Understanding Wikipedia as a Resource for Opportunistic Learning of Computing Concepts

Martin P. Robillard
martin@cs.mcgill.ca
School of Computer Science
McGill University
Montréal, QC, Canada

Christoph Treude
christoph.treude@adelaide.edu.au
School of Computer Science
University of Adelaide
Adelaide, SA, Australia

ABSTRACT

Posts on on-line forums where programmers look for information often include links to Wikipedia when it can be assumed the reader will not be familiar with the linked terms. A Wikipedia article will thus often be the first exposure to a new computing concept for a novice programmer. We conducted an exploratory study with 18 novice programmers by asking them to read a Wikipedia article on a common computing concept that was new to them, while using the think-aloud protocol. We performed a qualitative analysis of the session transcripts to better understand the experience of the novice programmer learning a new computing concept using Wikipedia. We elicited five themes that capture this experience: Concept Confusion, Need for Examples, New Terminology, Trivia Clutter, and Unfamiliar Notation. We conclude that Wikipedia is not well suited as a resource for the opportunistic learning of new computing concepts, and we recommend adapting information sharing practices in on-line programmer communities to better account for the learning needs of the users.

CCS CONCEPTS

• **Social and professional topics → Informal education**; **Computing literacy**.

KEYWORDS

self-regulated learning; Wikipedia; computing concepts

ACM Reference Format:
Martin P. Robillard and Christoph Treude. 2020. Understanding Wikipedia as a Resource for Opportunistic Learning of Computing Concepts. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE'20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366832

1 INTRODUCTION

Programmers are continually in situations where they must not only acquire technical information, but also discover what they do not know and learn new concepts. This problem is especially acute for novice programmers working outside a structured learning environment [21].

Helping novice programmers orient themselves in a knowledge curriculum that is both enormous and in continual evolution is challenging. Search tools can help, but their effectiveness depends on the ability of the user to issue proper queries and to assess the relevance and quality of the results [32]. Quality answers on on-line forums such as Stack Overflow can help users become aware of what they need to know to solve a programming task, but often the answers will refer to unfamiliar concepts. The purpose of our research is to help novice programmers learn computing concepts related to their task.

Wikipedia is a well-known source of information that can be used to opportunistically learn about new concepts, including computing topics. With the term *opportunistic*, we refer to an informal self-regulated learning context where a programmer attempts to learn a new concept based on emergent information needs related to a programming task [5]. As a resource for self-regulated learning, Wikipedia is controversial. Its own mission statement indicates that "it is not a textbook" [33], and correspondingly its articles do not typically satisfy pedagogical principles expected from learning resources for programmers [18]. The quality of some Wikipedia articles is also questioned [8]. Despite these obstacles, Wikipedia remains a prominent learning resource in computing because of, among others, the practice of *linking to Wikipedia* from on-line forums where programmers look for information. By linking to Wikipedia, entries on these forums, such as Stack Overflow or Reddit, "often use technical terms or acronyms and include a Wikipedia link in lieu of defining these terms" [30]. We recently found that an estimated one third of links to selected Wikipedia articles from Stack Overflow were provided to support opportunistic learning (see Section 2.1). Thus, a Wikipedia article will often be the first exposure to a new computing concept for a novice programmer. Our research question focuses on better understanding the experience of the novice programmer trying to learn a new computing concept by reading a Wikipedia article.

We conducted an exploratory study with 18 novice programmers by asking them to read a Wikipedia article on a common computing concept that was new to them. As target articles we used four computing concepts commonly referenced on Stack Overflow. The study participants were asked to think aloud during the study session. We conducted thematic analysis on the transcribed sessions and contribute a description and analysis of five major themes that summarize the salient aspects of the participants' experience.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366832

2 BACKGROUND

Our study is framed by existing linking practices on Stack Overflow and by prior work on self-regulated learning in computing and documentation usability.

2.1 Linking to Wikipedia

We collected data on the practice of linking to Wikipedia from Stack Overflow by extracting all links with the root domain *wikipedia.org* from questions and answers, using the SOTorrent dataset [2]. The dataset contains data on 43 666 554 Stack Overflow posts, 362 447 (0.8%) of which contain at least one link to Wikipedia. In total, there are 448 453 links to the online encyclopedia, pointing to 40 309 different link targets within Wikipedia. A similar ratio of links to Wikipedia articles has been reported for the *r/programming* community on Reddit [1].

