The Journal of Systems & Software 205 (2023) 111803

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

How are websites used during development and what are the implications for the coding process?☆

Omar Alghamdi [a,b,*], Sarah Clinch [a], Rigina Skeva [a], Caroline Jay [a]

[a] Department of Computer Science, University of Manchester, M13 9PL, United Kingdom
[b] College of Computing and Informatics, Saudi Electronic University, Riyadh, 6867, Saudi Arabia

ARTICLE INFO

Article history:
Received 23 August 2022
Received in revised form 15 March 2023
Accepted 14 July 2023
Available online 21 July 2023

Dataset link: https://doi.org/10.48420/c.6148593

Keywords:
Stack Overflow
Online code snippets
Problematic code
Human memory
Computer science education
Professional practice

ABSTRACT

Websites are frequently used to support the development process. This paper investigates how websites are used when writing code and programmers' perceptions of the potential impact of this on their behaviour and the quality of the resulting software. We interviewed 18 programmers (13 students enrolled in undergraduate computer science courses, and 5 experienced professionals), and analysed the data thematically. The findings were used to develop a survey, which was distributed to 276 programmers (251 students, 25 experienced professionals). The results indicate that use of websites, especially Stack Overflow, is viewed as an essential part of programming by both students completing coursework and professionals developing code in industry.

We also found that developers have experience of encountering a diverse set of problematic code snippets online, that copying code from websites without checking its quality or understanding how it worked is common, and that using online resources in this way had a potentially counter-productive effect on learning. Based on these findings, we make a number of recommendations, including better consideration of online code reuse in taught programmes, co-development and code-reuse practices in professional settings, and software licensing training for professional developers.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

© 2023 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

The internet has made information more accessible than ever before, with significant implications for how people learn and practice programming. Developers sometimes spend more time searching for information online than they do coding (Brandt et al., 2009), and website usage is thus viewed as an integral part of the software engineering process (Yang et al., 2017). Technical question and answer websites such as Stack Overflow are particularly popular; as of June 2022, Stack Overflow constitutes 22 million questions and 34 million answers, a clear indication of scale and popularity (Stack Exchange Inc., 2022).

Use of digital resources, particularly the Web and search engines, has led to concerns about a so-called 'Google effect' (Schooler and Storm, 2021; Sparrow et al., 2011). Individuals offload responsibility for retaining information to the Web (Risko and Gilbert, 2016), leading to reduced independent recall (Schooler and Storm, 2021; Sparrow et al., 2011) and a false sense that the information is known (Fisher et al., 2021). The role of memory in successful programming is relatively under-explored, but it is known that programmers use memory to maintain awareness of systems they develop, including the architecture, code intent, ownership and changes (Krüger and Hebig, 2020). In this paper, we take a lead from recent empirical results (Giebl et al., 2020), and consider whether programmers' use of the Web impacts their memory for programming concepts, and whether programmers' memory for code is dependent on prior experience (Giebl et al., 2020; Ichinco and Kelleher, 2017).

Copying code from online sources (also known as code cloning (Baxter et al., 1998)) is commonplace during development (Brandt et al., 2009), and is a matter of considerable debate. It has been argued that cloned code can impact software stability and maintenance both positively and negatively (Kapser and Godfrey, 2008; Krinke, 2008; Lozano and Wermelinger, 2008; Mondal et al., 2018). One factor in the impact of code cloning is the validity of the cloned code itself — prior research has shown that code snippets on Stack Overflow contain problems ranging

☆ Editor: Kelly Blincoe.

* Corresponding author at: Department of Computer Science, University of Manchester, M13 9PL, United Kingdom.
E-mail addresses: omar.alghamdi@manchester.ac.uk, oalghamdi@seu.edu.sa (O. Alghamdi), sarah.clinch@manchester.ac.uk (S. Clinch), rigina.skeva@manchester.ac.uk (R. Skeva), caroline.jay@manchester.ac.uk (C. Jay).

https://doi.org/10.1016/j.jss.2023.111803
0164-1212/© 2023 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

from security issues (Acar et al., 2016; Fischer et al., 2017; Licorish and Nishatharan, 2021; Meldrum et al., 2020; Meng et al., 2018) to outdated statements (Ragkhitwetsagul et al., 2021) and failure to meet language-appropriate styling (Bafatakis et al., 2019; Meldrum et al., 2020). Assessing code snippets suitability can be challenging, and requires reading and understanding of the code (Schröter et al., 2017); nevertheless, programmers often do not fully engage in this due to the time and effort involved (Maalej et al., 2014), a lack of knowledge (Escobar-Avila et al., 2019; Yu et al., 2019), or low readability of the snippet itself (Meldrum et al., 2020). Whilst prior efforts have provided measures of problematic online code (and their impacts for commercial software) based, for example, on data mining, to the best of our knowledge this is the first paper to identify those problems based on programmers' experience. Our work also differentiates these experiences based on programmer expertise.

We performed a qualitative interview study involving 18 programmers, including undergraduates and experienced professionals. This interview set out to capture and in-depth understanding of the resources (particularly online resources) employed during programming, how and why these were used, and their possible impacts on memory, code understanding, and resultant code. To test the prevalence of these findings, we conducted a survey with 251 undergraduate students, and 25 professional developers.

Our research questions were as follows:

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

RQ2 What is the relationship between resource use (during coding tasks) and human memory?
Specifically:

- RQ2.1 What is the role of human memory in programming?
- RQ2.2 What perceived weaknesses or challenges with human memory prompt people to use external resources?
- RQ2.3 What is the perceived impact of using resources on memory? Specifically, how does use of a dominant resource (the Web) impact programmers' perceptions of their ability to recall programming-related information?

RQ3 What are the potential impacts of resource use on programmers' code?
Specifically:

- RQ3.1 To what extent do programmers' reuse code that they encounter online? What safeguards do they put in place when they do this?
- RQ3.2 To what extent do programmers' encounter and identify problematic code online?
- RQ3.3 What are the perceived impacts of code reuse on code quality?

Our results confirm prevalent website use, particularly of Stack Overflow. They show that many developers copy and paste online code without full understanding of what that code does, and that they are motivated by gaps in their knowledge and time pressures. Despite an overall belief that they have a good memory, some programmers express concern about the potential for reliance on online resources to inhibit memory. By contrast, although problematic code is encountered online, programmers feel that experience allows them to differentiate between good and bad online content.

2. Background and related work

Our work draws on two distinct bodies of prior understanding. Firstly, we build on research from software engineering that seeks to understand the tools and mechanisms by which individuals complete programming tasks, and the resulting impact on code outputs. Secondly we consider the potential impact of this resource use on programmers themselves, by drawing on evidence from psychology relating to human memory. In particular, we consider the potential for negative effects to arise from use of the web as a transactive memory partner (i.e., cognitive offloading).

2.1. External resource use when software engineering

A diverse set of resources exist to support the learning of, and continued, programming activity. For students and novices, traditional textbooks have more recently been supplemented with web resources including video. Lausa et al. (2021) studied undergraduate students at public and private universities in the Philippines, asking about their use and perceptions of educational materials when programming. They found low rates of book/eBook use (<30%) and highest rates of use for YouTube (67–86%) and friends/peers (53–72%); their participants reported that books were more challenging to find code in compared to online resources/peers, but that some online resources were difficult to understand, and understanding code written by others was particularly challenging (Lausa et al., 2021).

In addition to the resources available to novices, professional programmers are likely to have access to experienced colleagues (Bai et al., 2020; Hucka and Graham, 2018; LaToza et al., 2006; Maalej et al., 2014). For example, in a survey of 1477 professionals, Maalej et al. (2014) report that they seek knowledge about software from other people, and the same proportion reported sharing their own knowledge by the same means. By contrast, artefacts such as documentation, comments and commit messages were used for knowledge seeking by 62% and for knowledge sharing by 56% (Maalej et al., 2014). Professionals are also more likely to have access to established codebases — one study of developers at Google observed developers making an average of twelve codebase queries per day (Sadowski et al., 2015).

Both students and professionals make substantial use of the web when programming (Acar et al., 2016; Brandt et al., 2009; Michaels et al., 2020; Xia et al., 2017). Specifically, search engines are used for activities such 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 (including e.g. syntax) (Hora, 2021; Xia et al., 2017).

The Stack Overflow Q&A website has been observed to dominate programming-related online search (Acar et al., 2016; Hora, 2021; Xia et al., 2017) and is often used in preference to official documentation (Ragkhitwetsagul et al., 2021) — for example, in their observation study of 60 industry developers, Xia et al. (2017) find that 63% of all searches involve a visit to the website (Xia et al., 2017). However, both free search and Stack Overflow use may make developers less efficient — in one study of developers assigned a security-related problem to solve, free search and use of Stack Overflow both resulted in more frequent queries and more webpages visited (Acar et al., 2016). Finding information on Stack Overflow may be particularly challenging for novices, who find that text and code snippets contain unnecessary details, are difficult to assess for relevance and are too complex for their current level of expertise (Chatterjee et al., 2020).

In this paper, we ask both student novices and experienced professionals about the resources used in their programming activities with a particular focus on Stack Overflow. Our interviews build a rich picture of how and why developers turn to the web as a resource, whilst our survey captures features of Stack Overflow use including frequency (and changes in frequency) and perceptions of its various features (e.g. multiple answers, voting, reputation).

2.1.1. Code reuse

A common pattern in developers' website use, particularly Stack Overflow, is that of identifying appropriate code snippets for reuse. For example, in one observational study of developers, over 10% of queries made by developers were related to reusable code snippets (Xia et al., 2017) — their developers were motivated to search for reusable code in order to reduce development time, although a number of factors made this problematic (difficulty formulating queries, poor support for code search, low code quality and developer trust).

Code reuse (also referred to as code cloning) is a well-established practice in software engineering, both from the web and from large commercial codebases, and can be both advantageous and harmful. Code reuse minimises duplicated effort and allows developers to draw on established code that may already have been subject to scrutiny and testing in order to reduce maintenance effort (Feitosa et al., 2020). For example, Kapser and Godfrey (2008) identify three motivations for code cloning: forking (a similar solution is used as a start point with the expectation that future development will diverge from the current path), templating (where functionality explicitly duplicates some existing code, but no common abstraction can be used to avoid duplication), and customisation (where existing code does not meet a new requirement and the existing code is cloned before modification) (Kapser and Godfrey, 2008). In these cases, code cloning can aid software maintainability, resulting in more stable code (Kapser and Godfrey, 2008; Krinke, 2008). However, other results suggest that stability can also be negatively impacted by code reuse (Lozano and Wermelinger, 2008; Mondal et al., 2018).

