comparable to professionals (39%); by comparison, those with prior study reported similar increases in use as the overall student sample (61%, $n = 155$). In both the student and professional samples, only a minority had reduced their use of Stack Overflow compared to three years ago (13% students; 17% professionals).

The remaining ten questions are regarding Stack Overflow specifications and used a 5-point Likert scale: strongly agree, agree, neutral, disagree and strongly disagree. In the question *"I cannot program without the Stack Overflow website"*, the reported dependency on Stack Overflow was considerably higher in the professional sample – 39% agreed/strongly agreed that they could not program without it, compared to 21% of students (Figure 4.6); reported dependency for students was roughly the same irrespective of prior experience (16-20%).

The full results of the section are shown in the figure Figure 4.7. Participants were asked the following questions: *"When searching for programming-related concepts on the Web, the Stack Overflow website is the most dominant result"*, *"I can find what I am looking for on Stack Overflow"* and *"I find it hard to tell if the question or answers on Stack Overflow are relevant to my programming tasks"*. Figure 4.7 shows that almost all respondents agreed/strongly agreed that Stack Overflow dominated search engine results (93% of students; 91% professionals), and the majority could easily find what they sought on the site (76% students; 83% professionals). Few respondents report difficulties determining if Stack Overflow content is relevant to their need (20% students; 9% professionals).

With regards to potential indicators of answer quality, this section asked four questions: *"I prefer to use the most upvoted solutions on the Stack Overflow website"*, *"I take the author's reputation into account when deciding how likely the answer will help"*, *"I can identify poor quality solutions on Stack Overflow because they will have been down voted"* and *"I am wary when reading unaccepted answers on Stack Overflow website"*. Both students (57%) and professionals (57%) were most likely to prefer upvoted answers. Students (and to a lesser degree professionals) also reported using downvotes as an indicator (58% of students, 39% professionals); 43.5% of professionals were neutral about using downvoted. Students were also more likely than professionals to be wary when considering unaccepted answers (51% students, 22% professionals). Author reputation was taken into account by a minority of both samples (17% students, 22% professionals).

Finally, with regard to the questions *"Having multiple different solutions, and others' comments on those solutions, is very helpful to me"* and *"I find that different answers or comments conflict with each other"*, around half of both samples noted conflict in that multiplicity (51% students; 48% professionals). However, the majority of respondents also reported that the plurality of content was helpful to them (96% students; 83% professionals).

4.2.3 Use and perceptions of online code snippets

This section is about the activity of adopting online code, and it started with eight questions about online code, followed by assessing programmers' encountering of ten problematic code. 237 students (94%), and all professionals, responded to questions about perceptions and use of online code snippets. The first question was: *"How often do you copy and paste a source code snippet from the web?"*, with a 5-point Likert scale: I do this most days, I do this at least three times a week, I do this at least weekly, I do this at least once a month, and I do this rarely or never. Figure 4.8 shows that the majority copied and pasted a source code snippet from the Web at least monthly (71% students, 64% professionals). As reported frequency increases (from monthly to several times a week), the proportion of professionals engaging in this behaviour (24%) exceeds that of students (14%).

The following questions used a 5-point Likert scale: strongly agree, agree, neutral, disagree and strongly disagree. The section asked participants about the circumstances in which copying online behaviour occurred (Figure 4.9, bottom), specifically: *"I copy code snippets to make up for gaps in my experience or knowledge"*, *"I copy code snippets only if I fully understand their contents"* and *"I copy code snippets only if they are consistent with my own code quality standards"*. Respondents generally agreed that they would copy and paste in response to a gap in their knowledge (54% of students, 64% professionals), but only if they fully understood the code (74% students, 64% professionals), and it met their own quality standards (66% students, 52% professionals).

Two questions were then asked to investigate this further: *"Copying and pasting code hinders programmers' understanding and learning"* and *"Copying and pasting*

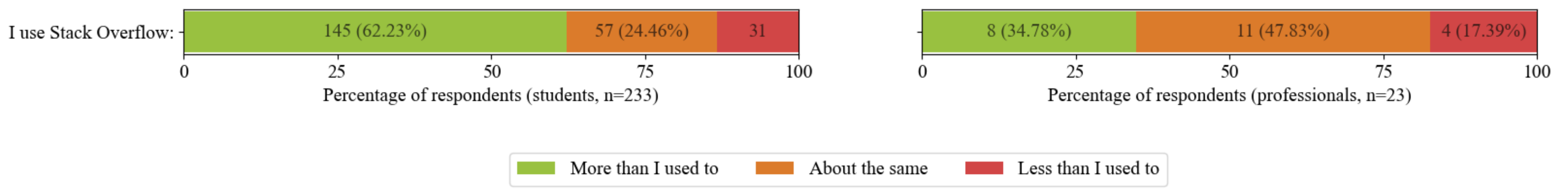


Figure 4.5: Reported change in Stack Overflow usage, compared to three years ago, for students (left) and professionals (right). Statements have been abbreviated, see Table 4.2 (S1) for full question text. For legibility reasons, percentages are omitted from low value labels (> 15%).

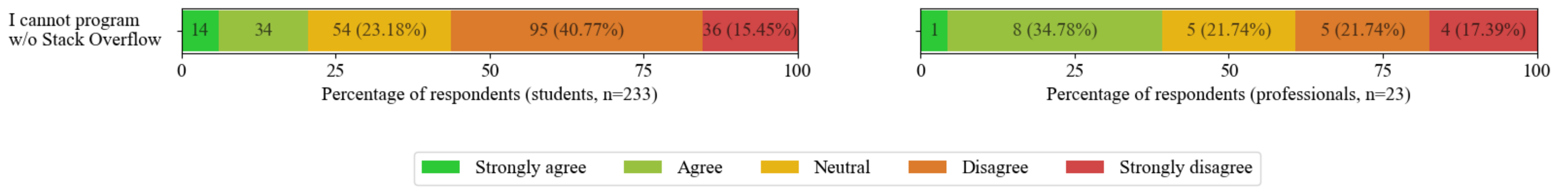


Figure 4.6: Reported dependency on Stack Overflow, for students (left) and professionals (right). Statements have been abbreviated, see Table 4.2 (S3) for full question text. For legibility reasons, percentages are omitted from low value labels (> 15%).

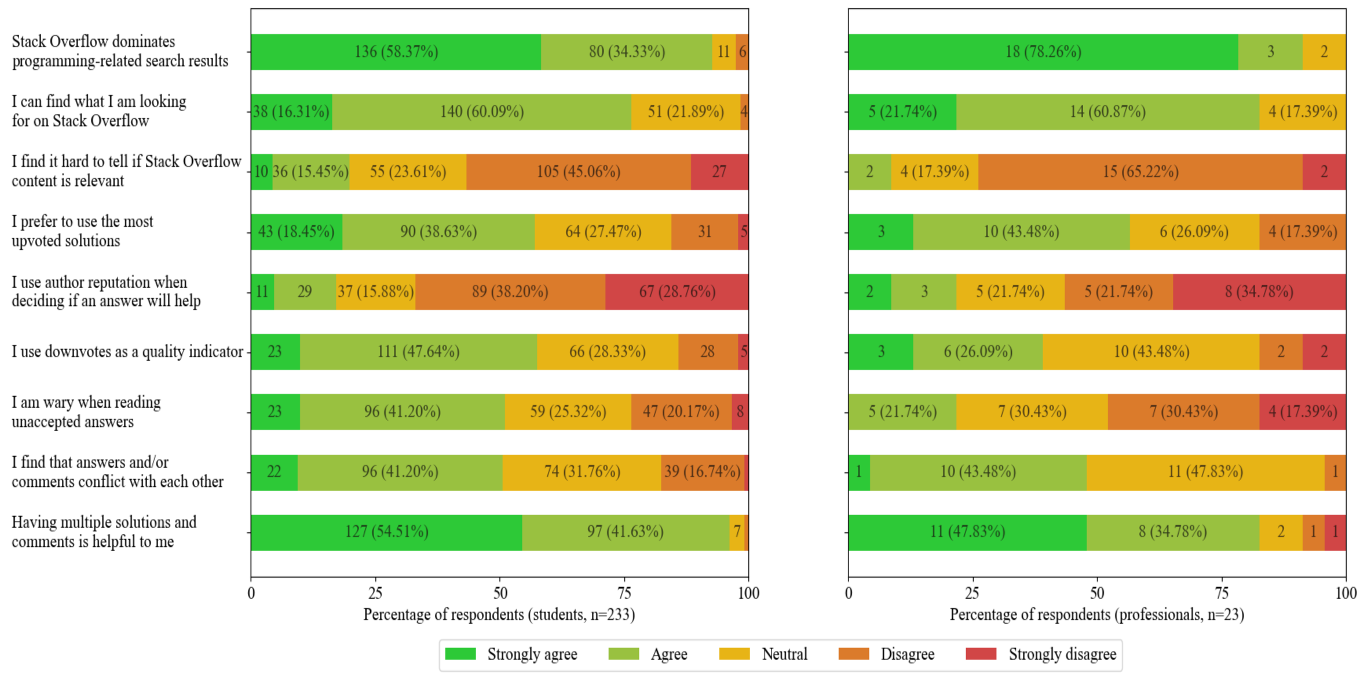


Figure 4.7: Level of agreement for students (left) and professionals (right) for statements about Stack Overflow. Statements have been abbreviated, see Table 4.2 (S2, S4-S11) for full question text. For legibility reasons, percentages are omitted from low value labels (> 15%).

code from websites reduces code quality". Within both samples, respondents held diverse opinions on the potential negative impacts of copy/pasting online code snippets – Figure 4.9 (center). Roughly equal proportions of students agreed (42%) and disagreed (39%) that copying and pasting code snippets negatively impacted code quality. Likewise, similar proportions of students agreed (42%) and disagreed (39%) that copying and pasting code negatively impacted programmer understanding. The majority (48%) of professionals disagreed with both statements, but there was still a significant minority in agreement (reduces code quality: 28% and understanding 36%).

When asked if the majority of online code snippets were of good quality, respondents tended to neither agree, nor disagree (neutral response: 41% of students, 56% professionals), but students were more likely to agree (27%) than professionals (12%). In a *"I trust code snippets found on the web"* question, both samples were more likely to agree that online code snippets were trustworthy (48% students; 44% professionals), although a significant majority still responded neutrally (35% students; 32% professionals). Results for both these questions are shown in Figure 4.9 (top)

The eleven problematic code issues extracted from the interview study were presented to both students and professionals. The question was *"To what extent have you found the following to be present in code snippets on the web?*. Using a 3-point Likert scale, respondents determined for each issue if they encountered the issue, have not encountered it but are aware of it, or are unaware of it and do not think of it as an issue. The issues were outdated code, code that does not compile/run, undocumented statements, inefficient/overly complex code, insufficient or incomplete code, code that is difficult to integrate, redundant statements, code with extraneous output, security issues, code with incorrect output and licence violations.

Of the eleven problematic code presented to respondents, seven had been directly experienced by a majority of both samples: code that is outdated (i.e. written for an non-current version of a programming language or library), does not compile/run, is difficult to integrate, contains undocumented statements, or redundant statements, is insufficient or incomplete, or is inefficient/overly complex (see Figure 4.10). For the remaining problems, one (extraneous output) had been encountered by professionals (64%); for the remainder respondents were generally aware that these issues were present in online code (students: $40-64\%$, professionals: $52-68\%$) despite not having directly encountered them. A substantial minority (37%) of students (and to a lesser degree, professionals 20%) were reportedly unaware of licence violations as a problem in online code snippets.

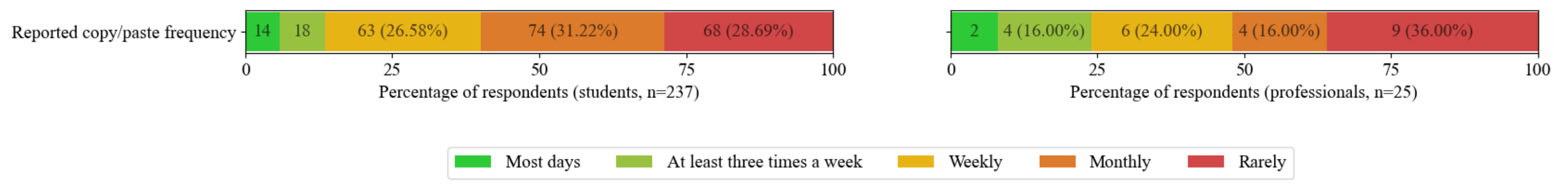


Figure 4.8: Frequency with which students (left) and professionals (right) report copying and pasting a code snippet from the Web. Statements have been abbreviated, see Table 4.2 (C1) for full question text. For legibility reasons, percentages are omitted from low value labels (> 15%).

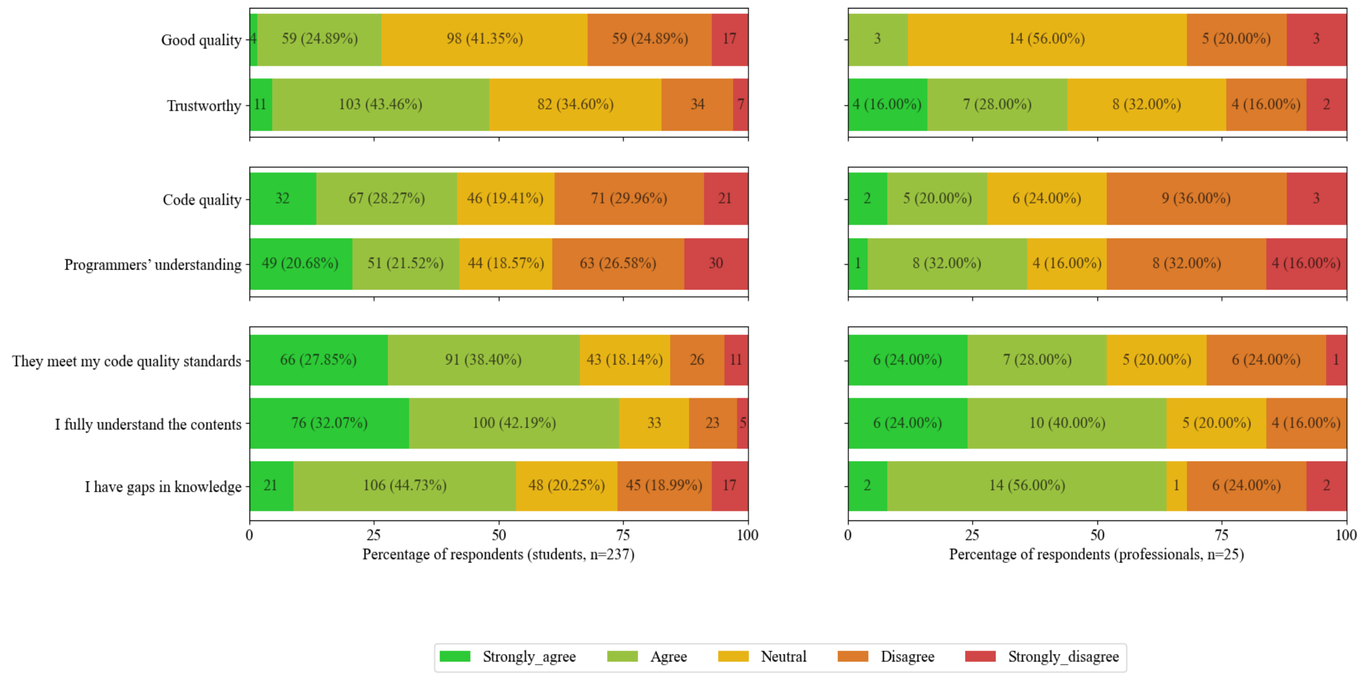


Figure 4.9: Level of agreement for students (left) and professionals (right) for statements about code snippets (top, C2 & C8), the negative impacts of copy and pasting code snippets (center, C6 & C7), and circumstances in which the respondents would copy and paste a code snippet into their own code (bottom, C3-5). Statements have been abbreviated, see Table 4.2 (C2-8) for full question text. For legibility reasons, percentages are omitted from low value labels (> 15%).

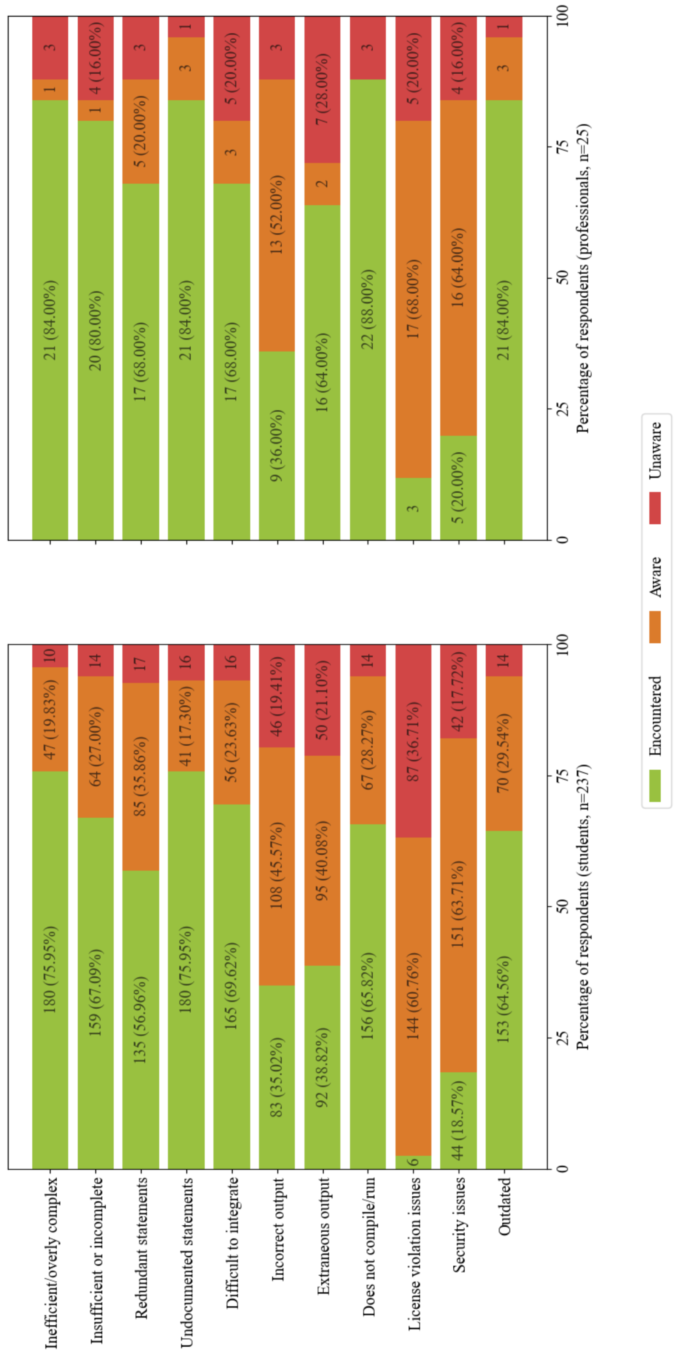


Figure 4.10: Issues encountered in online code snippets by students (left) and professionals (right); see Table 4.2 (C9-18) for full question text. For legibility reasons, percentages are omitted from low value labels (> 15%).

4.2.4 Programming and human memory

This section asked twelve questions grouped into two parts: using the programmers' memory during coding and remembering online contents. All the questions used a 5-point Likert scale: strongly agree, agree, neutral, disagree and strongly disagree. 232 students (92%) and twenty-three professionals (92%) responded to questions in this section. The results for first part questions for both students and professionals are displayed in Figure 4.11. In terms of using the programmers' memory during coding, the section started with two questions: *"Having a good memory is critical to successful programming"* and *"I have a good memory for programming concepts and syntax"*. Figure 4.11 shows that similar proportions of both samples agreed that having a good memory was critical to successful programming (54% students; 48% professionals). Students were most likely to report a good memory for programming concepts and syntax (78% of question respondents, compared with 57% of professionals). In addition, participants answered three questions: *"It is faster to remember programming-related information than it is to look it up"*, *"Being unable to remember programming concepts bothers me"* and *"I can program non-trivial applications using my memory alone"*. Respondents found that it is faster to remember programming-related information than it is to look it up (students: 58%; professionals: 48%). Conversely, students were also less likely to report being able to program using memory alone (48%, compared to 57% of professionals), and were more likely to report being bothered by an inability to remember programming concepts (62%, compared to 43% of professionals). In answering the question *"When solving a new programming problem, I am able to remember similar problems I have solved in the past"*, almost all respondents (students: 89%; professionals: 91%) reported that when solving a new programming problem, they are able to draw on memory for similar problems solved in the past.

In the second parts, it asked two questions *"Programming content on the Web is for reference not learning"* and *"There is no need to try and remember programming concepts because websites are always available"*. As seen in Figure 4.12, the majority of respondents felt that online programming content supported learning, rather than simply acting as a reference (students: 58%; professionals: 61%); likewise, 55% of students and 43% of professionals report that despite the ubiquitous availablity of relevant online content, there was still value in trying to remember programming concepts. The answer for the questions *"Looking at programming content on the Web confirms what I already know or reminds me of something I had forgotten"* revealed that almost

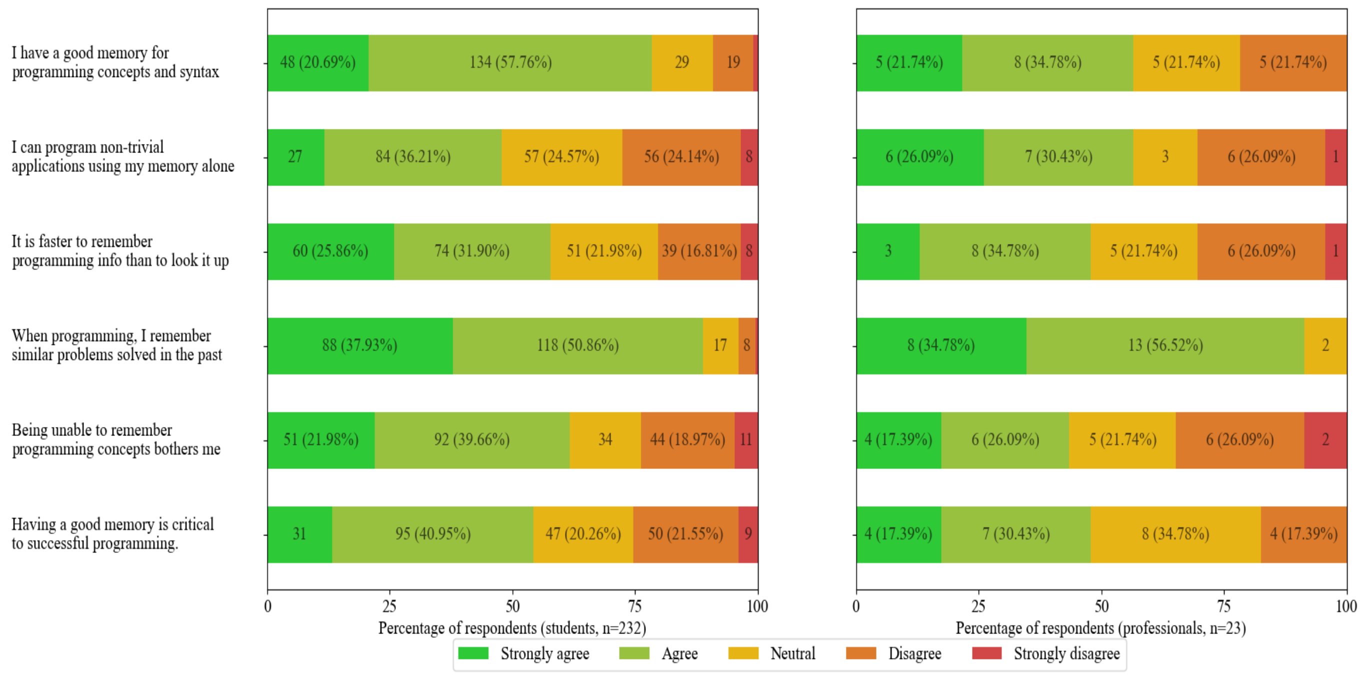


Figure 4.11: Level of agreement for students (left) and professionals (right) for statements about memory. Statements have been abbreviated, see Table 4.2 (M1-6) for full text. For legibility reasons, percentages are omitted from low value labels (> 15%).

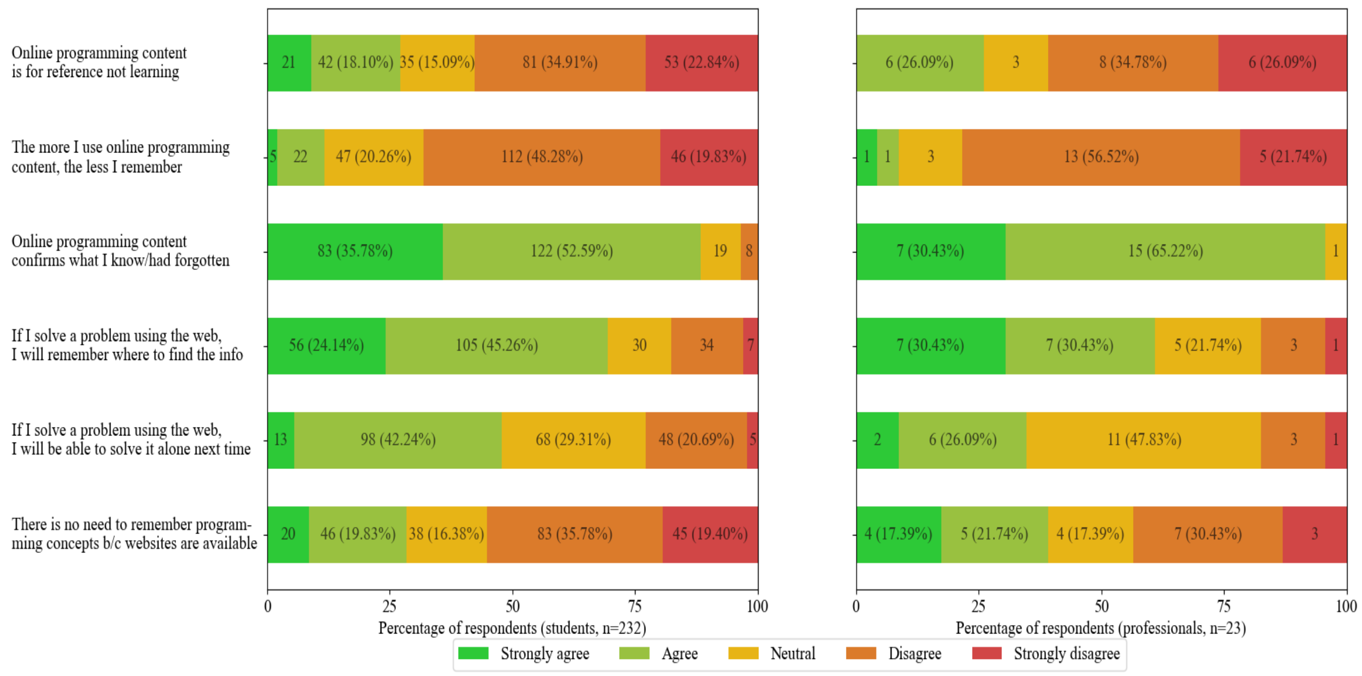


Figure 4.12: Level of agreement for students (left) and professionals (right) for statements about memory. Statements have been abbreviated, see Table 4.2 (M7-12) for full text. For legibility reasons, percentages are omitted from low value labels (> 15%).

all respondents indicated that online programming content helped to confirm or remind them of information that they already knew (students: 88%; professionals: 96%). Then, two questions about constituting knowledge from previously solved problem using the web: *"If I have previously solved a problem using the Web, I will be able to solve the same problem in the future without looking up the information again"* and *"If I have previously solved a problem using the Web, I will remember where to find the information needed to solve the problem next time"*. Where the content was unknown, responses did suggest that respondents generally did commit either the learned content or its source to memory – just over a third of professionals (35%) and almost half the students (48%) report that they would be able to solve the same problem in the future without looking up the information again, while around two-thirds of respondents would remember where to find the relevant information (students: 69%; professionals 61%). Overall, the answer to the questions *"The more I use programming content on the Web, the less I remember"* indicate that there was very little agreement that greater use of online programming content was negatively impacting the respondents' ability to remember (students: 12%; professionals: 9%).

4.3 Discussion

Following the results from the interview study in the previous chapter (Chapter 3), this section discusses the survey results and summarises the findings for each research question. It also compares the survey findings with the interview findings and provides the sample difference between interview and survey studies.

4.3.1 RQ1: What resources do programmers use to support the coding process? How and why do they use them?

Survey findings revealed that both students and professionals make significant use of online resources: with professionals also using the codebases they are working on, or have access to, as another primary resource. Student resource use is motivated by inexperience and time-bound assignments, leading them to value websites for their ease and speed of access, particularly when using search engines to identify relevant content. These constraints also explain their low rates of using books (also seen in other studies [Lausa et al., 2021]) and course tutors (when students complete work outside of teaching sessions). Whilst interviewed professionals noted that working closely with colleagues could be disruptive to others and not always possible, surveyed professionals actually were more likely than student respondents to report seeking help from friends or colleagues.

Student interviewees noted that friends were particularly valuable when Web searches did not return anything relevant to specific course assignments, or when they were struggling to understand and integrate online content. Overall, less than half (44%) of professional participants in the survey (compared with 20% of the students) reported frequently seeking help from colleagues, considerably fewer than those observed in prior studies ($>$ 80% [Maalej et al., 2014]). This difference, together with the conflict between survey and interview responses, suggests that the behaviour may be highly variable in professional samples, potentially influenced by the culture and practice of companies and development teams.

Both interview and survey findings indicated high rates of using Stack Overflow, mediated through an initial search engine. Again, this aligns with prior studies that have shown high return rates for Stack Overflow from technical searches [Hora, 2021, Xia et al., 2017] and extensive use of the website in development [Acar et al., 2016]. By contrast, interviewees rarely mentioned competitor Q&A websites, such as Reddit and Quora, and survey respondents reportedly rarely or never used them. Other websites, e.g. documentation and tutorials, were used but less frequently than Stack Overflow.

In the absence of significant development experience of their own, student interviewees reported using Stack Overflow to draw on the experiences of others. However, even experienced professionals described their dependency on Stack Overflow, and the survey findings confirmed that their frequency of using it remains largely static, compared to three years prior.