We selected a sample of four articles for detailed study. We selected these articles by considering entries in decreasing number of links to the article from Stack Overflow, but manually excluding articles that are declarative (i.e., about a standard, convention, protocol, etc.), too short (less than 600 words, or subjectively short because mostly used for examples), only applicable within a narrow technology context, non-technical (e.g., mostly about history), very closely related to a topic already selected, requiring knowledge and experience unrealistic for novice programmers (e.g., on distributed software architectures), or about a software product (e.g., Android) instead of a computing concept. We also excluded articles that had been explicitly flagged by the Wikipedia community as exhibiting a problem through a template message. In the end the four articles we selected are all linked from Stack Overflow more than 1 000 times: *Dependency Injection*, *Endianness*, *Levenshtein Distance*, and *Regular Expression*. The column *Rank* of Table 1 indicates the rank of each in the list of most frequently linked articles.

To determine whether the links from Stack Overflow to the Wikipedia articles corresponding to these four concepts were provided by the Stack Overflow user to support opportunistic learning, we manually annotated a random sample of 100 link references for each concept for a total of 400 link references. Both authors coded 30 link references per concept in a joint coding session, and the remaining 70 link references per concept were coded independently and assessed for inter-rater agreement using Cohen's kappa metric. We established three codes: *Yes* (the reader is assumed to need to learn about the concept), *No* (the reader is not assumed to need to learn about the concept, or we cannot tell if the reader needs or does not need to learn about the concept), and *Other* (an unusual context prevents either *Yes* or *No* to be meaningful, e.g., for mislabelled links). If the concept represented by the link is also explained in the text (e.g., "The *Levenshtein distance*, i.e. the minimum number of single-character edits (insertions, deletions or substitutions) ...") or if there is an explicit phrase indicating to the reader that they need to learn about the concept, we considered this as evidence that the link was provided to support opportunistic learning. In contrast, a mention of the concept in passing without additional explanation (e.g., "Also, instead of using the Hamming distance, consider the *Levenshtein distance*."), was considered as evidence that the link was not provided to support opportunistic learning.

Table 1 shows the results of this annotation. **On average, we found evidence that more than one third (37%) of links from Stack Overflow to selected Wikipedia articles were provided to support opportunistic learning.** Based on this observation, we sought to characterize the suitability of Wikipedia as a resource for opportunistic learning of computing concepts.

Table 1: Wikipedia articles studied

Wikipedia article	Rank	Yes	No	Other	Kappa
Dependency Injection	10	42%	58%	0%	0.62
Endianness	19	37%	61%	2%	0.70
Levenshtein Distance	22	39%	59%	2%	0.57
Regular Expression	26	29%	68%	2%	0.62

2.2 Related Work

The **cognitive aspects of programming** have been studied for decades. Mayrhauser and Vans contribute a review of early cognitive models of code understanding [31]. These help structure our understanding of the programming task, and include a provision for expert knowledge that takes into account general computing concepts. However, the models do not include features of concept acquisition or, more generally speaking, metacognition.

Metacognition relates to knowledge about one's own knowledge. Eteläpelto observes important differences in metacognitive knowledge and task awareness between novice and expert programmers [10]; Murphy and Tenenberg [25] and Hauswirth and Adamoli [15] focus on *metacognitive calibration* and argue about the importance of adequately estimating one's own level of knowledge in computing. Parham et al. surface the use of metacognitive strategies as part of the problem-solving process in programming tasks [27], and Bergin et al. [3] assess the impact of metacognition and other learning strategies on academic performance. Metacognition plays an important role in the formulation of part of our research goal, which is to help novice programmers become aware of relevant computing concepts (as a prerequisite to learning them). Recent work by Prather et al. also aims to increase metacognitive awareness of novice programmers [28], in their case through the use of an automated assessment tool.

Our research targets programmers who seek information and learn on their own, an activity characterized as **self-regulated learning (SRL)** [36], which takes place in a diversity of contexts in the case of computing [24]. Falkner et al. describe SRL strategies [12] and how they evolve [11] during students' undergraduate years. Their study focuses on traditional phases of the software development process (coding, design, testing), and does not integrate reflections on the use of documentation or the learning of new concepts. A number of other studies of SRL directly motivate our research goal. Boustedt et al. describe how students engaged in informal learning are worried about missing important information [4]; Isomöttönen and Tirronen describe the "challenge of supporting students' theoretical synthesis of the topics" [16]; Loksa and Ko argue that effective self-regulation must leverage existing background knowledge [22]. Based on interviews with stakeholders at companies, Zander et al. report that effective SRL strategies are expected in the workplace [34], which stresses the importance for novice programmers to develop effective information-seeking

habits early on. Brandt et al.'s studies [5] provide a vivid depiction of what opportunistic learning involves when programming.