Data mining studies have also shown how code propagates between Stack Overflow and larger code bases (e.g. open source and commercial projects) (Abdalkareem et al., 2017; Fischer et al., 2017; Yang et al., 2017), often with the intention of identifying common harmful code. Abdalkareem et al. (2017) examined the source of 1,496 Android apps finding that 377 had source code in common with Stack Overflow, and 22 were considered to be definite clones. For these 22 apps, just over 1% (mean) of the source was reused from Stack Overflow, but the introduction of code from Stack Overflow appeared to have increased the number of subsequent bug fix commits (i.e. the reused code had a potentially negative impact on functionality or other desirable qualities) (Abdalkareem et al., 2017). Similarly, Habchi et al. (2021) mined 180,013 code smells (potentially problematic code) from 324 open source Android applications and tracked their evolution — whilst the majority were removed, a manual study of 561 sampled code smell removal commits indicated that removals may occur as a side-effect of other activity. Interviews with 25 developers suggest that developers avoid refactoring code even if they are aware of a potential problem (Habchi et al., 2021). However, not all data mining studies agree that code reuse is responsible for introducing code problems — in their study of 1,244 open-source projects, Gkortzis et al. (2021) find that while project size is a good predictor of security vulnerabilities, code reuse predicts neither their presence or absence (i.e. there is no evidence that code reuse is either beneficial or harmful) (Gkortzis et al., 2021).

We extend this literature on code reuse by collecting data on patterns of reuse from both students and professional developers. Our interviews and survey describe the circumstances in which code reuse occurs (or does not) and on the perceived impact of reuse on both code outputs and the programmers themselves (specifically the degree to which they understand from and learn from copied code).

2.1.2. Problematic code snippets and assessments of quality

Code reuse is at its most problematic when (1) the code snippet targeted for reuse contains one or more problems, and (2) the programmer engaging in reuse is unable or unwilling to make a judgement about the poor quality (and suitability) of the snippet for reuse.

Data mining studies such as those discussed in (Section 2.1.1) have indicated a variety of problems in online code. For example, Fischer et al. (2017) mined 3,834 distinct security code snippets from Stack Overflow. After manual inspection and labelling of 1,360 training snippets, an SVM classifier found that 97.9% of mined snippets contained at least one security-related problem (Fischer et al., 2017). Further mining (Meng et al., 2018) and manual search (Bai et al., 2019) analyses of security-related Stack Overflow posts have identified multiple security vulnerabilities, including those present in accepted answers, upvoted answers and answers from individuals with high reputation (Meng et al., 2018). Lab study results also indicate that heavy use of Stack Overflow results programmers introducing more vulnerabilities to their security-related code (Acar et al., 2016). Note, however that security-related Q&A is only a subset of the content on Stack Overflow — in their analyses Licorish and Nishatharan (2021) find that security vulnerabilities were only present in a very small number of posts (Licorish and Nishatharan, 2021).

Problems with security libraries was just one subset of the problematic API use observed by Zhang et al. (2018) in their study of 217,818 Stack Overflow posts. Database and networking examples were also frequently found to be incomplete and to contain errors that could result in resource leaks and program crashes (Zhang et al., 2018). In addition, Yang et al. (2016) noted incomplete and unparsable code in Stack Overflow snippets more generally (Yang et al., 2016), while Zhou and Walker (2016) found 7,464 snippets (from 638,756) that contained deprecated calls with that deprecation noted in only 3% (based on manual analysis of a smaller sample). Other analysis has identified licensing as a potential issue (e.g. where code from a project under a particular license is posted to Stack Overflow with no indication of the associated licensing) (An et al., 2017; Ragkhitwetsagul et al., 2021), and noted the presence of outdated code (i.e. code that had been modified in its original project but not on Stack Overflow) — in some cases this outdated code contained bugs that had since been fixed in the original project (Ragkhitwetsagul et al., 2021). Nishi et al. (2019) also found duplication between tutorials and Stack Overflow (Nishi et al., 2019).

Research with Stack Overflow users indicate that the above are just a subset of the issues encountered. In their survey of 87 Stack Overflow visitors, Ragkhitwetsagul et al. (2021) find that 45% report finding code that worked only with older versions of a library or API, whilst 32% reported encounters with code that incorrectly claimed to solve the problem in the question. Furthermore, 46% reported that they encountered a mismatch between questions/answers and their exact problem (Ragkhitwetsagul et al., 2021). In another survey, Treude and Robillard (2017) presented programmers with code snippets (the majority sourced from Stack Overflow) – 29% of all code snippets (29% of those from Stack Overflow) were reported to be incomplete or inadequate, with issues including poor code structure/organisation, problematic variable naming, verbosity, and missing rationale (Treude and Robillard, 2017). Further surveys have also noted similar concerns about both quality and understanding for both Stack Overflow and other sources programming-related content on the web (Escobar-Avila et al., 2019; Wu et al., 2019), and indicate that developers lack sufficient understanding of licensing and code reuse (Wu et al., 2019).

External indicators (e.g. Stack Overflow's voting system, accepted answers and reputation) may help to reduce developers

use of problematic code (Meldrum et al., 2020; Nasehi et al., 2012), but code from unaccepted answers with few upvotes are still readily adopted (Wu et al., 2019) even when that code contains problems (Zhang et al., 2018). Movshovitz-Attias et al. gather answers from Stack Overflow and found that the main source of answers is the user with high reputation. One potential factor is programmers' ability to access code snippets for quality, a term that is loosely defined in both the literature and amongst programmers themselves (Börstler et al., 2017). Poor coverage of code quality is also observed in taught courses (Börstler et al., 2017; Kirk et al., 2020). For example, Kirk et al. (2020) examined the learning objectives of 141 University courses and found 70% make no mention of code quality (Kirk et al., 2020), with the result that students rarely resolve their code quality issues (Keuning et al., 2017).

Failure to fully understand code snippets also impacts the propagation of problematic code. Maalej et al. (2014) interviewed 28 and surveyed 1477 professionals, finding that programmers avoid understanding copied code for a variety of reasons including a focus on task completion (Maalej et al., 2014). Novices' lack of knowledge also hinders code understanding (Escobar-Avila et al., 2019). In an lab study of novices and professionals, LaToza et al. (2007) 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). As a result, developers may be better positioned to utilise code snippets even when they do not fully understand them — one survey of more experienced developers indicated that difficulties in understanding a code snippet (together with other factors such as perceived code quality) may actually lead developers to re-implement rather than copy code (Wu et al., 2019).

In this paper, we extend current understanding on both student and professional programmers encounters with problematic code snippets. Our interviews and surveys both indicate that both novices and professionals encounter a range of issues, but that the frequency and range of problems encountered is influenced by the frequency and range of programming tasks attempted.

2.2. Human memory

Human memory is the cognitive process by which we encode, store, and retrieve information (Baddeley et al., 2020; Robins et al., 2019). Although classifications of human memory vary, it is generally considered that long-term memory (memory lasting more than a few seconds) (Ericsson and Kintsch, 1995) includes both memory for experiences (episodic) and the body of knowledge that we build from those experiences (semantic) (Baddeley et al., 2020; Robins et al., 2019). For knowledge-based work, such as programming, it is generally accepted that the use of mental models and semantic memory is fundamental (Bidlake et al., 2020). Studies of programmers suggest that memory for code is a product of familiarity (Fritz et al., 2007) and expertise (Ichinco and Kelleher, 2017), and that programmers tend to remember abstractions (e.g. architecture and code intent) as well as external contextual information that supports co-development (e.g. ownership and changes) (Krüger and Hebig, 2020).

In addition to individual knowledge, researchers have suggested that our everyday relationships allow a form of group knowledge. In this *transactive memory* (Wegner et al., 1985), each individual in a group maintains specialised knowledge that other in-group members know how to access. Some researchers suggest that as technologies have become more central to our everyday activities, they have adopted the place of a transactive memory partner — that is, we selectively offload formation of semantic memories to these technologies (typically the internet), knowing that the information can easily be retrieved (Ferguson et al., 2015; Ward, 2013). This process of *cognitive offloading* (Risko and Gilbert, 2016) has evident advantages, allowing individuals access to a "super normal" (Ward, 2013) transactive partner whose knowledge far surpasses cognitive capacity. However, empirical studies suggest that the mere suggestion of internet availability can negatively impact human memory (Ferguson et al., 2015; Fisher et al., 2021; Sparrow et al., 2011).

Seminal studies in this space were performed by Sparrow et al. (2011) and led to the coining of the term "Google effect" — that is, when participants expect information to be retained by technology, they perform less well in encoding/recalling that information themselves (Sparrow et al., 2011). Since Sparrow et al.'s early work, potential negative impacts have been observed for a range of technologies and memory types (Clinch et al., 2021). Most relevant to this research is a number of studies exploring the interaction between search engine use and semantic memory. For example, Macias et al. found that information of the kind easily retrieved by search engines was more poorly remembered in a test of recall, particularly by those with better search engine skills (Macias et al., 2015), Ferguson et al. found that access to the internet reduces the likelihood that individuals will attempt to answer questions by drawing only on their personal knowledge (Ferguson et al., 2015), and Fisher et al. found that search engine use during a learning task reduced students' recall of material whilst simultaneously increasing their confidence that information had been retained (Fisher et al., 2021). This increased confidence results in inflated self-perceptions of knowledge (Fisher et al., 2015). Furthermore, since prior use of a search engine appears to increase the likelihood of future search engine use (Storm et al., 2017), negative impacts have the potential to increase over time.

Given heavy use of the internet during software engineering (Section 2.1), and the explicit perception of them as an "external memory aid" (Brandt et al., 2009, p1594), this paper considers the potential for Google effects to appear in the context of programming-related (as opposed to general) knowledge. To date there appear to be only a single study that examines search-engine induced memory impairment in this context. Giebl et al. performed a controlled experiment with 240 non-CS undergraduates, some of whom had prior programming experience and some who did not (Giebl et al., 2020). 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 internet, 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 internet were better able to retain concepts, and that this effect was strongest in those with prior programming experience (Giebl et al., 2020).

In this paper, we contribute to knowledge on how programmers' use their memory during programming tasks, and how this differs with expertise. Specifically, our interviews and surveys capture both students' and experienced professionals' perceptions of their memory use when programming, and of memory's importance to their successful execution of programming tasks. Furthermore, we consider the specific case of search-engine induced memory impairment when programming — we capture a variety of measures that describe programmers web dependence when programming, and the perceived impact of web use on their memory for programming-related concepts.