Comparing student and professional survey responses suggests that whilst similar proportions report frequently using Stack Overflow and successfully finding the content for which they look on the site, professionals are more likely to agree (and to disagree) that they need the website to program. A majority of the students responded neutrally to this question, indicating that dependence on Stack Overflow only emerges after a period of extended use. This increase may be attributed to a number of factors. As students grow in experience, they tackle more challenging problems, which Stack Overflow content may better address (in contrast to fundamentals, which books and tutorials readily serve). Successful problem-solving then acts as positive reinforcement and encourages habit formation.

Interview respondents noted the value of Stack Overflow features such as as multiple answers, comments and up/down votes. However, they also commented on conflicting answers where the answer needed to solve their problem was not the one upvoted or accepted. Current surveys show a similar tension, whilst most find having multiple answers and comments helpful, around half observed conflict, and a similar proportion reported the limited impact of upvotes/downvotes. Use of non accepted answers as an information source has also been observed in studies of source code [Wu et al., 2019].

Survey results indicated that professional developers make equally frequent use of GitHub. This behaviour did not emerge in the interviews (only one professional made any mention of Git, and none discussed GitHub), and it was only included in the survey as a result of mentions from student interviewees. Thus, high levels of reported use can be observed in the survey, but little can be said about how developers are leveraging these sites during their programming tasks. Greater use of GitHub amongst professionals may be symbiotic with their greater use of existing code. GitHub provides a means of accessing codebases but is likely easier to navigate for those who are already familiar with the process of navigating code that others have written.

4.3.2 RQ2: What is the perceived impact of using websites on programmers' memory?

The results suggest that use of online content is partly motivated by temporary inability to remember programming concepts and syntax or the perceived effort associated with recall of that information. Those with less experience (i.e. students) or impaired by working in multiple programming languages (e.g. those using a language new to them, or who regularly switch between languages) are particularly likely to perceive the barrier to Web use to be lower than that of their own memory. In other contexts, the time taken to switch from programming to information-seeking was a potential barrier, and roughly half of survey respondents reported finding it faster to remember programming-related information than to look it up. Their ability to do so is evidently dependent on their memory capability and sufficiency, with a majority of survey respondents reporting that they have a good memory for programming syntax/concepts, and they can program nontrivial applications using memory alone.

Interestingly, students and professionals differ on their rates of agreement with these two statements. More students report a good memory, but more professionals report the ability to program with memory alone (similar conflicts also appear in some of other memory-related survey results, such as using memory to solve problems previously solved with websites or locate the required information quickly, and the inconvenient of being unable to remember information). Differences between students and professionals in the latter statement seem intuitive and are likely due to the students' relative inexperience. Conversely, the idea that students overall would report better memory for programming concepts than professionals seems counterintuitive and conflicts with prior empirical evidence [Ichinco & Kelleher, 2017]. One possible explanation is that these students are making relative judgements, determining that their memory is good compared to peers. Another potential factor is the likely smaller set of syntax and concepts that students are expected to remember. Since students' programming experiences make only light-to-moderate demands on memory, they perceive that they are fulfilling this better than professionals whose memory carries a greater load.

In more generalised domains, there has been some concern that cognitive offloading may negatively impact independent ability to recall [Fisher et al., 2022, Sparrow et al., 2011]. The results suggest that whilst programmers may see websites as a substitute for memory when coding, they do not perceive using them as negatively impacting their abilities (only around 10% of survey respondents agreed that online content negatively impacts their ability to remember). However, not everything seen online is remembered. Only around one-third to one-half of the survey respondents indicated that they could independently remember how to solve a previously solved problem with the support of online resources. More common and consistent with other studies of digital information retrieval [Sparrow et al., 2011] was the report that they would remember the information locations. In this case, there is some evidence that programmers' perceptions may not be wholly accurate. Source attribution error may lead them to subsequently come to believe that the information was sourced from their own semantic memory, giving them false confidence in their abilities [Fisher et al., 2015, 2022]. This attribution error may explain some of the conflicting interview comments, where participants (particularly students) describe online content as both contributing to and impeding memory and learning.

Overall, results are encouraging, suggesting potentially little (perceived) impact of online resource use on memory for coding, but also suggesting some unawareness among participants of exactly what memory's role is in programming (i.e. limited metamemory). Whilst roughly 90% of survey respondents said that they could draw on their memory of prior programming experiences, only around half felt there was value in trying to remember programming concepts, and that memory was critical to programming. The interviews suggest that programmers may be differentiating between memory (for syntax) and "understanding", with prior work suggesting that programmers consider conceptual knowledge more important to remember [Krüger & Hebig, 2020]. Thus, disagreement about the importance of memory in programming may simply be down to the semantics of what programmers consider the term "memory" to include.

4.3.3 RQ3: How does use of websites affect programmers' code?

The results found that programmers reuse online code, copying and pasting snippets into their own source files. Student interviewees were motivated to reuse code to meet their course requirements. Around half of the survey respondents indicated that they copied code in response to a gap in knowledge; both interviews and surveys suggest that in other cases, online content acts as a prompt that refreshes existing knowledge. Amongst survey respondents, a greater proportion of students used a code snippet from the Web at least monthly, but the greater proportion of professionals used them at least three times a week or more frequently. This may be due to the more sporadic nature of student programming activities. Whilst some students may code regularly to develop and maintain skills, others may write code only in response to imminent deadlines resulting in extended periods with no coding activity (and thus no opportunity for online code reuse).

Levels of understanding for copied code snippets varied. Interview participants generally indicated that despite good reasons to try to understand code found online, it was not always possible because some code was beyond students' current competency (a problem also noted by survey respondents in [Escobar-Avila et al., 2019]). Also, time pressures that deadlines create mean that students could not spend the time needed to understand the code they were using. Overall, interviews and surveys showed that professionals and students understand online code they reuse, but survey responses indicated that professionals were more willing to copy code without fully understanding it. This finding is consistent with evidence from studies of program comprehension, in which novices expend more effort understanding code, and experts use a variety of strategies, including higher-level abstraction rather than detailed line-by-line understanding [LaToza et al., 2007, Maalej et al., 2014]. This finding may also reflect students' reasons for writing code in the first place. Students' coding assignments intend to facilitate learning, so that may have supported a greater desire for understanding. Furthermore, since a team of university researchers issued the survey, some students may have felt the need to report what they perceived to be the answer the researchers desired, rather than an honest one.

One potential risk of copying online snippets is introducing problematic code into developers' software [Abdalkareem et al., 2017a, Acar et al., 2016, Fischer et al., 2017]. The survey results suggest that both students and professionals give some attention to code quality when choosing whether or not to copy and paste code snippets from the Web. However, the interviews indicated that students may not always be well-placed to judge when a snippet contains problematic code (e.g. *"I don't know what code quality means"*, P4). This potentially explains students' split responses to the survey question about the negative impacts on code quality of using code snippets (C7). By contrast, professional interviewees felt more confident in identifying good- and poor-quality solutions and code on Stack Overflow, a skill that could mitigate negative impacts of code reuse.

Interview participants were asked whether, in their experiences of using online code snippets, they had observed issues. Interviewees reported eleven distinct issues with code, which are listed in Table 4.4. Some of the reported issues were previously noted in the literature (e.g. security vulnerabilities [Acar et al., 2016, Fischer et al., 2017, Licorish & Nishatharan, 2021, Meng et al., 2018], licence issues [Ragkhitwetsagul et al., 2019] and poor code quality [Ragkhitwetsagul et al., 2019, Treude & Robillard, 2017]). Based on the experience of both students and professional programmers, the list included issues that they may have encountered during coding with websites, not pertaining to specific programming languages or websites. In addition, encountering the issues does not necessarily engage the action of code copy-and-paste; it means that such code with issues resides online and will likely be adopted.

Survey participants were asked about each issue the interviews elicited. Awareness of the potential for encountering each issue was high (students: 63-96%; professionals 72-96%), and most issues had also been directly encountered all but four issues by $> 50\%$ of students, all but three issues by the same proportion of professionals). For ten issues (i.e. all but one), more professionals than students reported a direct encounter. The remaining issue, that code is difficult or impossible to integrate, is likely to be

	Interview Occurrences		**Survey Occurrences (%)**		
Issue	**Students**	**Professionals**	**Students**	**Professionals**	**See also**
Inefficient or overly complex code	✓	✓	75.95 (95.78)	84.00 (88.00)	[Treude & Robillard, 2017, Wu et al., 2019]
Undocumented code	✓	-	75.95 (93.25)	84.00 (96.00)	
Code that is difficult or impossible to incorporate/integrate into an existing project	✓	-	69.62 (93.25)	68.00 (80.00)	[Wu et al., 2019]
Insufficient or incomplete code	✓	-	67.09 (94.09)	80.00 (84.00)	[Treude & Robillard, 2017, Wu et al., 2019, T. Zhang et al., 2018]
Code that does not compile or does not run	✓	-	65.82 (94.09)	88.00 (88.00)	[Yang et al., 2017]
Outdated – code that does not work with the current version of a language or library	✓	✓	64.56 (94.09)	84.00 (96.00)	[Ragkhitwetsagul et al., 2019, Wu et al., 2019]
Redundant code	✓	-	56.96 (92.83)	68.00 (88.00)	[Nishi et al., 2019]
Code with extraneous output	✓	✓	38.82 (78.90)	64.00 (72.00)	
Code with incorrect output	✓	-	35.02 (80.59)	36.00 (88.00)	
Security issues	✓	✓	18.57 (82.28)	20.00 (84.00)	[Acar et al., 2016, W. Bai et al., 2019, Fischer et al., 2017, Licorish & Nishatharan, 2021, Meng et al., 2018]
Licence violations	✓	✓	2.53 (63.29)	12.00 (80.00)	[An et al., 2017, Ragkhitwetsagul et al., 2019, Wu et al., 2019]

Table 4.4: Comparison of issues observed in online code snippets by interview participants and survey respondents. Denoted interview occurrence means that one or more participants reported directly encountering this issue. Percentage of survey respondents who reported directly encountering the same issues are given without parenthesis. The total percentage of respondents who had encountered or were aware that the issue was present is given in parenthesis.

more frequently reported by students, simply due to their relative inexperience. It may not be that code on the platform generally is hard to integrate but, rather, that students have relatively little experience with integrating existing code into their own projects. Similarly, lower report rates by students of other issues likely result from (i) fewer overall interactions with the site than professionals, and (ii) a reduced ability to recognise these other issues (such as security concerns), due to a relative lack of experience. Prior research has also demonstrated poor coverage of code quality in taught courses [Börstler et al., 2017, Kirk et al., 2020], making experience critical in judging the quality of online code snippets.

Across both surveys and interviews, security issues and licence violations had markedly lower report rates than other issues. For students, once again, this is at least partially attributable to inexperience, in terms of being in circumstances where security/licensing issues would arise and having the ability to recognise the issues when encountering them. However, professionals' reported few encounters, despite prior studies that show evidence of developers both finding and utilising problematic code [Acar et al., 2016, Fischer et al., 2017, Meng et al., 2018, Ragkhitwetsagul et al., 2019].

Stack Overflow provides a number of quality indicators that could help inform users to assess answers and associated code snippets. As noted in Section 4.3.1, although around half of the survey respondents took them into account when reading Stack Overflow content, participants did not consider upvotes/downvotes sufficient. This aligns with prior work indicating that upvotes/downvotes are not sufficient to determine the quality of the answer (and by extension, the code snippet) [T. Zhang et al., 2018], and with the discovery by [Wu et al., 2019] that programmers retrieve answers from the not-high scores and unaccepted answers from the Stack Overflow. By contrast, only a minority of student and professional survey respondents considered the reputation metric. Students were more likely to report wariness when considering unaccepted answers on Stack Overflow, an indicator that professionals largely seemed to ignore, but current evidence suggests that no quality differences exist between accepted and unaccepted answers [T. Zhang et al., 2018].

Overall, participants appear to appreciate that online code-snippet quality is variable, even where their own experiences have been largely positive. Online code entailed different types of issues, extending the discovery by [Wu et al., 2019, W. Bai et al., 2019] that Stack Overflow has low-quality code causing code causes issues when adopted. However, the findings are unlike what [Escobar-Avila et al., 2019, Meldrum et al., 2020] mentioned that the programmers were pleased with the quality of the online code and that it is not a concern. The findings stressed students' vulnerability to facing online quality issues because they lack the skills and knowledge required to assess such encounters, agreeing with [W. Bai et al., 2019] that the lack of required experience means less ability to assess the code.

4.3.4 Sample differences between interview and survey

The interview and survey includes four samples: one professional and one student sample were involved in interviews, with a further pair of developer and student samples completing the survey (overlap between these samples is likely, particularly in the case of professionals where the survey deliberately distributed to previous interviewees). Some differences in findings between these samples were noted as follows.

4.3.4.1 Between interview and survey

1. Student interviewees reported use of online code repositories (e.g. GitHub) and of course materials, something that was not reflected survey responses.
2. Interviewees were more likely than survey respondents to report copying and pasting code that they did not understand.
3. Interviewees were more likely than survey respondents to comment on the potential for online resource use to negatively impact memory.
4. Rates of encountering problematic code on Stack Overflow were higher in surveys than would be suggested by the frequency of similar interview comments.

Regarding the items above, timing may be a factor in (1). Interviews took place earlier in the academic year, when students were perhaps more likely to be closely engaging with their course materials. For online repositories, there may also be an effect from the high proportion of first year University of Manchester undergraduates in the interviews. Whilst the perceptions and behaviours of these students are probably largely comparable to those at other universities, the University of Manchester requires students to submit many assignments through an internal GitLab instance. Examining University of Manchester survey respondents use of GitHub, more frequent (47%) and frequent/occasional use (84%) can be seen compared to the remaining student respondents (39% frequent, 75% frequent or occasional).

Items (2) and (3) may simply emerge from the longer, richer engagements that came from interviews, but they may warrant further investigation.

Item (4) perhaps reflects a difference in between abstractly being asked if the issues had been encountered versus asking participants to articulate specific experiences. Survey participants may have had a general sense that the issues must have been encountered, but in the absence of being able to draw on a specific example, interviewees passed over the topic more quickly, giving an impression of lower prevalence. As previously noted, measures of problematic code on Stack Overflow are also highly variable.

4.3.4.2 Between students and professionals

1. Developers make use of existing codebases and online code repositories comparable with their use of Stack Overflow; students do not.

2. Developers are more likely to report dependence on Stack Overflow, engage in frequent copy-and-paste and report a poorer memory for programming concepts

3. Students are more likely to try to understand a code snippet prior to reuse.

4. Professionals are both more likely to have encountered and to be more aware of licence violations on Stack Overflow.

Item (1) is largely explained by developers' greater access to existing codebases and an increased ability to understand large bodies of code written by others [Sadowski et al., 2015]. As noted in Section 4.3.2, Item (2) is potentially a result of greater demand on professional developers, and Section 4.3.3 notes that Item (3) may be influenced by students being particularly motivated to understand, and by response bias. Item (4) is explained by comments in interviews that highlight the potential legal repercussions of using incorrectly licenced code in commercial software.

4.3.5 Limitations and threats to validity

Study limitations are discussed according to the four threats to validity proposed by Runeson & Höst [2009]: construct validity, internal validity, external validity and reliability.

The interview responses were used to develop the survey questions, and these were subjected to a pilot study to minimise threats to construct validity. A threat to construct validity may have emerged as a property of interviewing student participants from University of Manchester. Whilst it was clearly indicated to participants that their responses would be treated anonymously, with no impact on their study or outcomes, students may still have been reluctant to divulge behaviours that they thought were negatively perceived by the academe. Students at UK universities are regularly advised against activities that might constitute plagiarism, such as copying and pasting from external sources. This, therefore, may have led students to minimise disclosure of these behaviours during the survey.

Although some differences in the perceptions and behaviours of students and professionals were noted, this research does not set out to examine causal relationships; thus, internal validity is of limited concern. However, the potential influence of external factors was noted, particularly on student responses. For many students, time spent on programming activities is driven by course requirements (e.g. deadlines), meaning that the survey responses could have been influenced by their current workload.

The survey used sampling methods that required participants to self-select (i.e. respond to advertisements), with potential implications for external validity. In particular, the recruitment materials used explicitly referred to the use of resources during programming activities, which may have led to particular patterns of self-selection/non-response that impacted the generalisability of the results.

The participants were recruited from two populations in the UK: undergraduate students studying in a computer science or software engineering programme, and software developers who had been in a professional role for at least one year. The choice was made to deliberately approach a large pool of computer science departments (168 from approximately 200 offering relevant courses) to maximise the representativeness of the survey sample. The final survey sample includes respondents from ten UK universities. Professional participants and respondents were also recruited as diversely as possible, with the sample drawn from a range of small and large organisations. On the basis of these recruitment strategies and the high degree of overlap between interview (from Chapter 3) and survey results, there is a confidence that the results are representative of the overall populations from which the samples were drawn.

Survey samples are dominated by students, a reflection of the relative ease of recruiting students when compared to professionals. This imbalance may have led to bias in the results. However, there is a considerable alignment in the results from students and professionals, implying that many behaviours and perceptions likely generalise across groups.

Multiple members of the research team were involved at every step of planning and analysing the survey to maximise the research reliability. As noted in Section 4.1.4, a small number of exclusions were made during the review of the survey data. Three further student responses were minimally cleaned during analysis. These students had indicated no prior programming experience prior to enrolling in their current programme of study and also indicated either prior study or hobby programming. In these cases, the response "no prior programming experience" was removed. Whilst it is possible that other inaccurate values were submitted, there was a limited incentive to deliberately provide inaccurate responses, and the relatively short duration of the survey should have minimised the effect of fatigue.

4.4 Conclusion

This chapter describes a survey that collected data about the use of online resources during programming tasks, and the perceived effects of that use on programmers' memory and code. The study analyses data from 276 respondents, including both student and professional programmers.

The study confirms that Web use is ubiquitous as a primary resource for coding, heavily dominated by use of Stack Overflow. Professional programmers also use GitHub, and other codebases, with equal frequency. Both students and developers arrive at Stack Overflow through a search engine (namely Google).

The results indicate that although use of online programming content often acts as a reminder for things already known, most programmers felt that they had a good memory for programming concepts and that this was not being diminished by use of the Web. Instead, students described engaging deliberate effort to learn from what they had seen online. Surveys' participants did, however, report a greater tendency to recall where they had found information, rather than the information itself.

In addition, participants do exploit the online code. Students, in particular, suffer from understanding and assessing the quality of the copied code. When using the websites, both participants types faced problematic code that deteriorated the quality of the code, and professionals are more susceptible than students to encountering problematic code. High proportions of participants had encountered problematic code online, and eleven specific classes of problem were highlighted based on the interviews outcomes (see Chapter 3 for the interviews outcomes, and table 4.4 for list of problematic code). Despite these problems, the majority of programmers reported engaging in frequent reuse of online code snippets and did not always understand the code they were reusing. Participants found limited value in the metrics that Stack Overflow provided to support the process of discernment between good and problematic online content (e.g. reputation, upvotes, downvotes, accepted answers).

Chapter 5

Strategies for Using Websites to Support Programming and their Impact on Source Code

The findings from previous chapters (Chapters 3 and 4) discovered that programmers used websites to reuse the code and encountered problematic code. Previous research explored online coding activities [Astromskis et al., 2017, Wang, 2017, Ciborowska et al., 2018]. In coding, searching is ubiquitously conducted to augment programmers' knowledge [Li et al., 2013] and for code comprehension and reuse [Hora, 2021, Maalej et al., 2014, Xia et al., 2017]. Others examined the code in relation to online activities, and found that complex code means more compiling and online searching [Astromskis et al., 2017], and using Stack Overflow delivered working but insecure code [Acar et al., 2016]. However, no works have studied programmers' online coding activities and investigated their implications for the resulting code. In addition, activities of using websites could influence the development of code and provide insights into the existence of problematic code. Existing investigations explored some of the mentioned problematic code by retrieving and examining online code [Ragkhitwetsagul et al., 2019, T. Zhang et al., 2018, Acar et al., 2016]. Despite the potential to detect problematic online code, little is known about whether these issues could exist in the code resulting while coding with the websites. This chapter addresses this gap by examining programming activities when coding with websites, and then examines how such activities may affect the produced source code. The purpose of such exploration is to examine the impacts of websites on coding and the resulting code.

This chapter presents an online experiment that used an observation method to record and observe programmers' coding sessions. The choice of the observation is widely used in software engineering research [Seaman, 1999], providing rich data. The study recruited ten undergraduate students to solve four Java programming tasks with varying levels of difficulties created based on their previous course. The experiment used Zoom software to conduct the experiment, record participants' screens and observe coding activities and behaviours within 50 minutes. The participants produced videos and source code files. The videos were analysed using reflexive thematic analysis [Braun & Clarke, 2006], and the collected participants' source code were analysed systematically to investigate the possible implications. In the end, the study interviewed participants to understand in more depth why they used particular activities and how these affected their code.

The results of the observations suggest that the predominant coding strategies were around searching and retrieving syntax from the Web, and involved tutorial websites. Participants produced three types of source code: correct code that works as required, non-executable and incorrect code. Each of these source code types was linked with the online discovered activities. Coding with the Web and encountering complex issues that increased task completion time and effort helped participants produce correct code. However, Web usage impacted the resulting code, producing either incorrect or non-executable code. Thus, using the Web during coding produced incorrect code.

The research questions for this chapter are:

RQ1 How do programmers use websites during programming?

RQ2 What are the effects of websites use on the resulting code?

5.1 Methodology

This chapter presents an online experiment the researcher conducted remotely, using three sequential phases: recording participants' videos, collecting their source code and interviewing them (refer to Figure 5.1 for more information about the experiment's phases). Such an approach allows triangulating multiple data sources to build a rich understanding of participants' behaviour [Seaman, 1999]. The experiment was conducted with undergraduate students recruited from the University of Manchester. Prior to recruitment, a pilot was used to ensure the clarity and validity of tasks and assess the experiment's overall duration. Procedures for the experiment followed the regulation from the University of Manchester regarding recording participants and using an online instrument, and the Department of Computer Science Ethics Committee at the University of Manchester reviewed and approved the study procedure [1]. All data were anonymised at the time of collection. A description follows of the detailed steps in the study phases for collecting and analysing data.

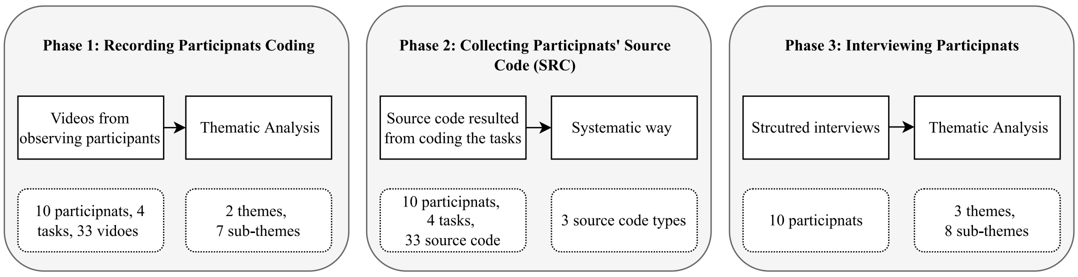

Phase 1: Recording Participnats Coding
Videos from observing participants
Thematic Analysis
10 participnats, 4 tasks, 33 vidoes
2 themes, 7 sub-themes
Phase 2: Collecting Participnats' Source Code (SRC)
Source code resulted from coding the tasks
Systematic way
10 participnats, 4 tasks, 33 source code
3 source code types
Phase 3: Interviewing Participnats
Strcutred interviews
Thematic Analysis
10 participnats
3 themes, 8 sub-themes

Figure 5.1: The three phases of the study with details

5.1.1 Task design

This study phase required participants to engage in a number of programming tasks appropriate to the participants' experience and expertise. Having limited the study's recruitment to a single cohort of undergraduate students in their second year at the University of Manchester, the tasks were designed based on taught course materials from their first year of study. These included a course textbook *"Java in two semesters"* [Charatan & Kans, 2019], lecture slides, workshops with instructions for programming exercises and lab manuals with sample solutions to those exercises. The process of designing the tasks included considering the programming exercises and examples that the textbook provided and the exercises the other course materials offered. In addition, the lecture slides, workshops and lab manuals were used to understand the covered and taught topics. The experiment presumed students to know concepts mentioned in lecture slides, workshops, or lab manuals. The initial set of 33 programming tasks was extracted from course materials. To ensure that online materials could support task completion, a preliminary search of Stack Overflow using task keywords ensured that answers were readily available. Stack Overflow website used because prior research indicated that Stack Overflow was a dominant resource for students seeking help on the Web [Acar et al., 2016, Brandt et al., 2009]. Tasks with fewer than 20 Stack Overflow

[1]Application reference: 2020-10022-16293.

answers were removed from the candidate pool, leaving 14 tasks for further consideration.

Additionally, the tasks were expected to be of varying levels of difficulty, to enable diverse behaviours when coding. Thus, the remaining 14 tasks were classified into three difficulty levels. Tasks that used only concepts/topics explicitly taught during the course unit lectures/exercises were classified as ‘easy’ (n=5); those that used a combination of explicitly taught and additional material (e.g. from the textbook) were classified as ‘medium’ (n=5); those that used contents not covered by the course but mentioned in the corresponding textbook and required substantial additional material were classified as ‘difficult’ (n=4). From the classified tasks, four tasks were selected with a variety of difficulty levels: two easy, one medium and one difficult. Table 5.1 provides a detailed description of each task, the examples of the code snippets available on Stack Overflow appear in Figures 5.2, 5.3, 5.4 and 5.5, and examples of the keywords used for searching on Stack Overflow appear in the Table 5.2.

5.1.2 Study instrument

The study used Zoom video conference software to conduct the experiment, record participants’ videos, collect their source code and interview them. The software provided a tool to record participants’ screens while coding and produced video files suitable for analysis. Using these settings assured that participants had a comfortable and easy way of coding without being aware of the recording. At the end of the coding, participants used the software to share their final source code. In addition, the instrument recorded the participants’ interviews.

5.1.3 Study procedure

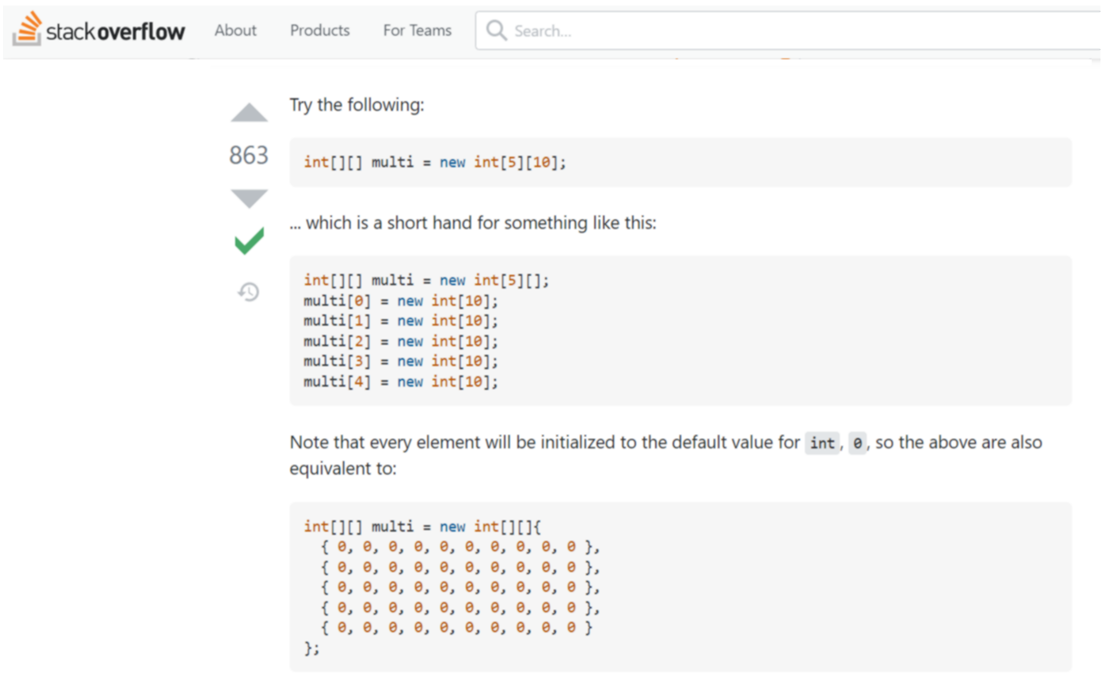
The study employed three phases using Zoom software.

5.1.3.1 Preliminaries

The researcher initiated a video call with each participant, and summarised the phases of the study. Once participants had a good understanding of the nature of their involvement, they provided a verbal consent. Then, participants received a document containing the four programming tasks.

Task no.	The task	Level	Task's topics
Task1	Write a program that contains a two-dimensional array with the following values: 10, 20, 30, 40, 50 and 60. Declare and initialise a two-dimensional array with three rows and two columns. Then, using nested loops, print the array contents one by one in the same order.	Easy	Java multi-dimensional arrays and nested loops were covered topics for participants.
Task2	Write and implement a program which converts a sum of money into a different currency. The user will enter the amount of money to be converted and the exchange rate. The program will contain separate methods for getting the sum of money from the user, getting the exchange rate from the user, calculating the conversion and displaying the result.	Easy	Java method definition and string input and output were covered topics for participants.
Task3	Write a program to enter and confirm a suitable code name for a spy agent. Declare a `String` variable and then get the user to enter a suitable name as a codename. Check that the codename meets the following requirements: 1. The codename is greater than 6 characters in length. 2. The codename starts with the word "Agent". 3. The codename ends with an "X" character. If one of the conditions above was not met, print "INVALID CODENAME" and ask the user to re-enter a code name.	Medium	Java `String` is covered, but its methods, such as `.startsWith` and `.endsWith`, were not explicitly taught in the first year Java course, but were covered in additional material.
Task4	Using Java threading feature, create three threads where each has a unique name. Then, each thread should print the numbers from 1 to 100 in sequence order. For example, thread A will print numbers 1 to 100, thread B will print numbers 1 to 100 and thread C will print numbers 1 to 100.	Difficult	Java threading topic is new for the participants as it was not explicitly taught in the first year Java course, but were covered in additional material.