An important part of self-regulated, and in our case opportunistic, learning is the **discovery and use of learning resources**. Gómez et al. provide complementary evidence for the practice of linking to Wikipedia on Stack Overflow [13], and Vincent et al. [30] contribute an analysis of the potential impact of linking to Wikipedia on Stack Overflow and Reddit. Studies of the content of learning resources for programmers also inform this work, and in particular Wikipedia [8, 19, 23]. Pedagogical examples are a major factor in the design of software documentation, so we note Trafton and Reiser's observation of the usefulness of examples [29], Novick and Ward's empirical confirmation that documentation users want examples [26], as well as attempts to integrate more examples in technology targeting novice programmers [35]. These are relevant to one of the themes elicited in the study.

3 METHOD

We explored the experience of novice programmers reading a Wikipedia article on computing concepts by conducting think-aloud sessions with participants drawn from the target population and analyzing the session transcripts using thematic analysis [6].

3.1 Participants

We recruited 18 participants through announcements disseminated at the University of Adelaide. Table 2 synthesizes their background. The participants are grouped by their target concept (see Section 2.1). We assigned the concepts to participants at the time of the study to ensure the assigned concepts were unknown to the participant, and to distribute a given concept across participants with different characteristics. We refer to the participants using a pseudonym, with the gender of the pseudonyms randomly assigned with the same 8/10 female/male ratio as in our set of participants.

The table synthesizes five factors which we considered, a priori, might help better understand the experience of the participants during the study sessions. We synthesized these factors by triangulating the participants' responses in the pre-study questionnaire with the video recording of the study sessions.

Discipline describes the main program of study of a participant. All participants were students at the University of Adelaide. *Computing* refers to any program of study where a significant portion of the course work is on computing topics. This includes core programs such as bachelor in computer science and software engineering, but also joint programs such as bachelor of mathematics and computer science. *Engineering* refers to all other engineering programs, such as electrical engineering, computer engineering, etc. *Other* refers to a non-technical program.

Stage captures how advanced a participant is in their program of study. We discretized the participants' background into three levels: freshman or junior ○, sophomore or senior ◓, and graduate student ●. We use the same progression of symbols for the levels of the following three characteristics, with a filled back circle representing the highest level of fulfillment for a characteristic.

English Language Proficiency (Lang.) qualifies a participant's level of competence in English. The University of Adelaide is an English-speaking institution and all participants were required to have basic competence in English to take part in the study. The levels distinguish participants for whom communicating in English required noticeable effort, versus participants who could read and communicate in English with only minor hesitation, versus participants who were fluent.

Table 2: Characteristics of the study participants

Name	Discipline	Stage	Lang.	Prog.	Forum
Dependency Injection					
Adam	Engineering	○	●	◓	◓
Ben	Computing	◓	◓	●	◓
Chris	Computing	○	●	◓	◓
Dora	Computing	◓	○	○	◓
Endianness					
Eric	Computing	●	●	◓	◓
Frank	Computing	○	●	◓	○
Gina	Computing	○	○	◓	◓
Harry	Engineering	◓	◓	◓	○
Levenshtein distance					
Iris	Engineering	◓	◓	◓	●
Julia	Engineering	◓	◓	◓	◓
Ken	Computing	●	◓	●	●
Lucy	Engineering	●	○	○	●
Mark	Other	○	●	○	○
Regular Expression					
Nicole	Computing	○	○	◓	◓
Oscar	Engineering	○	●	◓	◓
Paula	Engineering	◓	◓	○	◓
Quicy	Computing	●	○	○	◓
Rachel	Computing	○	○	○	○

Programming Experience (Prog.) qualifies a participant's relative level of programming experience. We distinguish between: Basic (just getting started with programming); Intermediate (programming for a few months to a year); Advanced (programming regularly for over a year).

Experience using Programming Forums (Forum) qualifies a participant's level of experience engaging with question and answer programming forums such as Stack Overflow. We distinguish between: Basic (a participant had not used such forums); Intermediate (a participant occasionally or regularly reads posts); Advanced (a participant uses forums almost every time they program and may have posted questions or answers).

As can be seen from the table, through our assignment procedure we were able to achieve a high degree of diversity across characteristics between concepts. In particular, all concepts group participants in all three levels of English proficiency.