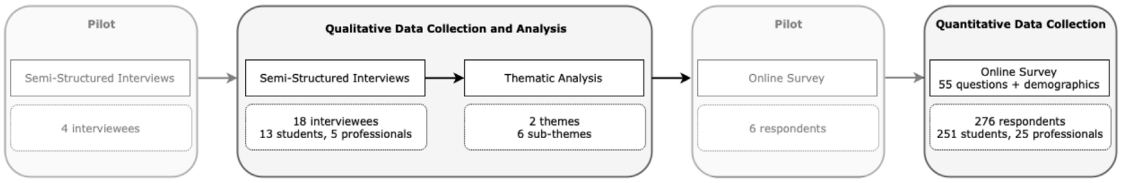

Fig. 1. Overview of research methodology. A sequential mixed methods approach combines semi-structured interviews whose qualitative data is thematically analysed; these themes form the basis for a larger quantitative data collection through online survey.

3. Methods

We take a sequential mixed methods approach (see Fig. 1) in which a set of semi-structured interviews ($N = 18$) are followed by a larger-scale online survey ($N = 276$). Both the interviews and the survey target a combination of undergraduate students enrolled on computer science courses, and professional programmers.

Procedures for both studies were reviewed and approved by the Department of Computer Science Ethics Committee at the University of Manchester.[1] All participants were provided with a briefing sheet prior to participation (available as supplementary material[2]), and were required to provide either written (for interviews) or checkbox (for survey) consent. All data was anonymised at time of collection.

3.1. Interviews

Semi-structured one-to-one interviews were conducted with both undergraduate students and professional programmers. Students were recruited from the University of Manchester; professionals were required to be working in the UK. Prior to recruitment, a pilot ($n = 4$) was used to ensure the clarity and validity of interview questions, and to assess overall interview duration.

3.1.1. Participants and recruitment

Eighteen participants were recruited: seven first year undergraduates, six second year undergraduates, and five professionals (see Table 1). All undergraduate students ($n = 13$) were students enrolled on a computer science programme at the University of Manchester. Students were interviewed during the first semester of the academic year, meaning that many of the first year students were extreme novices or hobbyist programmers with no formal training. By contrast, second year students had completed at least one year of undergraduate computer science tuition. Professional programmers were required to have been in paid employment in a job role involving programming for at least a year. All participants were required to self-identify as fluent English language speakers who were competent in at least one programming language (see Table 2).

Student participants were recruited via email: all students enrolled on a core first year programming course unit, as well as those enrolled on a core second year software engineering course unit, were invited to participate (˜500 students in total). Twelve students responded, with a further one recruited through snowball sampling: thus a total of thirteen students (referred to hereafter as P1-P13) agreed to participate in an interview.

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

All participants were rewarded with a £10 GBP Amazon gift certificate[3]

3.1.2. Data collection

All participants were asked to provide basic demographic information and to describe their programming experiences to date. Subsequent questions were designed to address the research questions presented in Section 1, i.e. they focused on (1) use of resources during programming tasks, (2) perceived impacts of that resource use on programmers' cognition, and (3) perceived impacts of resource use on programming activities and outputs Although the overall structure and line of questioning was similar, some adaptations were made for our two different populations: student participants were asked specific questions about their courses and available resources; whereas professional programmers were asked questions about their professional settings and approaches. The full set of interview questions is given in Table 5.

Interviews were conducted in English, either face-to-face or remotely between November 2019 and March 2020. Face-to-face interviews were carried out in a quiet place at University of Manchester. Remote interviews were conducted as audio or video calls using the *Skype* video conferencing software. Interviews lasted between 25 and 45 minutes, and were audio recorded before being transcribed verbatim by a commercial third party. Transcriptions were verified by the first author prior to analysis.

3.1.3. Analysis

Data were imported into the qualitative data analysis application *NVivo12*, and analysed using a process of inductive thematic analysis (Braun and Clarke, 2006, 2012), focusing on broad data-based thematic patterning.

To develop an initial familiarity, the first author read the raw data multiple times. Subsequent inductive coding of the excerpts by the first author led to a set of expansive codes that covered topics including (but not limited to) those articulated in our research questions. This full set of codes were then reviewed by the third author for consistency and clarity. The first author then grouped codes into coherent and comprehensive themes, and data extracts were coded to as many themes as relevant; the resulting themes were then revised iteratively through discussion between the first, second and fourth authors. The resulting candidate themes were then revisited in the context of the original data by the first and third authors to ensure good levels of compatibility with the original transcripts. These two authors then agreed a final set of themes. Inter-rater reliability between the first and third authors was calculated for 30% of the dataset, with near-full agreement (Cohen's $k = 0.84$).

3.2. Surveys

An online survey was distributed to undergraduate students studying UK computer science and software engineering programmes, and to professional programmers employed in the UK. Prior to recruitment, a pilot ($n = 6$) was used to ensure that the questions were clear and understandable, to eliminate any ambiguity, and to ensure that fluent but non-native English speakers could complete the survey within the expected time period (10–15 min).

[1] Reference: 2019-6829-12032

[2] https://doi.org/10.48420/c.6148593

[3] Approximately $14 USD.

Table 1
Participant demographics (interview study).
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.

Participant	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 study
P4	Male	18	Current: Year 1 undergraduate Prior: Middle School	None	None
P5	Male	18	Current: Year 1 undergraduate	None	Independent study
P6	Female	18	Current: Year 1 undergraduate	None	Independent study
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 study
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 study
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 study
P15	Female	34	None	Current: Senior developer Prior: ~10 years	None
P16	Female	45	Prior: Undergraduate & 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 2
Participants' experiences with programming languages (interview study), 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.

Participant	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↔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↔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
Participant demographics (online survey, student respondents).

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 (50.20%), 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: 9 *(Other values: Android Studio, Go, Haskell, Lua, Objective C, Rust, Swift, Typescript, Visual Basic.)*
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%).
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%)

3.2.1. Participants and recruitment

Two hundred and seventy six participants were recruited: 251 undergraduates, and 25 professionals (see Tables 3 and 4). Surveys were completed between January and June 2021, i.e. in the second half of the academic year.

Undergraduate participants were students enrolled on a computer science or software engineering programme at a UK University. Students were recruited predominantly online, through social media and emails sent from their department's director of undergraduate studies (or their delegate). One hundred and sixty-eight UK-based universities were contacted, and ten agreed to circulate our emails to students.

Professional participants were required to have been in paid employment in a UK-based job role involving programming for at least a year. Professional programmers were recruited through a combination of personal social networks and snowball sampling (e.g. interview participants were sent the link and invited to share it with others), and online recruitment in specialised forums such as the Society of Research Software Engineering.[4]

All participants were required to self-identify as fluent English language speakers who were competent in at least one programming language. Participants were rewarded with the option of entry into a prize draw to win a £50 GBP Amazon gift certificate[5]

3.2.2. Data collection

Online surveys were delivered using the Qualtrics survey tool. Two variants of the survey were created, one for students and one for professionals. The two surveys differed only in their initial questions, which collected basic demographic information and programming experiences to date. Subsequent questions (56) were developed based on the interview findings and were identical across both surveys; responses were given on three- (12 questions) four- (14 questions) or five-point (30 questions) scales. Questions were grouped into five topic-oriented sections to increase participant focus and minimise cognitive load (Lazar et al., 2010). Both questionnaires are included as supplementary material and summarised in Table 6.

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

[5] Approximately $70 USD.

Table 4
Participant demographics (online survey, professional respondents).

Question	Data		
Number of participants:	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 (80.00%), Female: 4 (16.00%), Non-binary: 0 (0.00%), Not disclosed: 1 (4.00%)		
Previous experience (multiple answers permitted: mean responses 1.36, median 1.00)	Casual/voluntary employment: 1 (4.00%), Full time employment: 19 (76.00%), Hobby programmer: 5 (20.00%), Internship: 2 (8.00%), Prior study (e.g. GCSE/A-Level): 6 (24.00%), No prior programming experience: 1 (4.00%),		
First line of code in any language (in years)	<1: 0 (0.00%), 2+, but <3: 1 (4.00%), 5-9: 4 (16.00%), 15-19: 2 (8.00%),	1+, but <2: 0 (0.00%), 3+, but <5: 4 (16.00%), 10-14: 1 (4.00%), 20+: 13 (52.00%),	
Most proficient programming language (Free text field, one participant listed 3 languages: overall mean responses 1.08)	C: 1 (4.00%), Fortran: 1 (4.00%), Objective C: 1 (4.00%), Ruby: 1 (4.00%)	C#: 3 (12.00%), Java: 4 (16.00%), Python: 11 (44.00%),	C++: 2 (8.00%), Javascript: 2 (8.00%), R: 1 (4.00%),
First line of code in proficient language (in years)	<1: 0 (0.00%), 2+, but <3: 2 (8.00%), 5-9: 7 (28.00%), 15-19: 2 (8.00%),	1+, but <2: 3 (12.00%), 3+, but <5: 2 (8.00%), 10-14: 4 (16.00%), 20+: 5 (20.00%).	
Competency in proficient language	Beginner: 0 (0.00%), Intermediate: 7 ((28.00%), Expert: 6 (24.00%)	Beginner↔Intermediate: 1 (4.00%), Intermediate↔Expert: 11 (44.00%),	

3.2.3. Analysis

From 311 raw survey responses (282 student; 29 developer), we excluded 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 therefore consisted of 276 responses (251 student; 25 developer). Likert scale responses were treated as ordinal data (Kitchenham and Pfleeger, 2003), and analysis was undertaken in Python, using the `pandas` and `matplotlib` libraries. Datasets and processing scripts are included as supplementary material.

4. Results

4.1. Interview study

Our thematic analysis revealed two main themes concerning website use, and its impact on memory and code (see Fig. 2); these themes encompass six sub-themes. Before presenting these themes, we describe the resources people reported using.

4.1.1. 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 codebases, course materials and the development environment (e.g., the auto-complete functionality of an IDE).

Book use varied considerably within our 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 books had been superseded by more convenient, and searchable, online resources.

Both students and professionals also discussed other people as a potentially valuable resource. For students this included their friends/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

Table 5
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.

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.
Q&A use	Have you ever used Internet Q&A websites (eg 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?

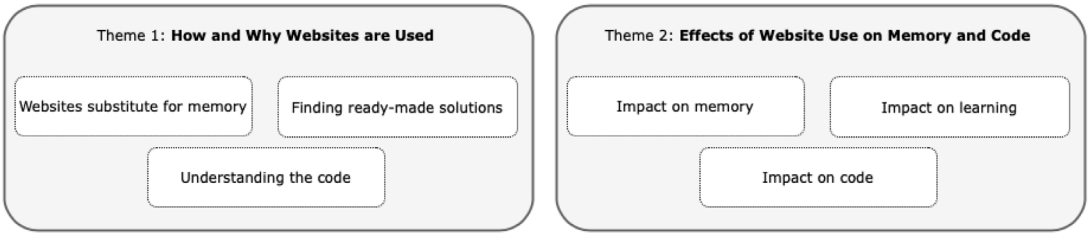

Fig. 2. Themes identified during analysis of interview data. The two primary themes are subdivided into a further three sub-themes.