Table 5.1: The study's tasks with their level and the topics covered from the course materials



Try the following:

```
int[][] multi = new int[5][10];
```

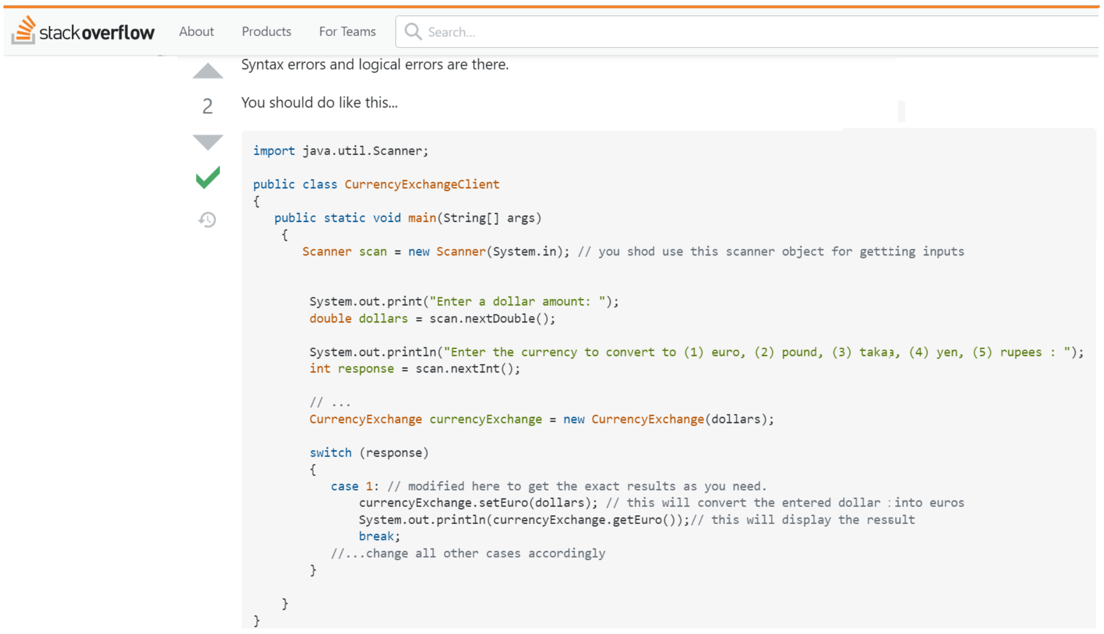
... which is a short hand for something like this:

```
int[][] multi = new int[5][];
multi[0] = new int[10];
multi[1] = new int[10];
multi[2] = new int[10];
multi[3] = new int[10];
multi[4] = new int[10];
```

Note that every element will be initialized to the default value for `int`, `0`, so the above are also equivalent to:

```
int[][] multi = new int[][]{
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};
```

Figure 5.2: An example of code snippets available on Stack Overflow for the first task presented on Table 5.1

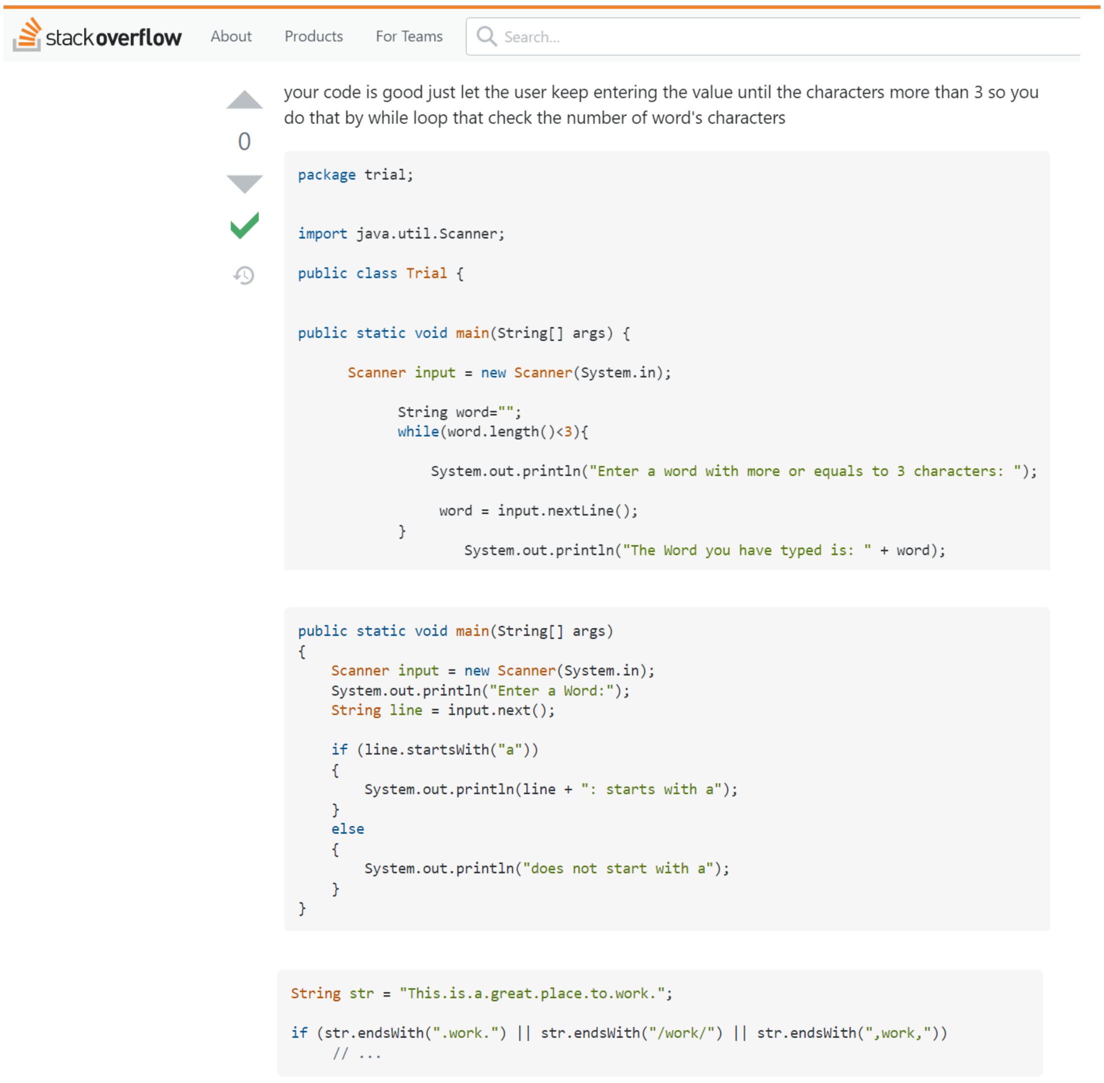


Syntax errors and logical errors are there.

You should do like this...

```
import java.util.Scanner;

public class CurrencyExchangeClient
{
   public static void main(String[] args)
    {
       Scanner scan = new Scanner(System.in); // you shod use this scanner object for gettting inputs


        System.out.print("Enter a dollar amount: ");
        double dollars = scan.nextDouble();

        System.out.println("Enter the currency to convert to (1) euro, (2) pound, (3) takaa, (4) yen, (5) rupees : ");
        int response = scan.nextInt();

        // ...
        CurrencyExchange currencyExchange = new CurrencyExchange(dollars);

        switch (response)
        {
           case 1: // modified here to get the exact results as you need.
               currencyExchange.setEuro(dollars); // this will convert the entered dollar :into euros
               System.out.println(currencyExchange.getEuro());// this will display the ressult
               break;
           //...change all other cases accordingly
        }

    }
}
```

Figure 5.3: An example of code snippets available on Stack Overflow for the second task presented on Table 5.1



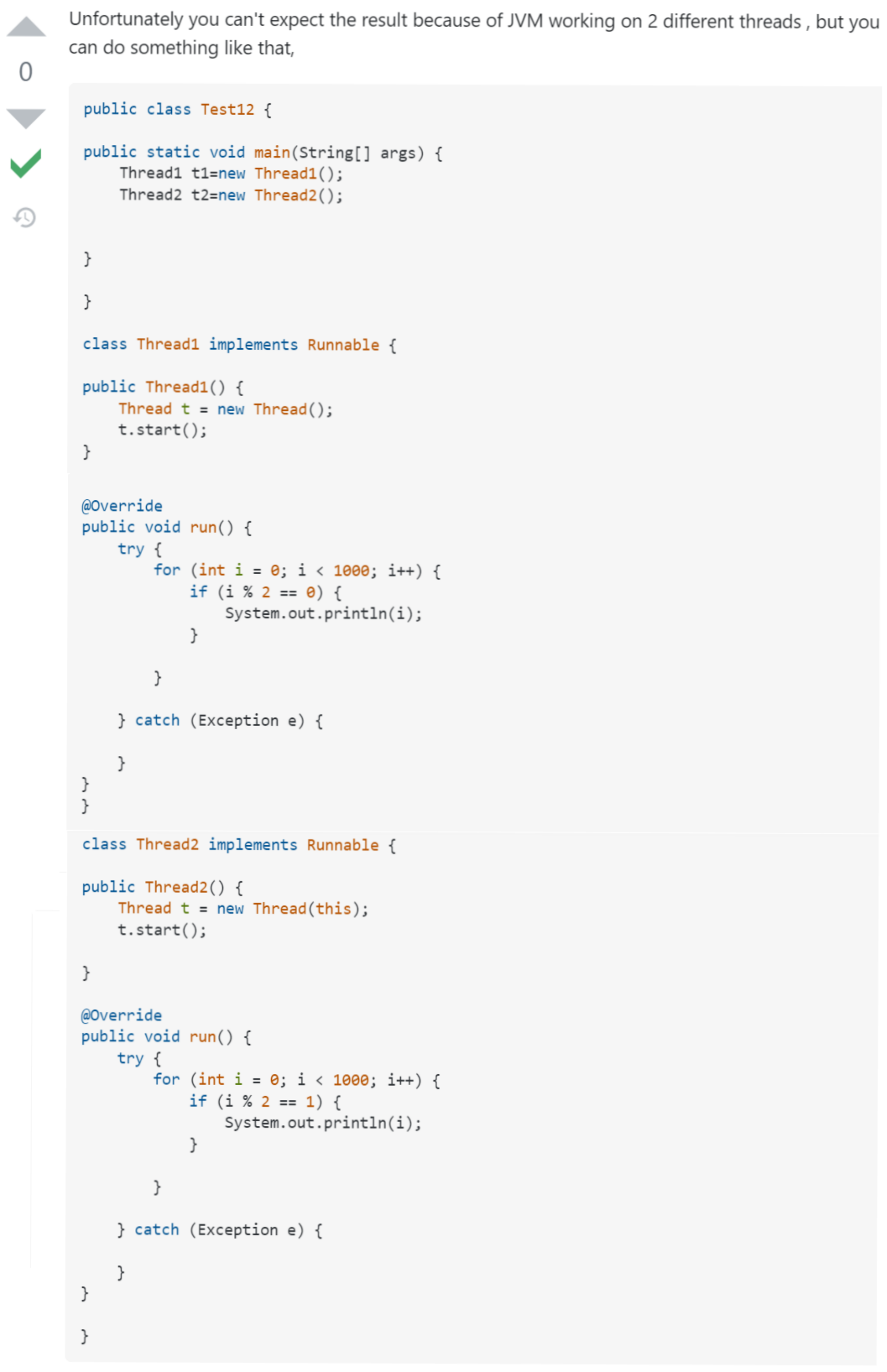
your code is good just let the user keep entering the value until the characters more than 3 so you do that by while loop that check the number of word's characters

```
package trial;


import java.util.Scanner;

public class Trial {


public static void main(String[] args) {

      Scanner input = new Scanner(System.in);

            String word="";
            while(word.length()<3){

                System.out.println("Enter a word with more or equals to 3 characters: ");

                 word = input.nextLine();
            }
                    System.out.println("The Word you have typed is: " + word);
```

```
public static void main(String[] args)
{
    Scanner input = new Scanner(System.in);
    System.out.println("Enter a Word:");
    String line = input.next();

    if (line.startsWith("a"))
    {
        System.out.println(line + ": starts with a");
    }
    else
    {
        System.out.println("does not start with a");
    }
}
```

```
String str = "This.is.a.great.place.to.work.";

if (str.endsWith(".work.") || str.endsWith("/work/") || str.endsWith(",work,"))
     // ...
```

Figure 5.4: An example of code snippets available on Stack Overflow for the third task presented on Table 5.1



Unfortunately you can't expect the result because of JVM working on 2 different threads , but you can do something like that,

```
public class Test12 {

public static void main(String[] args) {
    Thread1 t1=new Thread1();
    Thread2 t2=new Thread2();


}

}

class Thread1 implements Runnable {

public Thread1() {
    Thread t = new Thread();
    t.start();
}

@Override
public void run() {
    try {
        for (int i = 0; i < 1000; i++) {
            if (i % 2 == 0) {
                System.out.println(i);
            }

        }

    } catch (Exception e) {

    }
}
}
class Thread2 implements Runnable {

public Thread2() {
    Thread t = new Thread(this);
    t.start();

}

@Override
public void run() {
    try {
        for (int i = 0; i < 1000; i++) {
            if (i % 2 == 1) {
                System.out.println(i);
            }

        }

    } catch (Exception e) {

    }
}

}
```

Figure 5.5: An example of code snippets available on Stack Overflow for the fourth task presented on Table 5.1

The tasks	Keywords
First task	Two-dimensional array
Second task	Convert and exchange currency
Third task	Get user inputs, check inputs length and check the start and the end of the inputs.
Fourth task	Print numbers incrementally in Threads.

Table 5.2: Tasks keywords searched on Stack Overflow to check online availability of the tasks presented on Table 5.1.

5.1.3.2 Phase 1: Screen recordings

The researcher instructed participants that they should not overly concern themselves with the need to correctly solve all tasks, and that they could use any of their usual software, resources, or websites during task completion. Then, the researcher asked participants to share their screens and begin solving the tasks using their preferred IDE or editor. The researcher also notified them about the need to share the resulting source code at the end of the coding session. At this point, participants started to read and solve the tasks, and the researcher started screen-recording programmers while coding the tasks.

During coding sessions, the researcher remained connected to the Zoom meeting but muted the mic and turned off the video camera, to minimise distraction and provide participants with the sense of a normal setting. The researcher did not intervene or interrupt the coding session unless participants directly requested (e.g. to help resolve a technical issue or task ambiguity). The researcher did not provide any programming-related information that could help the participant solve tasks, even if the participant requested it.

Coding sessions ended with stopping participants after 50 minutes had elapsed, regardless of progress, or earlier if a participant indicated having completed all four tasks. Upon completion, the researcher stopped the screen sharing and recording and saved the recordings in approved local storage as an MP4 format file for subsequent analysis.

5.1.3.3 Phase 2: Source code collection

After completing the tasks, the researcher asked participants to share their final set of source code files for the four programming tasks. Participants received a Dropbox link through the Zoom chat to upload their source code for later analysis.

5.1.3.4 Phase 3: Interviewing participants

After completing the tasks, as a last step in the experiment, participants took part in an audio-recorded structured interview using Zoom software. The interview questions concerned four main aspects: participant demographics, use of the websites, use of online code and experience with the experiment. The demographic questions addressed participant age, gender and programming expertise/experience. The participants answered questions about their experiences completing the programming tasks and the degree to which their behaviour was representative of their usual programming activity. Participants also answered targeted questions about their use of websites during typical programming and the experiment's programming task phase, code cloning behaviours during the programming task phase and the relationship between Web-based information retrieval and existing knowledge (i.e. if they were looking things up to refresh their memory). Finally, the interview examined participants' experiences with the experiment, such as the tasks, time and capturing behaviours using video recordings (a complete list of the questions appears in Table 5.3). The audio-recorded interviews were transcribed for subsequent analysis.

5.1.4 Recruiting participants

The study recruited participants from *"COMP16412 Introduction to Programming 2"*: a second-year computer science course at the University of Manchester. The researcher contacted the course instructor to ask permission to recruit participants. Then, the researcher used two means of reaching prospective participants with study details, namely: sending emails to the students through the course instructor and posting an advert in the *"MondayMail"*: a weekly round-up (newsletter) for undergraduate University of Manchester students. The recruitment continued for two consecutive semesters during the 2020/2021 academic year: the first semester in 2020 and the second in 2021. The student interested in participating in the experiment emailed the researcher, who then emailed the participant with the participants' information sheet (available in the Appendix C.2) and the date and time available for the experiment. In total, 10 participants approached the researcher and took part in the study, and each received the reward of an Amazon voucher of £10 GBP gift certificate[2].

[2]Approximately $14 USD.

Section	**Questions**
Demographic and programming experience	Age Gender Level and number of years of experience. How do you self-estimate your programming experience? (Scale from 1 to 10, where Novice=0, Intermediate=5 and Professional=10). How many years did you spend programming? Which programming languages do you use for coding?
Web usage	What websites did you choose during the experiment and why? During your normal programming activity, did you prefer to use Stack Overflow? What is your specific approach to using the websites? Did you find the websites easy to use, helpful and correct? Did you feel frustrated during the use of the websites? When searching the websites, were you confident that you got the right answer? Have you searched online for problems you have previously encountered?
Code reuse	Did you copy the online code? Were you confident that you got the correct answers online? Could you possibly copy an online code that you had previously reused?
Memory	When you did not remember information, did Stack Overflow helps you remember information you already knew?
Experiment	Did you find the tasks difficult? Did you feel the study restrictions and the time crunch affected your performance? To what extent do the behaviours you used today reflect those you would typically use?

Table 5.3: The interview questions

ID	Gender	Age	Experience (Out of 10 where novice=0, intermediate=5, and professional=10)	Years of programming	Programming languages
P1	Male	20	5	Not available	Python, Java, C#, JavaScript and R
P2	Male	20	7	One semester	Python, Java, C, C#
P3	Female	19	5	One year	Python and Java
P4	Female	19	5	5 years	4 programming languages
P5	Male	20	5	8 years	Python, Java, JavaScript, typescript, C#, .NET, C, C++ and Swift.
P6	Female	20	5	6 years	Java, Python, C, C++, Web development
P7	Female	19	3	One year	Python
P8	Male	21	5	4 years	5 programming languages
P9	Male	20	5	3 years	Python, Java, C++ and pascal
P10	Female	22	5	5 years	Java along with 9 programming languages

Table 5.4: Participants' demographic and programming experience

Based on the demographic information in Table 5.4, five males and five females, between 19 and 22 years old, were recruited for the experiment. Participants rated their programming experience level as intermediate with eight reporting an average of 4 year of programming. Participants reported the programming languages or the number of languages in which they had expertise, with Java and Python recurring most.

5.1.5 Analysis

The first step in approaching the data was to analyse the interview data (phase 3), followed by video recordings (phase 1) and the source code (phase 2) (the interviews' transcripts, the video recordings and the participants' source code are available as supplementary material [3])

5.1.5.1 Interview analysis

While the interview was the last step the participants undertook in the experiment, it was the first data analysed. This was because the interview data provided more

[3] https://doi.org/10.48420/c.6366063.v2

Analysis phase	**Process description**
Familiarisation with the data	1. Transcribe the data. 2. Ensure familiarisation by listening the audio files, reading the excerpts multiple times and recording initial codes.
Generate an initial set of codes	1. Allocate each data to the relevant code using systematic way. 2. Inductive: data-driven approach without pre-knowledge of the data.
Searching for themes	1. Themes is a pattern that accrued constantly over the results. 2. Group similar codes into potential related themes.
Reviewing themes	1. Ensure the theme is consistent with the collected extracts. 2. Ensure the theme is consistent with the entire data.
Defining and naming themes	1. An iterative process of refine each theme and the story behind it. 2. Clear definition and name for each theme.
Producing the report	1. Choose vivid and compelling extracts then analyse them. 2. Check the final analysis with the research questions.

Table 5.5: The followed steps in analysing the interview data based on the reflexive thematic analysis approach [Braun & Clarke, 2006].

context to the participants' behaviours and a framework to assist in analysing the video recordings and source code. The analysis followed six steps based on the reflexive thematic analysis mentioned in [Braun & Clarke, 2006], as illustrated in Table 5.5. Before coding, the researcher read the transcriptions of the recorded interviews ($n = 10$) multiple times, to ensure familiarity. Then, the coding process used the qualitative data-analysis software *NVivo 12*, taking an inductive approach based on participants' data, producing multiple codes. This process was reviewed to ensure code names and contents were represented. Before moving to the next step, similar codes were grouped to help find the themes. Looking for common reoccurring patterns across the data produced many themes. Each theme's contents and names were reviewed against the purpose of the study, and the final themes were used to create the report. Three themes, along with multiple sub-themes, were reported.

To ensure the reliability of the coding, 30% of the data (i.e. three interview transcripts from three participants) were additionally coded by another researcher with a near-perfect agreement (Cohen's $k = 0.83$ and % of agreement=96%).

5.1.5.2 Video analysis:

The study collected videos of participants solving the tasks. Videos (screen recordings) were qualitatively analysed [Seaman, 1999] to observe and quantify participants' engagement with websites when solving the study tasks. In preparation for the analysis, the saved recorded videos were linked to the qualitative data analysis software *NVivo 12*, given the large size of the video files. Reflexive thematic analysis was used to analyse the videos in six steps (see Table 5.6). Coding the videos differs from the text because it was based on observing participants on screen. In the first step of the reflexive thematic analysis, the researcher watched the videos multiple times for content familiarisation. Then, the coding phase started with watching and coding the related behaviours, using a deductive approach. In contrast to the interviews, this approach was analyst-driven, using previously discovered findings (the interview findings from the previous chapter (Chapter 3) along with the interview from this study) as a framework for coding to find the relevant and interesting behaviours [Bakeman & Quera, 2012]. The choice of such an approach ensured behavioural coding of the videos that focused on the study's research questions and objectives, and helped the researcher to know what to expect when coding the videos. However, any interesting and related behaviours outside this framework were also captured to ensure unbiased data. Coding the videos was an iterative process involving many rounds, despite requiring more time and effort.

In more detail, the initial coding round aimed to observe behaviours based on the specified framework. The following coding round ensured that any interesting related behaviours outside the specified framework were coded. The third round was to extract behaviours related to the source code for later analysis. After this round, two criteria were established to ensure that the captured behaviours were meaningful and relevant. The criteria were that the behaviours had to be observed on the videos to constitute clear and reliable evidence and the behaviours should be measurable, in the sense that the videos provide enough information to support their serving as relevant evidence. The fourth round considered the mentioned criteria, checked the extracted behaviours and extracted any missing relevant behaviours. These multiple rounds ensured greater familiarity with the data and filtered behaviours down to relevant activities with multiple observed instances.

The next phase of the reflexive thematic analysis was grouping similar codes that form data patterns and share one overarching concept for the themes. For instance, codes that captured participants' behaviour toward online code adoption were grouped.

Analysis phase	**Process description**
Familiarisation with the data	1. Ensure familiarisation by watching the video, then recording initial codes.
Generate an initial set of codes	1. Systematically allocate each observed behaviour to the relevant code. 2. Deductive: analyst-driven approach based on previous findings.
Searching for themes	1. Themes is a pattern that accrued constantly over the results. 2. Group similar codes into potential related themes.
Reviewing themes	1. Ensure the theme is consistent with the observed behaviours. 2. Ensure the theme is consistent with the entire data.
Defining and naming themes	1. An iterative process of refine each theme and the story behind it. 2. Clear definition and name for each theme.
Producing the report	1. Provide vivid and compelling descriptions of the observed behaviours then analyse them. 2. Check the final analysis with the research questions.

Table 5.6: The followed steps in analysing the videos files based on the reflexive thematic analysis approach [Braun & Clarke, 2006].

The final set of themes was reviewed to ensure that no theme constituted a sub-theme to another theme or was not a theme in itself. Then, a thematic map was drawn to illustrate the final themes. Finally, a report on these behaviours was written.

A total of 33 videos (from a possible 40) were analysed. Five participants did not attempt Task 4, so no videos were captured for these participant-task pairings. A further two missing videos (P10 Tasks 1 and P10 Task 2) were attributed to a failure of the Zoom recording facility (see Table 5.7 for the collected videos).

Ultimately, the study performed inter-rater reliability with an independent researcher to ensure more data reliability. A 30% of videos (i.e. ten videos) were additionally coded by another researcher with almost perfect agreement (Cohen's $k = 0.87$ and % of agreement= 96%).

	Tasks			
Participant	T1	T2	T3	T4
P1	✓	✓	✓	✗
P2	✓	✓	✓	✗
P3	✓	✓	✓	✗
P4	✓	✓	✓	✗
P5	✓	✓	✓	✓
P6	✓	✓	✓	✓
P7	✓	✓	✓	✓
P8	✓	✓	✓	✓
P9	✓	✓	✓	✗
P10	✗	✗	✓	✓

Table 5.7: The collected videos files for each participants

5.1.5.3 Source code analysis:

The source code participants produced was analysed for the 33 tasks for which the video was captured (the collected source code can be found in Table 5.8). After finishing the coding sessions, participants shared these source code and were analysed closely to examine each code's status. The study used a systematic way to manually analysed each participant's source code and established a mixture of quantitative and qualitative measures:

1. **Quantitative: Compilation/Execution** - Does the source code compile and run using either an IDE or Command line in Windows?
2. **Quantitative: Correctness** - Does code execution produce the expected output?
3. **Quantitative: Online Clones** - The study analysed the instances of copying online code. This include the number of lines of code copied from online to the source code (without commented or removed code).
4. **Quantitative: Online Sources** - The URLs from which cloned code was sourced.
5. **Qualitative: Clone Purpose** - The perceived motivation for including cloned lines.

To ensure the reliability of the coding, 30% of the data (i.e. source code from three participants) were additionally coded by another researcher with near-perfect agreement (Cohen's $k = 0.83$ and % agreement= 93%).

	SRC			
Participant	T1	T2	T3	T4
P1	✓	✓	✓	✗
P2	✓	✓	✓	✗
P3	✓	✓	✓	✗
P4	✓	✓	✓	✗
P5	✓	✓	✓	✓
P6	✓	✓	✓	✓
P7	✓	✓	✓	✓
P8	✓	✓	✓	✓
P9	✓	✓	✓	✗
P10	✗	✗	✓	✓

Table 5.8: The collected source code (SRC) files for each participants

5.2 Results

The results of this chapter are presented based on the order of the analysis. The first part is the participants' interviews, followed by video behavioural coding and participants' written source code.

5.2.1 The interview

The thematic analysis produced three themes: reasons for the websites' usage, experiences using websites, and study specifications and issues.

5.2.1.1 Theme 1: Websites usage reasons

Participants used websites for various purposes during their coding; this theme will discuss these purposes in detail.

Websites as a reminder

Nearly all participants reported that the websites, such as Stack Overflow, could serve as a reminder for information even if it were already known.

> *I do not try to remember the code if I know where it is exactly. [...] I am just reminding myself from the sources. (P3)*

Such reminding process comprised confirming the syntax.

> *[...] it is often the same thing looking up just to check if you remember it correctly. (P6)*

Three participants thought remembering programming information was difficult because coding uses multiple programming languages.

> *I usually use the website to refresh my memory because I just do not remember all of the intricacies of every single language because each of them have slightly different syntax [...]. (P2)*

Other reason that affect remembrance was the recency of coding.

> *I have forgotten and not used it like for few months. (P9)*

Copy online code

All participants reported copying and pasting online code, even the previously encountered ones, due to the easy-to-find online code.

> *[Q: What about copying code that you have previously copied during websites search?]*
>
> *Yeah, I would copy it again; some code I know where I can find it, so whenever I cannot remember it by heart, I know exactly where to find and then just copied it and then it works. (P8)*

> *I look up the same thing every single time I program. (P4)*

Most participants felt that the websites urged them to adopt available online content without checking, where the ease of obtaining online code gave a sense of no need to understand the code.

> *[...] I went ahead and copied the first things I saw [...] I think this is very common from write code pull up a page get something from the internet and continue coding. (P8)*

> *[Q: Do you spare the effort of memorise the information because it is available online?]*
>
> *Yeah exactly, I know where to find it, so I do not want to learn it because it so easy to just pull up copy. (P8)*

The placement of syntax on the tutorial websites provided easy syntax extraction, as noted by three participants.

> *[...] if I am looking for syntax, I usually go to to those websites I mentioned before, and I am visually looking for these squares in different background, so I just filter what I see. If I see the bluish square with some code on it, I just look into it and usually I got what I need. (P10)*

Searching for information

Participants described the procedures they took when searching the website for programming material. At the start, they get familiarised with the requirements to help determine the following step.

> *I think my approach would usually be the same, read though the questions a few times and give it a go and see where I feel it does struggling and then start seeing if there anyone else done it in a smart way or whatever. (P9)*

Then, most participants started searching using the Google search engine without predetermining preference. Few strategies were used to benefit from the search results, such as structuring the search query in a more appealing way related to the required information.

> *I did not use a particular website, just try to figure most concise way to word my question and just see what comes up. (P9)*

When the results appeared, four participants followed three approaches to select the appropriate results. The first one was selecting of the most suitable websites that deliver the required answers.

> *[...] I just used the search and choose the one which seemed the most appropriate. (P2)*

> *[...] I chose whatever looks like it was answering my question, and most of it was fine. (P8)*

The second one was choosing of random websites at the top of Google search.

> *I used random websites as whatever came on the top of a Google search that the first thing I checked out. (P5)*

The last approach was looking for easier answers.

> *I just type my question and try to see from the description what answers are easier than go there. (P7)*

5.2.1.2 Theme 2: Websites experience

This theme tackles programmers' websites' preferences and potential concerns.

Websites choices

Half of the participants reported that tutorial websites provide clear, simple, understandable, self-sufficient and easy to locate examples that offset the need to read the corresponding contents.

> *The one I usually chose is "GeeksforGeeks" or "W3school". These two because it explains the best what I need to find, and the answers are on the beginning, so I do not need to scroll and read through everything. (P10)*

> *I think I used "GeeksforGeeks" at one point that usually quite useful because loads people post their like proper tutorials. (P5)*

> *"TutorialPoint" and "W3school" I generally use it a lot it because it straightforward and they provide pretty clear examples and do not rely too much on jargons and it is pretty easy to understand what you are looking for. (P6)*

More than half of the participants mentioned the use of Stack Overflow because it provides reliable and genuine examples aiding problem-solving.

> *[...] I use Stack Overflow for for this one specific problem I was having because those questions are more specific. (P7)*

> *I chose Stack Overflow because usually it is reliable. (P1)*

The usage of the Stack Overflow is approached with care, as P5 noted:

> *And there is Stack Overflow, but you need take that in with some caution sometime. (P5)*

Other than websites' features, one participant mentioned that the more experienced programmers' suggestions determined her websites' selection.

> *[Q: During the experiment, why did you choose such websites?]*

> *Because it was recommended by the Uni stuff, and older generation. (P3)*

Issues with the websites

Many participants reported issues when using the websites, as some online answers were functional, but they were not necessarily great.