3.2 Data Collection

We conducted all 18 individual sessions at the University of Adelaide using a laptop computer running screen and audio recording

software. Before the session, the participant was asked to fill in a pre-study questionnaire on their background and read a short description of the study. The instructions contained a specific request to "comment out loud about your learning process...". The first four participants were requested to read a selected Wikipedia article for 30min, after which we shortened the time to 20min because we realized that we could complete the task and reach thematic saturation within this time.[1]

All sessions were conducted by the first author. Each participant was asked to "learn as much as possible about a given concept" and allowed to use complementary Internet resources, such as translation services or other Wikipedia articles, to assist in their learning. During the session, the investigator reminded the participant to think aloud when necessary, and elicited additional comments through prompts and questions. At the end of the session, the investigator conducted a brief semi-structured debriefing.

The outcome of the data collection is a set of 18 videos of mean length 31.2 minutes (SD=5.6).

3.3 Data Analysis

The videos show a wide range of behavior and sophistication in the participant's approach to learning, similarly to the range observed by Kiili et al. in a study of students evaluating Internet sources [17].

We conducted a qualitative analysis of the videos collected. We structured our analysis based on the *thematic analysis* framework proposed by Braun and Clarke [6]. The numbers in the description below map to the phase numbers in the source cited. The analysis was conducted collaboratively by two investigators, referred to as Investigator A and Investigator B.

1. Transcription. Two investigators transcribed the videos (A: 14 videos; B: 4 videos) to prepare them for analysis, but also to gain familiarity with the raw data. For each video, the investigator noted the main actions (e.g., clicking on links, rolling over terms to extract definitions) and verbalizations of the participants, and made general notes on what they were observing.

2. Initial Codes. Investigator A reviewed all transcripts and coded all relevant data extracts using a flat structure of codes, with a number of iterations required before it was possible to elicit a stable catalog of codes. A data extract corresponds to a cohesive unit of action or verbalization by the participant, roughly equivalent to few sentences in the transcript. Investigator B then reviewed the codes and the two investigators jointly created a phase-final catalog of codes organized into two categories: *Experience* (eight codes) and *Meta* (also eight codes). The *Experience* codes capture direct observations of the participant's experience (e.g., "struggles with notation"), and the *Meta* codes capture verbalization that are reflections of the participants themselves (e.g., "too much information"). There was no explicit correspondence between the two types of codes in our open coding phase.

3. Searching for Themes. The investigators discussed the distribution of codes across transcripts and the relation between the codes, and identified six themes as salient. The themes mapped to a subset of the *Experience* codes, with the *Meta* codes used in a supporting role when defining the themes.

4/5. Reviewing and Defining Themes. Investigator A then reviewed once again all transcripts, this time identifying each extract with a theme, summarizing the themes, and linking the themes.

Member Checking. As recommended by Creswell [7], we additionally prepared specific description of the themes to present back to participants to elicit their feedback.

We finalized the definition of the themes after receiving the feedback from eight participants (see Section 4.2). As part of this process we also decided to drop one theme from our analysis. This theme, titled *Recalling Knowledge*, captured how participants tried to relate what they were reading about to what they knew already. When developing a rich description of this theme, we determined that it was simply a basic expression of the learning process and not specifically related to computer programming or the use of Wikipedia. Furthermore, it was the theme that had resonated the least with our participants.

3.4 Threats to Validity

Our research method follows a grounded, inductive approach with the following implications. The participants were asked to learn a concept as part of a study and not as the result of experiencing a need to learn, which creates a dissociation with common assumptions for adult learning [20]. However, we carefully selected the target concepts to match common information needs as expressed on Stack Overflow. The number of participants in our study is inevitably limited, which bears the risk that we may have missed important themes or modulations of a theme. However, interviewing 18 participants with different characteristics and studying four concepts distributed across participants provides a degree of assurance that aspects of the participants' experience that are salient enough to be observed in different contexts, will have been expressed. In the case of the concepts selected, we applied a strict selection procedure to ensure that the corresponding articles would be of a reasonable quality and not obviously inaccessible to the participants. A study of low-quality articles or articles wildly beyond the grasp of the participants would be likely to yield different results. The think-aloud protocol used to elicit insights from participants is not a procedure participants would naturally employ while learning a concept. As expected, the amount and usefulness of their verbalizations varied between participants and the threat is that we may have missed important insights due to some participants not properly verbalizing their thoughts. This threat is mitigated by the fact that all interviews were conducted by an investigator experienced with this research method, who used prompts and questions to elicit feedback. Finally, our use of thematic analysis to interpret the data implies that the themes elicited are impossible to completely detach from the investigator's influence. We controlled against deleterious investigator bias by triangulating data from multiple sources (screen recordings, questionnaire, semi-structured interviews), by having two investigators participate in the data analysis, and by asking our participants to verify our key outcomes.