Table 6
Questions asked in our online survey (excluding demographics). Response types are summarised in parentheses. All questions were established based on findings from the interview study.

Section	Question
Resource use	*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) *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 *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) *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.
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→Rarely) *To what extent do you agree with the following statements about the code snippets you find on the web?* (5-item scale: Strongly Agree→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 *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→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)
Programming and Human Memory	*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 *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.

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 Question and Answer 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)*

4.1.2. 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.

4.1.2.1. 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)*

4.1.2.2. Finding ready-made solutions. 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 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. (P6-Student)

4.1.2.3. 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)

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)

4.1.3. 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.

4.1.3.1. 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 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)

4.1.3.2. 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)*

4.1.3.3. 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 4.1.2.2) 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)*

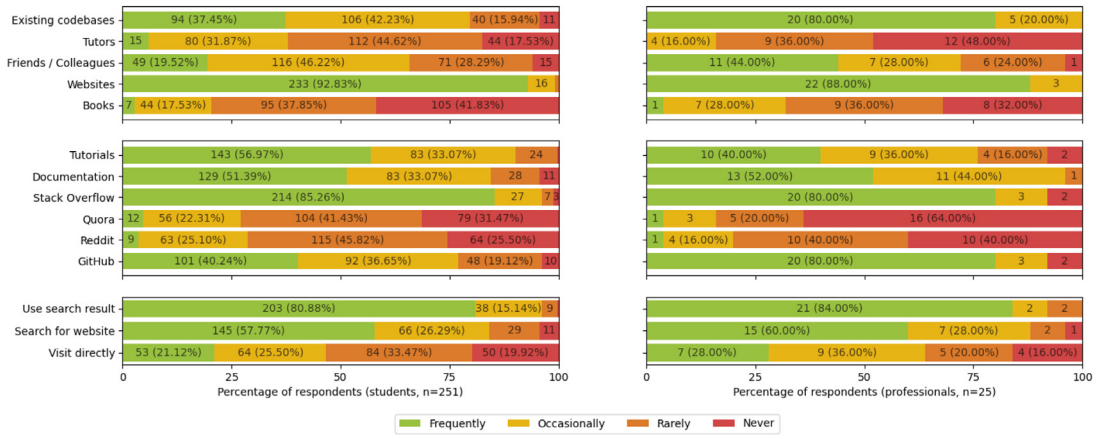

Fig. 3. Resource use (top, R1-5), website use (centre, R6-11), and website access mode (bottom, R12-14) by students (left) and professionals (right); see Table 6 for full question text. For the final three bars, the text 'Use search result' corresponds to the option 'I use a search engine and click whichever results look most relevant' (R14) and 'Search for website' to 'I use a search engine with the intention of finding content from a specific website' (R13). For legibility reasons, very low numbers of responses (<3) are unlabelled on the rightmost plot, likewise some percentages are omitted from value labels (<15).

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.

> *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-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 these 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 "piecemeal" software (P3) 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.

4.2. The survey

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%).[6] The student sample was tightly distributed in age (mean 21–22 years, median 20–21 years), whilst the professional sample 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). Full demographics for both samples are given in Tables 3 and 4.

4.2.1. Resource use

All respondents used at least one of our 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.

Fig. 3 (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).

Looking closer at website use (summarised in Fig. 3, center), we find that Stack Overflow is 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, and half of both samples report frequent use of documentation (students: 51%; professionals: 52%). Quora and Reddit use is low in both samples (frequent or occasional use by 27%/29% of students and 16%/20% of professionals for Quora/Reddit respectively).

Fig. 3, 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

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 (Fig. 3, 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. Fig. 4 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). 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 (Fig. 5). Reported dependency for students was roughly the same irrespective of prior experience (16–20%).

Fig. 6 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, 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). 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 presence of multiple answers and accompanying comments, 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

237 students (94%), and all professionals, responded to questions about perceptions and use of online code snippets. Fig. 7 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%). When asked about the circumstances in which this behaviour occurred (Fig. 8, bottom), 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).

Within both samples, respondents held diverse opinions on the potential negative impacts of copy/pasting online code snippets — Fig. 8 (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%).

[6] The proportion of males participating in this survey reflects high levels of male dominance in the UK IT workforce and in UK undergraduate computer science programmes. WISE reported that in 2019 women made up just 16% of the IT workforce (WISE, 2019); likewise UCAS reported that 17% of 2021 applicants to computer science were female (UCAS, 2021) (data for gender non-binary professionals/applicants were not readily available).

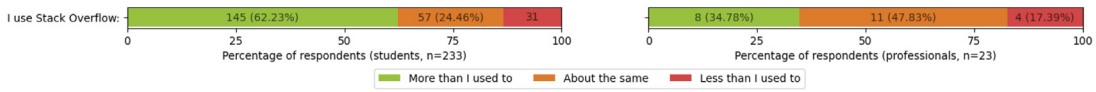

Fig. 4. Reported change in use Stack Overflow, compared to three years ago, for students (left) and professionals (right). See Table 6 (S1) for full question text. For legibility reasons, percentages are omitted from low value labels (<15%).

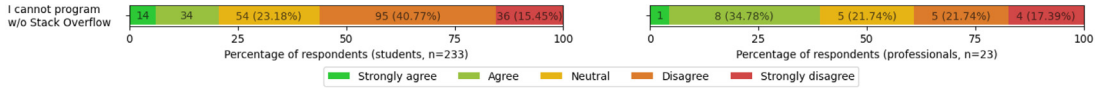

Fig. 5. Reported dependency on Stack Overflow, for students (left) and professionals (right). See Table 6 (S3) for full question text. For legibility reasons, percentages are omitted from low value labels (<15%).

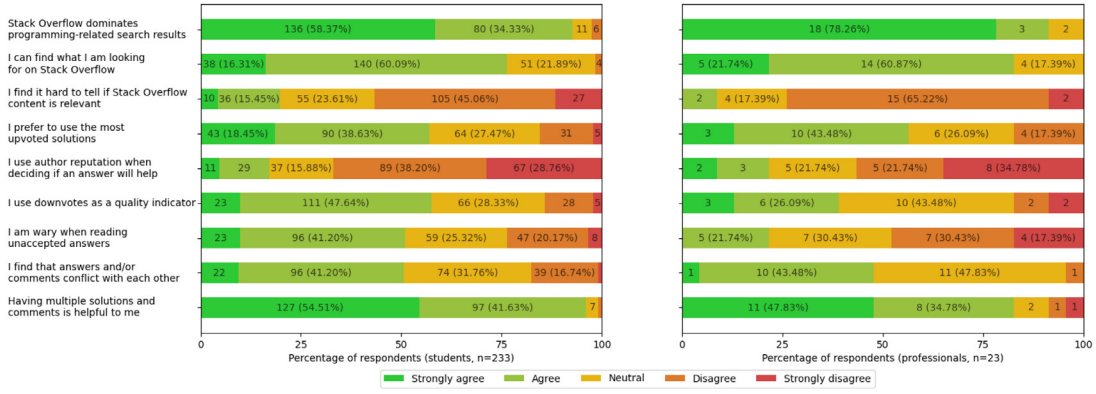

Fig. 6. Level of agreement for students (left) and professionals (right) for statements about Stack Overflow. See Table 6 (S2, S4-S11) for full question text. For legibility reasons, very low numbers of responses (<3) are unlabelled on the rightmost plot, likewise percentages are omitted from low value labels (<15%).

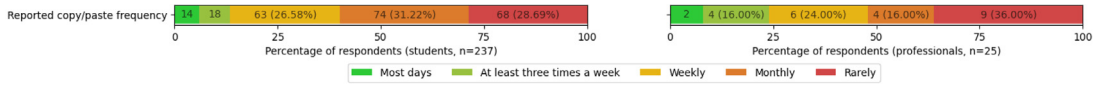

Fig. 7. Frequency with which students (left) and professionals (right) report copying and pasting a code snippet from the web. See Table 6 (C1) for full question text. For legibility reasons, percentages are omitted from low value labels (<15%).

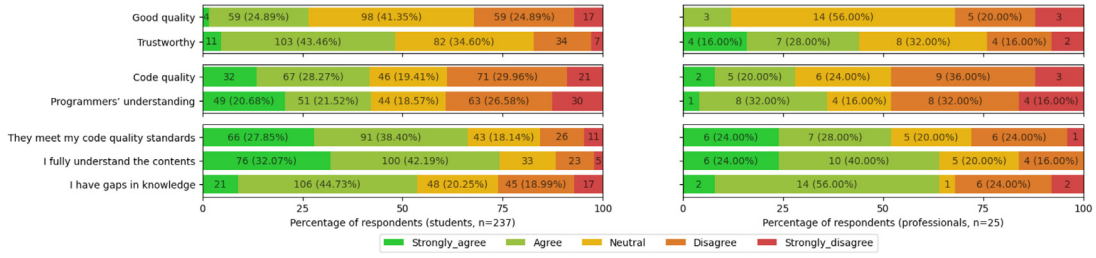

Fig. 8. Level of agreement for students (left) and professionals (right) for statements about code snippets (top, C8 & C2), the negative impacts of copy and pasting code snippets (centre, C7 & C6), and circumstances in which the respondents would copy and paste a code snippet into their own code (bottom, C5, C4 & C3). See Table 6 (C2-8) for full question text. For legibility reasons, percentages are omitted from low value labels (<15%).

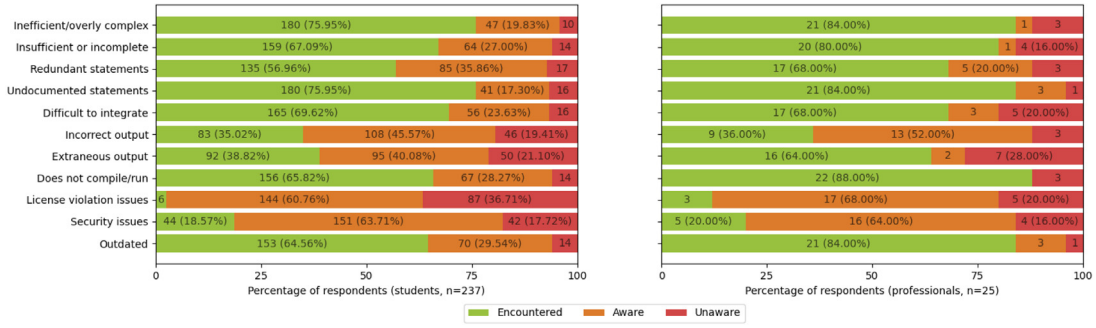

Fig. 9. Issues encountered in online code snippets by students (left) and professionals (right); see Table 6 (C9-18) for full question text. For legibility reasons, percentages are omitted from low value labels (<15%).