> *[Q: When searching the websites, were you confident that you got the right answer?]*
>
> *Getting answers that works, but it is not always the best answers I would say. (P4)*

Online answers additionally contributed to poor outcomes unsuitable for participants' intentions.

> *[Q: Did you find the websites easy to use, helpful and correct?]*
>
> *One give me a misleading data, so I would say no. (P8)*

Thus, participants reported the possibility of facing problematic code, such as outdated and incomplete code that misses important information.

> *Sometimes the code on Stack Overflow can be a bit outdated, new version of framework or languages. (P5)*

> *The things it annoys me is website not showing the entire piece of code, so the thing that missing is something outside of the immediate bit of code they are showing. Like import statements, if the import statement is wrong and I cannot work out what it is, they very rarely show that I guess. (P4)*

Participants were uncertain about their ability to integrate online code due to the associated problems.

> *Some problems are sometimes you are not working with your own code, so you do not know everything about it, you do not know how the code interact with the code you find online. (P1)*

Another impact was coding with unfamiliar programming aspects, where the websites did not help resolve such uncertainty.

> *[...] for the threading, I am not too familiar with threading, so I am not sure if it is what I wanted. (P8)*

5.2.1.3 Theme 3: Study Specifications and Issues

This theme addresses participants' thoughts and concerns regarding aspects of the experiment.

Video recording
More than half of the participants stated that their behaviours during the experiment followed their normal programming behaviours.

> *[Q: To what extends does the behaviours that you used today reflect the behaviours that you would normally use?]*
>
> *I think pretty accurately, depending on how often I have been coding, recently I probably forgot fewer very simple things, but I guess, yeah I think so. (P4)*

The online settings of the experiment is preferable than the lab-based because it relieves pressure from such experiment.

> *[Q: Do you feel me recording this experiment and being present affected your behaviour performance?]*
>
> *Maybe if it was face-to-face, but since it virtual then no. (P3)*

Nevertheless, some behaviours were either not captured or did not reflect the programmers' normal behaviours. Two participants expressed that recording could conceal some behaviour or introduce unwanted factors to their programming activities.

> *[...] I would probably open up my notes and some written notes which obviously that I cannot show on the screen. It is just a little pressure, it does not mean I will do something much different, it feels different, I am less relaxed. (P2)*

> *[Q: To what extends does the behaviours that you used today reflect the behaviours that you would normally use?]*
>
> *It sorts of similar but in smaller scale. Usually I have got two monitors set up. So one monitors open with like problems descriptions and the websites that help me out, and write code on the other one so just happen in parallel.*

Time constraints

Half of the participants felt that time constraints affected their performance in the experiment.

> *So today I try do it relatively fast because of the time thing. Normally I will take it a bit slower and maybe read more things. (P8)*

The other half of the participants indicated no stress from the restricted time, and using some approaches would assist regulating the time.

> *What I do when I started problems, I know it is time constrain I sort try to get the first few really quick or the one I spot easier, so I try manage the time better, and I know I have done it so time constrain does not stress me. (P5)*

Task Difficulty

Participants reported that the tasks were not difficult but required more searching.

> *[...] I Google a lot more usually than I did today because the tasks was not challenging. (P5)*

In contrast, four participants thought the fourth task could be challenging because they had no previous information, which may lead to not approaching the task.

> *I have not got the time to read it [fourth task], but I am not familiar with threading, so that why I left it. (P3)*

Two participants thought the problems were not from the task but from forgetting the information and time.

> *My Java was a bit rusty, slight, I did not remember a lot of things how to do in Java in particular. But no, I think if I have a bit more time I will finish them all. (P2)*

5.2.2 Behavioural coding

The reflexive thematic analysis revealed two main themes representing participants' strategies. The first theme is acquiring knowledge that encompasses the online searching process from the start until the end of the programming session. The second theme is utilising knowledge to write the code that comprises how participants benefit from the search results by reusing the code. Next is an overview of the resources and activities, followed by the themes.

5.2.2.1 Overview of the resources and compiling activities

Participants used Web-based and non Web-based resources to search and reuse code, and compiled the code to observe the outcomes. All the participants searched the websites at various stages and processed the results. Most participants started their searches using the Google search engine and chose the first presented search results regardless of the website's name or the attached contents. In addition, most participants chose the tutorial websites to retrieve syntax, including "GeeksforGeeks", "W3school", "JavaTpoint", "Java671", "BeginnerBook" and "TutorialPoint". The choice of tutorial websites changed based on the tasks. Other than tutorial websites, two websites were used rarely by participants, including documentation for syntax, Stack Overflow for resolving errors and blogs for understanding.

Participants used non Web-based resources during coding. They consulted the document containing the four programming tasks during coding. In the first task, eight participants visited the document (P1, P2, P3, P4, P5, P6, P7, P8). Most visits were commenced once, except for three participants as P3 three times, P7 four times and P8 eight times. Similarly, all participants in the second task, except P4 and P10, visited the requirements document, and their visits were low, unlike P3 and P6 who visited the document six and ten times respectively. In the third task, nearly all the participants visited the requirements document, but not P5 and P9. As compared to the other tasks, the third task possessed the most visits. P2, P4, and P6 visited the requirements document six, twice, and five times, and P1, P3, P7, P8, and P10 visited fourteen, fifteen, nine, thirteen and seven times respectively. Lastly, the requirements document visits in the fourth task were not high as P5, P6, P7, P8 and P10 visited three, five, four, four and twice respectively. Other non Web-based resources were course-related materials, previous code and IDE. P1 used coursework resources to copy the `Console` syntax in the second task, and P4 used previous code to exploit syntax while coding the tasks like class declaring, "Main" method, `.println` statement and `Console` syntax. The used IDE provided syntax suggestions in the form of auto-complete and used by P2 and P7 to get the "Main" method in the first and third tasks.

Besides using the resources, participants edited and compiled their code, going through a cycle of observing and fixing errors post-compilation. All participants in the first task had to compile their code and change it according to the outcomes, and P1 and P2 faced the most errors. In the second task, all participants, except P5 and P9 who did not face any errors, compiled their code and commenced fixing the produced errors, and P2, P4 and P6 faced the most errors. In the third task, all participants, except

P9, compiled their code, and P1, P3, and P8 faced a high number of run-time, syntax and logic errors and spending considerable time resolving these errors. Although participants in the third task made numerous attempts to produce executable code, more than half of the participants produced non-executable code. In the fourth task, P7 faced logic errors that caused time to resolve, while others faced fewer errors, like P6 and P8, or no errors, such as P5 and P10.

5.2.2.2 Theme 1: Acquiring knowledge

Participants searched the Web for syntax, to clarify understanding and to fix errors. Searches yielded useful knowledge, but also caused problems.

Looking for syntax: The recordings of all the participants showed that they searched the websites for syntax either before or while coding. Before writing the code, P6, P7, P8 and P10 in the fourth task and P3 and P8 in the first task searched for specific terms such as threading, *"two-dimensional array"* and the "Main" method. Search for syntax during coding was noticed by all participants who divided the tasks into meaningful searchable pieces and searched for syntax in all the tasks. Specifically, all participants in the first task, except P10, searched for *"two-dimensional array"* and basic syntax like `For` loop, "Main" method and array. In the second task, all participants, except P2 and P10, searched regarding converting data types and getting user inputs such as `Scanner` and `Console`. P4 and P9 searched for basic syntax, including Java function structure and `.println`, similar to the previous task. All participants in the third task searched syntax to retrieve `String` methods like `.length`, `.startsWith`, `.endsWith`, `.charAt` and `.substring`. The basic syntax was again followed by P9 for `While` loop and P6 for `AND` symbol, in accordance with previous tasks. In the fourth task, the searching activities were superficial toward understanding the threading syntax. Four participants (P5, P6, P8, P10) searched for threading topics in general, creating threads, assigning thread names and importing threads. P8 used the whole question text to search at the task's start.

Increasing understanding: Participants sought programming information related to the task along with searching for syntax. P1 and P4 on the second task and P7 on the third task searched for syntax clarifications, such as the use of the `.equals` method for character type, the difference between `Float` and `Double` and the use of currency in Java. P7 and P10 also searched to gather information about the threading feature in the fourth task.

Looking to fix errors In order to resolve errors encountered during coding and compilation, participants resorted to the Web for assistance. During coding, P1 faced errors regarding non-correct syntax in the first and third tasks, such as "Main" method, *"two-dimensional array"*, `.startsWith` and `.equals`. P9, in the second task, also encountered an error regarding the function structure, and searched for a Java function example. Regarding errors after code compiling, P2 and P7 faced errors in the first task caused by incorrect declaring of *"two-dimensional array"* and *"foreach"* loop. Five participants (P1, P2, P4, P6, P7) in the second task faced errors, and P4 encountered the most. The errors were due to converting data types, missing elements such as using static in the method, issues with user inputs and misplacing data types caused by online suggestions. The third task has a few `String` errors from P4 on `.length` and P2 on declaring variables. Lastly, P7 and P8 in the fourth task searched online to resolve errors on exceptions, confusion between threads and threads and thread names.

Experiencing problems during acquiring knowledge: It can be observed that some online searches were unhelpful, causing additional time and effort. Eight participants struggled with online search across coding stages in all the tasks, and half of them returned to the previously visited Web pages. At the first task, P2 accessed various websites and searched for the nested *"foreach"* loop, resulting in more search time and attempts, then eventually changing the chosen method. In another instance, P2 searched for *"two-dimensional array"*, and the outcomes did not help succeed in printing array elements accurately but motivated further searches that also did not help fix the issue. Similarly, P7 searched repeatedly for a *"two-dimensional array"* before writing it correctly. P9 searched for the "Main" method and the `.println` statements without beneficial outcomes and resolved without searching. While coding the second task, P4 struggled the most as she searched five times for `Float` and `Double`, truncate `Float`, Decimal format, and currency without valuable outcome; she sometimes returned to the previously searched results and chose another website. P6 and P7 searched for `Scanner` syntax, but their searches were incomplete and caused further searches, and P4 and P9 conducted fruitless searches when converting data types from `String` to `Float`.

Online searching issues continued in the third task. Five participants faced issues related to `String`. P2 was affected the most as he searched for the *"GetChar"* method in Java but adopted `String` declaring instead, then continued searching for the equivalent of *"GetChar"* from C++ in Java and accessed three websites without reaching an answer. The previously adopted `String` declaring resulted in issues when compiling, causing two further search attempts, which was finally resolved by adopting a new code from the Web; P10 shared a similar struggle while searching for the *"GetChar"* method. In addition, search instances conducted by P3, P9 and P10 caused further search attempts searching for `String` methods. P8 searched for unnecessary syntax like *"regex"* and *"Matcher"*. Lastly, three participants (P7, P8, P10) in the fourth task searched for how to start one thread after another and count threads, then faced unproductive content, causing them to reformulate their query and repeat the search.

5.2.2.3 Theme 2: Utilising knowledge to write the code:

Through seeking online knowledge, participants utilised the knowledge to develop their code. The following sub-themes will list participants' observed activities when copying online code.

Exploiting search results to copy code: All the participants copied online code snippets using two ways: a clipboard as a normal way and a visual way by looking at online code while writing it. There also appeared to be a further copy instance where participants examined particular code snippets from the websites and then wrote it in their code afterwards. This copy instance will be labelled as a mental way. These three copy instances were conducted throughout all the tasks with variance. In the first task, seven participants copied online code: normally (P1, P3, P8), mentally (P4, P7) and visually (P2, P9). The most copied syntax were *"two-dimensional array"*, "Main" method, array assigning, length of the array and getting array elements. Similarly, all participants, except P10, copied online code when solving the second task: normally (P1, P3, P4, P7, P8), mentally (P4, P6, P7) and visually (P2, P5, P9). Participants copied syntax like declaring and importing `Scanner`, getting user entry input, declaring a variable and converting syntax such as `String` to `Float` or `Integer` or the other way around. The way of copying changed in the third task, where all participants, except P5 and P10, copied online code: normally (P1, P3, P8), mentally (P1, P4, P6, P7, P8) and visually (P2, P3, P9). The most copied syntax in the third task were `.startsWith`, `String .length`, `.endsWith`, getting user input, `.charAt`, `.substring`, *"Matcher"*, declaring `String` and import and declare `Scanner`. At last, all five participants (P5, P6, P7, P8, P10) who solved the fourth task exhibited copy-and-paste actions using the normal way. The copied code includes a class header, printing outputs and declaring the variables. Participants within all the tasks copied basic syntax from online, such as "Main" method, `For` loop and `.println` statement.

Developing code knowledge: Syntax was used across and within the same task repeatedly. Participants became familiar with most of these syntax they searched online and used previously, which reduced the need to repeat the search for the syntax. In the first task, P1 and P2 copied `For` loop and `.println` statement within the same task. In the second task, there were instances of copying from within the same task and from the first task. All participants exhibited copying code within the task, such as `Scanner` and `Console` methods, `.println` statement, `While` loop and converting data types. The "Main" method and class declaration were retrieved from the first task by P1 and P6. In a similar way, P1 and P6 in the third task copied "Main" method and class declaration from the first task. Eight participants (P1, P3, P4, P6, P7, P8, P9, P10) returned to the second task to copy either `Console` or `Scanner` syntax. P1 and P7 required extra elements from the second task, such as converting `String` to `Integer`, class declaring, method declaring, `.println` statement and "Main" method. Syntax was also copied within the third task, including `.println` statement, `Scanner` and `Console` getting user input, `String` length, declare `String` variable and converting `String` to an `Integer`. In the fourth task, P6 copied the "Main" method and class declaration from the first task, but P7 and P10 copied the "Main" method from the third task. The thread declaring and `Run()` method were copied within the task.

Implementing complicated approaches: Participants followed suggestions imposed by either websites or their code practices that caused more coding time and effort. Videos observed these approaches but did not necessarily appear in the final source code. Five participants (P2, P3, P4, P5, P9) in the first task followed online complex suggestions. In particular, P2 copied an online code that printed the array's contents using the `toString` method, which caused printing memory places, not the array contents. Three participants (P3, P4, P5) used online suggestions to manually assign values to the array without using a loop, and P5 placed the array contents using three variables, each holding two array elements. P2 created four loops instead of two based on the online suggestions. A similar observation is valid for P1 but not from the online suggestions. In the second task, four participants (P3, P6, P8, P9) copied the online code for user entry that used a `String` data type, which then converted it to `Integer` data type. P5 and P1 followed the same path but without online suggestions. Six participants (P2, P3, P4, P5, P8, P10) followed their code practices in the third task. In particular, P2 and P3 faced issues declaring multiple `String` variables and P4 and P10 made multiple unnecessary `if` statements that could be eliminated. P8 chose *"regex"* and *"Matcher"*, which resulted in several searches, debugging and more time. As most of the code was copied online in the fourth task, the online suggestions were problematic only for P7, who used three `Run()` methods with different names and loops and then added an extra *"InterruptedException"* exception causing more searches to get it correct.

5.2.3 Source code

Participants' source code were analysed and linked with the videos observations. The Table 5.9 shows that nearly all participants provided source code files compiled and executed successfully (28/33 files, 85%), and just over half (19/33, 58%) were considered to be correct (i.e. they met the requirements described in the task). Across the four tasks, participants produced at least two executables (mean: 2.80, median 2.50) and one correct source file (mean: 1.88, median 2.00). Progressively fewer source code files were correct as the tasks progressed (max: 9, min: 2). All but one of the supplied source files (P5 Task 3) contained cloned code, with each participant including an average of 2.3 (mean, P1) to 8.50 (mean, P6) cloned lines per file (medians ranged from 1.5, P5, to 6.0, P4 & P10) (online copy and paste activities are presents in Appendix C.1).

5.2.3.1 Correct:

Nineteen source code files compiled, ran and met the task requirements. Nine participants produced correct code in the first task, five participants in the second task (P2, P4, P5, P7, P9), three participants in the third task (P6, P8, P10) and two on the fourth task (P7, P8). There were instances of unnecessary syntax retrieved from the Web on the first and fourth tasks, which increased the complexity of the correct source code. P2, in the first task, used four loop statements instead of two to solve the *"two-dimensional array"* based on online suggestions. P5 copied online code and identified the *"two-dimensional array"* using three array locations, and P3 printed the locations of the array along with the contents. In the fourth task, P7 followed the online suggestions and included multiple `Run()` methods for each thread, `For` loop and `.println`

	Not provided	Non-executable	Incorrect	Correct	Online Clones
Task 1	1: P10	0	0	9: P1-P9	9/9 files: 1–5 lines/file $\bar{x}$: 2.78, $\tilde{x}$: 3.0
Task 2	1: P10	0	4: P1, P3, P6, P8	5: P2, P4, P5, P7, P9	9/9 files: 2–7 lines/file $\bar{x}$: 4.67, $\tilde{x}$: 5.0
Task 3	0	5: P2-P5, P9	2: P1, P7	3: P6, P8, P10	9/10 files: 2–6 lines/file $\bar{x}$: 3.80 (4.22), $\tilde{x}$: 4.0 (4.0)
Task 4	5: P1-P4, P9	0	3: P5, P6, P10	2: P7, P8	5/5 files: 6–24 lines/file $\bar{x}$: 12.00, $\tilde{x}$: 10.0

Table 5.9: Results of source code analysis. In the first four columns, the overall number of source files in each category is followed by the list of participants whose source fell into each category. In the final column, the number of files and lines containing cloned source are given, together with the mean ($\bar{x}$) and median ($\tilde{x}$) number of cloned lines per file (values in brackets for task 3 indicate averages for files containing cloned code).

statements.

Each of the correct source code were mapped with video recordings to understand the activities participants used to produce the correct code. Regarding searching activities, syntax searches were apparent throughout coding sessions for all participants, where they located items related to the task. For example, *"two-dimensional array"* was copied in the first task, converting methods in the second task, string methods in the third task and creating threads using full code snippets for the fourth task. Other than syntax, a few searching occasions were attributed to clarifications and resolving errors. In addition, the online search resulted in multiple complicated issues faced by more than half of the participants who produced the correct source code, which caused non-beneficial results and motivated further search attempts. Furthermore, all participants referenced the requirements document to check the requirements and exhibited medium referencing in the first task, low in the second task and high in the third task.

In terms of copying online code activities, nearly all participants who produced correct source code copied online code in varied ways, including normal, mental and visual, but the normal way is overwhelming. On the contrary, the copy from non Web-based resources is a rare strategy done by one participant. The copied code were reused again from either previous or current tasks by half of the participants. In addition, participants in twelve correct code instances ran into complex issues, half of them from their choice and the other half from the online suggestions. At last, compiling the code and fixing the resulting errors caused six instances of high time spent fixing the errors.

The instances of online copied code were analysed for each correct code with the help of participants' video recordings (details of copy and paste activities can be found Appendix C.1). The average online copied code was calculated using the number of code copies for each participant divided by the number of participants. The first, second and third tasks have lower online copy instances than the fourth task.

5.2.3.2 Non-executable:

The non-executable source code status was only observed on the third task. Half of the third task participants (P2, P3, P4, P5, P9) failed to produce a functional compiled code. By analysing each non-executable source code, multiple reasons caused the code to be non-executable. Participants thought their code carried enough information while the code was incomplete, contained incorrect elements, or missed critical elements. In particular, P2 missed `System.in` that caused incorrect `Scanner`, used `return()` in the method while the method is void, used variable before declaring it and wrote `else if` statement without a condition. Similarly, P3 used `else` without a condition and missed the `Scanner` import, and P2 and P3 copied from online incomplete `Scanner` syntax and encountered difficulty when adopting the code. In addition, P4 wrote four non-executable `if` statements, along with an incomplete `.println` statement. At last, the P5 and P9 source code contained incomplete "Main" method and missed a semicolon.

The activities that led to non-executable code were explored for a more in-depth understanding. Participants excessively searched for syntax during coding, except for P5. Other searches were for learning and resolving errors. Searching produced issues as P2 conducted a repeating search with no successful output, and P3 and P9 conducted unnecessary searches. Most participants referred to the requirements document moderately, except P5 and P9. In the code activities, online code was copied by all participants, except P5, following mainly the visual way. Most participants made choices based on their practices that complicated their coding, used non Web-based code and built knowledge from second task code. In the end, compiling the code took an average time except for P3, while P9 did not debug the task.

By analysing the instances of online copied copy for each participant in the third task, the average copy is four instances where they used tutorial websites to retrieve (appendix C.1 summarises the online copying code activates).

5.2.3.3 Incorrect:

It was noticeable that there were many instances of solving the task without paying attention to the specified requirements. Missing the requirements occurred more in the second task (P1, P3, P6, P8) and fourth task (P5, P6, P10), but less in the third task (P1, P7). Multiple reasons caused the source code to not comply with the specified requirements. In the second task, P1 and P6 did not address `Float` or `Double` user entries caused by copying online code for P1, and P3 and P8 missed doing methods for each requirement. In the third task, the `String` inputs in P1 source code did not function because it accepts non-string entries without processing, and P7 missed the `While` loop for continuous user entry. In the fourth task, P5, P6 and P10 printed threads but not in sequence order, and all solved the task using the Web.

Participants in this source code type conducted some common behaviours. In syntax search, the online search was generally prevalent, where participants looked for tasks related to syntax. Searching for errors and facing complex issues were not high during searching activities. Nearly all the participants resorted to requirements document with moderate to high access. For the code activities, copying the code has normal observations where participants copy online code normally and build knowledge from previous code with some complex issues. One observation of the compiling process occurred to the P1 in the third task, where he spent 5 minutes resolving the errors.

Regarding the instances of online copy details, the average of copying online code was five in the second task and two in the third task; however, high instances of copies were in the fourth task with an average of 15 instances. Regarding the copied code, copying the whole code was noticed in the fourth task to deliver the requirements (complete information visible in Appendix C.1).

5.3 Discussion

5.3.1 RQ1: How do programmers use online resources during programming?

The observations revealed that the predominant coding strategies were searching and retrieving syntax from the Web while others were learning and resolving errors. These results stress the importance of searching during coding, in accordance with results from [LaToza et al., 2006, Singer et al., 2010, Li et al., 2013]. Participants in the experiment referred to various resources, including websites, the requirements document, course-related materials, previous code and the IDE; they predominantly searched using websites. These results reflect those of [Wang, 2017] who found that programming search online for even easy task. Participants preferred referring to the tutorial websites to seek syntax, consistent with the findings of [G. R. Bai et al., 2019]; conversely, other studies suggest that programmers use Stack Overflow for coding [Acar et al., 2016, Abdalkareem et al., 2017b]. Participants started their search by decomposing the tasks into searchable parts. During their coding with the Web, participants faced problems and difficulties that increased their search time and efforts and led to implementing convoluted ways of writing the code.

Participants conducted two interesting searches (noted from searching observations): first, the search for Java basic syntax, such as "Main" method, `.println` statements and loops syntax; second, the search for the previously taught syntax, such as *"two-dimensional array"* and user-entry methods. A possible explanation for the former could be that participants disregard remembering the basic syntax and preferred to fetch it online. Interviews with participants showed that they relied on websites to remind them of this basic information, which they perhaps could have retrieved from memory if they had not had access to the Web. The above reason could also affect the latter search (searching for the previously taught syntax) along with the duration between the experiment and the course taught to the participants. Fetching both syntax types from online could mean that participants in the study preferred not to trust their memory of the syntax itself but, rather, to rely on their ability to find it online, supporting the notion of using the websites as a reminder [Brandt et al., 2009]. In addition, participants searched websites for such items as the equivalent of *"GetChar"* from C++ in Java, suggesting that previous knowledge (especially knowledge of another programming language) plays a role in the search process. In some cases, participants searched continuously the websites for a method without appearing to get relevant results. This could be explained by the fact that participants received irrelevant results. The challenges faced when searching online, such as accessing unhelpful Web pages, are consistent with those of [Escobar-Avila et al., 2019].

Participants searched the websites and used the knowledge, such as the code, to augment their missing elements. Using online code is one reason for referring to the websites [Abdalkareem et al., 2017b]. Participants used more online code on the fourth task than other tasks, suggesting that the more difficult or novel the tasks, the more they needed to use online code. Using the Web for the unfamiliar task comes with uncertainty about the information obtained, as the interviews showed. In addition, participants copied online code, including such basic and taught syntax as `.println` and `Scanner`, using three different ways: using copy-and-paste functionality; observing it and retyping it; memorising it and then retyping it. They also reported in the interviews that the reason for using online code is the ease with which they can locate it. Searching for basic syntax every time the user requires it appears easier than committing it to memory.

An interesting observation was the use of similar syntax across tasks without repeating the search, suggesting that programmers take advantage of their initial search and do not repeat it. Participants in the interview supported this observation, reporting that they engaged in such constant copying of known code. The activity of repeatedly reusing similar syntax aligned with those of using templates for structural purposes [Kim et al., 2004] and the idea of syntax familiarity [Jacques & Kristensson, 2021]. Overall, the results in this chapter offer an in-depth exploration of seeking and utilising online knowledge while coding. It can be assumed that searching websites and reusing the available code are integrated into coding activities. Websites could pose challenges that call for additional coding time and effort.

5.3.2 RQ2: What are the effects of Web use on the resulting code?

Web usage while coding helped participants produce correct code that meets requirements. Participants who produced correct code overcame online search issues and faced complicated approaches. This finding is similar to [Astromskis et al., 2017], who showed that source code complexity implies more compiling and looking for online help. However, these aspects do not always guarantee producing correct code. Web usage impacted the code that participants produced with instances that were either incorrect or non-executable code. Producing such source code could be attributable to not only Web usage but also other factors, such as poor programming practices. Nevertheless, in many instances, participants did not exploit online content efficiently. Sometimes, online copied code is unsuitable or not required by the task. Examples include the manual assignment of array values, not printing array contents, unsuitable data types for user entries, redundant code and unnecessary multiple `Run()` methods. Participants appeared to trust the code without giving it much scrutiny, at the expense of efficiency and effectiveness. It is not possible to know whether better code would have been written were the Web not available, but it is certainly the case that wholesale or unthinking inclusion of online code did not always result in satisfactory code. In the interviews, participants reported that websites urged them to use code as presented, but sometimes this resulted in negative outcomes. The difficulty of reusing online code is also observed superficially in [Escobar-Avila et al., 2019, Xia et al., 2017]. The findings of this study suggest that while reusing online code can be effective, it can also cause complex problems that require time and effort to resolve.

It is interesting to note that the instances of non-executable code appeared only on the third task. One possible explanation is that the third task was challenging, requiring more time and efforts than the other tasks. In addition, the Web and access to the requirements document did not help to reach code that was correct and compliant with requirements. Participants referred moderately to highly to the requirements document for clarification, but such access did not promise to deliver correct code according to requirement specifications. It seems possible that these results are due to websites misleading programmers by giving the impression that the proposed contents meet the requirements or by confusing the requirements with the online presented content; such possible explanation was reported in the interview. Participants' high frequency of referencing the requirements document could signify their struggle to understand the tasks. The finding of this source code type agrees with [Acar et al., 2016] findings, showing that using Stack Overflow leads participants to produce functional but insecure code while using official documentation produces secure but non-executable code. Overall, accessing the websites and requirements document does not guarantee correct code. Participants had a free choice of resources they could access to support their coding; however, such access did not necessarily help them resolve the missing elements and produce correct code.

Other factors could play a role in producing source code types, such as task difficulty, time constraints and previous experience. While participants had previously learned how to achieve most of the tasks, difficulty appeared to be a factor—production of valid code diminished as task difficulty increased. However, participants reported in the interview that the tasks were not difficult to code, indicating that they may have been unaware that code did not meet requirements. Participants noted that planning the time helps in coding the tasks. All the participants shared a similar background, a few with high-level experience. That experience did not reflect positively on the outputs, such as for P5, emphasizing that the noticeable impacts occur regardless of experience.

The instances of participants' observations, source code and interviews showed multiple cases of problematic code aligned with problematic code that previous chapters (Chapters 3 and 4) and previous literature had identified. Observations of coding strategies provided exemplars of *Code with extraneous output*, such as the use of `toString` method in 5.2.2.3. Some instances of copying online code were incomplete (see 5.2.3.2) similar to *Fragment code*, or added unnecessary content that increased the complexity of the code (such as four loop statements instead of two 5.2.2.3) similar to *Inefficient or overly complex code*. Non-executable source code produced by participants is similar to the previously noted *Code that does not compile or does not run*, and incorrect source type is similar to *Code with incorrect output*. Interviews led to further reports of *Code that does not work with the current version of a language or library* and of *Insufficient or incomplete code* (also noted in [Treude & Robillard, 2017, Ragkhitwetsagul et al., 2019]). A possible explanation for propagation of problematic online snippets to the participants' code outputs is that Web use prompts the inclusion of found content without checking, which the interviewees in this study also mentioned.

5.3.3 Limitations

Participants' source code contained relatively few lines, providing little room to analyse the impact of website use. The study's design introduced no baseline with which to compare coding with or without using the websites. Asking participants to code without websites would not have been meaningful, and participants may have been reluctant to take part in a study designed that way. Also, repeating the study with the same participants would introduce previous exposure to the tasks.

In addition, to minimise threats to construct validity, a pilot study was used to validate and refine the tasks and the study's phases. One threat to construct validity could have been a concern about using appropriate tasks to reflect upon the coding activities. Task design ensures participant familiarity by using previous materials, and ensures online content support solving the tasks. Other threat may have emerged as a property of recruiting student participants from the University of Manchester. Whilst it was clearly indicated to participants that their data would be treated anonymously, to prevent impact on their study or outcomes, students may still have been reluctant to divulge behaviours that they thought academe would perceive negatively. Students at UK universities are regularly advised against activities that might constitute plagiarism, such as copying and pasting from external sources, and this therefore may have led students to minimise disclosure of these behaviours during their coding sessions and interviews.

This study does not set out to examine causal relationships, and the internal validity is of limited concern. However, potential influence by external factors could have included experiment time, settings and researcher availability influencing their coding. The study uses multiple data sources to ensure triangulation, including observations, source code and interviews. In addition, steps were taken to involve multiple members of the research team at every step. An inter-rater reliability method was further conducted to increase the reliability of the findings.

5.4 Conclusion

The online experiment in this chapter explored programmers' Web activities during coding tasks, analysing the resulting source to uncover possible consequences for the code. Recordings and source code for 10 programmers solving four programming tasks were collected and participants were interviewed. Recordings and interviews were thematically analysed and source code analysed through a combination of quantitative and qualitative measures.