[1] Three distinct concepts were assigned to the first four participants. The one not included was *Regular Expression*.

Table 3: Learning Computing Concepts with Wikipedia: Main Themes

Theme	Description	Evidence
Concept Confusion	The reader encounters a new concept which seem to be a familiar concept that they have already learned. Initially, it seems the new concept could be referring to what they already know, but this is not the case. The ambiguity creates a distraction and leads the reader to make false assumptions about what they are trying to understand.	
Need for Examples	The reader would like to see examples to understand the concept. They explicitly look for example applications of the concept. Seeing the concept in action helps them both get an initial understanding of the concept and solidify their knowledge.	
New Terminology	Wikipedia articles can be dense with new terminology. Many of the terms will be unfamiliar to the novice reader. The presence of new terms makes it harder to understand parts of the article that refer to it. The amount of new terms can be distracting, and even overwhelming, because for each it is necessary to decide how much one needs to know about this term to keep making sense of the article. Encountering too many new terms can be discouraging.	
Trivia Clutter	Because of their encyclopedic nature, Wikipedia articles can include a lot of detailed but peripheral information that does not help to understand a concept. This information constitutes clutter when the page is used as a learning tool, as it must be identified as irrelevant and skipped. This problem is especially acute for readers not fluent in the language of the article.	
Unfamiliar Notation	Some information in the article can use a specialized notation. Examples include source code, modeling languages, and mathematical notation. Because the notation is not the topic of the article, it is not explained. Readers unfamiliar with the notation must guess what it means and remain unsure of what they are learning about.	

4 FINDINGS

Table 3 describes the main themes we identified and summarizes the amount of supporting evidence for each theme. The summary of the evidence is provided as a bar chart where each bar represents one participant, and the height of the bar represents the number of distinct data extracts related to the theme. The length of the chart's area spans a maximum of 18 bars. The diagram thus supports assessing, at a glance, how supporting evidence is distributed. For example, the theme *Trivia Clutter* is developed based on data from one third of the participants, where one participant provided many of the usable extracts, with one or two extracts from the other sessions. The bars are ordered in decreasing height.

4.1 A Closer Look at the Themes

To the extent possible given the space, we provide rich descriptions of each theme using a selection of corresponding data extracts.

Concept Confusion. Although only observed with four participants, *Concept Confusion* provides a striking illustration of the disorientation that can be experienced. Noting the epsilon symbol (ϵ) in the article on Levenshtein Distance, Iris incorrectly thought it referred to the molar attenuation coefficient, as seen in a previous course: "Epsilon, what I meant is, I learned it in electrical, so it's a value that's fixed for every material". Paula was also confused by the epsilon symbol, in her case as used in the article on Regular Expression: "Isn't [epsilon] the 'subset' symbol?". Common words used in a specialized context, such as "distance", "word", and "cost", caused similar confusion: "I don't really understand what is measuring the distance between two sequences. I'm thinking [of] coordinates..."—*Iris*; "Words [in the sense of a number of bits]. I think they meant keywords!"—*Frank*.

Need for Examples. The majority of our participants clearly stated or demonstrated that they needed examples to understand the material. For instance, when asked to explain the meaning of a section of the article, Ben, Eric, Iris, Julia, Nicole, and Paula provided an explanation strictly in terms of an example from the article. In two other cases, when asked a prompting question, the participants searched for examples, e.g.: "It's probably more useful just to look at the examples"—*Adam*. Finally, in other cases, participants stated the value of examples directly: "I don't really understand the Jaro-Winkler distance. Usually if I want to understand something it's from an example"—*Julia*; "I'd like to see if there's a basic example somewhere, just to solidify what I was reading"—*Chris*

New Terminology. A majority of our participants struggled with the amount of new terminology they encountered in the article. This struggle was identified through participants verbalizing what they did not understand, but also by observing them roll over Wikilinks and search for terms using a search engine. The theme is accurately captured by Adam: "to read this whole article and understand everything, as a beginner [...] it would be impossible because, every couple of sentences you've got new links that you have to click on to understand this concept. But to understand that you have to go to that article and click on all sorts of links to understand that, and it's just a huge web of information. [...] And all of this wouldn't completely make sense unless you understand most of these [other concepts]."