Table 7
Summary statistics for the number of different types of code issues that survey respondents had directly encountered, had not encountered but were aware of, and those they were unaware of (see Table 6, C9-18).

		Encountered	Aware	Unaware
Students	Mean (Std.)	5.71 (2.52)	3.92 (2.37)	1.38 (1.66)
	Median (Q1–Q3)	6 (4-8)	4 (2-5)	1 (0-2)
	Min–Max	0-11	0-11	0-11
Professionals	Mean (Std.)	6.88 (2.35)	2.56 (1.19)	1.56 (2.26)
	Median (Q1–Q3)	7 (6-9)	2 (2-3)	1 (0-2)
	Min–Max	1-10	1-5	0-8

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%). 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 Fig. 8 (top)

Seven of the eleven problems 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 (Fig. 9). 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 license violations as a problem in online code snippets.

Professionals were slightly more likely to report encounters with code problems than students (Table 7). From 237 student respondents, twenty-six (10.97%) reported encountering fewer than three problems (from a possible eleven), and seven (2.95%) said they had never encountered any of them. At the other extreme, thirty-five (14.77%) reported encountering more than eight problems (from a possible eleven), and three (1.27%) said they had encountered all of them. By contrast all of the twenty-five professionals had encountered at least one problem and only two (8.00%) had encountered fewer than three; seven (28.00%) had encountered more than eight, and none had encountered them all. Overall awareness of at least some potential problems in online code was high, only two students and no professionals were unaware of more than eight of the potential issues.

Looking more closely at the relationship between problematic code encounters and perceptions of code quality, we find that respondents (students and professionals) who have encountered more than eight of the code problems are very much more likely to disagree that the majority of code snippets are of good quality (C8: strongly disagree/disagree 52%) than those who had encountered fewer than three (C8: strongly disagree/disagree 14%). A smaller difference is seen in the proportion of each group reporting agreement with this same statement (C8: strongly agree/agree 19% vs. 29%). Agreement that copying and pasting online code snippets reduces code quality is also stronger in the group with more than eight different problems encountered (C7: strongly agree/agree 45%) compared to those encountering fewer than three problems (strongly agree/agree 29%) but disagreement rates are similar in both groups (C7: strongly disagree/disagree: 40% and 39% for >8 and <3 respectively).

4.2.4. Programming and human memory

232 students (92%) and twenty-three professionals (92%) responded to questions about the interaction between programming activities, use of online programming content, and human memory.

Fig. 10 (top) 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), and that it is faster to remember programming-related information than it is to look it up (students: 58%; professionals: 48%). Conversely, however, 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). 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.

As seen in Fig. 10 (bottom), 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 availability of relevant online content, there was still value in trying to remember programming concepts. Almost all respondents indicated that online programming content helped to confirm or remind them of information that they already knew (students: 88%; professionals: 96%). 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 there was very little agreement that greater use of online programming content was negatively impacting the respondents' ability to remember (students: 12%; professionals: 9%).

5. Discussion

In this section, we discuss the study results and summarise the findings for each research question.

5.1. RQ1: What resources do people use to support the coding process? How and why do they use them?

We found evidence 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 by 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

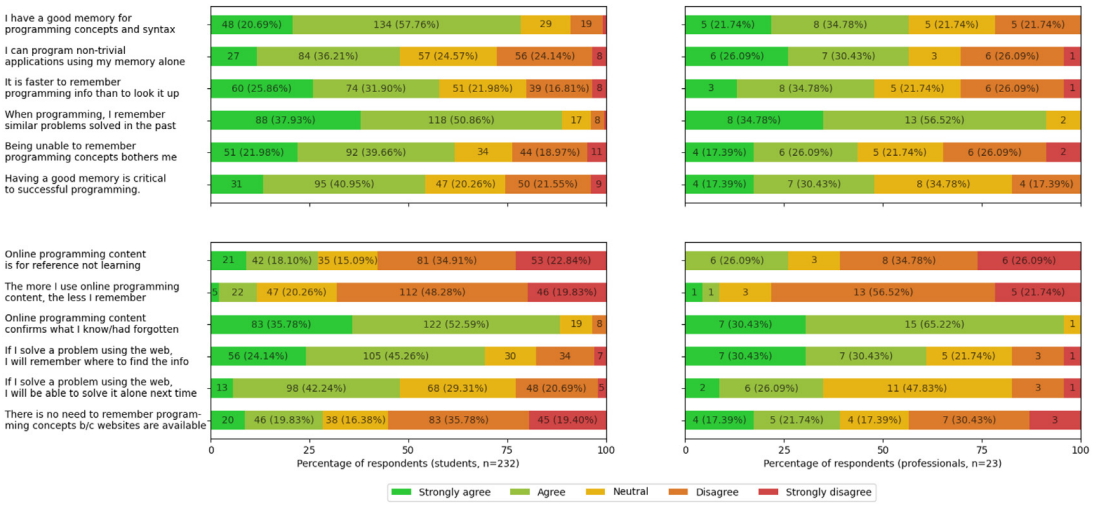

Fig. 10. Statement agreement for students (left) and professionals (right). Statements have been abbreviated, see Table 6 (M1-12) for full text. For legibility reasons, very low numbers of responses (<3) are unlabelled on the rightmost plot, likewise percentages are omitted from low value labels (<15%).

also explain their low usage rates for book use (also seen in other studies (Lausa et al., 2021)), and course tutors (when students complete work outside of teaching sessions). Whilst professionals noted in interviews that working closely with colleagues could be disruptive to others and was not always possible, surveyed professionals actually were more likely to report seeking help from friends/colleagues when compared with student respondents. Student interviewees noted that friends were particularly valuable when web searches did not have anything relevant to specific course assignments, or when they were struggling to understand and integrate online content. Overall though, less than half (44%) our professionals (compared with 20% of students) reported frequently seeking help from colleagues, considerably lower than those observed in prior studies (>80% (Maalej et al., 2014)). This difference, together with the conflict in 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.

Online resources were often accessed via a search engine, with all but one interview participant (P15) reporting use of Google (e.g. *"I just Google it"*, P12; *"Google is your best friend"*, P18). This behaviour is consistent with prior studies (Maalej et al., 2014; Xia et al., 2017), and our interviewees often described queries 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.

Our interview and survey findings both indicate high rates of use of Stack Overflow, mediated through an initial search engine; this again aligns with prior studies that have shown high return rates for Stack Overflow for technical searches (Hora, 2021; Xia et al., 2017), and extensive use of the website in development (Acar et al., 2016). In contrast to Stack Overflow, competitor Q&A websites like Reddit and Quora were rarely mentioned by interviewees and were reportedly rarely or never used by survey respondents. Other websites like documentation and tutorials were used but not as frequently as 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 describe their dependency on Stack Overflow, and our survey findings confirm that their frequency of use for Stack Overflow remains largely static compared to three years prior. Comparison of student and professional survey responses suggest that whilst similar proportions report frequent use of Stack Overflow, and success finding the content they look for on the site, professionals are more likely to agree (and to disagree) that they need the website to program — a majority of 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 may be better addressed by Stack Overflow content (in contrast to fundamentals, which are readily served by books and tutorials). Successful problem solving then acts as a positive reinforcement and encourages habit formation.

Interview respondents noted the value of Stack Overflow features such as multiple answers, comments and up/down votes. However, they also commented on conflict between answers, and experiences where the answer needed to solve their problem was not the one that had been upvoted or accepted. Our surveys show a similar tension, whilst most find having multiple answers and comments helpful, around half observed conflict and a similar proportion reported that 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).

Our survey results indicate that professional developers make equally frequent use of GitHub. This behaviour did not emerge in our interviews (only one professional made any mention of Git, and none discussed GitHub), and was only included in our survey as a result of mentions from student interviewees. Thus, we observe high reported use, but can make little comment on exactly 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 access to codebases, but is likely to be easier to navigate by those who are already familiar with the process of navigating code written by others.

5.2. RQ2: What is the relationship between resource use (during coding tasks) and human memory?

In this study we focus of programmers' perceptions of their memory and resource use. This encompasses understanding of when programmers choose to engage memory in preference to, or supported by, resources. As well as exploration of any perceived negative impacts of resource use on memory for concepts. This research does not set out to empirically measure changes in recall quantity or quality associated with resource.

5.2.1. What is the role of human memory in programming? (RQ2.1)

Roughly half our survey respondents report that it is faster for them to remember programming-related information than 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 that they are able to program non-trivial applications using memory alone. Interestingly, rates of agreement for these two statements differs for students and professionals, with more students reporting a good memory but more professionals reporting being able to program with memory alone (similar conflicts are also seen in some of our other memory-related survey results). Differences between students and professionals in the latter statement seem intuitive and are likely due to the relative inexperience of students. By contrast, the idea that students' would, on the whole, report better memory for programming concepts than professionals seems counter-intuitive, and conflicts with prior empirical evidence (Ichinco and Kelleher, 2017). One possible explanation is that these students are making relative judgements, determining that their memory is good compared to peers, something that is also thought to occur in other cases of counter-intuitive memory reporting (De Winter et al., 2015). Another potential factor is the likely smaller set of syntax and concepts students are expected to remember (i.e. a more "lenient lifestyle" (De Winter et al., 2015)) — since their programming experiences make only light-to-moderate demands on memory, they perceive that they are fulfilling this better than professionals whose memory is under greater load.

5.2.2. What perceived weaknesses or challenges with human memory prompt people to use external resources? (RQ2.2)

Our 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), those impaired by working in multiple programming languages (e.g. those using a new-to-them language, 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 cases, the time taken to switch from programming to information seeking was a potential barrier.

5.2.3. What is the perceived impact of using resources on memory? (RQ2.3)

Interviews suggest that one approach taken by our programmers is to attempt to solve a problem first, and then to utilise online resources when encountering difficulties. Empirical evidence suggests that this is likely to have a positive effect, with those who attempt a programming task independently better able to remember concepts found online (Giebl et al., 2020). Others use the web as a reminder for content they are already familiar with, cognitively offloading storage of the syntactic details (a behaviour also seen in prior studies (Brandt et al., 2009)).