The observations revealed that the vast majority of Web searches were around syntax, involving tutorial websites. Syntax search comprised breaking down the task into searchable chunks, usually for syntax that was basic and had been taught previously. Searching the Web could produce unfavourable results that require repeating the search. Additionally, participants copied code in three ways: using copy-and-paste functionality; observing it and retyping it; memorising it then retyping it. The copied syntax was not always appropriate or necessary for solving the task.

Participants produced source code in three categories: correct (i.e.working according to the requirements); non-executable; and incorrect (i.e. compiling/running but non-compliant with the requirements). Participants used various strategies during their coding using the websites, and these strategies were linked with participants' source code to investigate their outputs for any implications. Although coding with the Web and encountering complex issues that increased task completion time and effort helped participants produce correct code, Web usage impacted the resulting code, producing either incorrect or non-executable code. Thus, using the Web during coding produced incorrect code.

Chapter 6

Discussion and Conclusion

Websites have become a source of knowledge for programmers to improve their code and resolve issues, enriching their experiences and learning. The research in this thesis explores the potential consequences of Web usage on programmers' memory and code, contributing to knowledge by examining how website use is integrated into the coding process and investigating possible implications. Interviews with students and professional programmers (see Chapter 3) indicated that websites were considered the most preferred coding resource. While programmers perceived little impact on their memory from coding with the Web, such usage caused issues with code. In a survey study that generalised the interview study outcomes (see Chapter 4), students and professional programmers confirmed their preference to code using websites and reuse online code. They also indicated problematic online code, but consider little perceived implications on their memory. In the final study, an effort was made to empirically examine in-practice coding with the websites and the implications for the resulting code (see Chapter 5). The experiment results found that websites were still the programmers' choice, and the implications for code, such as bugs, were also presented in their resulting code.

This final chapter considers the implications of the combined results on coding with the websites by revisiting the three main research questions (Sections 6.1-6.3) and proposing recommendations that could help mitigate negative impacts of programmers' Web use (Section 6.4). The thesis concludes with consideration of limitations and constructive avenues for future research (Section 6.5), and some closing remarks (Section 6.6).

6.1 What resources do programmers use to support the coding process? How and why do they use them?

All three studies indicate that websites were a primary resource when coding. This finding is consistent with previous studies [Acar et al., 2016, Brandt et al., 2009, Xia et al., 2017, G. R. Bai et al., 2019]. The results do, however, show differences in resources used by programmers with different levels of experience, namely students and those with at least a year of professional coding experience. Students chose the Web in preference to course-related resources and books (also seen in [Lausa et al., 2021, Bringula, 2017]), perceiving access the former to be lower effort, easier to search and to be a more readily available source of answers. On the other hand, professionals' exposure to resources within their industry-based environment provides the advantage of accessing more related resources. Professionals used both the Web and existing code libraries as their primary resources (also reported by [Sadowski et al., 2015]). Professionals also considered their colleagues a valuable resource (also observed by [Maalej et al., 2014]), unlike students whose peers were likely to have similar levels of inexperience.

Programmers started their pursuit of knowledge using search engines, specifically Google (also seen in other studies [Maalej et al., 2014, Xia et al., 2017]), and such online search requires programmers' time and effort (consistent with [LaToza et al., 2006, Li et al., 2013, Gao et al., 2020, Koenzen et al., 2020]). Programmers used their search to find and reuse online syntax that helped them to achieve their tasks and shape their outcomes. However, observations also showed that programmers struggled to find relevant results, resulting in repeated search attempts. These difficulties in searching online are consistent with [Escobar-Avila et al., 2019, de Dieu et al., 2022, Wang, 2017].

Survey and interview participants expressed a preference for Stack Overflow (again this is consistent with previous studies [Bhasin et al., 2021, Acar et al., 2016, Fuchs et al., 2014, Han et al., 2020]), with professionals using online repositories (e.g. GitHub) alongside. Participants also reported that tutorial websites offered them content that is more concrete and trusted to be working. Secondary online sources included tutorials and documentation, used to establish and reuse syntax (a behaviour also observed in the final study). The change in websites could be attributed to the difference in the coding activities. The search for solutions to programming issues is likely available in

Stack Overflow, and programmers could associate websites' usage with solving problems. Nevertheless, the need for ready syntax would be available on the tutorial websites. In general, the use of resources evolves throughout the advance of programmers' experiences.

6.2 What is the perceived impact of using websites on programmers' memory?

The role of memory in programming has been established in prior work [Krüger et al., 2018, Fritz et al., 2007, Kelleher & Ichinco, 2019]. Evidence of "Google effects" associated with general use of the Web [Fisher et al., 2022, Sparrow et al., 2011, Schooler & Storm, 2021] therefore led us to explore the potential for effects specifically associated with programmers online resource use.

Results in this thesis show that participants perceive recall of information in memory to be faster than retrieval of the same information from the Web. However, difficulties retrieving information from memory prompted participants to seek information online, and those participants repeatedly referred to online sources for that same pieces of known information. Despite reporting that Web use reminded them of missing information, participants reported that in some cases the location of information on the Web was retained in preference to the information itself. It is interesting to note that programmers constantly rely on websites to resolve even known problems, choosing not to retain such information in memory since it will be available online. While the interviews in Chapter 3 suggested conflicts over whether the Web impeded or helped programmers' memory, overall participants across the three studies perceived little negative impact on their memory from coding with the Web.

There are numerous potential explanations for differences between this thesis's findings and previous research on memory inhibition associated with Web use. Programmers may be more task-focused than participants in controlled studies of memory, and the relatively trivial information recalled in prior studies likely bear little resemblance to the activities our participants associated with memory for coding-related information. Programmers may also use memory differently than the users; they could engage the memory during coding, which could cause programmers to retain a semantically shared common understanding between various languages (also reported by [Kim et al., 2004]), given the complexity of the coding process with multiple programming languages. Moreover, our results capture perceptions of memory rather than measures of recall itself, and participants expressed minimal reflection on the role of memory in coding indicating that these perceptions may not be representative of actual cognition. Inaccuracy of programmers' evaluations of their memory may also be suggested by some of the seemingly contradictory results in our studies – for example, while professionals reported more ability to program with memory alone, students were more likely to report a good memory for programming concepts (alternative explanations for these discrepancies were explored in Section 4.3.2). Furthermore, given that use of the Web is associated with false confidence in ones' own memory and source attribution error [Fisher et al., 2015, 2022], programmer perceptions of negative impacts may be rendered inaccurate as a result of those very same impacts.

Interactions between coding activities, Web use and human memory are under-explored in the literature.

Findings in this thesis align well with what is known about memory use in coding activities (i.e. that memory is a valuable resource) [Ichinco & Kelleher, 2017, Krüger & Hebig, 2020], but less well with prior indications that use of the Web can negatively impact memory for information [Sparrow et al., 2011, Schooler & Storm, 2021, Fisher et al., 2022]. However, since this investigation centres on programmers' perspectives, it's possible that future studies go on to observe the same negative effects seen in Web use more generally. Other factors could play a role on memory effects and programmers' perceptions, for example coding with multiple programming languages could increase complexity of information retention/retrieval with implications for potential "Google effects".

6.3 How does use of websites affect programmers' code?

Participants in all three studies reported that they often reused code from the Web during coding (also observed in prior work [Umarji et al., 2008, Hucka & Graham, 2018, Xia et al., 2017, Hora, 2021]). This code reuse was attributed to availability of suitable code, and a need to extend and refresh knowledge. Observations showed that participants searched and reused online syntax using three distinct ways: using a clipboard, observing the code and retyping it, and memorising the code then retyping it. Participants also reused more online code when solving more challenging tasks.

Online code reuse may affect code accuracy [Abdalkareem et al., 2017a]. Survey and interview participants reported encountering problematic online code, with high awareness of eleven issues and encounters with most of them. Some of these problems have previously been noted in the literature (e.g. security vulnerabilities [Acar et al., 2016, Fischer et al., 2017, Licorish & Nishatharan, 2021, Meng et al., 2018], licence issues [Ragkhitwetsagul et al., 2019], and poor code quality [Ragkhitwetsagul et al., 2019, Treude & Robillard, 2017]). Code snippets with these problems were also documented in search results and interviews during the in-practice study (*Code that does not work with the current version of a language or library*; *Inefficient or overly complex code*; *Code with extraneous output*; *Insufficient or incomplete code*). Moreover, problematic code propagated it into participants code outputs with negative consequences – e.g. encounters with snippets of *Code that does not compile or does not run* resulted in participants producing *Non-executable code*).

Programmers' experience level appears to play a role in recognition of problematic code. Surveyed professionals reported more direct encounters with problematic online code than their student counterparts (students' reported more encounters with *"Code that is difficult or impossible to integrate"*, difficulties that may have arisen from inexperience). Higher rates of problematic code encounters among professional programmers is likely the result of both increased exposure to online code, and increased ability to recognise problematic snippets. Programmers' experience further correlates with practices surrounding reuse of online code. Surveyed students reported making efforts to understand reused code, encountering difficulties understanding reused code and presumed that copying hinders their understanding. By contrast, professionals were more willing to copy online code without understanding it. These differences may be attributed to the fact that students are reusing code during the completion of assessed tasks (and may therefore be anticipating questions from teaching staff about their code), and/or increased confidence levels among experienced professionals.

To the best of our knowledge, this thesis provides the first directed observational study of website use during coding and its impacts on programmers' resulting code. Findings from the observational study, supported by reports from the surveys and interviews, indicate that online content may influence programmers by suggesting unsuitable content. Trust in online content, demands on time, inexperience, and inability to assess code quality (see also [W. Bai et al., 2019]) may lead programmers to reuse problematic code resulting in increased coding time/effort and incorrect code outputs. There are, however, clear benefits to Web use; our studies do show instances of successful online resource use, especially amongst more experienced programmers (similar to the finding of [Sojer & Henkel, 2011]). The findings in this thesis can support the careful reuse of online code by contributing to the body of knowledge that describes programmers' struggles when coding with websites, and the factors that make those struggles more likely/consequential. Results also suggest a further indirect and long-term consequence of code reuse; use of online (and potentially problematic) code may reduce learning and promote poor practice, particularly among novices.

6.4 Recommendations:

The findings in this thesis have applications for stakeholders across software engineering, including programmers in general, Web content authors, students, professional programmers, and organisations. The following sections distill these findings into some concrete recommendations.

6.4.1 To all programmers

- Programmers should continue in their cautious approach to Stack Overflow and online snippets with respect to the potential for introducing bugs into code. This includes:
  - Awareness and acknowledgement of the potential to encounter problematic code online.
  - Caution with respect to use of metrics and measures of esteem (e.g. reputation, upvotes) as a proxy for the validity or quality of solutions and associated code snippets.
  - Reflection and scrutiny of outputs produced with the assistance of online code snippets with respect to known problem areas (code smells, security vulnerabilities, licensing issues).
- When coding with websites, programmers should continuously evaluate whether content aligns with their goals and requirements.
  - Programmers should note that mere access to the websites for seeking and utilising knowledge does not necessarily mean reaching the correct code that is accurate and according to the requirements.
- Programmers should appreciate the time taken to reach their answers when seeking knowledge from the websites and avoid rushing to a conclusion, especially

students to meet a deadline; therefore, they need to acknowledge the time required for such a process to increase their learning outcomes.

6.4.2 To content authors

- Content authors should consider the problematic code from the programmers' perspective and fully explain their online posted code.
- Content authors should consider that users with various experiences may consider their posted code.

6.4.3 To novice programmers and students

- Students/novices should make considered and appropriate resource selection for the task at hand.
  - Students/novices should use a diverse resource set, rather than limiting themselves to the Web. For students this should include resources provided/signposted as part of their teaching.
  - When presented with a novel programming problem, students/novices should seek to solve the problem independently before resorting to the Web.
- Students/novices should consider online code as a learning resource, and should adopt practices to understand, reflect on, and learn from the code they are reusing.
- Students/novices should be aware of their limitations, particularly when using code snippets that go beyond their current expertise. Specifically, they should consider their ability to identify problematic online code, and the potential ramifications of using unidentified problematic code.

6.4.4 To educators

- Educators should providing training in online information seeking, including how to search effectively, source selection, and appropriate expectations for online content.

- Educators should train students to engage in judicious code reuse, equipping them to make sound judgements about the suitability of code snippets. This includes supporting them in determining when (and which parts of) code snippets are relevant and recognising problematic code.

6.4.5 To professional programmers

When writing code for paid employment/as part of a professional voluntary role:

- Developers working with security-related libraries (or writing other code for which online code snippets are known to be impacted by security vulnerabilities [Fischer et al., 2017, Meng et al., 2018]) should be extra vigilant when considering code reuse. They should: familiarise themselves with common relevant problems in online code snippets and ensure full understanding of the reused snippet.

- Developers should be transparent with their colleagues/management when limitations on time, expertise, or other resources is leading them to engage in code reuse.

6.4.6 To organisations

- Development teams should explore opportunities for co-development (i.e. pair programming), irrespective of individuals' development experience.

- Teams should take care to avoid creating a culture in which individuals are reluctant to seek advice from others.

- Professional organisations should ensure developers have adequate training in software licensing and its implications for code reuse.

- Organisations with large internally-developed code libraries should consider mechanisms for making those libraries a viable resource for their software developers. This may introduce a need for better documentation and/or indexing.

- Professional organizations have a responsibility to promote understanding of potentially problematic code, and to stress any liability caused by reusing problematic code online.

6.5 Limitations and Future Work

This thesis focuses on the use of websites during coding activities, with particular focus on Stack Overflow, and on reuse of online code. This limited scope leaves plenty of room for further study, particularly with regard to other resources elicited by participants in studies one (Chapter 3) and two (Chapter 4) (e.g. students' course-related resources and code libraries used by professional programmers) – in some cases these are well explored in the literature, but others are poorly understood.

Studies in this thesis recruited students who enrolled in computer science courses (all three studies), together with smaller samples of professionals (studies one [Chapter 3] and two [Chapter 4] only). In studies one and two, the intent was to sample a student population and thus the sample is inherently representative. In study three (Chapter 5), however, computer science students are used as a proxy for programmers as a whole. Our justification for doing so is that (a) students are more readily available, and (b) in our earlier studies (Chapters 3 and 4) students have been largely comparable with professionals. Despite this, there is need for a more nuanced study of programming activity that targets and distinguishes between the diverse set of populations that engage in programming activities (e.g. recent graduates, established professionals, hobby programmers).

A further limitation of our samples is the use of volunteer sampling, selected due to the relative ease of recruiting sufficiently large samples. This sampling method may have introduced unanticipated biases resulting in an unrepresentative sample. Our samples were also recruited within the UK, limiting our ability to generalise findings to programmers more generally.

Consideration of the role of, and interactions between, programmers' memory and code (in Chapters 3 and 4) is one of the more novel aspects of the work reported in this thesis. As a result, this work is exploratory and based on programmers' reported experiences rather than objective measures. Prior research on the use of the Web for general knowledge suggests that an experimental paradigm could be used to provide objective measures. However, since our findings suggest little perceived effect, future experimentation in this area will likely benefit from continued use of subjective measures alongside objective data. Moreover, knowledge acquisition for programming tasks typically takes place over an extended period, making it a challenging target for controlled lab studies.

Regarding the methods used, the thesis started with the interview study (Chapter 3) followed by the survey study (Chapter 4) to explore memory and code. This design provided an in-depth understanding of the matter from programmers' perspectives. While the design could start with the survey and then follow with the interview, the followed approach in this thesis provided more understanding of the phenomena with the ability to select related points for generalisation. Furthermore, the experiment mentioned in Chapter 5 was designed to be qualitative in nature. Although collecting or analysing ready set of code would have been easier to set up (e.g. no need for extensive time to analyse the videos), the current design of the experiment was chosen to ensure more exploration and understanding of the coding with the websites, and the qualitative approach ensures the richness of information.

This thesis contributes to the understanding of the potential for problematic code as a result of websites use during development. Future work could explore mitigation strategies designed to prevent Web use from negatively impacting code outputs. This could include the design of tools (e.g. software, IDE and browser plugins) that identify programmers' copy-and-paste activities when using websites and suggests follow-on activities such as the review of copied code for understanding, fit with requirements, and quality control. Problematic snippets identified in research studies (e.g. [Fischer et al., 2017]) may also have a role in such tool development. Our recommendations (Section 6.4) also suggest a role for education, training and organisational policies in mitigation, and future research should explore the effectiveness of these.

This thesis' explorations of interactions between coding activities, Web use and programmers' memory are an early step in a largely unexplored research space. Future work could draw on and apply theory and evidence from cognitive psychology, with a view to better understanding these interactions. For example, programmers in our studies noted resorting to the Web for information when their use of multiple programming languages made it difficult to recall language-specific syntax. This may be explored further with consideration for psychological theories of interference, which suggests that similar memories compete making it harder to recall relevant information.

6.6 Conclusion

The studies in this thesis indicate that websites are the primary resource programmers use when coding; that they seek and reuse online knowledge; and use of online resources has implications for code outputs. The first two studies provide quantitative and qualitative analysis of how the Web is used during coding, and of encounters and reuse of online code. The third study provides further understanding of website use during coding activities and the propagation of problematic snippets to code outputs.

Together, the studies show that experienced and novice programmers start their online search using Google, and often end up directed to Stack Overflow. Professionals are more likely to draw on larger codebases (both commercial codebases and code from GitHub), whilst students largely limit themselves to Web tutorials, Q&A and code snippets. Alongside Web use, programmers report drawing on their memory during coding tasks, with greater experience increasing the likelihood that programming tasks could be completed using memory alone. Despite some evidence of memory impairment in generalised Web use (i.e. the Google effect), programmers in our studies perceived little implications on their memory. Our studies do, however, suggest implications for code produced when websites are used as a primary resource. Some of the reported online problematic code appeared on programmers' outputs when coding with the websites, indicating that using the websites did not guarantee that programmers reached the correct code. Participants' reports of encountering problematic online code indicated a high possibility of such code being present online.

Programmers will inevitably use the websites during development, and their usage will likely introduce issues to their code. The findings in this thesis are contributing towards our understanding of how this occurs, which in turn is critical for the development of appropriate mitigation strategies including education programs, organisational policy and effective tools.

References

Abdalkareem, R., Shihab, E., & Rilling, J. (2017a). On code reuse from StackOverflow: An exploratory study on Android apps. *Information and Software Technology*, *88*, 148–158. doi: 10.1016/j.infsof.2017.04.005

Abdalkareem, R., Shihab, E., & Rilling, J. (2017b). What do developers use the crowd for? a study using Stack Overflow. *IEEE Software*, *34*(2), 53–60. doi: 10.1109/MS.2017.31

Abtahi, P., & Dietz, G. (2020). Learning rust: How experienced programmers leverage resources to learn a new programming language. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–8). ACM. `https://dl.acm.org/doi/10.1145/3334480.3383069`.

Acar, Y., Backes, M., Fahl, S., Kim, D., Mazurek, M. L., & Stransky, C. (2016). You get where you're looking for: The impact of information sources on code security. In *2016 IEEE Symposium on Security and Privacy (SP)* (pp. 289–305). IEEE. `http://ieeexplore.ieee.org/document/7546508/`. doi: 10.1109/SP.2016.25

An, L., Mlouki, O., Khomh, F., & Antoniol, G. (2017). Stack Overflow: A code laundering platform? In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 283–293). IEEE. doi: 10.1109/SANER.2017.7884629

Astromskis, S., Bavota, G., Janes, A., Russo, B., & Di Penta, M. (2017). Patterns of developers" behaviour: A 1000-hour industrial study. *Journal of Systems and Software*, *132*, 85–97. `https://linkinghub.elsevier.com/retrieve/pii/S016412121730136X`. doi: 10.1016/j.jss.2017.06.072

Bai, G. R., Clee, B., Shrestha, N., Chapman, C., Wright, C., & Stolee, K. T. (2019). Exploring tools and strategies used during regular expression composition tasks. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)* (pp. 197–208). IEEE. doi: 10.1109/ICPC.2019.00039

Bai, W., Akgul, O., & Mazurek, M. L. (2019). A qualitative investigation of insecure code propagation from online forums. In *2019 IEEE Cybersecurity Development (SecDev)* (pp. 34–48). IEEE. `https://ieeexplore.ieee.org/document/8901567/`.

Bakeman, R., & Quera, V. (2012). Behavioral observation. In H. Cooper, P. M. Camic, D. L. Long, A. T. Panter, D. Rindskopf, & K. J. Sher (Eds.), *H.Cooper (Ed), APA Handbook of Research Methods in Psychology* (Vols. 1, Foundations, planning, measures, and psychometrics, p. 207—225). APA. doi: 10.1037/13619-013

Baltes, S., & Treude, C. (2020). Code duplication on Stack Overflow. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)* (pp. 13–16). IEEE. doi: 10.1145/3377816.3381744

Bateson, A. G., Alexander, R. A., & Murphy, M. D. (1987). Cognitive processing differences between novice and expert computer programmers. *International Journal of Man-Machine Studies*, *26*(6), 649–660. doi: 10.1016/S0020-7373(87)80058-5

Becker, B. A. (2016). A new metric to quantify repeated compiler errors for novice programmers. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 296–301). ACM. https://dl.acm.org/doi/10.1145/2899415.2899463.

Bhasin, T., Murray, A., & Storey, M.-A. (2021). Student experiences with github and Stack Overflow: An exploratory study. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)* (pp. 81–90). IEEE. doi: 10.1109/CHASE52884.2021.00017

Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., & Klemmer, S. R. (2009). Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1589–1598). ACM. doi: 10.1145/1518701.1518944

Brandt, J., Guo, P. J., Lewenstein, J., & Klemmer, S. R. (2008). Opportunistic programming: How rapid ideation and prototyping occur in practice. In *Proceedings of the 4th International Workshop on End-user Software Engineering* (pp. 1–5). ACM. doi: 10.1145/1370847.1370848

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. doi: 10.1191/1478088706qp063oa

Braun, V., & Clarke, V. (2013). *Successful qualitative research: A practical guide for beginners*. Sage Publications.

Bringula, R. P. (2017). Influence of usage of e-books, online educational materials, and other programming books and students' profiles on adoption of printed programming textbooks. *Program: Electronic Library and Information Systems*, *51*(4), 441–457. doi: 10.1108/PROG-06-2015-0046

Börstler, J., MacKellar, B., Störrle, H., Toll, D., van Assema, J., Duran, R., ... Kleiner, C. (2017). "I know it when I see it" Perceptions of Code Quality: ITiCSE '17 Working Group Report. In *Proceedings of the 2017 ITiCSE Conference on Working Group Reports - ITiCSE-WGR '17* (pp. 70–85). ACM. http://dl.acm.org/citation.cfm?doid=3174781.3174785.

Carr, N. (2008). Is Google making us stupid? *Teachers College Record*, *107*(2), 89–94. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1744-7984.2008.00172.x.

Carter, A. S., Hundhausen, C. D., & Adesope, O. (2015). The normalized programming state model: Predicting student performance in computing courses based on programming behavior. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research - ICER '15* (pp. 141–150). ACM. `http://dl.acm.org/citation.cfm?doid=2787622.2787710`.

Carter, A. S., Hundhausen, C. D., & Adesope, O. (2017). Blending measures of programming and social behavior into predictive models of student achievement in early computing courses. *ACM Transactions on Computing Education*, *17*(3), 1–20. `http://dl.acm.org/citation.cfm?doid=3135995.3120259`.

Cecutti, L., Chemero, A., & Lee, S. W. S. (2021). Technology may change cognition without necessarily harming it. *Nature Human Behaviour*, *5*(8), 973–975. `http://www.nature.com/articles/s41562-021-01162-0`.

Charatan, Q., & Kans, A. (2019). *Java in two semesters: Featuring JavaFX* (Fourth edition ed.). Springer. doi: 10.1007/978-3-319-99420-8

Chattopadhyay, S., Nelson, N., Nam, T., Calvert, M., & Sarma, A. (2018). Context in programming: an investigation of how programmers create context. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering* (pp. 33–36). ACM. `https://dl.acm.org/doi/10.1145/3195836.3195861`.

Chen, M., Fischer, F., Meng, N., Wang, X., & Grossklags, J. (2019). How reliable is the crowdsourced knowledge of security implementation? In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 536–547). IEEE. doi: 10.1109/ICSE.2019.00065

Ciborowska, A., Kraft, N. A., & Damevski, K. (2018). Detecting and characterizing developer behavior following opportunistic reuse of code snippets from the web. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)* (pp. 94–97). ACM. doi: 10.1145/3196398.3196467

Connaway, L. S., Dickey, T. J., & Radford, M. L. (2011). "If it is too inconvenient I'm not going after it:" Convenience as a critical factor in information-seeking behaviors. *Library & Information Science Research*, *33*(3), 179–190. doi: 10.1016/j.lisr.2010.12.002

Creative commons: Attribution-sharealike 4.0 international (cc by-sa 4.0). (n.d.). Retrieved January 04, 2023, from `https://creativecommons.org/licenses/by-sa/4.0/`

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). Sage Publications.

de Dieu, M. J., Liang, P., & Shahin, M. (2022). How do developers search for architectural information? An industrial survey. In *2022 IEEE 19th International Conference on Software Architecture (ICSA)* (pp. 58–68). IEEE. doi: 10.1109/ICSA53651.2022.00014

Do I have to worry about copyright issues for code posted on Stack Overflow? (2008). Retrieved February 02, 2023, from `http://meta.stackexchange.com/questions/12527/do-i-haveto-worry-about-copyright-issues-for-code-posted-onstack-overflow`

Détienne, F. (2001). *Software design — Cognitive aspects.* Springer. doi: 10.1007/978-1-4471-0111-6

Ericsson, K. A., & Kintsch, W. (1995). Long-term working memory. *Psychological review*, *102*(2), 211. doi: 10.1037/0033-295X.102.2.211

Escobar-Avila, J., Venuti, D., Di Penta, M., & Haiduc, S. (2019). A survey on online learning preferences for computer science and programming. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)* (pp. 170–181). IEEE. `https://ieeexplore.ieee.org/document/8802102/.`

Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., & Oivo, M. (2018). Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineering*, *23*(1), 452–489. doi: 10.1007/s10664-017-9523-3

Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., & Fahl, S. (2017). Stack Overflow considered harmful? the impact of copy and paste on Android application security. In *2017 IEEE Symposium on Security and Privacy (SP)* (p. 121-136). IEEE. doi: 10.1109/SP.2017.31

Fisher, M., Goddu, M. K., & Keil, F. C. (2015). Searching for explanations: How the internet inflates estimates of internal knowledge. *Journal of Experimental Psychology: General*, *144*(3), 674. doi: 10.1037/xge0000070

Fisher, M., Smiley, A. H., & Grillo, T. L. H. (2022). Information without knowledge: the effects of internet search on learning. *Memory*, *30*(4), 375–387. `https://www.tandfonline.com/doi/full/10.1080/09658211.2021.1882501`. doi: 10.1080/09658211.2021.1882501

Fritz, T., Murphy, G. C., & Hill, E. (2007). Does a programmer's activity indicate knowledge of code? In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering* (pp. 341–350). ACM. doi: 10.1145/1287624.1287673

Fuchs, M., Heckner, M., Raab, F., & Wolff, C. (2014). Monitoring students' mobile app coding behavior data analysis based on IDE and browser interaction logs. In *2014 IEEE Global Engineering Education Conference (EDUCON)* (pp. 892–899). IEEE. `http://ieeexplore.ieee.org/document/6826202/`.

Gallardo-Valencia, R. E., & Sim, S. E. (2011). What kinds of development problems can be solved by searching the web? A field study. In *Proceedings of the 3rd International Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation* (pp. 41–44). IEEE. doi: 10.1145/1985429.1985440

Gao, G., Voichick, F., Ichinco, M., & Kelleher, C. (2020). Exploring programmers' api learning processes: Collecting web resources as external memory. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 1–10). IEEE. doi: 10.1109/VL/HCC50065.2020.9127274

Giebl, S., Mena, S., Sandberg, R., Bjork, E. L., & Bjork, R. A. (2022). Thinking first versus Googling first: Preferences and consequences. *Journal of Applied Research in Memory and Cognition*. doi: 10.1037/mac0000072

Giebl, S., Mena, S., Storm, B. C., Bjork, E. L., & Bjork, R. A. (2021). Answer first or Google first? using the Internet in ways that enhance, not impair, one's subsequent retention of needed information. *Psychology Learning & Teaching*. `http://journals.sagepub.com/doi/10.1177/1475725720961593`.

Grinschgl, S., Meyerhoff, H. S., & Papenmeier, F. (2020). Interface and interaction design: How mobile touch devices foster cognitive offloading. *Computers in Human Behavior*, *108*, 106317. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0747563220300716` doi: https://doi.org/10.1016/j.chb.2020.106317

Grinschgl, S., Papenmeier, F., & Meyerhoff, H. S. (2021). Consequences of cognitive offloading: Boosting performance but diminishing memory. *Quarterly Journal of Experimental Psychology*, *74*(9), 1477–1496. doi: 10.1177/17470218211008060

Han, L., Chen, T., Demartini, G., Indulska, M., & Sadiq, S. (2020). On understanding data worker interaction behaviors. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 269–278). ACM. doi: 10.1145/3397271.3401059

Heersmink, R. (2016). The internet, cognitive enhancement, and the values of cognition. *Minds and Machines*, *26*(4), 389–407. doi: 10.1007/s11023-016-9404-3

Hora, A. (2021). Googling for software development: what developers search for and what they find. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)* (pp. 317–328). IEEE. doi: 10.1109/MSR52588.2021.00044

Höst, M., Regnell, B., & Wohlin, C. (2000). Using students as subjects—a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, *5*, 201–214. doi: 10.1023/A:1026586415054

Hucka, M., & Graham, M. J. (2018). Software search is not a science, even among scientists: A survey of how scientists and engineers find software. *Journal of Systems and Software*, *141*, 171–191. doi: 10.1016/j.jss.2018.03.047

Ichinco, M., & Kelleher, C. (2017). Towards better code snippets: Exploring how code snippet recall differs with programming experience. In *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 37–41). IEEE. doi: 10.1109/VLHCC.2017.8103448

Jacques, J. T., & Kristensson, P. O. (2021). Studying programmer behaviour at scale: A case study using amazon mechanical turk. In *Companion Proceedings of the 5th International Conference on the Art, Science, and Engineering of Programming* (pp. 36–48). ACM. doi: 10.1145/3464432.3464436

Jadud, M. C. (2005). A first look at novice compilation behaviour using BlueJ. *Computer Science Education*, *15*(1), 25–40. doi: 10.1080/08993400500056530

Jadud, M. C. (2006). Methods and tools for exploring novice compilation behaviour. In *Proceedings of the Second International Workshop on Computing Education Research* (pp. 73–84). ACM. doi: 10.1145/1151588.1151600

Kahn, A. S., & Martinez, T. M. (2020). Text and you might miss it? Snap and you might remember? Exploring "Google effects on memory" and cognitive self-esteem in the context of Snapchat and text messaging. *Computers in Human Behavior*, *104*, 106166. doi: 10.1016/j.chb.2019.106166

Kapser, C. J., & Godfrey, M. W. (2008). "Cloning considered harmful" considered harmful: Patterns of cloning in software. *Empirical Software Engineering*, *13*(6), 645–692. `http://link.springer.com/10.1007/s10664-008-9076-6`.