Trivia Clutter. Wikipedia articles typically include numerous "small facts" (trivia) which, although relevant for an encyclopedia entry, are of dubious help for learning a concept. Participants were free to chose what to focus on during the study session, and many engaged naturally in an information selection and filtering process: "If it doesn't matter I'll usually skip all this"—*Eric*. Six participants were explicit enough about this process to allow us to document and analyze it. The issue of trivia is best illustrated through the reference to the tangential application of Levenshtein distance to that of *linguistic distance*. When reading the corresponding paragraph, Julia reacted with "[I don't understand] how Levenshtein Distance applies

Table 4: Results of Member Checking. Letters are the initial of the participants' pseudonym (Table 2). Values are: – (Neutral/Don't Know); • (Agree), •• (Strongly Agree)

Theme	A	D	F	I	J	K	L	N
Concept Confusion	•	•	•	•	••	•	–	•
Recalling Knowledge	–	•	–	•	••	–	–	•
Need for Examples	••	••	••	•	••	••	–	•
New Terminology	••	•	••	•	•	••	–	•
Trivia Clutter	••	•	••	••	••	–	–	•
Unfamiliar Notation	•	••	••	••	••	–	–	•

in linguistics"; Mark had a similar reaction: "It's a concept I never heard before [...]. I'm not sure why it's relevant". Topics identified as trivia by our participants include etymology, history, long lists of names, and related algorithms and concepts.

Unfamiliar Notation. Each of the four target articles included material in some specialized notation, including the Unified Modeling Language (UML) (Dependency Injection), source code (all four articles), mathematics (Levenshtein Distance), and other formal languages (Regular Expression). Use of this notation translated into an obstacle for many participants, of which we were able to collect evidence for eight. For example, Adam, Chris and Dora had to guess their way through UML diagrams: "I've seen UML diagrams like this [class diagram], but not seen one like this [sequence diagram]. So I don't know, this diagram."—*Adam*; Likewise, Adam, Chris, Harry, and Mark faced code fragments in an unfamiliar programming language: "I don't know what the union [variable in C code] is, because I never used it"—*Harry*; As Lucy went through the article on Levenshtein Distance, she indicated "I have no idea about that math defintion".

4.2 Feedback from Participants

We sent a follow-up survey to all 18 participants to help assess the credibility of our themes. The survey asked them for each theme: "To what degree do you agree that this theme resonates with you?" and provided respondents with a five-point Likert scale for agreement (from "strongly disagree" to "strongly agree"). For each theme, we included the description shown in Table 3 and an example from the study. Table 4 shows the results. The header for each participant column corresponds to the initial of the participant's pseudonym. Eight of the participants responded to the survey, and no participant disagreed with any of the themes. The median agreement was at least *Agree* on the five-point scale for all themes. For transparency, we also include the feedback we collected for the preliminary theme *Recalling Knowledge*, which we subsequently elided.

5 DISCUSSION AND CONCLUSION

By interviewing 18 participants, we elicited five themes that describe the experience of novice programmers attempting to learn a computing concept by reading a Wikipedia article. **Although they are individually self-explanatory, the themes are best interpreted as an ensemble**, with meaningful links among them.

Concept Confusion overlaps with *Unfamiliar Notation* if the confusion is caused by the use of symbols or other notation: "Isn't this [epsilon] the "subset" symbol?"—*Paula*. However, the themes are distinct because *Concept Confusion* emphasizes the distraction caused by misunderstanding, whereas *Unfamiliar Notation* captures the sense of disorientation caused by the obvious absence of prerequisite declarative knowledge. *Concept Confusion* can also be seen as a special case of *New Terminology*, but where the new term is not immediately identified as new, as illustrated by Frank's confusion over the use of the term "word" in the context of computer architecture (see Section 4.1).

Need for Examples captures the need of learners to have examples. However, in the context of Wikipedia, longs lists of examples can lead to the problem of *Trivia Clutter* if the examples are numerous and trivial and do not support an educational function: "I think this is not a very important concept for this page. Because this is just examples"—*Gina*. The obstacle to learning that *Trivia Clutter* represents is also compounded if it leads to *Concept Confusion*, *New Terminology*, or *Unfamiliar Notation*. Not only does the reader need to determine that content is irrelevant for learning a concept, but they also need to struggle and overcome confusion in their process to make this determination, as illustrated by the case of linguistic distance (see Section 4.1).

Finally, while the majority of participants expressed a *Need for Examples*, such examples can cause them to face *Unfamiliar Notation* if they are not knowledgeable in the notation used in examples, e.g., the programming language.