In more generalised domains, there has been some concern that cognitive offloading may negatively impact independent ability to recall (Fisher et al., 2021; Sparrow et al., 2011), and encourages further reliance on online content (Storm et al., 2017). Our results suggest that whilst programmers may see websites as a substitute for memory, they do not perceive that use as negatively impacting their abilities (only around 10% of survey respondents agreed that online content negatively impact their ability to remember). However, not everything seen online is remembered, with only around one-third to a half of survey respondents indicating that they would be able to independently remember how to solve a problem that they had previously solved 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 where to find the information again. 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 a false confidence in their abilities (Fisher et al., 2015, 2021). This attribution error may explain some of the conflict in our interview comments whereby participants (particularly students) describe online content as both contributing to, and impeding, memory and learning.

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

5.3. RQ3: What are the potential impacts of resource use on programmers' code?

This study cannot be used as evidence of a causal relationship between online resource use and programmer outputs. Instead, we focus on the *potential* for harms or benefits by trying to better understand code reuse practices, programmers' experiences of online code, and programmers' perceptions of the relationship between their resource use and code.

5.3.1. To what extent do programmers' reuse code that they encounter online? What safeguards do they put in place when they do this? (RQ3.1)

Our findings indicate that programmers do reuse code that they encounter online, copying and pasting snippets into their own source files; moreover, student interviewees reported explicitly seeking out code snippets with this intention. Student interviewees were motivated to reuse code in order meet their course requirements. Around half of 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. Within our 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 is perhaps due to the more sporadic nature of student programming activities — while 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).

Our participants identified a variety of potential negative impacts (e.g. on code outputs, on learning, on course outcomes). One approach to mitigate or safeguard against these was to develop understanding of how the code snippets they were reusing functioned (either at a macro level, or line-by-line). Levels of understanding for copied code snippets varied – interview participants generally indicated that there were good reasons to try and understand code found online, but this was not always possible

– some code was beyond students' current competency, and time pressures created by deadlines meant that students were unable to spend the time needed to understand the code they were using. Both interviews and surveys show that both professionals and students understand online code when they reuse it, but our survey responses indicated that professionals were more willing to copy code that they did not fully understand. This finding is consistent with evidence from studies of program comprehension in which novices expend more effort in understanding code, with experts using a variety of strategies including higher level abstraction rather than detailed line-by-line understanding (LaToza et al., 2007; Maalej et al., 2014). The finding may also be a reflection of students reasons for writing code in the first place; since students coding assignments were intended to facilitate learning, there may have been a greater desire for understanding. Furthermore, since the survey was issued by a team of university researchers, some students may have felt the need to report what they perceived to be the desired answer, rather than an honest one.

5.3.2. To what extent do programmers' encounter and identify problematic code online? (RQ3.2)

One potential risk when copying online snippets, is the introduction of problematic code into developers' software (Abdalkareem et al., 2017; Acar et al., 2016; Fischer et al., 2017). Our 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, our interviews indicate 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 our survey question about negative impacts of code snippet use on code quality (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 against negative impacts of code reuse.

We asked interview participants whether, in their experiences of using online code snippets, they had observed issues previously noted in the literature (e.g. security vulnerabilities (Acar et al., 2016; Fischer et al., 2017; Licorish and Nishatharan, 2021; Meng et al., 2018), license issues (Ragkhitwetsagul et al., 2021), and poor code quality (Ragkhitwetsagul et al., 2021; Treude and Robillard, 2017)). Our interviewees reported eleven distinct issues with code, which we list in Table 8. Survey participants were asked about each issue elicited in the interviews. 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 of the 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 more frequently reported by students simply due to their relative inexperience — i.e., it may not be that code on the platform generally *is* hard to integrate, but rather than students have relatively little experience at integrating existing code into their own projects. Similarly, it is likely that lower report rates by students for other issues results from (i) fewer interactions with the site than professionals,[7] and (ii) a reduced ability to recognise these other issues (such as a 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 our surveys and interviews, security issues and license violations had markedly lower report rates than other issues. For students, this again is at least partially attributable to inexperience (both in terms of being in circumstances where security/licensing issues would arise, and in terms of ability to recognise the issues when encountered). For professionals however, reported encounters are low despite prior studies that show evidence that developers do both find, and utilise, problematic code (Acar et al., 2016; Fischer et al., 2017; Meng et al., 2018; Ragkhitwetsagul et al., 2021).

Stack Overflow provides a number of quality indicators that could help inform users when assessing answers and associated code snippets. As noted in Section 5.1, although around half of survey respondents took them into account when reading Stack Overflow content, our participants did not consider upvotes/downvotes sufficient — this aligns with prior work indicating that upvotes/downvotes are not sufficient to determine answer (and by extension, code snippet) quality (Zhang et al., 2018). By contrast the reputation metric, which has previously been shown to be indicative of good quality answers (Movshovitz-Attias et al., 2013), was considered by only a minority of student and professional survey respondents. Students were more likely to report wariness when considering unaccepted answers on Stack Overflow, an indicator that our professionals largely seemed to ignore, but current evidence suggests that there are no quality differences between accepted and unaccepted answers (Meldrum et al., 2020).

5.3.3. What are the perceived impacts of code reuse on code quality? (RQ3.3)

Our survey results show some interesting tensions between respondents' perceptions (C8) and experiences (C9–C19) of code snippet quality, and of their belief that reuse of online code reduces code quality (C7).

On average, students had encountered slightly fewer code problems than professionals (mean/median 6 for students compared to 7 for professionals), and were more likely to agree that code snippets were of good quality. While there were no difference in levels of disagreement between professionals and students, disagreement with this statement appears to be strongly aligned with the number of problematic code encounters — those who have encountered more problems are less likely to agree that the majority of snippets are of good quality.

Consulting (or even copy and pasting) poor quality online code does not guarantee poor-quality developer output. In interviews, participants described refactoring and correcting code they had copied from the web — although this was only possible where they recognised problems in the copied code and also had time available to make modifications. Despite this, among professionals there was little difference in the rates of agreement that copy and paste reduced code quality than those that had indicated concerns about the quality of online code (C8: agree/strongly agree 28% vs C7: disagree/strongly disagree 32%). Moreover, greater proportions of student survey respondents agreed that copy and paste reduced code quality than had indicated concerns about the quality of online code (C8: agree/strongly agree 42% vs C7: disagree/strongly disagree 32%). Together, these results suggest that neither professionals nor students believe that problems in online code snippets will be eradicated when those snippets are copied and refactored into subsequent projects.

[7] Not every encounter with Stack Overflow will result in a problematic code snippet, thus a reasonable level of site usage may be needed before encountering any issues. Note however that measures of the prevalence of problematic code on Stack Overflow are highly variable; for example Digkas et al. report that the "vast majority" of examined snippets had no rule violations (as measured by tool SonarQube (Digkas et al., 2019)), whereas Bafatakis et al. report that less than seven percent of sampled Python did not violate coding style rules (Bafatakis et al., 2019).

Table 8
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.

Issue	Interview occurrences		Survey occurrences (%)		See also
	Students	Professionals	Students	Professionals	
Inefficient or overly complex code	✓	✓	75.95 (95.78)	84.00 (88.00)	Treude and 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 and Robillard (2017), Wu et al. (2019), Yang et al. (2016), Zhang et al. (2018)
Code that does not compile or does not run	✓	-	65.82 (94.09)	88.00 (88.00)	Bafatakis et al. (2019), Yang et al. (2016)
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. (2021), Wu et al. (2019), Zhou and Walker (2016)
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), Bai et al. (2019), Fischer et al. (2017), Licorish and Nishatharan (2021), Meng et al. (2018)
License violations	✓	✓	2.53 (63.29)	12.00 (80.00)	An et al. (2017), Ragkhitwetsagul et al. (2021), Wu et al. (2019)

5.4. Sample differences

Our research 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 we deliberately distributed the survey to previous interviewees). We note some differences in our findings between these samples as follows.

5.4.1. 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 to report copying and pasting code that they did not understand, when compared with survey respondents.
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 code issues on Stack Overflow were higher in surveys than would be suggested by the frequency of similar interview comments.

We believe that 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 our 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, we do see more frequent (47%) (and frequent/occasional use, 84%) compared to the remaining student respondents (39% frequent, 75% frequent or occasional).

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

(4) perhaps reflects a difference in between abstractly being asked if they issues had been encountered versus asking participants to articulate specific experiences. Survey participants may have a general sense that the issues must have been encountered, but in the absence of being able to draw on a specific example, our 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.

5.4.2. Between students and professionals

1. Developers make use of existing codebases, and online code repositories that is comparable with their use of Stack Overflow, students do not.
2. Developers are more likely to report dependence on Stack Overflow, are more likely to engage in frequent copy and paste, and report a poorer memory for programming concepts
3. Students are more likely to try and understand a code snippet prior to reuse.
4. Professionals are both more likely to have encountered and more aware of license violations on Stack Overflow.

(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 5.2, (2) is potentially a result of greater demand on professional developers, and Section 5.3 notes that (3) may be influenced by students being particularly motivated to understand, and by response bias. (4) is explained by comments in our interviews that highlight the potential legal repercussions of using incorrectly licensed code in commercial software.

5.5. Limitations and threats to validity

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

To minimise threats to *construct validity*, we used a pilot study to validate and refine the initial interview questions. Interview responses were then used to develop the survey questions, and these too were then subject to a pilot study. One threat to construct validity may have emerged as a property of interviewing

student participants from our own university. Whilst we 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 therefore may have led students to minimise disclosure of these behaviours during their interviews.

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

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

We recruited UK participants from two populations: undergraduate students studying on a computer science or software engineering programme, and software developers who had been in a professional role for at least one year. Student interview participants were recruited exclusively from the University of Manchester, and our interview sample did not include the most experienced student programmers (third and fourth years). To maximise the representativeness of our survey sample, we deliberately approached a large pool of computer science departments (168 from approximately 200 offering relevant courses). Our final survey sample includes respondents from ten UK universities. Concepts emerging from interviews with first and second years were adopted in the survey (which spanned all years of undergraduate study). 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 our interview and survey results, we are confident that results are representative of the overall populations from which the samples are drawn.

Both our interview and 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 our results. We do, however, note considerable alignment in the results from our students and professionals, implying that many behaviours and perceptions likely generalise across groups. To address any potential bias, and the previously described gaps in our sampling, future work may want to explore approaches for gaining a richer perspective on the different perceptions of students at differing stages of study, and to engage professional developers at scale. Furthermore, whilst our samples cover two very distinct groups of programmers (students and professionals), other studies may target still more populations that engage in programming activities — these may be subdivisions of our own populations (e.g. recent graduates vs. very established professionals, professionals in a variety of sectors) or populations omitted in this work (e.g. hobby programmers). Moreover, given the UK-centric nature of this research, further data is needed to understand the degree to which the perceptions and behaviours articulated by our samples are reflective of other populations, particularly given the differences seen in international degree-level education programmes.