Kelleher, C., & Ichinco, M. (2019). Towards a model of API learning. In *2019 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 163–168). IEEE. doi: 10.1109/VLHCC.2019.8818850

Kelly, M. O., & Risko, E. F. (2022). Revisiting the influence of offloading memory on free recall. *Memory & Cognition*, *50*(4), 710–721. doi: 10.3758/s13421-021-01237-3

Kim, M., Bergman, L., Lau, T., & Notkin, D. (2004). An ethnographic study of copy and paste programming practices in OOPL. In *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE '04.* (pp. 83–92). IEEE. doi: 10.1109/ISESE.2004.1334896

Kirk, D., Crow, T., Luxton-Reilly, A., & Tempero, E. (2020). On assuring learning about code quality. In *Proceedings of the Twenty-Second Australasian Computing Education Conference* (pp. 86–94). ACM. `https://dl.acm.org/doi/10.1145/3373165.3373175`.

Kitchenham, B., & Pfleeger, S. L. (2002). Principles of survey research part 3: Constructing a survey instrument. *ACM SIGSOFT Software Engineering Notes*, *27*(2), 20–24. doi: 10.1145/511152.511155

Ko, A. J., Myers, B. A., Coblenz, M. J., & Aung, H. H. (2006). An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on Software Engineering*, *32*(12), 971–987. `http://ieeexplore.ieee.org/document/4016573/`.

Koenzen, A. P., Ernst, N. A., & Storey, M.-A. D. (2020). Code duplication and reuse in Jupyter notebooks. In *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 1–9). IEEE. doi: 10.1109/VL/HCC50065.2020.9127202

Krinke, J. (2008). Is cloned code more stable than non-cloned code? In *2008 Eighth IEEE International Working Conference on Source Code Analysis and Manipulation* (pp. 57–66). IEEE. `http://ieeexplore.ieee.org/document/4637539/`. doi: 10.1109/SCAM.2008.14

Krueger, C. W. (1992). Software reuse. *ACM Computing Surveys (CSUR)*, *24*(2), 131–183. doi: 10.1145/130844.130856

Krüger, J., Wiemann, J., Fenske, W., Saake, G., & Leich, T. (2018). Do you remember this source code? In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)* (pp. 764–775). IEEE. doi: 10.1145/3180155.3180215

Krüger, J., & Hebig, R. (2020). What developers (care to) recall: An interview survey on smaller systems. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 46–57). IEEE. doi: 10.1109/ICSME46990.2020.00015

LaToza, T. D., Garlan, D., Herbsleb, J. D., & Myers, B. A. (2007). Program comprehension as fact finding. In *Proceedings of the the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering* (p. 361–370). ACM. doi: 10.1145/1287624.1287675

LaToza, T. D., Venolia, G., & DeLine, R. (2006). Maintaining mental models: A study of developer work habits. In *Proceedings of the 28th International Conference on Software Engineering* (pp. 492–501). ACM. doi: 10.1145/1134285.1134355

Lausa, S., Bringula, R., Catacutan-Bangit, A. E., & Santiago, C. (2021). Information-seeking behavior of computing students while programming: Educational learning materials usage, satisfaction of use, and inconveniences. In *2021 IEEE Global Engineering Education Conference (EDUCON)* (pp. 761–765). IEEE. doi: 10.1109/EDUCON46332.2021.9453905

Lazar, J., Feng, J. H., & Hochheiser, H. (2017). Surveys. In *Research Methods in Human-computer Interaction.* Morgan Kaufmann. doi: 10.1016/B978-0-12-805390-4.00005-4

Li, H., Xing, Z., Peng, X., & Zhao, W. (2013). What help do developers seek, when and how? In *2013 20th Working Conference on Reverse Engineering (WCRE)* (pp. 142–151). IEEE. doi: 10.1109/WCRE.2013.6671289

Licorish, S. A., & Nishatharan, T. (2021). Contextual profiling of Stack Overflow java code security vulnerabilities initial insights from a pilot study. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)* (pp. 1060–1068). IEEE. doi: 10.1109/QRS-C55045.2021.00160

Loh, K. K., & Kanai, R. (2016). How has the internet reshaped human cognition? *The Neuroscientist*, *22*(5), 506–520. doi: 10.1177/1073858415595005

Lozano, A., & Wermelinger, M. (2010). Tracking clones' imprint. In *Proceedings of the 4th International Workshop on Software Clones* (pp. 65–72). ACM. doi: 10.1145/1808901.1808910

Lu, X., Kelly, M. O., & Risko, E. F. (2020). Offloading information to an external store increases false recall. *Cognition*, *205*, 104428. doi: 10.1016/j.cognition.2020.104428

Maalej, W., Tiarks, R., Roehm, T., & Koschke, R. (2014). On the comprehension of program comprehension. *ACM Transactions on Software Engineering and Methodology*, *23*(4), 1–37. `https://dl.acm.org/doi/10.1145/2622669`.

Macias, C., Yung, A., Hemmer, P., & Kidd, C. (2015). Memory strategically encodes externally unavailable information. In *Cogsci.* `https://cogsci.mindmodeling.org/2015/papers/0255/paper0255.pdf`.

Meldrum, S., Licorish, S. A., Owen, C. A., & Savarimuthu, B. T. R. (2020). Understanding Stack Overflow code quality: A recommendation of caution. *Science of Computer Programming*, *199*, 102516. `https://www.sciencedirect.com/science/article/pii/S0167642320301246`.

Meng, N., Nagy, S., Yao, D., Zhuang, W., & Arango-Argoty, G. (2018). Secure coding practices in java: Challenges and vulnerabilities. In *Proceedings of the 40th*

International Conference on Software Engineering (pp. 372–383). IEEE. doi: 10.1145/3180155.3180201

Misu, M. R. H., & Satter, A. (2022). An exploratory study of analyzing JavaScript online code clones. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension* (pp. 94–98). ACM. doi: 10.1145/3524610.3528390

Mondal, M., Rahman, M. S., Roy, C., & Schneider, K. (2018). Is cloned code really stable? *Empirical Software Engineering*, *23*(2), 693–770. doi: 10.1007/s10664-017-9528-y

Nasehi, S. M., Sillito, J., Maurer, F., & Burns, C. (2012). What makes a good code example?: A study of programming Q&A in StackOverflow. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)* (pp. 25–34). IEEE. doi: 10.1109/ICSM.2012.6405249

Ndukwe, I. G., Licorish, S. A., Tahir, A., & MacDonell, S. G. (2023). How have views on software quality differed over time? Research and practice viewpoints. *Journal of Systems and Software*, *195*, 111524. doi: 10.1016/j.jss.2022.111524

Nishi, M. A., Ciborowska, A., & Damevski, K. (2019). Characterizing duplicate code snippets between Stack Overflow and tutorials. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)* (pp. 240–244). IEEE. doi: 10.1109/MSR.2019.00048

Nygren, H., Leinonen, J., & Hellas, A. (2017). Tracking students' internet browsing in a machine exam. In *Proceedings of the 6th Computer Science Education Research Conference* (pp. 91–95). ACM. doi: 10.1145/3162087.3162103

Ormerod, T. (1990). Human Cognition and Programming. In J. Hoc (Ed.), *Psychology of Programming* (pp. 63–82). Academic Press. `http://www.sciencedirect.com/science/article/pii/B9780123507723500094`. doi: 10.1016/B978-0-12-350772-3.50009-4

Parnin, C., & Rugaber, S. (2012). Programmer information needs after memory failure. In *2012 20th IEEE International Conference on Program Comprehension (ICPC)* (pp. 123–132). IEEE. doi: 10.1109/ICPC.2012.6240479

Pfleeger, S. L., & Kitchenham, B. (2001). Principles of survey research part 1: Turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes*, *26*(6), 16–18. doi: 10.1145/505532.505535

Piwek, P., & Savage, S. (2020). Challenges with learning to program and problem solve: An analysis of student online discussions. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 494–499). ACM. doi: 10.1145/3328778.3366838

Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G., & Oliveto, R. (2019). Toxic code snippets on StackOverflow. *IEEE Transactions on Software Engineering*, *47*(3), 560–581. https://ieeexplore.ieee.org/document/8643998/. doi: 10.1109/TSE.2019.2900307

Risko, E. F., & Gilbert, S. J. (2016). Cognitive offloading. *Trends in Cognitive Sciences*, *20*(9), 676–688. https://www.cell.com/trends/cognitive-sciences/abstract/S1364-6613(16)30098-5.

Rose, D. E., & Levinson, D. (2004). Understanding user goals in web search. In *Proceedings of the 13th international conference on world wide web* (p. 13–19). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/988672.988675 doi: 10.1145/988672.988675

Roy, C. K., Cordy, J. R., & Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, *74*(7), 470–495. doi: 10.1016/j.scico.2009.02.007

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, *14*(2), 131–164. doi: 10.1007/s10664-008-9102-8

Sadowski, C., Stolee, K. T., & Elbaum, S. (2015). How developers search for code: a case study. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (pp. 191–201). ACM. http://dl.acm.org/citation.cfm?doid=2786805.2786855. doi: 10.1145/2786805.2786855

Schooler, J. N., & Storm, B. C. (2021). Saved information is remembered less well than deleted information, if the saving process is perceived as reliable. *Memory*, *29*(9), 1101–1110. doi: 10.1080/09658211.2021.1962356

Schröter, I., Krüger, J., Siegmund, J., & Leich, T. (2017). Comprehending studies on program comprehension. In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)* (pp. 308–311). IEEE. doi: 10.1145/3328778.3366838

Seaman, C. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, *25*(4), 557–572. doi: 10.1109/32.799955

Sheatsley, P. B. (1983). Questionnaire construction and item writing. In *Handbook of Survey Research* (pp. 195–230). Academic Press. `https://www.sciencedirect.com/science/article/pii/B9780125982269500124`. doi: 10.1016/B978-0-12-598226-9.50012-4

Shneiderman, B. (1976). Exploratory experiments in programmer behavior. *International Journal of Computer & Information Sciences*, *5*(2), 123–143. `http://link.springer.com/10.1007/BF00975629`.

Shneiderman, B., & Mayer, R. (1979). Syntactic/semantic interactions in programmer behavior: A model and experimental results. *International Journal of Computer & Information Sciences*, *8*(3), 219–238. doi: 10.1007/BF00977789

Siegmund, J., Kästner, C., Liebig, J., Apel, S., & Hanenberg, S. (2014). Measuring and modeling programming experience. *Empirical Software Engineering*, *19*(5), 1299–1334. doi: 10.1007/s10664-013-9286-4

Simon, H. A. (1980). Problem solving and education. In D. Tuma & F. Reif (Eds.), (pp. 81–96). Lawrence Erlbaum Associates.

Singer, J., Lethbridge, T., Vinson, N., & Anquetil, N. (2010). An examination of software engineering work practices. In *CASCON First Decade High Impact Papers* (pp. 174–188). ACM. doi: 10.1145/1925805.1925815

Sojer, M., & Henkel, J. (2011). License risks from Ad Hoc reuse of code from the Internet. *Commun. ACM*, *54*(12), 74–81. `http://doi.acm.org/10.1145/2043174.2043193`. doi: 10.1145/2043174.2043193

Sparrow, B., Liu, J., & Wegner, D. M. (2011). Google effects on memory: Cognitive consequences of having information at our fingertips. *Science*, *333*(6043), 776–778. http://www.sciencemag.org/cgi/doi/10.1126/science.1207745. doi: 10.1126/science.1207745

StackExchange data explorer. (2023). Retrieved January 04 2023, from www.data.stackexchange.com/

Stack Overflow privileges. (2023). Retrieved January 04, 2023, from https://stackoverflow.com/help/privileges

Starke, J., Luce, C., & Sillito, J. (2009). Searching and skimming: An exploratory study. In *2009 IEEE International Conference on Software Maintenance* (pp. 157–166). IEEE. doi: 10.1109/ICSM.2009.5306335

Storm, B. C., James, K. K., & Stone, S. M. (2021). Pretesting can be beneficial even when using the internet to answer questions. *Memory*, *30*(4), 388–395. doi: 10.1080/09658211.2020.1863990

Storm, B. C., & Stone, S. M. (2015). Saving-enhanced memory: The benefits of saving on the learning and remembering of new information. *Psychological Science*, *26*(2), 182–188.

Sue, V. M., & Ritter, L. A. (2012). *Conducting online surveys.* Sage Publications. doi: 10.4135/9781506335186

Treude, C., Barzilay, O., & Storey, M.-A. (2011). How do programmers ask and answer questions on the web? (NIER track). In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)* (pp. 804–807). ACM. doi: 10.1145/1985793.1985907

Treude, C., & Robillard, M. P. (2017). Understanding Stack Overflow code fragments. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 509–513). IEEE. http://ieeexplore.ieee.org/document/8094452/.

Umarji, M., Sim, S. E., & Lopes, C. (2008). Archetypal internet-scale source code searching. In *Open Source Development, Communities and Quality* (pp. 257–263). Springer. doi: 10.1007/978-0-387-09684-1_21

Wang, Y. (2017). Characterizing developer behavior in cloud based ides. In *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (pp. 48–57). IEEE. doi: 10.1109/ESEM.2017.27

Ward, A. F. (2013). Supernormal: How the internet is changing our memories and our minds. *Psychological Inquiry*, *24*(4), 341–348. `http://www.tandfonline.com/doi/abs/10.1080/1047840X.2013.850148.`

Watson, C., Li, F. W., & Godwin, J. L. (2013). Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *2013 IEEE 13th International Conference on Advanced Learning Technologies* (pp. 319–323). IEEE. doi: 10.1109/ICALT.2013.99

Wegner, D. M. (1987). Transactive memory: A contemporary analysis of the group mind. *Theories of Group Behavior*, 185–208. `https://dtg.sites.fas.harvard.edu/DANWEGNER/pub/Wegner%20Transactive%20Memory.pdf.`

Wegner, D. M. (1995). A Computer Network Model of Human Transactive Memory. *Social Cognition*, *13*(3), 319–339. `https://www.proquest.com/docview/848853543/abstract/906244CFD5C74059PQ/1.`

What is the license for the content i post? (2023). Retrieved January 04, 2023, from `https://stackoverflow.com/help/licensing`

Wong-Aitken, D., Cukierman, D., & Chilana, P. K. (2022). ”It depends on whether or not i'm lucky” how students in an introductory programming course discover, select, and assess the utility of web-based resources. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (pp. 512–518). ACM. `https://dl.acm.org/doi/10.1145/3502718.3524751.`

Wu, Y., Wang, S., Bezemer, C.-P., & Inoue, K. (2019). How do developers utilize source code from StackOverflow? *Empirical Software Engineering*, *24*(2), 637–673. `http://link.springer.com/10.1007/s10664-018-9634-5.`

Xia, X., Bao, L., Lo, D., Kochhar, P. S., Hassan, A. E., & Xing, Z. (2017). What do developers search for on the web? *Empirical Software Engineering*, *22*(6), 3149–3185. doi: 10.1007/s10664-017-9514-4

Yang, D., Martins, P., Saini, V., & Lopes, C. (2017). Stack Overflow in github: any snippets there? In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)* (pp. 280–290). IEEE. doi: 10.1109/MSR.2017.13

Zhang, H., Wang, S., Chen, T.-H., Zou, Y., & Hassan, A. E. (2021). An empirical study of obsolete answers on Stack Overflow. *IEEE Transactions on Software Engineering*, *47*(4), 850–862. doi: 10.1109/TSE.2019.2906315

Zhang, T., Upadhyaya, G., Reinhardt, A., Rajan, H., & Kim, M. (2018). Are code examples on an online Q&A forum reliable?: a study of api misuse on Stack Overflow. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)* (pp. 886–896). IEEE. doi: 10.1145/3180155.3180260

Zhang, T., Yang, D., Lopes, C., & Kim, M. (2019). Analyzing and supporting adaptation of online code examples. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (pp. 316–327). IEEE. doi: 10.1109/ICSE.2019.00046

Appendix A

Supplemental materials for Chapter 3

A.1 Recruitment letters used to recruit participants

A.1.1 For instructors:

Sample Recruitment Letter or Email for Course Instructor

Dear Course Instructor,

My name is Omar Alghamdi and I am a PhD student from the Department of Computer Science at the University of Manchester. I am writing to you on your role as director of teaching. My research is about the role of the Internet on programming. It will investigate the role of websites on the programmer and investigate any impacts.

I’m writing to ask your permission to recruit students from your course in my study. The study needs participants from a computer science course to fill out an online questionnaire. The questionnaire will not take more than 10-15 minutes to complete. I will send you complete information with the questionnaire questions.

Participation in the study is completely optional. A student who wishes to participate in the study will have no positive or negative impact on either their studies or their relationship with the university/department

Thank you very much.

Sincerely,

A.1.2 For students:

Dear student,

My name is Omar Alghamdi and I am a PhD student from the Department of Computer Science at the University of Manchester. I am writing to invite you to participate in my research study about the role of the Internet on programming. You're eligible to participate in this study because you are currently studying on a computer programming course

If you decide to participate in this study, you will be invited to answer few questions about how you program. The interview will take 45 minutes and will be audio recorded. In recognition of the time spent we will provide a small gift card.

Remember, this is completely voluntary. You can choose to be in the study or not. If you wish to participate in the study, you will have no positive or negative impact on either your studies or your relationship with the university/department.

If you'd like to participate or have any questions about the study, please email at *omar.alghamdi@postgrad.manchester.ac.uk*

Note: to participate in the study, you should be a second-year student who is enrolled in any programming courses. For further information, Please read the attachment (Participant information sheet).

Thank you very much.

Sincerely,

A.2 Participants information sheet and consent form

MANCHESTER 1824
The University of Manchester

Participant Information Sheet

The role of the Internet on Programming

This is a research project undertaken at the Department of Computer Science at the University of Manchester. Before you decide whether to take part, it is important for you to understand why the research is being conducted and what it will involve. Please take time to read the following information carefully before deciding to take part. You may discuss the study with others if you wish. Please contact us if there is anything that is not clear or if you would like more information. Thank you for taking the time to read this.

About the research

- **Who will conduct the research?**

  Primary Investigators

Name: Omar Alghamdi and Dr. Sarah Clinch
Address: Department of Computer Science,
University of Manchester, Oxford Rd,
Manchester. M13 9PL.
Email: omar.alghamdi@postgrad.manchester.ac.uk and sarah.clinch@manchester.ac.uk

- **What is the purpose of the research?**

  During programming, programmers may often seek help from a wide range of websites. This research will investigate programmers' interactions with websites during their programming, and whether such interactions alter the programmer's cognition. The research will also investigate the possible impacts on the code produced.

- **Will the outcomes of the research be published?**

  The outcome of the research will be published as an academic paper in a PhD thesis and/or at academic conferences.

- **Who has reviewed the research project?**

  The research has been reviewed by Department of Computer Science Ethics Committee.

What would my involvement be?

- **What would I be asked to do if I took part?**

  You will answer a set of questions discussing how you find help when you program. You will be asked to indicate which websites you use and whether you find them helpful. You will also be asked about the way that you remember and use code that you find online. Demographic data will be collected during the interview.

Audio recording is essential to participation in the study. You should feel comfortable with the recording process at all times, and are free to stop or pause recording at any time.

We anticipate that participation will take no more than forty five minutes of your time.

Benefits and Risks

There will be no perceived risks from taking part in the study and this will have no impact on your grade or your reputation with the University or any of the teaching staff. Your data will be stored anonymously with no links made between your data and your personal identity.

- **Will I be compensated for taking part?**

  You will be compensate with a £10 Amazon voucher.

- **What happens if I do not want to take part or if I change my mind?**

  It is up to you to decide whether or not to take part. If you do decide to take part, you will be given this information sheet to keep and will be asked to sign a consent form. If you choose to take part, you are still free to withdraw at any time without giving a reason and without detriment to yourself. However, you have three working days to inform the researcher that you wish to withdraw from the study. After three working days, we will anonymize your file and send it for transcription. It will not be possible to remove your data from the project once it has been anonymised as we will not be able to identify your specific data. This does not affect your data protection rights.

  If you decide not to take part, you do not need to do anything further.

Data Protection and Confidentiality

- **What information will you collect about me?**

  To participate in this research project, we will need to collect demographic information. Specifically, we will need to collect: your age, your gender your, course name (if applicable).

  The consent form will ask for your name and signature, which will remain only on the consent form.

  In term of the audio recording, the recordings will consist of voice only, and obtained during the interview session.

- **Under what legal basis are you collecting this information?**

  We are collecting and storing personal identifiable information in accordance with data protection law which protects your rights. These state that we must have a legal basis (specific reason) for collecting your data. For this study, the specific reason is that it is “a public interest task” and “a process necessary for research purposes”.

- **What are my rights in relation to the information you will collect about me?**

  You have a number of rights under data protection law regarding your personal information.

  If you would like to know more about your different rights or the way we use your personal information to ensure we follow the law, please consult our Privacy Notice for Research. (http://documents.manchester.ac.uk/display.aspx?DocID=37095)

- **Will my participation in the study be confidential and my personal identifiable information be protected?**

  Only the research team at the University of Manchester will have access to your information, which will be anonymised. Any identifying information will be removed and replaced with a random ID number. Your consent form will be retained for five years inside the University encrypted storage.

  In accordance with data protection law, the University of Manchester is the data controller for this project. This means that we are responsible for making sure your personal information is kept secure, confidential and used only in the way you have been told it will be used. All researchers are trained with this in mind, and your data will be looked after in the following way:

  - Your data will be anonymous. Only non-identified demographic will be captured.
  - The data will be stored on an encrypted device and will be backed up to a secure store
  - The interview will be held and recorded at the Kilburn building. Then, the audio files will be transferred to the University research data storage (RDS). The audio files will be shared with external transcriber.
  - The data will be shared for research purposes, but you will not be identifiable from the data
  - The data will be used in future research studies
  - The data will be stored no more than five years.
  - The data will not be transferred outside the EU.
  - The data will not be shared with any other organisation.
  - The research will not keep contact details for use in future studies
  - The recordings will be used to create transcripts by a third party transcriber, who is a University of Manchester approved service.
  - The audio files will be deleted after transcribing them.

  Please also note that individuals from The University of Manchester or regulatory authorities may need to look at the data collected for this study to make sure the project is being carried out as planned. This may involve looking at identifiable data. All individuals involved in auditing and monitoring the study will have a strict duty of confidentiality to you as a research participant.

What if I have a complaint?

- **Contact details for complaints**

If you have a complaint that you wish to direct to members of the research team, please contact:

Dr. Sarah Clinch
Department of Computer Science, The University of Manchester
Oxford Road, Manchester. M13 9PL.
0161 275190 or sarah.clinch@manchester.ac.uk

If you wish to make a formal complaint to someone independent of the research team or if you are not satisfied with the response you have gained from the researchers in the first instance then please contact

The Research Governance and Integrity Officer, Research Office, Christie Building, The University of Manchester, Oxford Road, Manchester, M13 9PL, by emailing: research.complaints@manchester.ac.uk or by telephoning 0161 275 2674.

If you wish to contact us about your data protection rights, please email dataprotection@manchester.ac.uk or write to The Information Governance Office, Christie Building, The University of Manchester, Oxford Road, M13 9PL at the University and we will guide you through the process of exercising your rights.

You also have a right to complain to the Information Commissioner's Office

(https://ico.org.uk/make-a-complaint/) about complaints relating to your personal identifiable

information Tel 0303 123 1113

Contact Details

If you have any queries about the study or if you are interested in taking part then please contact the researcher:

Omar Alghamdi
Department of Computer Science, The University of Manchester
Oxford Road, Manchester. M13 9PL.
omar.alghamdi@postgrad.manchester.ac.uk

Participant Consent Forms

The role of the Internet on Learning to Program

Interview Study

If you are happy to participate please complete and sign the consent form below

	Activities	Initials
1	I confirm that I have read the attached information sheet (**Version 1.1, Date 15/10/2019**) for the above study and have had the opportunity to consider the information and ask questions and had these answered satisfactorily.	
2	I understand that my participation in the study is voluntary and that I am free to withdraw at any time without giving a reason and without detriment to myself. I understand that it will not be possible to remove my data from the project once it has been anonymised and forms part of the data set. I agree to take part on this basis.	
3	I agree to the interviews being audio recorded.	
4	The study will use an external transcriber to transfer the audio files to text files. I am happy to share my data with a third-part transcriber	
5	I agree that any anonymised data collected may be used in teaching or publications (e.g. including academic journals)	
6	I understand that data collected during the study may be looked at by individuals from The University of Manchester or regulatory authorities, where it is relevant to my taking part in this research. I give permission for these individuals to have access to my data.	
7	I agree to take part in this study.	

Data Protection

The personal information we collect and use to conduct this research will be processed in accordance with data protection law as explained in the Participant Information Sheet and the Privacy Notice for Research Participants.
(http://documents.manchester.ac.uk/display.aspx?DocID=37095)

_________________________ _________________________ ______________
Name of Participant Signature Date

_________________________	_________________________	_____________
Name of the person taking consent	Signature	Date

1 copy for the participant, 1 copy for the research team (original)

A.3 Thematic analysis theme tables

Table A.1: Interview coding table (0.0 Overview of resources used)

	0.0 Overview of resources used
P1	
P2	In Stack Overflow there'll be people who have used those libraries for ages, and they will have the best way of doing something in a library.
P3	
P4	
P5	I often [. . .] use GitHub and look at projects from other people, so example projects that maybe go towards the same direction, and I look at how they solve that issue and if that might be applicable for my problem as well.
P6	Sometimes if I've got a big task ahead of me and I don't know any, of it or if I'm completely unfamiliar with the language, I buy a book on it, just to try and follow that through. I've got lots of iterations of it, I always rely on what I have saved above what I actually remember.
P7	
P8	[I] usually just Google and often the first thing that comes up is Stack Overflow but I don't generally search Stack Overflow specifically. It's just what comes up on Google. If I want a bit more detail, I guess documentation.
P9	If I have any problem that I can't solve, I'm using the Stack Overflow.
P10	I think it is quite useful to get help from an actual human because you can explain your question even better, and they can obviously interpret it a bit better than someone on Stack Overflow. And they also will get to kind of speak you through it and show you the kind of step-by-step. Stack Overflow is very, very good [. . .] I would say Stack Overflow is the best resource for getting help in programming in general. [. . .] I think Stack Overflow is definitely the best in my opinion. Well, it's what I use the most. I think it is very popular with a lot of people as well.

P11	It’s quite nice on Stack Overflow how you’ll get lots of different answers to the same question. Stack Overflow is good [. . .] if you just want to see the code [. . .] whereas GeeksforGeeks it will give you the code but also actually it kind of explained what they’ve done, why they’ve done it, time complexities of the code, so it is quite nice having that much more detail.
P12	
P13	
P14	Stack Overflow has its problems, but we would be lost without it, you have to have it, there’s no two ways about it.
P15	
P16	
P17	When you’ve actually got something a bit new [. . .] then you can keep dipping into it, even on a daily basis.
P18	

Table A.2: Interview coding table (1.0 How and why websites are used)

	1.0 How and why websites are used		
	1.1 Websites as a substitute for memory	**1.2 Finding ready-made solutions**	**1.3 Understanding the code**
P1			
P2		I just copy and pasted blocks of code until it worked so that wasn’t great but it did work in the end but I still have no idea what it does.	I mean, it helps if you like try and understand but if you need to get something done and the code’s there that does that, then take it.
P3			
P4	I use my memory and the information it is increasing so [. . .], so I have to use the website less now.		

P5	When I start a task I base it on my previous knowledge, on my previous experience [. . . when I] make the code very messy or very ambiguous or there’s a major issue that I can’t resolve, that’s the point where I go to websites and try to look at solutions.		
P6		I copied the code and then I ran it under their purpose. I changed out the bits I didn’t need, swapped things round. [. . .] So, I just ended up stripping away all the code that was not relevant to mine and then changing the parameters in the function.	

P7	I think I think I just don't have enough experience like to remember stuff yet. I mean, I don't look for every single thing but most of the times I have to look	[. . .] for the second coursework it was. . . you were supposed to make a game and I want to make Tetris in the beginning and then I've realised that to make that I would have to actually copy some of the stuff from online and I just didn't want to do that so I just changed and I'm not going to do Tetris anymore. So I just try to make it as much as mine as possible. It's really basic but I get really confused sometimes with for and while loop and then I keep. . . like every time I have to use it I just have to like look at it again and make sure which one I need to use.	Sometimes it gets a bit annoying because some of the things that they suggest can be really difficult to understand and then I feel really overwhelmed about it.

P8	I’ll Google if I’ve forgotten how to do a specific thing like how this function works or what is the syntax of this? [. . .] I mean it’s more just remembering syntax of things that I don’t use very often or remembering like what’s the specific way of doing this thing that I did two months ago that I would definitely Google because I can’t remember. I try to remember it first just because it’s tedious to have to go over and actually Google it so I might try it from memory, see if it works. If it doesn’t, I’ll Google it.		

P9	[…] when I'm doing the task I want the task to be quite challenging for me so this means that I need to use actually Stack Overflow to learn or any courses to learn the things that I don't know to accomplish the task.		I'm understanding it like the cut code of Stack Overflow is just mostly either too general for me or it is just on random data so I need to understand it first to actually implement it. But still I bear in mind that the actual Googling things could be really beneficial for me in the way that I can learn something, I can reinforce my knowledge or I can revise it.

P10	Because I'm not very experienced in it at all [. . .] I will always look it up.	So, to use their code I had to do string dot value of, and then put it in a char array and blah, blah, blah, and do all this rubbish just to be able to shoehorn that code into my system, where in fact it probably really wasn't efficient because I'd just gone to all the trouble to have to change the types and mess around with all that. I'm not very experienced in it at all, really, I've only started it at the start of this year, I will always look it up. I've probably been guilty of it in the past when I'm just trying to rush to get an assignment done, and [. . .] I don't care, even if it not works, but I just want to send it off and I don't want [to be] late. I don't think there's much value in just taking someone else's code and putting it in your system.	If I quickly want to confirm something that I think I already know, or to kind of search for things that I just literally don't understand. For example, errors that I might see, I can't interpret what that means and there's not really that amount of me staring at an error that's going to make me understand it. It's just the kind of thing you have search up, but I would say I use my memory more so than online resources.