Our main conclusion is to confirm Wikipedia's unsuitability as a resource for opportunistic learning of computing concepts. Although Wikipedia does not claim to be a learning resource in the first place [33], we (see Section 2.1) and others [13, 30] have observed that it is common to link to Wikipedia to guide readers to definitions of concepts they might need to learn. Our study contributes a thematic categorization with rich descriptions of evidence showing how the content of the Wikipedia article for four popular computing concepts is not pedagogical.

Our findings have implications for both members of online programmer communities and for researchers. In the short term, linking practices in programming Q&A forums would benefit from an explicit distinction between simple *concept references* and *resource recommendations for learners*. Concept references can continue to link to Wikipedia, but resource recommendations should favor tutorial and other sites explicitly targeted at learners. For example, the site *regular-expressions.info* [14] is a more appropriate learning resource than the Wikipedia article *Regular Expression* for that concept. Given that the number of potential computing concepts can easily exceed the number of high-quality tutorial web sites that it is possible to maintain, the crowd-sourced development of computing dictionaries, such as FOLDOC [9], has the potential to help orient learners while avoiding the cognitive load caused by detailed treatment and extensive trivia found of some Wikipedia articles.

This paper contributes additional texture to our knowledge of information needs for novice programmers and the obstacles they face when trying to fulfill them. To continue to advance the goal of helping novice programmers learn computing concepts opportunistically, it will be necessary to also understand which sources they currently consult and why. Finally, there is potential for a line of research on automatically synthesizing concept explanations and examples from web resources such as Stack Overflow and Wikipedia.

ACKNOWLEDGMENTS

We are grateful to the study participants for their contribution. We thank the members of the Computer Science Education Research Group (CSER) in the School of Computer Science at the University of Adelaide for their advice and assistance, and Alison Li, Deeksha Arya, and Mathieu Nassif from the Knowledge and Software Technology Group at McGill University for recommendations on related work and feedback on the paper. This work has been supported by the Australian Research Council's Discovery Early Career Researcher Award (DECRA) funding scheme (DE180100153).

REFERENCES

[1] Maurício Aniche, Christoph Treude, Igor Steinmacher, Igor Wiese, Gustavo Pinto, Margaret-Anne Storey, and Marco Aurélio Gerosa. 2018. How Modern News Aggregators Help Development Communities Shape and Share Knowledge. In *Proceedings of the 40th ACM/IEEE International Conference on Software Engineering*. 499–510.

[2] Sebastian Baltes, Lorik Dumani, Christoph Treude, and Stephan Diehl. 2018. SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts. In *Proceedings of the 15th International Conference on Mining Software Repositories*. 319–330. version: 2018_09_23.

[3] Susan Bergin and Ronan Reilly. 2005. Examining the Role of Self-Regulated Learning on Introductory Programming Performance. In *Proceedings of the 1st International Workshop on Computing Education Research*. 81–86.

[4] Jonas Boustedt, Anna Eckerdal, Robert McCartney, Kate Sanders, Lynda Thomas, and Carol Zander. 2011. Students' Perceptions of the Differences Between Formal and Informal Learning. In *Proceedings of the 7th International Workshop on Computing Education Research*. 61–68.

[5] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 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*. 1589–1598.

[6] Virginia Braun and Victoria Clarke. 2006. Using Thematic Analysis in Psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101.

[7] John W. Creswell. 2003. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches* (2nd ed.). Sage Publications.

[8] Peter K. Dunn, Margaret Marshman, and Robert McDougall. 2017. Evaluating Wikipedia as a Self-Learning Resource for Statistics: You Know They'll Use It. *The American Statistician* 73, 3 (2017), 1–8.

[9] Denis Howe et al. 2019. FOLDOC: Free On-Line Dictionary of Computing. https://foldoc.org/. Accessed 19 August 2019.

[10] Anneli Eteläpelto. 1993. Metacognition and the Expertise of Computer Program Comprehension. *Scandinavian Journal of Educational Research* 37, 3 (1993), 243–254.

[11] Katrina Falkner, Claudia Szabo, Rebecca Vivian, and Nikolas Falkner. 2015. Evolution of Software Development Strategies. In *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering*, Vol. 2. 243–252.

[12] Katrina Falkner, Rebecca Vivian, and Nickolas Falkner. 2014. Identifying Computer Science Self-Regulated Learning Strategies. In *Proceedings of the 19th Annual Conference on Innovation and Technology in Computer Science Education*. 291–296.

[13] Carlos Gómez, Brendan Cleary, and Leif Singer. 2013. A Study of Innovation Diffusion Through Link Sharing on Stack Overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories*. 81–84.