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

As noted in Section 3.2.3, a small number of exclusions were made during review of the survey data. Three further student responses were minimally cleaned during analysis – these were students who had indicated no prior programming experience prior to enrolling on their current programme of study and also indicated one of 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 limited incentive to deliberately provide inaccurate responses, and the relatively short duration of the survey should have minimised the effect of fatigue. All quantitative data analysis scripts have been made available for validation and reuse.

5.6. Recommendations and future work

In response to the our findings, we reflect on applications (i.e. lessons for student programmers, educators, and industry). We also highlight areas where additional research is needed to better develop understanding.

5.6.1. To novice programmers and students

We make no attempt to correlate our student programmers' grades with their perceptions and behaviours. However, our results suggest some practice amongst students that could be helpful in encouraging understanding and development. Specifically, a number of our interviewees reported applying their knowledge in problem solving prior to web search. Likewise, some interviewees described the effort taken to ensure understanding of a code snippet prior to reuse (this process was sometimes realised through refactoring). We suggest that these examples be held up as good practice for both students in formal education and novices developing their skills independently.

- R1 When presented with a novel programming problem, students should seek to solve the problem independently before resorting to web use.
- R2 Students should view copy-and-paste as an opportunity for learning. Refactoring of online code snippets should be undertaken with the explicit goal of ensuring that code written by more-experienced others is fully understood.

When presented with a specific list of issues, significant majorities of students report both awareness and direct experience of problematic online code. However, discussion of these in interviews were often shallow, and centred on a fairly small number of examples of bad practice covered in their courses (e.g. poor variable naming, lack of comments). With support from their educators, we therefore recommend that

- R3 When incorporating online code snippets into a project, students should engage in deliberate assessment of the code to identify potential problems.

5.6.2. To educators

Our results give a very strong indication that website use, particularly Stack Overflow use, is in heavy use by computer science students. It is neither practicable nor desirable to dissuade students' from use of such online resources, particularly since our research with professionals suggest the sites will continue to be an important part of development post-graduation. Instead, we suggest that educators need to consider the implications of Stack Overflow use on curricula, assessment and teaching. Specifically, our results prompt us to recommend that:

R4 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 to determine when (and which parts of) code snippets are relevant, and to recognise problematic code.

5.6.3. To professional programmers and industry

In interviews, professionals (even those with decades of experience) expressed a desire for close working with colleagues, both to draw on expertise and because "it's quicker" (P16). However, professional programmers expressed a greater reluctance to "bother" colleagues with their problems (P15). Agile practices such as pair- and mob-programming[8] provide opportunities for co-development but many organisations are reluctant to radically transform their development processes. A cost–benefit analysis of co-development is beyond the scope of this work, but our findings indicate that:

R5 Development teams should explore opportunities for co-development, irrespective of individuals' development experience.

R6 Teams should take care to avoid creating a culture in which individuals are reluctant to seek advice from others.

Our professionals were generally able to identify problematic code snippets, but expressed particular uncertainty around licensing. During interviews, none of the professionals expressed any concern about the potential implications for license issues in a commercial setting, instead discussion centred on the introduction of buggy code into their systems. Thus, we recommend that:

R7 Developers should continue in their cautious approach to Stack Overflow and online snippets with respect to the potential for introducing bugs into code.

R8 Professional organisations should ensure developers have adequate training in software licensing and its implications for code reuse.

R9 Organisations with large internally-developed codebases should consider mechanisms for making those codebases a viable resource for their software developers. This may introduce a need for better documentation and/or indexing.

Our results also show few professionals report encounters with security vulnerable code snippets. We did not ask our developers if they regularly used security libraries, so cannot say whether this low response rate reflects the true rate of encounter. We do however recommend that:

R10 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 familiarise themselves with common relevant problems in online code snippets and be attentive when sourcing code online.

5.7. Future work

Our findings on resource use largely align with prior work and are consistent across our data. However, our results omit some resources — most notably video-content, was not mentioned in our interviews and thus was not an area of exploration in our survey. Given the presence of other studies indicating a growth in online video use by student programmers (Lausa et al., 2021), and the potential for these videos to act as a source of code snippets (Khandwala and Guo, 2018; Yadid and Yahav, 2016), such videos would be a target for future research. Furthermore, our results show some conflict around the use of existing codebases, and of GitHub in particular. Repeating the more expansive survey questioning that we made for Stack Overflow, for both video and these latter two sources, may be helpful in understanding how student and professional developers leverage a larger set of dominant resources.

Building on the above, we also propose further study of copy-and-paste behaviours in programming task completion. Observation methods and instrumentation of development environments may be particularly useful in building a rich picture of how copy-and-paste behaviours are realised in everyday programming activities, and whether these differ based on the utilised web source type. Observation and think-aloud studies may also be well-suited to understanding programmers' responses when presented with problematic code snippets. Future study should further focus on examining programmers' resulting code to look for any potential implications, such as problematic online code, while coding using the web.

Differences observed between our samples, and between questions on our survey related to human memory, raise a number of questions that warrant further exploration. What is the role of memory and metamemory in the execution of programming tasks? To what extent do self-reports of programming-related memory competency and objective measures of programming-related memory align? How does programming experience interact with memory for programming-related content? Is there evidence of a 'Google effect' (Sparrow et al., 2011; Storm et al., 2017) when using the web, and particularly Stack Overflow, during development?

6. Conclusion

We collected qualitative and quantitative data about use of online resources during programming tasks, and the perceived effects of that use on programmers' memory and code. Our article analyses data from 18 interviewees and 276 survey respondents, including both student and professional programmers.

We find that web use is ubiquitous as a primary resource for coding, and that this is 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).

Our 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 in order to learn from what they had seen online. Our participants did, however, report a greater tendency to recall where they had found information, rather than the information itself.

High proportions of our participants had encountered problematic code online, and we highlight eleven specific classes of problem that were elicited in our interviews and survey. Despite these problems, the majority of our programmers reported engaging in frequent reuse of online code snippets, and did not always understand the code that they were reusing. Qualitative data suggests that professional programmers felt able to discern between good and problematic online content. Metrics provided by Stack Overflow to support this process of discernment (e.g. reputation, upvotes, downvotes, accepted answers) were found to be of limited value to our participants.

In response to these findings, we recommend that student programmers prioritise independent problem solving, and see any web use as an opportunity to learn through refactoring and

[8] https://mobprogramming.org

understanding unfamiliar code. Students (supported by educators) must develop skills to assess online code snippets for relevance and for potential problems. By contrast, our recommendations for professionals are that they foster a more collaborative working culture that encourages co-development and provides tools to enable search and reuse of company codebases. Cautious reuse of code snippets from Stack Overflow and the wider Web can continue, but professionals need further training in software licensing and its implications for code reuse.

CRediT authorship contribution statement

Omar Alghamdi: Conceptualization, Methodology, Validation, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization, Project administration, Funding acquisition. **Sarah Clinch:** Conceptualization, Methodology, Validation, Data curation, Writing – review & editing, Visualization, Supervision, Project administration. **Rigina Skeva:** Methodology. **Caroline Jay:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://doi.org/10.48420/c.6148593 How are websites used during development and what are the implications for the coding process? : Interview and Survey Study (Original data).

Acknowledgements

The authors would like to thank Lorraine Underwood, and all the directors of undergraduate studies, who helped recruit to participants, and all the participants who kindly participated in our study. This research was sponsored by Saudi Electronic University, College of Computing and Informatics, Kingdom of Saudi Arabia.

References

Abdalkareem, R., Shihab, E., Rilling, J., 2017. On code reuse from stackoverflow: An exploratory study on android apps. Inf. Softw. Technol. 88, 148–158.
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, IEEE, pp. 289–305.
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). IEEE, pp. 283–293.
Baddeley, A.D., Eysenck, M.W., Anderson, M.C., 2020. Memory, third ed. Routledge, Taylor & Francis Group.
Bafatakis, N., Boecker, N., Boon, W., Salazar, M.C., Krinke, J., Oznacar, G., White, R., 2019. Python coding style compliance on stack overflow. In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories. MSR, IEEE, pp. 210–214.
Bai, W., Akgul, O., Mazurek, M.L., 2019. A qualitative investigation of insecure code propagation from online forums. In: 2019 IEEE Cybersecurity Development (SecDev). IEEE, ISBN: 978-1-5386-7289-1, pp. 34–48.
Bai, G.R., Kayani, J., Stolee, K.T., 2020. How graduate computing students search when using an unfamiliar programming language. In: Proceedings of the 28th International Conference on Program Comprehension. In: ICPC '20, ACM, ISBN: 978-1-4503-7958-8, pp. 160–171.
Baxter, I., Yahin, A., Moura, L., Sant'Anna, M., Bier, L., 1998. Clone detection using abstract syntax trees. In: Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272). IEEE Comput. Soc, Bethesda, MD, USA, ISBN: 978-0-8186-8779-2, pp. 368–377.
Bidlake, L., Aubanel, E., Voyer, D., 2020. Systematic literature review of empirical studies on mental representations of programs. J. Syst. Softw. 165, 110565.
Börstler, J., MacKellar, B., Störrle, H., Toll, D., van Assema, J., Duran, R., Hooshangi, S., Jeuring, J., Keuning, H., 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. ACM Press, ISBN: 978-1-4503-5627-5, pp. 70–85.
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. ACM, New York, NY, USA, ISBN: 9781605582467, pp. 1589–1598.
Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3 (2), 77–101.
Braun, V., Clarke, V., 2012. Thematic analysis. In: APA Handbook of Research Methods in Psychology, Vol 2: Research Designs: Quantitative, Qualitative, Neuropsychological, and Biological. American Psychological Association, ISBN: 978-1-4338-1005-3, pp. 57–71.
Chatterjee, P., Kong, M., Pollock, L., 2020. Finding help with programming errors: An exploratory study of novice software engineers' focus in Stack Overflow posts. J. Syst. Softw. 159, 110454.
Clinch, S., Cortis Mack, C., Ward, G., Steeds, M., 2021. Technology-mediated memory impairment. In: Dingler, T., Niforatos, E. (Eds.), Technology-Augmented Perception and Cognition. Springer International Publishing, ISBN: 978-3-030-30457-7, pp. 71–124.
De Winter, et al., 2015. On the paradoxical decrease of self-reported cognitive failures with age. Ergonomics 58 (9), 1471–1486.
Digkas, G., Nikolaidis, N., Ampatzoglou, A., Chatzigeorgiou, A., 2019. Reusing code from StackOverflow: The effect on technical debt. In: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications. SEAA, pp. 87–91.
Ericsson, K.A., Kintsch, W., 1995. Long-term working memory. Psychol. Rev. 102 (2), 211–245.
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). IEEE, pp. 170–181.
Feitosa, D., Ampatzoglou, A., Gkortzis, A., Bibi, S., Chatzigeorgiou, A., 2020. Code reuse in practice: Benefiting or harming technical debt. J. Syst. Softw. 167, 110618.
Ferguson, A.M., McLean, D., Risko, E.F., 2015. Answers at your fingertips: Access to the internet influences willingness to answer questions. Conscious. Cogn. 37, 91–102.
Fischer, F., Böttinger, K., Xiao, H., Stransky, C., Acar, Y., Backes, M., Fahl, S., 2017. Stack overflow considered harmful? the impact of copy& paste on Android application security. In: 2017 IEEE Symposium on Security and Privacy. SP, IEEE, pp. 121–136.
Fisher, M., Goddu, M.K., Keil, F.C., 2015. Searching for explanations: How the internet inflates estimates of internal knowledge. J. Exp. Psychol.: General 144 (3), 674.
Fisher, M., Smiley, A.H., Grillo, T.L.H., 2021. Information without knowledge: the effects of internet search on learning. Memory 1–13.
Fritz, T., Murphy, G.C., Hill, E., 2007. Does a programmer's activity indicate knowledge of code? 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. In: ESEC-FSE '07, Association for Computing Machinery, pp. 341–350.
Giebl, S., Mena, S., Storm, B.C., Bjork, E.L., Bjork, R.A., 2020. Answer first or google first? Using the internet in ways that enhance, not impair, one's subsequent retention of needed information. Psychol. Learn. Teach. 20 (1), 58–75.
Gkortzis, A., Feitosa, D., Spinellis, D., 2021. Software reuse cuts both ways: An empirical analysis of its relationship with security vulnerabilities. J. Syst. Softw. 172, 110653.
Habchi, S., Moha, N., Rouvoy, R., 2021. Android code smells: From introduction to refactoring. J. Syst. Softw. 177, 110964.
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, IEEE, pp. 317–328.
Hucka, M., Graham, M.J., 2018. Software search is not a science, even among scientists: A survey of how scientists and engineers find software. J. Syst. Softw. 141, 171–191.
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). IEEE, ISBN: 978-1-5386-0443-4, pp. 37–41.
Kapser, C.J., Godfrey, M.W., 2008. "Cloning considered harmful" considered harmful: Patterns of cloning in software. Empir. Softw. Eng. 13 (6), 645–692.
Keuning, H., Heeren, B., Jeuring, J., 2017. Code quality issues in student programs. In: Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education. ACM, ISBN: 978-1-4503-4704-4, pp. 110–115.