P11	I would say my memory isn't that great but when it comes to coding you don't need, I would say, a fantastic memory [. . .] because as long as you can understand the concept, you can always implement it in code.	Usually it won't be for like an entire block of code. It will just be for a specific function [. . .] from there I can see if that specific function works.	I feel that if I copy and paste it, I don't necessarily always understand it.
P12			
P13		Probably before it didn't used to matter much, but now there's so many website which has come up, now it's becoming difficult for programmers, even mostly when people are work in a company apparently they depend on these websites as well, to find out solution for their problem as well.	I try to copy their snippets of code, try it in my program, see how the results vary and see what exactly is happening in their code.
P14		In my company there's no code reviews; so I could get code from somewhere and just say it's mine. Nobody knows any better.	
P15			

P16	It goes away somewhere in my brain and next time, If I did it a few times, I'd remember it. If I didn't, I'd just keep on looking at my code and then eventually... cause I'd only get it off the website the once and then it'd be in my code, and cause I know I'd done it, that's where I'd refer to it again. That's exactly how we work. [...] I know as you get more experienced in a language, you know the base of the stuff. Then it tends to be either something that you've not done before, or something that you've not done for ages and you've forgotten, or something that needs to be found out how to do because nobody's really done it before.		
P17	The websites are just there, on tap, it is too easy to go there.	Copy and paste is quick and easy.	
P18			

Table A.3: Interview coding table (2.0 Effects of website use on memory and code)

	2.0 Effects of website use on memory and code		
	2.1 Impact on memory	**2.2 Impact on learning**	**2.2 Impact on code**
P1			Sometimes it's contradictory; someone would write something, someone would write something else. If the stakes are low, yes, I could test it and nothing would happen.

P2	I'll often remember a Stack Overflow post if I've used it before [. . .] saving the location to memory, and I always remember that. Moving average in NumPy, I've searched for that at least five times [. . .] I can never remember the correct way of doing it so I pretty much always just look up the blog posts for that or the Stack Overflow post.	It helps if you like to try and understand, but if you need to get something done and the code does that, then take it. Someone has written it and put it online. You can take that and use it as your own so.	[Q: Did you face any difficulty when you copy and paste the codes from the websites?] Yes, so especially with different Python versions, so back at GCSE I didn't understand there was Python 2 and Python 3 so I just copied Python 2 code with no print, brackets around print and stuff like that and I was like, why doesn't it work? What do you mean, long isn't a function? Just, like, yes, so I didn't understand that then. So I know that the code that I copied [. . .] especially when I look back at it [. . .] it was like really, really bad. Like I had like repeated if statements in a row. A lot of the code on Stack Overflow for doing sockets does not work, so that took a while. They just had like inputting a get parameter and putting it on the webpage. And I was like, wait, you can put a script tag in the get parameter and it works.

P2 (Cont'd)			Creative Commons is a fairly open licence, isn't it, so I'm pretty sure you're allowed to just copy it. But there's presumably restrictions on where you can use the code. So none of the code I write is for commercial purposes so I'm presumably exempt from that.
P3	Python has a lot of built-in functions, and I've used lots of them and I still don't remember lots of them. [Q: You are not sure about the availability of the information in your memory?] R: Yeah. In some cases, I would be able to fix the problem again on my own but I'm not sure I always would.	But with certain things, you do have to read a lot more, and that can be a bit of a deterrent [. . .] well it means that it will take me longer and it can slow things down which does deter from certain things. I think it is quite important because it changes completely how you code in a way but it's more of a long term thing. [. . .] I might read that and then I'll understand it and then I might permanently change the way that I program. If I have a problem in a specific thing that I am using, then a specific solution won't necessarily implant in my mind because I don't understand necessarily why it works.	If you are just dealing with every issue as you come to it, it can make code that's a bitstream of consciousness [. . .] just random snippets [. . .] it can make the code a lot less elegant.

P4	If you use it excessively you will not improve your memory, you will keep using it a lot all the time.		I do not know what code quality means. [Q: How do you feel about that copy and paste as a general problem?] Maybe it is good to reduce time, but it is bad maybe to reduce your creativity and problem solving. Maybe if you think about it yourself you'll get a better answer for your problem than copying it. [Q: You mentioned that you didn't like the code that you last used from the website. Why is that?] The output of it wasn't what I wanted to do. Sometimes they post answers for Python two and we usually use Python three.

P5	I think it could have a negative impact if you move to the internet right away.	When I started out with Python I have sometimes found myself to copy or adopt a solution that I didn’t really necessarily understand so it worked and I knew to a certain degree why it worked but I couldn’t fiddle with the code too much to adapt it to my issues. I just had to go with what it was but once I got better at using Python, I also learned to understand code snippets more properly so when I adopt code snippets, I can actually mould them so that they suit my issues more properly. I don’t think it’s a negative feeling when you don’t really understand it.	There is a lot of buggy code snippets out there that are presented as answers. My programs never were too much security based.

P6	I just forget how to add things to a list, and then it's really embarrassing when I look it up, and it is like, oh, append.		Sometimes the code wouldn't work and then they wouldn't run. I like the full tutorial ones because I know they're going to work at the end. Whereas on Stack Exchange, you're not sure it's always going to work. I'm sure I take a lot online that's code redundancy, but I'm sure a lot in my own code was that too.
P7		So probably the ones I'm using, I'm pretty sure that those are not the optimum solutions, those are not the best ones but like I could definitely find better ones but I just wouldn't understand that.	It's really basic but I get really confused sometimes with for and while loop [. . .] and then I keep like every time I have to use it I just have to like look at it again and make sure which one I need to use. I feel like it's a bit of like lying to yourself if you are copy and pasting it because it's not you that's actually making it.
P8			

P9	I: Do you think the availability of the information on the websites will be a problem for you and affect your experience in programming in the long term? R: Yes, I think so. I'm trying to actually understand and remember things because I think that this is just for me to get better in programming and better in technologies that I am using. [. . .] Googling things could be really beneficial for me in the way that I can learn something, I can reinforce my knowledge or I can revise it.	My extreme use of Stack Overflow could be actually a downside in my career [. . .] I mean that I'm used to really easy approach to it so that I'm not used to reading documentation, learning some more to actually search in documentation but I am just writing a question and this question is on the website.	I'm understanding it like the cut code of Stack Overflow is just mostly either too general for me or it is just on random data so I need to understand it first to actually implement it. Googling things could be really beneficial for me in the way that I can learn something, reinforce my knowledge, or revise it. There were problems that the actual variables had quite weird scopes, there were problems that using the MVVM design pattern that the developers were putting the logic into the view where it shouldn't be, which is quite a huge mistake.

P10		I feel like if you don't get taught a language formerly, a lot of it is just trying to put building blocks together from resources online. I do think that it [copying and pasting online code] kind of hinders people's understanding a little bit [. . .] in certain situations it can be bad because it can stop the learning aspect of it and just force the more I just want to get a grade aspect.	To use their code I had to do string dot `valueOf`, and then put it in a char array and blah, blah, blah, and do all this rubbish just to be able to shoehorn that code into my system, where in fact it probably really wasn't efficient because I'd just gone to all the trouble to have to change the types and mess around with all that. You assume it's right but maybe it isn't and it would probably take you longer to try and work out why that isn't right than to just write your own code in the first place. Sometimes the code fragment will look like it works and you'll try and dry run it and maybe you'll skip over the little thing that makes it not work [. . .] put in my code and then it doesn't work. Yes, a lot of the time I feel like people on there [the web] have really, really bad code practices. I don't know if it's just because they maybe haven't been formally taught or maybe if that's just their style [. . .] I just don't necessarily think that's the clearest way to write things.

P10 (Cont'd)			I found this code fragment and it literally made no sense. I can't even stress. All the variable names were a letter, it had no comments.
P11		I do always try to like avoid just directly copy and pasting.	Sometimes the best or most voted answer isn't necessarily the one I'm looking for [...] sometimes you do have to look slightly harder for what you are actually looking for.
P12	I was working on more than one language, it's not necessary that you will remember syntax always, because you get confused between, what is the syntax in Python and in C++ [...] you can just use sites like W3school.	Unless you understand your code, then if someone tells you that there is an error in this part, then you have to be dependent on the other person, and find their error code, it's completely illogical to do that.	

P13	[Q: Do you think your memory is capable to depend on when programming?] Not really, because I think now the situation has become a bit worse, because anything you go to find, you find out online. So you're more inclined to go online and get the stuff out.	Are you learning the stuff? Because it's not copy paste at the end, you need to know what you did at the end, because that's what the marker ask you.	What I look for is which one fits to my problem, and I try to implement that one. I don't really go through which code quality issues. They [online code] are not commented well. You can't just take out some stuff on random websites [. . .] you don't even know if it's trusted source, what source they come from. I try to look out for, like, more than code quality issues, I try and find out if it's in the required programming language [. . .] check if it's best way.
P14	I am nearly 50 years old. Now when I was a young boy, then my memory was fantastic, I could remember things, no problem. Now [. . .] I don't tend to remember things anymore.		

P15			I mean my code quality was less good when I was a newer developer obviously. I don't think that could be blamed just on Stack Overflow. I think it's to do with just how deeply you understand things you're doing in general .
P16	It [online information] goes away somewhere in my brain, [. . .] and If I did it a few times, I'd remember it. If I didn't, I'd just keep on looking at my code. There's a lot in the memory that you need. I wouldn't say that the internet takes that away from you because [. . .] personally, you only use it that once to get it and get it working and then it's yours, you claim it.		One of the problems I've come across is that the code sort of works but it's volatile and it doesn't work all the time, so you don't know there's a problem. So that's to do with versioning, a lot of the problems, and what's on the internet is outdated but you don't know because it still sort of works. So that's a big problem that I came across.
P17			

P18			On Stack Overflow, you have some developers, inexperienced developers, or junior developers, who will put answers on there which come with a weak data structure. Having done this myself for many years, I can spot weak answers and strong answers.

Appendix B

Supplement materials for Chapter 4

B.1 Survey questions

B.1.1 Demographic questions

Demographic Questions:

Q1: In which year were you born? [*Students and Professionals-mandatory*]

Options from 1920 till 2003

Q2: What is your gender? [*Students and Professionals-mandatory*]

- Male.
- Female.
- Non-binary/third gender.
- Prefer not to disclose.
- Prefer not to say.

Q3: What is your current year of study? [*Students-mandatory*]

- Year 1 – UG.
- Year 2 – UG.
- Year 3 – UG.
- Year 4 - Integrated MSc.

Q4: Which university are you studying at? [*Students-mandatory*]

List of UK universities.

Q5: Which of the following best describes your programming experience before starting your degree program? [*Students and Professionals-mandatory*]

- Hobby programmer.
- Completed programming internship.
- Casual employment in a programming-related role (including voluntary or charity work).
- Full time employment in a programming-related role.
- Prior study of computer science for a qualification (e.g. GCSE / A-Level).
- No prior programming experience.

Q6: When did you write your first line of code (in any programming language)? [*Students and Professionals-mandatory*]

- Less than 1 year ago.
- At least 1, but less than 2 years ago.
- At least 2, but less than 3 years ago.
- At least 3, but less than 5 years ago.
- 5-9 years ago.
- 10-14 years ago.

- 15-19 years ago.
- 20+ years ago.

Q7: In which programming language would you consider yourself to be most proficient? [*Students and Professionals-mandatory*]

Open text.

Q8: When did you write your first line of code in the language with which you are most proficient? [*Students and Professionals-mandatory*]

- Less than 1 year ago.
- At least 1, but less than 2 years ago.
- At least 2, but less than 3 years ago.
- At least 3, but less than 5 years ago.
- 5-9 years ago.
- 10-14 years ago.
- 15-19 years ago.
- 20+ years ago.

Q9: How would you describe your competency in the language with which you are most proficient? [*Students and Professionals-mandatory*]

- Beginner.
- Between Beginner and Intermediate.
- Intermediate.
- Between Intermediate and Expert.
- Expert.

Q10: Enter your email [*Students and Professionals-Optional]*.

Open text.

B.1.2 Full questions

Survey Questions:

Resources Use

Q11: Which of the following resources do you use when programming? [*4-point Likert scale: frequently, occasionally, rarely, never*]

R1. Books.
R2. Websites.
R3. Friends / Colleagues.
R4. Course tutors.
R5. Existing codebases (code you have written or worked with in the past).

Contingency questions survey: proceed to Q12-13 just when the answer to Q11b was frequently or occasionally and skip if else.

Q12: Which of the following websites do you use when programming? [*4-point Likert scale: frequently, occasionally, rarely, never*]

R6. GitHub.
R7. Quora.
R8. Reddit.
R9. Stack Overflow.
R10. Programming language documentation (e.g. https://docs.python.org/, https://docs.oracle.com/en/java/).
R11. Tutorial Websites (e.g. https://www.w3schools.com, https://www.geeksforgeeks.org).

Q13: How do you access information from these websites when programming? [*4-point Likert scale: frequently, occasionally, rarely, never*]

R12. I visit the site directly.
R13. I use a search engine with the intention of finding content from a specific website.
R14. I use a search engine and click whichever results look most relevant.

Use and perception of Stack Overflow

Contingency questions survey: proceed to S1 just when the answer to Q12d was frequently or occasionally and skip if else.

S1. Comparing to three years ago, I used Stack Overflow [*3-point Likert scale: less than I used to/ More less than I used to/ About the same*].

Q14: To what extent do you agree with the following statements about the Stack Overflow website? [*5-point Likert scale: strongly agree, agree, neutral, disagree, strongly disagree*]

S2. When searching for programming-related concepts on the web, the Stack Overflow website is the most dominant result.

S3. I cannot program without the Stack Overflow website.
S4. I can find what I am looking for on Stack Overflow.
S5. I find it hard to tell if the question and/or answers on Stack Overflow are relevant to my programming tasks.
S6. I prefer to use the most upvoted solutions on the Stack Overflow website.
S7. I take the author's reputation into account when deciding how likely the answer will help.
S8. I can identify poor quality solutions on Stack Overflow because they will have been down voted.
S9. Having multiple different solutions, and others' comments on those solutions, is very helpful to me.
S10. I find that different answers and/or comments conflict with each other.
S11. I am wary when reading unaccepted answers on Stack Overflow website.

Use and perception of online code snippets

C1. How often do you copy and paste a source code snippet from the web? [*5-point Likert scale: I do this most days / I do this at least three times a week / I do this at least weekly / I do this at least once a month / I do this rarely or never*].

Q15: To what extent do you agree with the following statements about the code snippets you find on the web? [*5-point Likert scale: strongly agree, agree, neutral, disagree, strongly disagree*]

C2. I trust code snippets found on the web.
C3. I copy code snippets to make up for gaps in my experience / knowledge.
C4. I copy code snippets only if I fully understand their contents.
C5. I copy code snippets only if they are consistent with my own code quality standards.
C6. Copying and pasting code hinders programmers' understanding and learning.
C7. Copying and pasting code from websites reduces code quality.
C8. The majority of online code snippets are of good quality.

Q16: To what extent have you found the following to be present in code snippets on the web? [*3-point Likert scale: I have encountered this problem myself / I am aware that this is a problem but have not encountered it myself / I am unaware of or don't think this is a problem*]

C9. Code that does not work with the current version of a language or library.
C10. Security issues.
C11. License violation issues.
C12. Code that does not compile or does not run.
C13. Code with extraneous output (e.g. unwanted prints).
C14. Code with incorrect output (e.g. 5 + 1 = 7).
C15. Code that is difficult or impossible to incorporate into an existing project.
C16. Undocumented code.
C17. Redundant code.
C18. Insufficient or incomplete code.

C19. Inefficient or overly complex code (the problem could be solved much more simply another way).

Memory questions

Q17: To what extent do you agree with the following statements about human memory when programming? [*5-point Likert scale: strongly agree, agree, neutral, disagree, strongly disagree*]

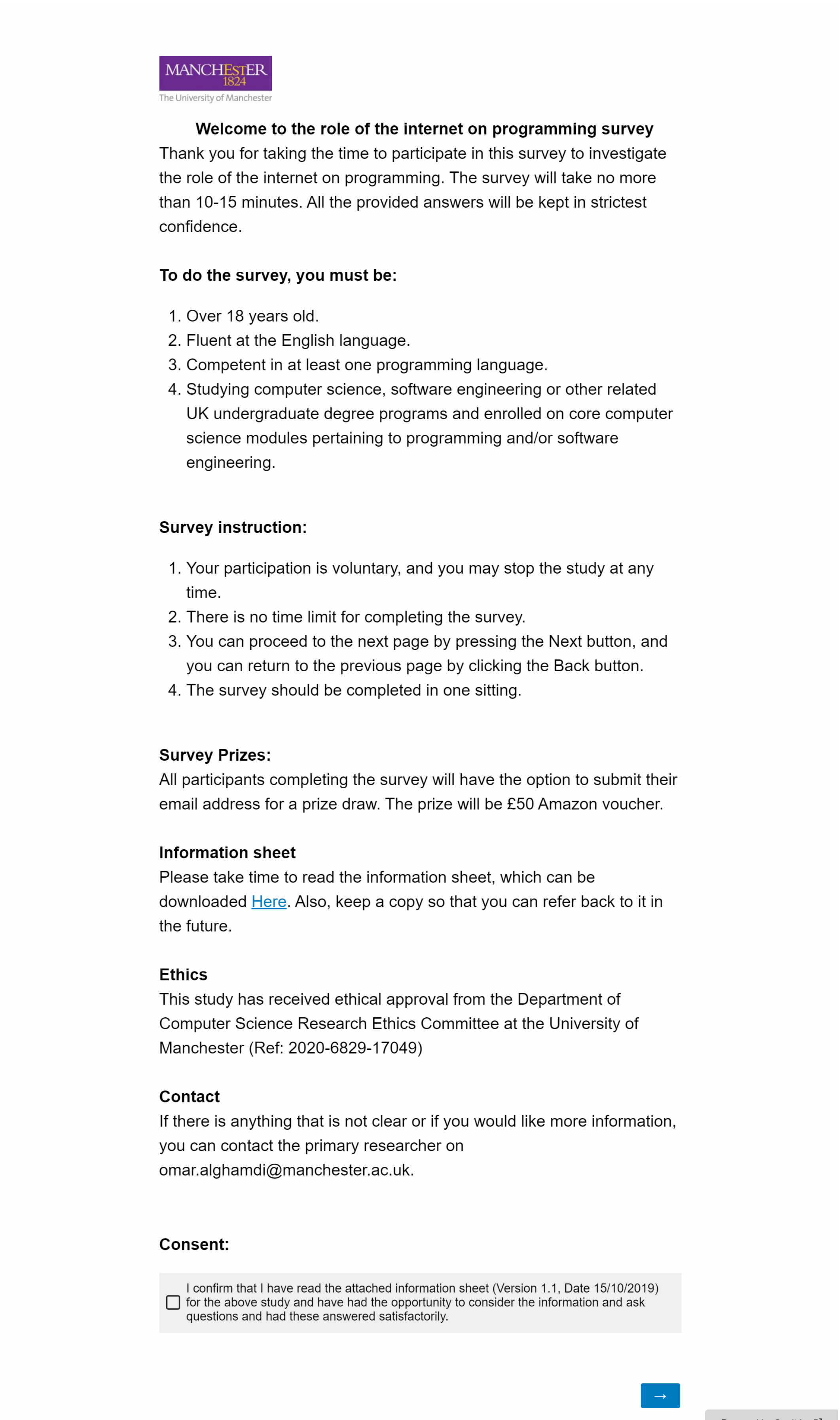
M1. Having a good memory is critical to successful programming.
M2. I have a good memory for programming concepts and syntax.
M3. When solving a new programming problem, I am able to remember similar problems I have solved in the past.
M4. It is faster to remember programming-related information than it is to look it up.
M5. I can program non-trivial applications using my memory alone.
M6. Being unable to remember programming concepts bothers me.

Q18: To what extent do you agree with the following statements about remembering content you find on the web? [*5-point Likert scale: strongly agree, agree, neutral, disagree, strongly disagree*]

M7. There is no need to try and remember programming concepts because websites are always available.
M8. If I have previously solved a problem using the web, I will be able to solve the same problem in the future without looking up the information again.
M9. If I have previously solved a problem using the web, I will remember where to find the information needed to solve the problem next time.
M10. Looking at programming content on the web confirms what I already know or reminds me of something I had forgotten.
M11. The more I use programming content on the web, the less I remember.
M12. Programming content on the web is for reference not learning.

B.2 Versions of the first page of the survey in Qualtrics

B.2.1 Students



MANCHESTER 1824
The University of Manchester

Welcome to the role of the internet on programming survey

Thank you for taking the time to participate in this survey to investigate the role of the internet on programming. The survey will take no more than 10-15 minutes. All the provided answers will be kept in strictest confidence.

To do the survey, you must be:

1. Over 18 years old.
2. Fluent at the English language.
3. Competent in at least one programming language.
4. Studying computer science, software engineering or other related UK undergraduate degree programs and enrolled on core computer science modules pertaining to programming and/or software engineering.

Survey instruction:

1. Your participation is voluntary, and you may stop the study at any time.
2. There is no time limit for completing the survey.
3. You can proceed to the next page by pressing the Next button, and you can return to the previous page by clicking the Back button.
4. The survey should be completed in one sitting.

Survey Prizes:
All participants completing the survey will have the option to submit their email address for a prize draw. The prize will be £50 Amazon voucher.

Information sheet
Please take time to read the information sheet, which can be downloaded Here. Also, keep a copy so that you can refer back to it in the future.

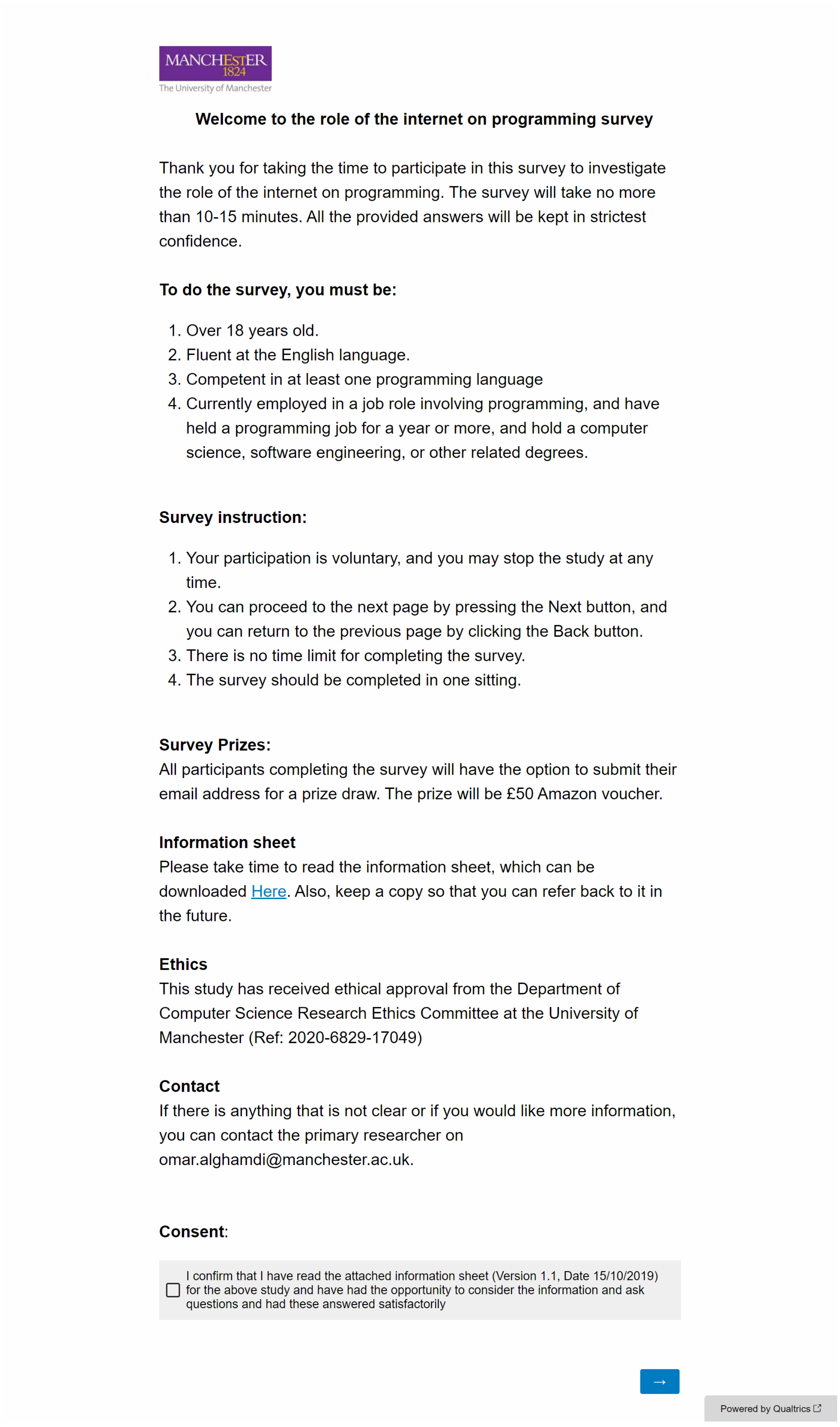
Ethics
This study has received ethical approval from the Department of Computer Science Research Ethics Committee at the University of Manchester (Ref: 2020-6829-17049)

Contact
If there is anything that is not clear or if you would like more information, you can contact the primary researcher on omar.alghamdi@manchester.ac.uk.

Consent:

☐ I confirm that I have read the attached information sheet (Version 1.1, Date 15/10/2019) for the above study and have had the opportunity to consider the information and ask questions and had these answered satisfactorily.

B.2.2 Professionals



MANCHESTER
1824
The University of Manchester

Welcome to the role of the internet on programming survey

Thank you for taking the time to participate in this survey to investigate the role of the internet on programming. The survey will take no more than 10-15 minutes. All the provided answers will be kept in strictest confidence.

To do the survey, you must be:

1. Over 18 years old.
2. Fluent at the English language.
3. Competent in at least one programming language
4. Currently employed in a job role involving programming, and have held a programming job for a year or more, and hold a computer science, software engineering, or other related degrees.

Survey instruction:

1. Your participation is voluntary, and you may stop the study at any time.
2. You can proceed to the next page by pressing the Next button, and you can return to the previous page by clicking the Back button.
3. There is no time limit for completing the survey.
4. The survey should be completed in one sitting.

Survey Prizes:
All participants completing the survey will have the option to submit their email address for a prize draw. The prize will be £50 Amazon voucher.

Information sheet
Please take time to read the information sheet, which can be downloaded Here. Also, keep a copy so that you can refer back to it in the future.

Ethics
This study has received ethical approval from the Department of Computer Science Research Ethics Committee at the University of Manchester (Ref: 2020-6829-17049)

Contact
If there is anything that is not clear or if you would like more information, you can contact the primary researcher on omar.alghamdi@manchester.ac.uk.

Consent:

☐ I confirm that I have read the attached information sheet (Version 1.1, Date 15/10/2019) for the above study and have had the opportunity to consider the information and ask questions and had these answered satisfactorily

→

Powered by Qualtrics

B.3 Participants information sheet and consent form

Participant Information Sheet

The role of the Internet on Programming

This is a research project undertaken at the Department of Computer Science at the University of Manchester. Before you decide whether to take part, it is important for you to understand why the research is being conducted and what it will involve. Please take time to read the following information carefully before deciding to take part. You may discuss the study with others if you wish. Please contact us if there is anything that is not clear or if you would like more information. Thank you for taking the time to read this.

About the research

- **Who will conduct the research?**

  Primary Investigators

  Name: Omar Alghamdi and Dr. Sarah Clinch
  Address: Department of Computer Science,
  University of Manchester, Oxford Rd,
  Manchester. M13 9PL.
  Email: omar.alghamdi@postgrad.manchester.ac.uk and sarah.clinch@manchester.ac.uk

- **What is the purpose of the research?**

  During programming, programmers may often seek help from a wide range of websites. This research will investigate programmers' interactions with websites during their programming, and whether such interactions alter the programmer's cognition. The research will also investigate the possible impacts on the code produced

- **Will the outcomes of the research be published?**

  The outcome of the research will be published as an academic paper in a PhD thesis and/or at academic conferences.

- **Who has reviewed the research project?**

  The research has been reviewed by Department of Computer Science Ethics Committee.

What would my involvement be?

- **What would I be asked to do if I took part?**

  You will answer a set of questions discussing how you find help when you program. You will be asked to indicate which websites you use and whether you find them helpful. You will also be asked about the way that you remember and use code that you find online. Demographic data will be collected during the questionnaire.

We anticipate that participation will take no more than 10-15 minutes of your time

Benefits and Risks

There will be no perceived risks from taking part in the study and this will have no impact on your grade or your reputation with the University or any of the teaching staff. Your data will be stored anonymously with no links made between your data and your personal identity. You have the option to provide your email for the prize draw. However, if you choose not to provide your email, you will not be eligible to enter the prize draw. The email address will be stored in a secure place and will be removed after completing the prize draw.

- **Will I be compensated for taking part?**

  Students: The prize draw will take place in July at the end of each academic year the study was conducted. The study will give out a total of 50 pound prize as Amazon vouchers. There will be three vouchers: one will be 30 pounds, and two will be 10 pounds.

  Professional programmers: 50 pounds as one voucher per a year.

- **What happens if I do not want to take part or if I change my mind?**

  It is up to you to decide whether or not to take part. If you do decide to take part, you will be given this information sheet to keep and will be asked to sign a consent form. If you choose to take part, you are still free to withdraw at any time without giving a reason and without detriment to yourself. You can withdraw from the study any time before submitting your responses and closing the survey window. After submitting your survey response, we will anonymize your file. It will not be possible to remove your data from the project once it has been anonymised as we will not be able to identify your specific data. This does not affect your data protection rights. If you enter your email for the prize draw and decided to withdraw from the study, you may need to contact the researcher for that

  If you decide not to take part, you do not need to do anything further.

Data Protection and Confidentiality

- **What information will you collect about me?**

  To participate in this research project, we will need to collect demographic information. Specifically, we will need to collect: your age, your gender your, course name (if applicable), and your email address.