[14] Jan Goyvaerts. 2019. Regular-Expressions.info. https://www.regular-expressions.info. Accessed 19 August 2019.

[15] Matthias Hauswirth and Andreea Adamoli. 2017. Metacognitive Calibration When Learning to Program. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. 50–59.

[16] Ville Isomöttönen and Ville Tirronen. 2013. Teaching Programming by Emphasizing Self-Direction: How Did Students React to the Active Role Required of Them? *ACM Transactions on Computing* 13, 2 (2013).

[17] Carita Kiili, Leena Laurinen, and Miika Marttunen. 2008. Students Evaluating Internet Sources: From Versatile Evaluators to Uncritical Readers. *Journal of Educational Computing Research* 39, 1 (2008), 75–95.

[18] Ada S. Kim and Andrew J. Ko. 2017. A Pedagogical Analysis of Online Coding Tutorials. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 321–326.

[19] Charles Knight and Sam Pryke. 2012. Wikipedia and the University, a Case Study. *Teaching in Higher Education* 17, 6 (2012), 649–659.

[20] Malcom S. Knowles, Elwood F. Holton III, and Richard A. Swanson. 2005. *The Adult Learner* (6th ed.). Elsevier.

[21] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A Study of the Difficulties of Novice Programmers. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. 14–18.

[22] Dastyni Loksa and Andrew J. Ko. 2016. The Role of Self-Regulation in Programming Problem Solving Process and Success. In *Proceedings of the 12th ACM International Computing Education Research Conference*. 83–91.

[23] Teun Lucassen and Jan Maarten Schraagen. 2010. Trust in Wikipedia: How Users Trust Information from an Unknown Source. In *Proceedings of the 4th Workshop on Information Credibility*. 19–26.

[24] Robert McCartney, Jonas Boustedt, Anna Eckerdal, Kate Sanders, Lynda Thomas, and Carol Zander. 2016. Why Computing Students Learn on Their Own: Motivation for Self-Directed Learning of Computing. *ACM Transactions on Computing Education* 16, 1 (2016), 2:1–2:18.

[25] Laurie Murphy and Josh Tenenberg. 2005. Do Computer Science Students Know What They Know?: A Calibration Study of Data Structure Knowledge. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. 148–152.

[26] David G. Novick and Karen Ward. 2006. What Users Say They Want in Documentation. In *Proceedings of the 24th Annual ACM International Conference on Design of Communication*. 84–91.

[27] Jennifer Parham, Leo Gugerty, and D. E. Stevenson. 2010. Empirical Evidence for the Existence and Uses of Metacognition in Computer Science Problem Solving. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*. 416–420.

[28] James Prather, Raymond Pettit, Brett A. Becker, Paul Denny, Dastyni Loksa, Alani Peters, Zachary Albrecht, and Krista Masci. 2019. First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 531–537.

[29] J. Gregory Trafton and Brian J. Reiser. 1993. Studying Examples and Solving Problems: Contributions to Skill Acquisition. In *Proceedings of the 15th conference of the Cognitive Science Society*. 1017–1022.

[30] Nicholas Vincent, Isaac Johnson, and Brent Hecht. 2018. Examining Wikipedia With a Broader Lens: Quantifying the Value of Wikipedia's Relationships with Other Large-Scale Online Communities. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 566:1–566:13.

[31] Anneliese von Mayrhauser and A. Marie Vans. 1995. Program Comprehension During Software Maintenance and Evolution. *Computer* 28, 8 (1995), 44–55.

[32] Ryen W. White, Susan Dumais, and Jaime Teevan. 2009. Characterizing the Influence of Domain Expertise on Web Search Behavior. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining*. 132–141.

[33] Wikipedia. 2019. What Wikipedia is not. https://en.wikipedia.org/wiki/Wikipedia:What_Wikipedia_is_not. Accessed 10 July 2019.

[34] Carol Zander, Jonas Boustedt, Anna Eckerdal, Robert McCartney, Kate Sanders, Jan Erik Moström, and Lynda Thomas. 2012. Self-directed Learning: Stories from Industry. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*. 111–117.

[35] Rui Zhi, Thomas W. Price, Samiha Marwan, Alexandra Milliken, Tiffany Barnes, and Min Chi. 2019. Exploring the Impact of Worked Examples in a Novice Programming Environment. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 98–104.

[36] Barry J. Zimmerman and Manuel Martinez Pons. 1986. Development of a Structured Interview for Assessing Student Use of Self-Regulated Learning Strategies. *American Educational Research Journal* 23, 4 (1986), 614–628.