Khandwala, K., Guo, P.J., 2018. Codemotion: expanding the design space of learner interactions with computer programming tutorial videos. In: Proceedings of the Fifth Annual ACM Conference on Learning At Scale. pp. 1–10.

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. ACM, New York, NY, USA, ISBN: 978-1-4503-7686-0, pp. 86–94.

Kitchenham, B., Pfleeger, S.L., 2003. Principles of survey research: Part 6: Data analysis. ACM SIGSOFT Softw. Eng. Notes 28 (2), 24–27.

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. IEEE, ISBN: 978-0-7695-3353-7, pp. 57–66.

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.

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. In: ESEC-FSE '07, Association for Computing Machinery, New York, NY, USA, ISBN: 9781595938114, pp. 361–370.

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. In: ICSE '06, Association for Computing Machinery, New York, NY, USA, ISBN: 978-1-59593-375-1, pp. 492–501.

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.

Lazar, J., Feng, J.H., Hochheiser, H., 2010. Research Methods in Human-Computer Interaction, second ed. John Wiley & Sons Inc., ISBN: 978-0-470-72337-1.

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.

Lozano, A., Wermelinger, M., 2008. Assessing the effect of clones on changeability. In: 2008 IEEE International Conference on Software Maintenance. pp. 227–236.

Maalej, W., Tiarks, R., Roehm, T., Koschke, R., 2014. On the comprehension of program comprehension. ACM Trans. Softw. Eng. Methodol. 23 (4), 1–37.

Macias, C., Yung, A., Hemmer, P., Kidd, C., 2015. Memory strategically encodes externally unavailable information. In: CogSci.

Meldrum, S., Licorish, S.A., Owen, C.A., Savarimuthu, B.T.R., 2020. Understanding stack overflow code quality: A recommendation of caution. Sci. Comput. Program. 199, 102516.

Meng, N., Nagy, S., Yao, D., Zhuang, W., Arango-Argoty, G., 2018. Secure coding practices in java: Challenges and vulnerabilities. In: 2018 IEEE/ACM 40th International Conference on Software Engineering. ICSE, IEEE, pp. 372–383.

Michaels, R., Tula, N., Ramisetty-Mikler, S., Nurmuradov, D., Bryce, R., 2020. An empirical study of how novice programmers search the web for help. J. Comput. Sci. Coll. 42.

Mondal, M., Rahman, M.S., Roy, C., Schneider, K., 2018. Is cloned code really stable? Empir. Softw. Eng. 23.

Movshovitz-Attias, D., Movshovitz-Attias, Y., Steenkiste, P., Faloutsos, C., 2013. Analysis of the reputation system and user contributions on a question answering website: StackOverflow. In: Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. ASONAM '13, Association for Computing Machinery, New York, NY, USA, pp. 886–893.

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, IEEE, pp. 25–34.

Nishi, M.A., Ciborowska, A., Damevski, K., 2019. Characterizing duplicate code snippets between Stack Overflow and tutorials. In: Proceedings of the 16th International Conference on Mining Software Repositories. IEEE Press, pp. 240–244.

Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G., Oliveto, R., 2021. Toxic code snippets on stack overflow. IEEE Trans. Softw. Eng. 47 (3), 560–581.

Risko, E.F., Gilbert, S.J., 2016. Cognitive offloading. Trends in Cognitive Sciences 20 (9), 676–688.

Robins, A.V., Margulieux, L.E., Morrison, B.B., 2019. Cognitive sciences for computing education. In: Robins, A.V., Fincher, S.A. (Eds.), The Cambridge Handbook of Computing Education Research. In: Cambridge Handbooks in Psychology, Cambridge University Press, Cambridge, UK, ISBN: 978-1-108-49673-5, pp. 231–275.

Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. 14 (2), 131–164.

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. In: ESEC/FSE 2015, ACM, New York, NY, USA, ISBN: 9781450336758, pp. 191–201.

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.

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, IEEE, pp. 308–311.

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.

Stack Exchange Inc., 2022. Stack exchange data explorer. https://data.stackexchange.com, rev 2021.6.16.81, accessed: 2022-06-16.

Storm, B.C., Stone, S.M., Benjamin, A.S., 2017. Using the internet to access information inflates future use of the internet to access other information. Memory 25 (6), 717–723.

Treude, C., Robillard, M.P., 2017. Understanding stack overflow code fragments. In: 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, ISBN: 978-1-5386-0992-7, pp. 509–513.

UCAS, 2021. UCAS record: 82% increase in UK women applying to IT degrees. https://www.fenews.co.uk/education/ucas-record-82-increase-in-uk-women-applying-to-it-degrees/, (Accessed: 2022-07-08).

Ward, A.F., 2013. Supernormal: How the internet is changing our memories and our minds. Psychol. Inq. 24 (4), 341–348.

Wegner, D.M., Giuliano, T., Hertel, P.T., 1985. Cognitive interdependence in close relationships. In: Compatible and Incompatible Relationships. Springer, pp. 253–276.

WISE, 2019. 2019 Workforce statistics – one million women in STEM in the UK. https://www.wisecampaign.org.uk/statistics/2019-workforce-statistics-one-million-women-in-stem-in-the-uk/, (Accessed: 2022-07-08).

Wu, Y., Wang, S., Bezemer, C.-P., Inoue, K., 2019. How do developers utilize source code from Stack Overflow? Empir. Softw. Eng. 24 (2), 637–673.

Xia, X., Bao, L., Lo, D., Kochhar, P.S., Hassan, A.E., Xing, Z., 2017. What do developers search for on the web? Empir. Softw. Eng. 22 (6), 3149–3185.

Yadid, S., Yahav, E., 2016. Extracting code from programming tutorial videos. In: Proceedings of the 2016 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. pp. 98–111.

Yang, D., Hussain, A., Lopes, C.V., 2016. From query to usable code: An analysis of stack overflow code snippets. In: 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories. In: MSR '16, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450341868, pp. 391–401.

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). IEEE, pp. 280–290.

Yu, C.S., Treude, C., Aniche, M., 2019. Comprehending test code: An empirical study. In: 2019 IEEE International Conference on Software Maintenance and Evolution. ICSME, pp. 501–512.

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, IEEE, pp. 886–896.

Zhou, J., Walker, R.J., 2016. API deprecation: A retrospective analysis and detection method for code examples on the web. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. In: FSE 2016, Association for Computing Machinery, New York, NY, USA, ISBN: 9781450342186, pp. 266–277.

Omar Alghamdi is a Ph.D. candidate in the Department of Computer Science at the University of Manchester, as a member of the Human–Computer Systems group. His research work focuses on the implications of online resources on programmers' memory and on the code that is produced and used online. He is experienced in mixed-methods research, involving a combination of qualitative and quantitative studies with programmers.

Sarah Clinch is a Senior Lecturer in the Human–Computer Systems group at The University of Manchester. They received the ir Ph.D. in Computer Science from Lancaster University. Clinch is known internationally for their work on networked digital displays (including authorship of a Synthesis Lecture on the topic). Their current research centres on the role of pervasive technologies to influence cognition and behaviour. Clinch is on the editorial board for IEEE Pervasive and PACM IMWUT and was an inaugural member of the ACM Future of Computing Academy.

Dr. Rigina Skeva is a lecturer in the Department of Computer Science at the University of Manchester, as a member of the Human–Computer Interactions research group. Her research work focuses on the design and evaluation of Virtual Reality Therapies for the treatment of psychological conditions and the study of the cognitive responses of humans in their interactions with technological systems. She is experienced in mixed-method research, involving a combination of qualitative and experimental studies and patient and public involvement and engagement approaches.

Caroline Jay is a Professor of Computer Science and Head of Research in the School of Engineering at the University of Manchester. She is qualified as both a Psychologist (B.A., CPsychol) and Computer Scientist (M.Sc., Ph.D.), and undertakes research crossing these domains. She is Research Director of the Software Sustainability Institute, and a keen advocate for open and reproducible science. Caroline is a Fellow of the Alan Turing Institute, where she undertakes research looking at the relationship between the environment and human health.