  The consent form will ask for your name and signature, which will remain only on the consent form.

- **Under what legal basis are you collecting this information?**

  We are collecting and storing personal identifiable information in accordance with data protection law which protects your rights. These state that we must have a legal basis (specific reason) for collecting your data. For this study, the specific reason is that it is “a public interest task” and “a process necessary for research purposes”.

- **What are my rights in relation to the information you will collect about me?**

  You have a number of rights under data protection law regarding your personal information.

  If you would like to know more about your different rights or the way we use your personal information to ensure we follow the law, please consult our Privacy Notice for Research.

- **Will my participation in the study be confidential and my personal identifiable information be protected?**

  Only the research team at the University of Manchester will have access to your information, which will be anonymised. Any identifying information will be removed and replaced with a random ID number. Only the research team will have access to the key that links this ID number to your personal information. Your consent form will be retained for five years inside the University encrypted storage.

  In accordance with data protection law, the University of Manchester is the data controller for this project. This means that we are responsible for making sure your personal information is kept secure, confidential and used only in the way you have been told it will be used. All researchers are trained with this in mind, and your data will be looked after in the following way:

  - Your data will be anonymous. Only non-identified demographic will be captured.
  - The data will be stored on an encrypted device and will be backed up to a secure store
  - The data will be shared for research purposes, but you will not be identifiable from the data
  - The data will be used in future research studies
  - The data will be stored no more than five years.
  - The data will not be transferred outside the EU.
  - The data will not be shared with any other organisation.
  - The research will not keep contact details for use in future studies

  Please also note that individuals from The University of Manchester or regulatory authorities may need to look at the data collected for this study to make sure the project is being carried out as planned. This may involve looking at identifiable data. All individuals involved in auditing and monitoring the study will have a strict duty of confidentiality to you as a research participant.

What if I have a complaint?

- **Contact details for complaints**

  Dr. Sarah Clinch
  Department of Computer Science, The University of Manchester
  Oxford Road, Manchester. M13 9PL.

0161 275190 **or** sarah.clinch@manchester.ac.uk

If you wish to make a formal complaint to someone independent of the research team or if you are not satisfied with the response you have gained from the researchers in the first instance then please contact

The Research Governance and Integrity Officer, Research Office, Christie Building, The University of Manchester, Oxford Road, Manchester, M13 9PL, by emailing: research.complaints@manchester.ac.uk or by telephoning 0161 275 2674.

If you wish to contact us about your data protection rights, please email dataprotection@manchester.ac.uk or write to The Information Governance Office, Christie Building, The University of Manchester, Oxford Road, M13 9PL at the University and we will guide you through the process of exercising your rights.

You also have a right to complain to the Information Commissioner's Office about complaints relating to your personal identifiable information Tel 0303 123 1113

Contact Details

If you have any queries about the study or if you are interested in taking part then please contact the researcher(s)

Omar Alghamdi
Department of Computer Science, The University of Manchester
Oxford Road, Manchester. M13 9PL.
omar.alghamdi@manchester.ac.uk

Participant Consent Forms

The role of the Internet on Learning to Program

Questionnaire Study

If you are happy to participate please complete and sign the consent form below:

	Activities	Initials
1	I confirm that I have read the attached information sheet (**Version 1.1, Date 15/10/2019**) for the above study and have had the opportunity to consider the information and ask questions and had these answered satisfactorily.	
2	I understand that my participation in the study is voluntary and that I am free to withdraw at any time without giving a reason and without detriment to myself. I understand that it will not be possible to remove my data from the project once it has been anonymised and forms part of the data set. I agree to take part on this basis.	
3	I agree that any anonymised data collected may be used in teaching or publications (e.g. including academic journals)	
4	I understand that data collected during the study may be looked at by individuals from The University of Manchester or regulatory authorities, where it is relevant to my taking part in this research. I give permission for these individuals to have access to my data.	
5	I agree to take part in this study.	

Optional consent
Note: During the prize draw, the winners will be selected and contacted through the Emails. The Emails will be withdrawn after the prize draw completion. However, not accepting the following consent will withdraw you from the prize draw and study finding notifications.

6	I agree that the researchers may retain my contact details in order to enter a prize draw.	

Data Protection

The personal information we collect and use to conduct this research will be processed in accordance with data protection law as explained in the Participant Information Sheet and the Privacy Notice for Research Participants.

___________________________	___________________________	_______________
Name of Participant	Signature	Date
___________________________	___________________________	_______________
Name of the person taking consent	Signature	Date

1 copy for the participant, 1 copy for the research team (original)

B.4 Recruitment letters used to recruit participants

B.4.1 For instructors to recruit students:

Dear

Greeting,

My name is Omar Alghamdi, and I am a PhD student from the Department of Computer Science at the University of Manchester. My research is about the role of the internet on programming, and it will understand the role of websites on programming and investigate any impacts.

I am writing to you to ask your permission to distribute my survey to undergraduate students on the computer sciences at the University. The survey will take 10-15 minutes, and any CS undergraduate student can participate. As a 'Thank You' for completing this survey, students will be able to enter a prize draw to win £50 as Amazon vouchers.

Note: This study has received ethical approval from the University of Manchester’s Research Ethics [Committee 2019-6829-12032].

Please let me know if you have any questions

Thank you very much.

Sincerely,
Omar

B.4.2 For students:

B.4.2.1 Email Recruitment letter

Dear Students,

My name is *Omar Alghamdi*, a *PhD student* from the *Department* of *Computer Science* at the University of Manchester. I am writing to invite you to participate in my research study about *the role of the Internet in programming*.

You are *invited to answer few questions about how you program.* The questionnaire will take *10-15 minutes.* In recognition of the time spent, you will be able to *enter the prize draw*.

Participation is entirely voluntary. If you wish to participate in the study, you will have no positive or negative impact on either your studies or your relationship with the university/department.

If you'd like to participate, please visit the **link-** http://bit.ly/TheSurvey1

If you have any questions about the study, please email me at *omar.alghamdi@manchester.ac.uk*

Thank you very much.

Sincerely,

B.4.2.2 Poster

Are you studying computer science?

Researchers at the University of Manchester are looking for participants for an online survey about use of the web when programming.

Duration: 10-15 minutes

Reward: Prize draw entry

Complete the survey either by visiting the link:

bit.ly/TheSurvey1

OR

Scan the QR Code

SCAN ME

More info?

omar.alghamdi@manchester.ac.uk

MANCHESTER 1824

The University of Manchester

Participate In Research

This study has received ethical approval from the Department of Computer Science Research Ethics Committee at the University of Manchester (Ref: 2020-6829-17049)

B.4.2.3 For ”Monday Mail” students:

Dear Students,

Greetings,

My name is Omar Alghamdi, and I am a PhD student running a survey to understand the programmers use of the web and investigate any impact. The survey will take 10-15 minutes, and any CS undergraduate student can participate. As a 'Thank You' for completing this survey, you will be able to enter a prize draw. The survey link is: http://bit.ly/TheSurvey1

B.4.3 For professional programmers:

B.4.3.1 Email Recruitment letter

My name is Omar Alghamdi, and I am a PhD student running a survey to understand the programmers use of the web and investigate any impact. The survey will take 10-15 minutes, and any UK based programmer can participate. As a 'Thank You' for completing this survey, you will be able to enter a prize draw to win £50 as Amazon vouchers. The survey link is: http://bit.ly/TheSurvey2

B.4.3.2 Poster

Do you work as a programmer or software engineer?

Researchers at the University of Manchester are looking for participants for an online survey about use of the web when programming.

Duration: 10-15 minutes

Reward: Prize draw entry

Complete the survey either by visiting the link:

bit.ly/TheSurvey2

OR

Scan the QR Code

SCAN ME

More info?

omar.alghamdi@manchester.ac.uk

MANCHESTER 1824

The University of Manchester

Participate In Research

This study has received ethical approval from the Department of Computer Science Research Ethics Committee at the University of Manchester (Ref: 2020-6829-17049)

B.4.4 For social media:

Do you use websites while programming?

Can you spare 10 minutes to help us to understand programmers' use of the web?

Participants must be 18+ and fluent in English. To participate in our online survey, visit:

shorturl.at/iuFHU or email omar.alghamdi@manchester.ac.uk for more information.

#UniversityofManchester #research #SoftwareEngineering #programming

B.5 Full demographics for students and professionals

Question	Data
Number of participants	251 students.
Year of Birth (Age)	Mean: 1999 (21-22 years), Median: 2000 (20-21 years), Mode: 2000 (20-21 years), Std.: 2.41 years, IQR: 1999-2001, Range: 1982-2002 Not disclosed: n=14 (5.58%).
Gender	Male: 185 (73.71%), Female: 55 (21.91%), Non-binary: 6 (2.39%), Not disclosed: 5 (1.99%).
Year of study	1st: 83 (33.07%), 2nd: 66 (26.29%), 3rd: 88 (35.06%), 4th: 14 (5.58%).
Place of study	Swansea University: 64 (25.50%), Lancaster University: 51 (20.32%), Queen Mary, University of London: 45 (17.93%), University of Manchester: 43 (17.13%), University of Nottingham: 28 (11.16%), Newcastle University: 8 (3.19%), University of Warwick: 6 (2.39%), University of St Andrews: 4 (1.59%), Other UK University (2 institutions): 2 (0.80%).
Previous experience (multiple answers permitted: mean responses 1.46, median 1.00)	Prior study (e.g. GCSE / A-Level): 167 (66.53%), Hobby programmer: 123 (49.00%), No prior programming experience: 35 (13.94%), Casual/voluntary employment: 21 (8.37%), Internship: 17 (6.77%), Full time employment: 4 (1.59%).
First line of code in any language (in years)	< 1: 7 (2.79%), 1+, but < 2: 11 (4.38%), 2+, but < 3: 18 (7.17%), 3+, but < 5: 62 (24.70%), 5-9: 126 (50.2%), 10-14: 24 (9.56%), 15-19: 3 (1.20%) 20+: 0 (0.00%).
Most proficient programming language (Free text field, 21 participants listed > 1 language: overall mean responses 1.10)	C: 9 (3.59%), C#: 17 (6.77%), C++: 17 (6.77%), CSS: 3 (1.20%), HTML: 5 (1.99%), Java: 126 (50.20%), JavaScript: 17 (6.77%), Kotlin: 2 (0.80%), PHP: 6 (2.39%), Python: 66 (26.29%), Other (Android Studio, Go, Haskell, Lua, Objective C, Rust, Swift, TypeScript, Visual Basic): 9 (3.58%).
First line of code in proficient language (in years)	< 1: 40 (15.94%), 1+, but < 2: 33 (13.15%), 2+, but < 3: 59 (23.51%), 3+, but < 5: 62 (24.70%), 5-9: 52 (20.72%), 10-14: 5 (1.99%), 15-19: 0 (0.00%), 20+: 0 (0.00%).

Table continued on next page

Table B.1: Students demographics

Competency in proficient language	Beginner: 7 (2.79%), Beginner↔Intermediate: 33 (13.15%), Intermediate: 120 (47.81%), Intermediate↔Expert: 82 (32.67%), Expert: 9 (3.59%).

Question	**Data**
Number	25 professional programmers
Year of Birth (Age)	Mean: 1983 (37-38 years), Median: 1985 (35-36 years), Mode: 1988, 1997 (23-24, 32-33 years), Std.: 9.69 years, IQR: 1977-1990, Range: 1963-1998
Gender	Male: 20 (80%), Female: 4 (16%), Non-binary: 0 (0.00%), Not disclosed: 1 (4%)
Previous experience (multiple answers permitted: mean responses 1.38, median 1.00)	Casual/voluntary employment: 1 (4%), Full time employment: 19 (76%), Hobby programmer: 5 (20%), Internship: 2 (8%), Prior study (e.g. GCSE / A-Level): 6 (24%), No prior programming experience: 1 (4%),
First line of code in any language (in years)	< 1: 0 (0.00%), 1+, but < 2: 0 (0.00%), 2+, but < 3: 1 (4%), 3+, but < 5: 4 (16%), 5-9: 4 (16%), 10-14: 1 (4%), 15-19: 2 (8%), 20+: 13 (52.00%)
Most proficient programming language (Free text field, one participant listed 3 languages: overall mean responses 1.08)	C: 1 (4%), C#: 3 (12%), C++: 2 (8%), Fortran: 1 (4%), Java: 4 (16%), Javascript: 2 (12%), Objective C: 1 (4%), Python: 11 (44%), R: 1 (4%), Ruby: 1 (4%)
First line of code in proficient language (in years)	< 1: 0 (0.00%), 1+, but < 2: 3 (12%), 2+, but < 3: 2 (8%), 3+, but < 5: 2 (8%), 5-9: 7 (28%), 10-14: 4 (16%), 15-19: 2 (8%), 20+: 5 (20%)
Competency in proficient language	Beginner: 0 (0.00%), Beginner↔Intermediate: 1 (4%), Intermediate: 7 (28%), Intermediate↔Expert: 11 (44%), Expert: 6 (24%)

Table B.2: Professional demographics

Appendix C

Supplement materials for Chapter 5

C.1 Analysis of online copied code

Table C.1: The analyses of copied code from the websites. Note: this does not necessarily mean that the copied appeared at the final code, but copied during coding.

#P	Tasks	#C&P	Purpose of C&P	Source of C&P (based on C&P purpose)
P1	1	2	1-"Main" method method syntax, 2-Array declaration	1-Protechtrainning.com, 2-math.hws.edu
	2	3	1- `Console`: user entry syntax and assign it to `String` variable. 2-Convert `String` entry to `Integer`. 3-Convert `String` entry to `Float`.	1-Course material, 2-jaxenter.com, 3-java67.com
	3	2	1-`String` length, 2-`.startsWith` method	1-www.javatpoint.com, 2-beginnersbook
P2	1	2	1-Array declaration, 2-Import array, 3-Printing array outcomes.	1-java67.com, 2,3-JavaTpoint.com

	2	4	1-`Scanner` import, 2-`Scanner` identify object, 3-`Scanner` assign the entry to variable, 4-static method	1,2,3-JavaTpoint.com, 4- Stack Overflow
	3	5	1-`Scanner` import, 2-`Scanner` identify object, 3-`Scanner` assign the entry to variable, 4-`.println` statement, 5- `String` length	1,2,3,4,5- JavaTpoint (tutorials)
P3	1	5	1- The "Main" method method, 2- Array declaration and assigning values, 3-First and second `For` loop declarations, 4- Print the output	1,2,3,4-geeksforgeeks.org
	2	6	1- Declare `Scanner` object, 2- Read user input, 3-`.println` statement, 4- Convert `String` to `Integer`, 5- Convert `Integer` to `String`.	1,2,3- w3schools.com, 4,5- javatpoint.com
	3	4	1- Declaring `String` and getting user entry, 2-Declare `Integer` variable to get `String` length from user entry, 3- Declare char to get character at specific location, 4- Method to get the start of `String`.	1- w3schools.com. 2- guru99.com, 3- w3schools.com, 4-beginnersbook.com

P4	1	4	1- “Main” method method, 2-Declare `For` loop, 3- Declare array with contents, 4- Assign contents to the array variable	1- Previous ready code, 2-W3school, 3-geeksforgeeks.org
	2	7	1- Import *“DecimalFormat”*, 2- Import *“RoundingMode”*, 3- Declare new *“DecimalFormat”* object, 4- Select *“RoundingMode”* function, 5- Print rounding format, 6-Comma between variable in method, 7- Convert `String` to `Float`	1,2-docs.oracle.com, 3,4,5-stackoverflow.com, 6-W3school, 7-Java67.com
	3	6	1-Extract `String` “substring” method, 2-Get the ends `String` character, 3- `String` length, 4- “Main” method method, 5- Declare `Console`, 6- Get user values and assign it to `Console`	1,2-stackoverflow.com, 3- W3school, 4,5,6- Previous code.
P5	1	1	Array declaration	Java67
	2	2	1- Convert `String` to `Double`, 2-Catch NumberFormatException Exception	1- www.baeldung.com, 2- stackoverflow.com
	3	NON	NON	NON

	4	11	1-Declare new class that extend *“Thread”*, 2- Declare `Run()` method, 3- Print thread information, 4- Catch exception, 5- Print exception error, 6- Define Main class, 7- Declare `Integer`, 8- Declare `For` loop, 9- Declare object from Thread class, 10- Set thread name, 11- Start thread.	1, 2, 3, 4, 5, 6, 7, 8, 9, 11- geeksforgeeks.org, 10- javaconceptoftheday.com
P6	1	1	Declaring `For` loop	W3School
	2	5	1- Import `Scanner`, 2- Declare `Scanner` object, 3- `.println`, 4- Declare `String` and get user entry as `String`, 5- Parse `String` to `Integer`	1,2,3,4- w3schools.com, 5- beginnersbook.com
	3	4	1- Method to get the start of `String`, 2- “&” operator, 3- `String .endsWith` function, 4- `String` length,	1- beginnersbook.com, 2- tutorialpoint.com, 3-beginnersbook.com, 4-educative.io
	4	24	All the code were copied to fulfil the Task 4 functions.	tutorialspoint.com
P7	1	3	1- Declaring Array, 2- Loop and nested loop	1,2- GeeksforGeeks

	2	4	1- Import and declare `Scanner`, 2- Completing `Scanner` declaring, 3- `.println`	1- geeksforgeeks.org, 2,3- w3schools.com
	3	3	1- `String` length, 2- `String` "substring", 3- `String .charAt` method.	1- tutorialpoint.com, 2- javatpoint.com, 3- java67
	4	9	1- Declare *"Thread"* class, 2- Declare `Run()` method, 3- `.println` statement, 4- Create thread object, 5- Start thread object, 6- Lock elements, 7- *"wait()"*, 8- Throw Exception, 9- Import interrupted Exception	1, 2, 3, 4, 5- beginnersbook.com, 6,7,- javabypatel.blogspot.com, 9- docs.oracle.com
P8	1	3	1- Declare class, 2- Declare "Main" method method, 3- Declaring array.	1,2- tutorials.jenkov.com, 3- geeksforgeeks.org
	2	5	1- Import `Scanner`, 2- Declare `Scanner` object, 3- `.println`, 4- Declare `String` and get user entry, 5- Convert `String` into `Float`	1,2,3,4- w3schools.com, 5- java67.com

	3	6	1- Declare “Pattern” object with specification, 2- Declare “Matcher” object with specifications, 3- Declare “Boolean” variable to get *“matcher.find()”* status, 4- Define `String` variables to hold *“regex”* value: start and end of the pattern, 5- Declaring `Scanner`, 6- Assign `Scanner` value to variable.	1,2,3- w3schools.com, 4-tutorialspoint.com, 5,6- Previous code.
	4	6	1- Declare class with *“Thread”* extending, 2- Use *“@Override”*, 3- Declare *“Run()”* method, 4- `For` loop printing the threads’ output, 5- `.println`, 6- Import *“java.lang.Thread”*	1,2,3,4,5-freecodecamp.org, 6- stackoverflow.com
P9	1	4	1- Declare array, 2- Declare `For` loop, 3- Length of the array, 4- Getting 2d array element (column).	1- GeeksforGeeks.org, 2- W3school.com, 3-GeeksforGeeks.org, 4- stackoverflow.com.

	2	6	1- Import `Scanner`, 2- Declare `Scanner` object, 3- Declare `String` variable to get user entry, 4- Transfer `String` into `Float`, 5- `.println` statement, 6- Transfer `String` into `Float`	1,2,3- inf.unibz.it, 4-java67.com, 5-GeeksforGeeks.org, 6-JavaTpoint.com
	3	3	1- `String .startsWith` function, 2- Get `String .endsWith` function, 3- While loop	1-beginnersbook.com, 2- guru99.com, 3- w3school.com
P10	3	2	1- `String .substring`, 2- `String .charAt`	1- howtodoinjava.com, 2- javatpoint.com
	4	10	1- Declare *“Thread”* class, 2- Declare `Run()` method, 3- `.println`, 4- Error catching , 5- `.println` for error, 6- Declare “Main” method class and Main method , 7- Declare `Integer`, 8- Declare *“Thread”* class object, 9- Start thread	1,2,3,4,5,6,7,8,9- https- //www.geeksforgeeks.org/- multithreading-in-java/

C.2 Participants information sheet and consent form

MANCHESTER
1824
The University of Manchester

Participant Information Sheet

The role of the Internet on programming

This is a research project undertaken at the Department of Computer Science at the University of Manchester. Before you decide whether to take part, it is important for you to understand why the research is being conducted and what it will involve. Please take time to read the following information carefully before deciding whether to take part and discuss it with others if you wish. Please contact us if there is anything that is not clear or if you would like more information. Thank you for taking the time to read this.

Please take as much time as you need before deciding to participate in the study.

About the research

- **Who will conduct the research?**

  Primary Investigators

  Name: Omar Alghamdi and Dr. Sarah Clinch and Dr. Caroline Jay
  Address: Department of Computer Science,
  University of Manchester, Oxford Rd,
  Manchester. M13 9PL.
  Email: omar.alghamdi@postgrad.manchester.ac.uk and sarah.clinch@manchester.ac.uk and Caroline.Jay@manchester.ac.uk

- **What is the purpose of the research?**

  We'd like to understand more about how programmers write code to solve programming problems, and the role of different websites in that process.

- **Will the outcomes of the research be published?**

  The outcome of the research will be published as an academic paper, in a PhD thesis, and/or at academic conferences. The research may be also be used for teaching purposes.

- **Who has reviewed the research project?**

  The research has been reviewed by the Department of Computer Science Ethics Committee.

What would my involvement be?

- **What would I be asked to do if I took part?**

  You would be asked to participate in a video or audio call using the Zoom video conferencing software. The call will be between you and one researcher, it will be recorded, and you will be asked to share your screen during the call. The call will take no more than one hour of your time.

During the call, we will ask you to solve four programming tasks using the Java programming language. All programming tasks will be based on course materials from your first year; they may be challenging but the concepts should not be completely unfamiliar to you. This portion of the call will last up to forty-five minutes. We will ask you to share the screen as you complete the tasks, and to share with us the resulting Java files once the call is over. Feel free to do the tasks in any order and to move between them as you wish. However, at the end of this part of the study (45 minutes), we will ask you to stop all work on the tasks. You should focus only on the provided tasks and aim to complete them all in the allocated time, but do not worry if one or more tasks are still incomplete. Please do not change your code once the allocated time is ended.

At the end of the call, we will ask you a set of interview questions. These questions will take no more than fifteen minutes.

➢ **Will I be compensated for taking part?**

You will be compensated with a £10 Amazon voucher.

➢ **What happens if I do not want to take part or if I change my mind?**

It is up to you to decide whether or not to take part. You are free to take as much time as you wish before deciding. If you do decide to take part, you will be given this information sheet to keep and will be asked to provide verbal consent at the start of the Zoom call. You are free to withdraw your consent at any time during the call without giving a reason and without detriment to yourself. You may also choose to withdraw your data for up to two days following the call – you can contact the research team by email to do this. After two days, we will transcribe and anonymise your data and it will no longer be possible to remove it from the dataset. You should retain your assigned ID to be able to withdraw from the study. This does not affect your data protection rights.

<u>Data Protection and Confidentiality</u>

➢ **What information will you collect about me?**

In order to participate in this research project we will need to collect information that could identify you, called “personal identifiable information”. Specifically we will need to collect:

- Age
- Gender
- Sex

In term of the video/audio recording, the recordings will consist of:

- Screen recording video: record the screen while solving the tasks.
- Voice only: obtained during the interview session.

➢ **Under what legal basis are you collecting this information?**

We are collecting and storing personal identifiable information in accordance with data protection law which protects your rights. These state that we must have a legal basis (specific reason) for collecting your data. For this study, the specific reason is that it is "a public interest task" and "a process necessary for research purposes".

➢ **What are my rights in relation to the information you will collect about me?**

You have a number of rights under data protection law regarding your personal information.

If you would like to know more about your different rights or the way we use your personal information to ensure we follow the law, please consult our Privacy Notice for Research.

➢ **Will my participation in the study be confidential and my personal identifiable information be protected?**

Only the research team at the University of Manchester will have access to your information, which will be anonymised. Any identifying information will be removed and replaced with a random ID number.

In accordance with data protection law, The University of Manchester is the Data Controller for this project. This means that we are responsible for making sure your personal information is kept secure, confidential and used only in the way you have been told it will be used. All researchers are trained with this in mind, and your data will be looked after in the following way:

- Your data will be anonymised (associated with a random ID instead of your name or other personal information). Any sensitive or personal information revealed during the call (e.g. on your shared computer screen) will be obscured as part of the anonymisation process.
- Recordings will be made using the Zoom video-conferencing software. We will ask you to turn off all video recording except for screen sharing, so the resulting recording will include your screen and all call audio.
- Data will be stored on an encrypted device and will be backed up to a secure data store.
- Anonymised transcripts and screen recordings will be shared with other organisations for research purposes, but you will not be identifiable from the data.
- Your data may be used in future research studies.
- The data may be shared with any other organisation.
- The interview audio files will be transcribed by the researcher, and any disclosure of personal or sensitive information during the calls will be redacted.

When you agree to take part in a research study, the information about you may be provided to researchers running other research studies in this organisation. The future research will be of a similar nature to this research project and will concern programming field. Your information will only be used by this organisation and researchers to conduct research in accordance with UREC studies: The University of Manchester's Research Privacy Notice**.** The information will only be used for the purpose of publication and cannot be used to contact you regarding any other matter.

Please also note that individuals from The University of Manchester or regulatory authorities may need to look at the data collected for this study to make sure the project is being carried out as planned. This may involve looking at identifiable data. All individuals involved in auditing and monitoring the study will have a strict duty of confidentiality to you as a research participant.

What if I have a complaint?

➢ **Contact details for complaints**

If you have a complaint that you wish to direct to members of the research team, please contact:

Dr. Sarah Clinch
Department of Computer Science, the University of Manchester
Oxford Road, Manchester. M13 9PL.
0161 275190 or sarah.clinch@manchester.ac.uk

If you wish to make a formal complaint to someone independent of the research team or if you are not satisfied with the response you have gained from the researchers in the first instance then please contact

The Research Ethics Manager, Research Office, Christie Building, The University of Manchester, Oxford Road, Manchester, M13 9PL, by emailing: research.complaints@manchester.ac.uk or by telephoning 0161 275 2674.

If you wish to contact us about your data protection rights, please email dataprotection@manchester.ac.uk or write to The Information Governance Office, Christie Building, The University of Manchester, Oxford Road, M13 9PL at the University and we will guide you through the process of exercising your rights.

You also have a right to complain to the Information Commissioner's Office about complaints relating to your personal identifiable information Tel 0303 123 1113

Contact Details

If you have any queries about the study or if you are interested in taking part then please contact the researcher:

Omar Alghamdi
Department of Computer Science, the University of Manchester
Oxford Road, Manchester. M13 9PL.
omar.alghamdi@postgrad.manchester.ac.uk

Participant Consent Form

The role of the Internet on programming

Experimental Study

If you are happy to participate please complete and sign the consent form below

	Activities	Initials
1	I confirm that I have read the attached information sheet (**Version 1.1, Date 30/06/2020**) for the above study and have had the opportunity to consider the information and ask questions and had these answered satisfactorily.	
2	I understand that my participation in the study is voluntary and that I am free to withdraw at any time without giving a reason and without detriment to myself. I understand that it will not be possible to remove my data from the project once it has been anonymised and forms part of the data set. I agree to take part on this basis.	
3	I agree that any anonymised data collected may be used in teaching or publications (e.g. including academic journals).	
4	I agree that any anonymised data collected may be shared with researchers at other institutions.	
5	I agree to the interviews being audio recorded.	
6	I agree to the screen being video recorded.	
7	I agree to share my source code file with other researchers.	
8	I understand that data collected during the study may be looked at by individuals from The University of Manchester or regulatory authorities, where it is relevant to my taking part in this research. I give permission for these individuals to have access to my data.	
9	I agree to take part in this study.	

Data Protection

The personal information we collect and use to conduct this research will be processed in accordance with data protection law as explained in the Participant Information Sheet and the Privacy Notice for Research Participants.

__________________________	__________________________	______________
Name of Participant	Signature	Date
__________________________	__________________________	______________
Name of the person taking consent	Signature	Date

[Insert details of what will happen to the copies of consent form e.g. 1 copy for the participant, 1 copy for the research team (original), 1 copy for the medical notes]

C.3 Text used to recruit participants from social media

Are you a programmer? Do you love coding?

Then, helps us understand programmers' behaviours by solving few programming tasks.

The study is completely online via Zoom and will take no more than one hour of your time.

Please consider partaking and let your friends know.

Note: You must be an undergraduate second-year student to be able to participate in the study.

Please email: omar.alghamdi@manchester.ac.uk for more information

C.4 Script used for verbal consents

Verbal consent script

The role of the Internet on programming

The verbal script:

Hi,

My name is Omar Alghamdi, and I am a PhD student in the Computer Science department. I am conducting research about understanding the role of the internet on the programming, and I am interested in your experiences as a programmer. The purpose of the research is to understand the behaviours and processes that emerge during programming. Your participation will involve solving programming tasks and a short interview, and it will last for an hour. This research has no known risks. This research will benefit the academic community because it helps us to understand possible programmer's behaviours and processes. Your identity or personal information will not be disclosed in any publication that may result from the study. Data that are taken during the study will be stored in a secure location, and may be used for teaching or publication. The data may also be shared with other organizations. Please, feel free to ask questions and seek further information before consenting to take part in the study.

If you are happy to participate, please agree verbally to the following:

	Activities
1	Do you confirm that you have read the attached information sheet (**Version 1.3, Date 09/08/2020**) for the above study and have had the opportunity to consider the information and ask questions and had these answered satisfactorily?
2	Do you understand that your participation in the study is voluntary and that you are free to withdraw at any time without giving a reason and without detriment to yourself? Do you understand that it will not be possible to remove your data from the project once it has been anonymised and forms part of the data set? Do you agree to take part on this basis?
3	Do you agree that any anonymised data collected may be used in teaching or publications (e.g. including academic journals)?
4	Do you agree that any anonymised data collected may be shared with researchers at other institutions?

5	Do you agree to the interviews being audio recorded?
6	Do you agree to the screen being video recorded?
7	Do you agree to share your source code file with other researchers?
8	Do you understand that data collected during the study may be looked at by individuals from the University of Manchester or regulatory authorities, where it is relevant to your taking part in this research? Do you give permission for these individuals to have access to your data?
9	Do you agree to take part in this study?

Name:

For generating IDs:

1. First three letters of your **surname**:
2. Tow digit of the **Day** you were born: