Noname manuscript No.
(will be inserted by the editor)

How Programmers Find Online Learning Resources

Deeksha M. Arya · Jin L.C. Guo · Martin P. Robillard

Received: date / Accepted: date

Abstract When learning a new technology, programmers often have to sift through multiple online resources to find information that addresses their questions. Prior work has reported that information seekers use a number of different strategies, including following *scents*, or indicators, to locate appropriate resources. We present a qualitative and quantitative investigation of how programmers learning a new technology employ these strategies to navigate between online resources and evaluate the pertinence of these resources. We performed a diary and interview study with ten programmers learning a new technology, to study how users navigate from the question they have to the resource that satisfies this need. Based on our observations, we propose a resource-seeking model that represents the online resource seeking behaviour of programmers when learning a new technology. The model is comprised of six components that can be divided into two groups: Need-oriented components, i.e. Questions, Preferences, and Beliefs, and Resource-oriented components, i.e. Resources, Cues, and Impression Factors. We identified nine relations between these components and studied how the components are associated. We report on the characteristics of the components and the relationships between them, and discuss the importance of search customization and other implications of our observations for resource creators and search tools.

Deeksha M. Arya
McGill University
E-mail: deeksha.arya@mail.mcgill.ca

Jin L.C. Guo
McGill University
E-mail: jguo@cs.mcgill.ca

Martin P. Robillard
McGill University
E-mail: martin@cs.mcgill.ca

Keywords Software Documentation · Information Seeking · Online Learning Resources · User Study · Diary Study · Qualitative Analysis · Quantitative Analysis

1 Introduction

Resources to learn a new technology have become more available with the increase in information sharing websites. Search engines assist users in reducing the time and effort spent navigating the complex landscape of online software resources to find relevant information. Nevertheless, programmers still face difficulties in refining the set of results that the search engines retrieve to locate the exact resource that might solve their needs (Escobar-Avila et al., 2019).

Through an interview and observational study, Teevan et al. (2004) observed that participants used two information seeking strategies - *orienteering* and *teleporting* when performing searches. Orienteering refers to using small steps to move towards a perceived destination that contains the information needed. In contrast, teleporting involves a direct movement to the target destination. Teevan et. al found that participants primarily used the orienteering strategy, possibly because of its cognitive ease, allowing participants to be in control of their location and of context during their search.

The effort required in manually filtering resources during navigation may be attributed to the fact that there are many aspects that information seekers consider. Information foraging theory suggests that users typically follow *information scents* that help identify the information most suited to their needs (Pirolli and Card, 1999). A *scent* refers to the perception of the value of information given by cues, such as citations or ratings. Information scents have proved useful in foretelling user actions such as when a link to a web page is followed or from which page a user will leave a website (Pirolli and Fu, 2003). Despite previous investigations on users' behavior during information retrieval, we are not aware of any model to represent what the orienteering strategy entails for programmers when they search for technology learning resources in a natural setting, what kind of "scents" programmers use, and how the programmers make decisions about resources to access.

We investigate the question of *how to represent the behaviour and rationale of programmers as they search for online resources when learning a new technology.* We conducted a diary study with ten programmers to closely examine the kinds of resources they refer to for different queries and how they navigate to that resource. Programmers search for information for many different reasons (Gallardo-Valencia and Sim, 2011; Xia et al., 2017), but we focus our study on programmers who are in the process of learning a new technology.

We propose a **resource-seeking model** to characterize the process of finding online learning resources. Our model is comprised of two groups of components: *Need-oriented* and *Resource-oriented*. Need-oriented components are Questions, Preferences, and Beliefs. These components of the model cap-

ture information or context about a programmer's information need. Resource-oriented components are related to the resources that programmers access and include Resources, Cues, and Impression Factors. Our model describes nine relations between the six individual components, for example, that a Cue *is used to select* a Resource. This model surfaces the formerly implicit components that guide the search for technical information. Awareness and understanding of the interplay between these components is helpful for resource seekers to effectively form search queries, and to resource designers to facilitate the seeking process by presenting information purposefully. As part of our contribution, we also release a public data set of 795 instances of the components of the model and 662 links between them (see Section 4.3).

In Section 2, we discuss related literature in information needs and information seeking. We introduce our model's conceptual framework in Section 3. We describe our study design in Section 4. Sections 5 and 6 provide more detailed descriptions of the components of the model and in Section 7 we describe the relations between the components. We discuss the implications of our observations in Section 8. We summarize our findings in Section 9.

2 Related Work

The earliest information foraging model, suggested by Pirolli and Card (1999), is decomposed into three sequential theoretical models, i.e. the patch model, scent model, and diet model. They described that the information that can solve users' needs may be contained in multiple information "patches". To find the information they need, users perform "scent-following" wherein they use cues or hints to decide how much time to allocate to moving between or within patches. The diet model describes how users select, make sense of, and consume information related to their needs.

Our work focuses on investigating the "scents" that programmers follow to make decisions about which "patch" of information, i.e. learning resource, could be pertinent to their information needs. Our work builds on two posited components of information seeking: the information need, and the resource. We study the behaviour of programmers in decision making from posing a Question to accessing a pertinent Resource. We discuss the literature on understanding the *information needs* that programmers have, and their *online information seeking* behaviour to resolve these needs.

2.1 Information Needs

Prior work has mainly focused on the needs of developers in every-day development and maintenance activities. Based on their experience, Erdos and Sneed (1998) suggested that there are seven questions that a programmer must have the answer to, to be able to maintain a software program. Sillito et al. (2006) found that when changing software, programmers ask questions that can be

grouped into four categories related to knowledge about the entity graph of code components. These works primarily focus on code behaviour (e.g. *Where is a particular variable declared?* (Erdos and Sneed, 1998)), whereas the Questions we observe in our study cover a broader scope, including any aspect that a programmer may want to know about when *learning* the technology, such as underlying concepts or installation-related questions.

Ko et al. (2007) performed an observational study to determine what kind of information developers generally look for, and categorized search instances into 21 information needs. Gallardo-Valencia and Sim (2011) asked 25 developers from a company to self-report over a period of 15 days their web searches. They found five main types of problems that induce searching for information online. Duala-Ekoko and Robillard (2012) performed a study in which twenty participants were asked to think aloud as they worked on two programming tasks in a familiar programming language but using unfamiliar APIs. The authors analysed the screen recordings with the verbalized thought process and observed that the participants had twenty types of questions. Rao et al. (2019) studied users' web search behavior for software engineering tasks by analyzing the logs of millions of search queries to the search engine Bing, and found six categories of intent for searches.

Some overlap exists between the work on information needs and categories of the Question component in our study. For example, *In what situations does this failure occur?* (Ko et al., 2007), *the existence of errors in software* (Gallardo-Valencia and Sim, 2011) and *Debug* (Rao et al., 2019) correspond to the *Debug* category in our work (see Section 5.1). Despite this strong correspondence, we chose not to reuse previous categorization. We focus on the questions of *programmers* who are *learning* a new technology. Although the categories may be similar, the context while searching for this information differs from everyday information look-up. Hence, we did not want to assume that prior taxonomies would completely encompass questions by programmers who are learning a technology, or alternatively that the programmers would have questions in all existing categories.

Erdem et al. (1998) recognized that questions are composed of multiple factors, and proposed a model to represent the questions that programmers have while trying to understand software. The model identifies a question by three components - its topic (the subject of the question), its question type (e.g. *who*, *what*, *where*), and its relation type (i.e. the kind of information that is requested). In addition to the questions programmers learning a new technology had, we studied the requirements the programmer had about the information they were seeking (Preferences), and why these requirements existed (Beliefs).

2.2 Information Seeking

Teevan et al. (2004) proposed that search engines should support the common orienteering strategy participants generally take, i.e. using small steps towards

finding the information needed. They suggested that search engines could provide meta-information, cues and context of search results to prompt users. To gain insight into scent-following behaviour, Pirolli and Fu (2003) measured information scent of a web page as the mutual relevance of its contents. They found that it is useful in foretelling user actions like from which page a user will leave a website.

Brandt et al. (2009) studied *why* programmers search for information by performing an in-lab study with 20 programmers. They observed that programmers searched online to clarify existing knowledge, remind themselves of details, or to learn by trying code snippets. In the latter case, participants used primarily aesthetic aspects, such as the existence of advertisements on the web page, to quickly judge whether to read through the page. The authors also analysed a web query data sample containing 101,289 queries from 24,293 programmers, to gain a deeper insight into the search process. They found an association between the types of pages visited and the type of queries performed. For example, they found that code-only queries resulted in more API documentation accesses, and natural language queries to more tutorial accesses. We also perform statistical tests to determine the association between types of questions and resources accessed. However, our categories of questions revolve around the content of the question, as opposed to Brandt et al.'s study which focuses on the format of the search query.

From a survey of 74 individuals at an IBM enterprise customers event, Earle et al. (2015) found that 72 participants had preferences of the type of documentation they would use. Escobar-Avila et al. (2019) surveyed 205 Computer Science students and practitioners to determine their habits in learning programming and its related concepts. More than 55% in both populations said they preferred visual/auditory formats for learning, and only about 3% indicated they prefer textual mediums. We investigate such preconceived Preferences that programmers have, expanding beyond documentation type to include the information style and presentation expected from a resource.

Prior work has also studied the seeking behavior for code within a target resource. Lawrance et al. (2008) proposed the Programmer Flow by Information Scent (PFIS), an algorithm to describe how programmers navigate through source code during debugging based on a bug report. This algorithm involves measuring the "proximity" of each possible area of source code the programmer could go to (e.g. package, class, method, or variable) with the bug report content, and calculating the probability of a visit based on multiple simulations of traversal. Lawrence et al. compared the algorithm's prediction of which piece of code will be visited to observed human behaviour and found that PFIS predicted human navigation close to aggregated human decisions. Srinivasa Ragavan et al. (2016) also focused on navigation between code artifacts by studying how programmers compare similar pieces of code to determine which one is applicable to a particular task.

Piorkowski et al. (2015) studied how the *intent* of a developer, i.e. to fix a bug or to learn to help someone else fix a bug, affects the type of information sought. They performed a user study with eleven participants, split into two

equal sized groups with tasks of the two different intents. The authors observed that there was a large overlap between *cues* that participants with both intentions used to navigate the package explorer, editor, stack trace, and search results during debugging. They observed that a majority of cues used were code output or domain related. Liu et al. (2021) performed a formative study with fifteen programmers to understand how they reuse programming decisions regarding technologies used in particular scenarios, made by other programmers. Liu et al. elicited three major *facets* that help assess whether reuse is appropriate in programming: *context of the prior decision*, *trustworthiness of the web source and the author*, and the *thoroughness of the knowledge for reuse*. In contrast, our work focuses on programmers' rationale when navigating available online resources for information about a technology, irrespective of the particular format.

Nadi and Treude (2020) studied the navigational cues in finding relevant answers in a Stack Overflow post. They reported that *essential sentences*, i.e. ones which users can use to determine whether an answer is worth reading or not, highlighted by most participants mainly contained explanations, or specified a library or a code component. Marques et al. (2020) determined that sentences within an artifact, perceived by their participants as relevant to a task, contained common semantic meanings that could help determine what information within the artifact is relevant. They also observed that while participants used different search strategies, they used implicit clues to find the information they needed. For example, they would judge the value of text based on visual cues like whether it was in bold, or was concise. We complement prior literature on "scent-following" within and across artifacts, and focus on cues that programmers use when deciding *between* online software technology resources on which one(s) to access.

We complement the within-artifact information seeking literature by focusing on the search process on the web and what it entails. Sadowski et al. (2015) performed a case study at Google, via a survey conducted nearly every time a participant accessed their internal search website to gain insight on why and how programmers at the company performed searches. They augmented this study method with an analysis of log data to determine quantitative measures of search session, such as how many terms were in queries or the average number of clicks that lead to a successful search. They reported micropatterns of observed search sessions. For example, they found that programmers who are *very familiar* with code typically follow the micropattern: one or more searches followed by one or more clicks. Bai et al. (2020) performed a follow-up task-oriented lab study with graduate students and compared their results with the observations of Sadowski et al. (2015).

As opposed to studying the interplay of querying and clicking-on-resources in successful search, we focus on studying the decisions that programmers make in *choosing* between different artifacts in the specific context of learning a new technology. We formulate our observations into a model comprised of six components that represent this resource seeking behavior. Our work lies

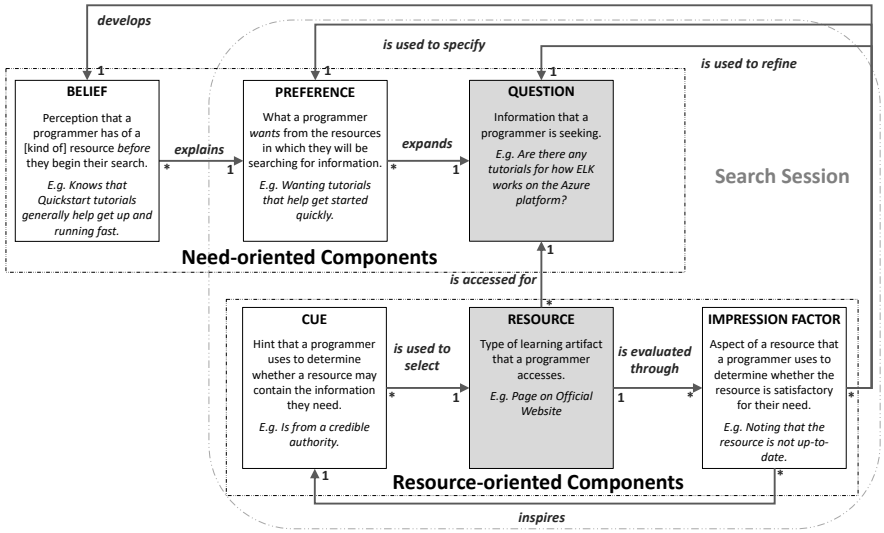

Fig. 1: The online software technology resource-seeking model of users learning a new technology. The components shaded in grey, i.e. Questions and Resources, are posited components, while the rest emerged from our analysis. The numbers and asterisk annotated on the arrows indicate the cardinality of the relation. For example, multiple Cues can *be used to select* one Resource, and multiple Resources can *be accessed for* a single Question.

within *document request*, a direction that requires advancement to build better quality, user-catered documentation according to Robillard et al. (2017).

3 Resource-Seeking Model

We propose a model for representing how programmers seek online resources when learning a new technology. This model consists of six components: three need-oriented components (Questions, Preferences, and Beliefs), and three resource-oriented components (Resources, Cues, and Impression Factors). All the components, except for Beliefs, occur within the scope of a single *search session*. A search session is a time window in which a programmer searches for and navigates through one or more resources online to meet their information needs. We illustrate the components and relations in the resource-seeking model in Figure 1 using excerpts from our data set. The data collection and analysis we followed is described in Section 4.2.

3.1 Need-oriented Components

Since searching for a resource only arises when a programmer has some information they need to find, every search session *must* contain at least one instance of a Question. A Question refers to the search query that the programmer uses, and acts as the starting point as they begin their search.

Preferences refer to a programmer's pre-existing expectations or requirements of the resource they are looking for. In their search, a programmer may specifically look for an article on the blogging website *Medium*. A Preference *expands* a Question by providing more context for the search.

A Belief is a pre-existing opinion about certain resource or type of resource. It *explains* a Preference, as it justifies the reason the preference arises. A programmer may explain that they want to watch a video because "[...] a Youtube video [is] easier to follow than a textual post that might contain more jargon I don't follow."

3.2 Resource-oriented Components

Within a search session, programmers access at least one Resource to find the information they need, making it the other essential component in addition to Questions. A Resource refers to a learning resource that is accessed by a programmer to find the information they need. A programmer may visit multiple Resources within a single search session. We define a typed relation between instances of these two essential components: a Resource *is accessed for* a Question. For example, a programmer may search for "What does the valgrind error summary mean?" To answer this question, a programmer may click on a video (the Resource) from a search results page.

When presented with links and/or previews of resources, programmers use Cues, i.e. hints or characteristics of the resources, to make decisions about whether to access them. For example, a programmer may click on a resource because "[...] it would be the best source, since it is from the original makers of the [target programming] language". Hence, a particular Cue *is used to select* a Resource.

The Impression Factor of a resource is the aspect of the accessed resource that a programmer uses to evaluate the resource. A Resource *is evaluated through* the Impression Factor. For example, a programmer may use the date of the last update of a resource to assess whether it might be out of date. The Impression Factor also plays an important role in the feedback loop for the search process. It *is used to refine* the Question or *is used to specify* a new Preference as part of query refinement. For example, a programmer, upon realizing their query is returning only scientific papers, may choose to add "Medium" to the search query, and subsequently look only for resources hosted on the Medium website.

An Impression Factor can also be used to *inspire* a new Cue when searching for more resources. After finding a Resource that "skipped a lot of basic information", a programmer may be inspired to click the next Resource if it from a website that hosts "entry-level tutorials for technologies". An Impression Factor may also help

develop a Belief that influences subsequent search sessions, if a programmer forms a strong impression of a Resource.

4 Study Design

The resource-seeking model emerged from a diary and observational study we conducted, in which we closely examined how programmers find learning resources online. Diary studies provide a balance between observational studies in natural settings, observational studies in a lab environment, and surveys (Lazar et al., 2017b). Our data collection approach is modeled after previous self-reporting studies, such as those conducted by Gallardo-Valencia and Sim (2011) and Xie and Joo (2012), that are followed by a reflection interview in which participants may be asked to recreate some of their searches during the study period.

We focus on programmers learning a new technology. We use the term *programmer* to denote people who write code in any capacity. This population includes *developers*, who are professionally employed to build and maintain software. While recruiting participants for the study, we ensured that they had prior programming experience, and were just beginning to learn a technology new to them. We ensured that no participants were learning only from in-person or online courses, pre-defined training material, or research papers, where their searches would be guided by instructors or training material that contained pre-defined learning objectives. We also enlisted only those participants whose learning happened on a regular basis, i.e. at least daily for a minimum of three days a week, so that the searches performed during the study would be part of a regular learning process, instead of an exceptional occurrence.

4.1 Data Collection

Each participant filled a form with demographic questions. We asked each participant to fill in a diary entry for every search for information made online over a period of five days, regarding the technology they were learning. We requested that participants document every step in their search process. Figure 2 shows the diary template we provided. Participants were requested to send their completed diary entry (or entries) to us at the end of each study day. Thereby, we were able to immediately clarify with the participants any ambiguity or request for more details in the diary entries, if necessary.

After each participant completed five study days, the first author conducted an hour-long open-ended interview with the participant. The interviewer asked the participant to recreate two selected diary entries from the study week. While repeating the steps in the entries, the interviewer encouraged the participant to describe their thought process aloud (Jääskeläinen, 2010). We did this for two reasons. First, the interviewer could verify the accuracy of the diary entries, clarify any ambiguities, and correct any mistakes

Name:
Date:

What I am hoping to learn with my search:
I have searched for this information before: Y/N

Steps taken:

1. <Insert Step 1>
   Thought that guided this step:
2. <Insert Step 2>
   Thought that guided this step:
3. <Insert Step 3>
   Thought that guided this step:

[Please add more steps here if necessary]

I found the information I needed: Y/N

Approx. amount of time taken to find information: <X>

Comments and Notes:

Fig. 2: Diary entry template for each search session

in reporting during the interview. Second, we could gather rich descriptions of the search sessions from the participants, which were not present in the diary entries. Based on the course of the discussion, the interviewer asked follow-up questions regarding the participant's information seeking process. After the interview, we asked the participant to complete a questionnaire about their experience of looking for learning resource online and participating in the study. We offered a compensation of up to $100 CAD for completing the study. The study is approved by the Research Ethics Board Office at McGill University (file number: 20-07-039).

Eleven participants took part in the study, providing a total of 131 diary entries. One participant completed only two of the five study days, and submitted two diary entries. Since we received less than three diary entries (on average, one per day for the minimum criteria of learning three days a week) from this participant, we omitted their data in our study. Table 1 describes our participants' demographics.

Of the remaining 129 diary entries, we filtered out 14 entries from our data set because they were not searches for technical information about the technology (e.g. one entry was about industry perspectives of the technology), contained insufficient information about the steps to reproduce entirely, or accessed only resources beyond the scope of the study (such as research papers). Despite filtering these 14 entries, no participant had less than three valid diary entries. Our final data set comprises of 115 diary entries from ten participants.

Table 1: Participant Demographics. "Initial" learning phase indicates 0-4 weeks of learning so far, "Intermediate" is any time beyond 4 weeks.

ID	Occupation	Prog. Exp. (Years)	Target Technology	Learning Phase
P2	Software Engineer	14	Elasticsearch-Logstash-Kibana (ELK)	Initial
P3	Research Assistant	6	Data visualization in Python	Intermediate
P4	Master's Student	4	Genetic Algorithms for Automatic Software Repair	Initial
P5	Research Assistant	6	Object Oriented Programming in C++	Intermediate
P6	Software Engineer	6	Selenium using Python	Initial
P7	Undergaduate Student	2	C#/Unity	Initial
P8	Master's Student	9	Torch + Lua	Intermediate
P9	Software Engineer & Master's Student	10	DRM and VA-API	Initial
P10	Software Developer Trainee	5	SQL	Initial
P11	Software Developer	6	Microservice Mesh with Envoy and Istio	Initial

Prog. Exp. — Programming Experience

Our participant sample size follows that of prior diary and observational studies including work done by Teevan et al. (2004) (15 participants), Meng et al. (2019) (11 participants), and Chattopadhyay et al. (2020) (10 participants). We discuss qualitatively observations based on the participants' detailed experiences during this study. The sample size also ensured the study could be completed in a realistic time period. Each participant's study period is five working days during which time we kept in touch with the participants, answering questions they had about the study, reviewing and requesting for clarifications in diary entries when necessary, and also performing iterations of coding of the entries. Analysing the transcripts of the hour-long interviews for relevant insights and useful anecdotes took an entire day each. Still, the 115 diary entries we collected from ten participants allowed us to make numerous repeated observations of components and connections between them (see Figure 4).

4.2 Data Analysis

We refined the data collection method as observations emerged in the analysis (Lazar et al., 2017a). This way, we were able to clarify ambiguities and ensure that the diary entries and interviews of participants remained within the context of our study.

Figure 3 illustrates the process we followed to obtain our data set. The first author open coded 25 diary entries from five participants, chosen in a

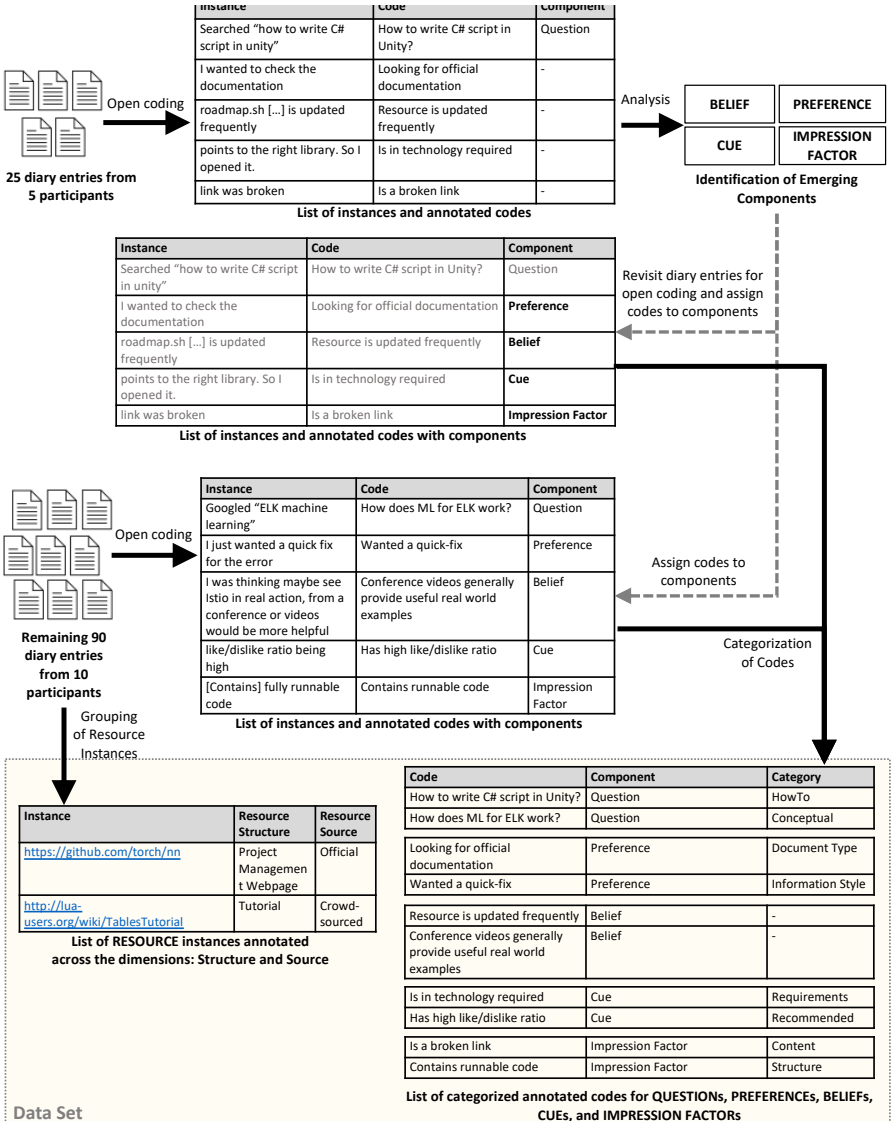

Fig. 3: Process followed to obtain the data set for this study. The tables obtained in the last steps (represented by the last row in the figure), together form our data set.

stratified manner such that the sample consisted of one entry from each day from each participant. The open coding process included annotating the diary entry's content for the information that the participant needed (Question), and their thought process during the search session. We additionally identified the web links to resources in the diary entries. As a result, we created 132

codes in total. Upon analysing the created codes, we identified four emerging components, namely, Preferences, Beliefs, Cues, and Impression Factors (see Section 3). The first author then revisited the 25 diary entries to verify the open coding, identify missed instances of the components, and assign created codes to one of the five components Question, Preferences, Beliefs, Cues, or Impression Factors. As the study progressed with new participants, the first author open coded the new diary entries and associated these codes to the appropriate component.

To characterize each component, the first author performed card-sorting of all the codes within each of Question, Preference, Cue, and Impression Factor. They created a coding guide for the different categories that they observed. To alleviate the bias of a single annotator, the other two authors used this coding guide to categorize the codes, and we performed inter-rater reliability tests on their categorization. The Cohen's Kappa between the two latter authors for each component was 0.91 for Questions, 0.74 for Preferences, 0.71 for Cues, and 0.79 for Impression Factors indicating at least substantial agreement in all four cases (Landis and Koch, 1977). All three authors resolved the disagreements via a collective discussion.

For Resources, all three authors together discussed and grouped the *instances* based on two dimensions: *structure* and *source*. We use the term *instance*, in the remainder of the article, to refer to each individual quotation in the diary entries that have been coded. It is possible that multiple instances have the same code if they are nearly identical in their semantics. We did not categorize Beliefs further because they are described rarely by our participants.

We noticed that in the diary entries, there existed meaningful connections between instances of different components. For example, multiple Resources were accessed to answer a single Question, and different Cues were used to select each of the Resources. Hence, for each search session, we identified these connections. We named the relation between two components according to the semantics of their connections.

We used statistical tests to investigate the association between different components in our data set, for example whether the *Authoritative* Cue was predominantly used to access *Official* documentation. We tested the null hypothesis that there is no association between any two categories of components (Sprent, 2011). We discuss the details of the statistical analysis in Section 7.

4.3 Replication Package

Our data set comprises of the coded Questions, Resources, Preferences, Beliefs, Cues, and Impression Factors and the connections between instances of these components. Figure 4 shows the number of instances of each component and their relations within our data set. Additionally, we share the documents needed to replicate the study including the demographic form,

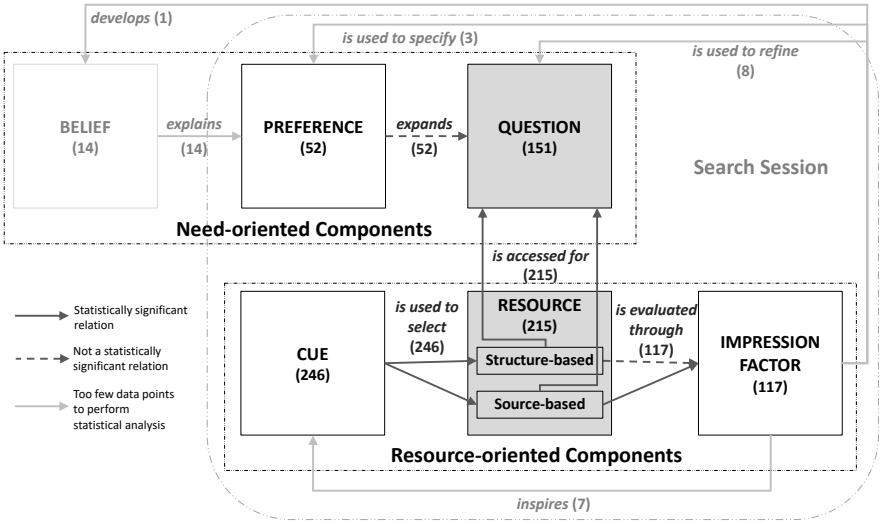

Fig. 4: Frequencies of the components and relations of the resource seeking model, in our data set.

diary entry template, questionnaire, and coding guide as supplementary material of this manuscript. The replication package is available online at `https://doi.org/10.5281/zenodo.7504510`.

5 Need-Oriented Components

We consider Questions, Preferences, and Beliefs as need-oriented because they involve aspects related to what the programmer is searching for.

5.1 Questions

We organized Questions into four categories (see Table 2). The most frequent (55 of 151) instances we observed are task-oriented ***HowTo***s, while the least frequent (4 of 151) instances are ***Debug*** questions. Both these categories have been identified in prior work by Rao et al. (2019) and Gallardo-Valencia and Sim (2011) as types of information need.

The ***Conceptual*** Question takes four different forms. Participants asked *what is* questions when they wanted to understand the fundamental knowledge about a particular component, e.g. "What are graph objects (component) in plotly (library)?" [P3]. Some participants also wanted to understand how a certain concept could be *applied* in a concrete context. For example, P4 searched for "How is genetic programming applied to automatic software repair?". Four of the participants had questions about the difference between two components or technologies, such

Table 2: Question categories. The last column corresponds to the number of instances of each category in our data set.

Category	Description	Example	#
HowTo	Questions that ask how to achieve a particular functionality. This also includes how to navigate the development environment, for e.g. to build an application. This does not include how to fix errors, which should be categorized as *Debug*.	*How to get the dot product between two word tensors using nn.dotproduct?*	55
Conceptual	Questions related to the conceptual understanding of technology and its components. This includes whether a component exists, what a component is, its syntax, what the difference between some components are, and how multiple technologies interact.	*What is "shallow copy"?*	53
Document Type	Questions related to finding a useful resource for learning or determining what information a particular resource or kind of resource provides. The question includes looking for a particular resource, or a kind of resource. It also includes questions about what information a particular resource provides about the technology.	*What is a tutorial I can use to learn about plotly?*	25
Debug	Questions related to why an error occurs or how to fix it. This includes trying to understand what an error message means.	*Why does the stale element reference error occur?*	14
Misc.	Questions that can not be exclusively grouped into one of the other categories, or does not contain enough context to identify the appropriate category.	*What are some best practices when coding in Python in the Selenium framework?*	4

as "What is the difference between softmax and softmin functions?" [P8]. P6 had four *syntax* related questions like "What is the syntax for do-while loop in Python?".

The ***Document Type*** category refers to a search intended to find a particular resource. In some cases, the participants knew exactly the website or resource that they are looking for: "What information about Genetic programming for automatic software repair can be found on blogging website Medium?" [P4]. In other cases, the participants only had an idea of the *kind of* resource they are looking for. For example, "What is a tutorial I can use to learn about plotly?" [P3]. We observed that the *Document Type* searches occurred in two scenarios. Participants began a search session with a general query for useful resources. When searching for "What are some tutorials on the Elastic stack?", P2 mentioned they "wanted to see what's out there". In the second case, participants specified the type of resource they want in the middle of a search session, normally after an unsuccessful search for information. They recounted past experiences and narrowed their search within familiar resources. For example, when searching "Genetic programming

Table 3: Preference categories. The last column corresponds to the number of instances of each category in our data set.

Category	Description	Example	#
Resource Type	Specifying the resource or kind of resource needed.	*Looking for a Github resource*	27
Information Style	Specifying that the information should be structured or presented in a particular manner. This includes organization, level of granularity, depth, recency of information, and if code or data examples are wanted.	*Wants a resource that provides a high-level overview*	25

Automatic Software Repair" in the Youtube search box, P4 mentioned "After trying my luck with Google search, I wanted to see if there were any resources on Youtube".

5.2 Preferences

We elicit two categories of Preferences (see Table 3). ***Resource Type*** indicates the specification of a particular resource or kind of resource. P2 specified in one search session that they "wanted in-depth API documentation" [P2], as opposed to resources that market the visualizations created using the technology. ***Information Style*** refers to the specification of the characteristics of the information. For example, P2 said in another entry: "I found a lot of useful search results ranging from specific API fields and tactics to more high-level overviews. Since I'm still learning, I went for the higher level overviews.". Both categories of Preferences are nearly equally frequent in our dataset (27 and 25, respectively).

The Preference plays an important role in the searching process because it describes the bias that the programmer has when looking for information. For example, a participant who is looking for a particular resource type may entirely ignore other resource types, despite them containing the information they need: "I found two results that looked promising as they were both related to torch and looked like documentation rather than Q&A by public. The remaining ones were on Stack Overflow which I ignored since I want to rely on the doc." [P8]

Together, Questions and Preferences constitute the complete picture of what a participant is looking for, thus indicating the information need.

5.3 Beliefs

A Belief justifies the existence of a certain Preference, by explaining *why* the Preference exists. P4 explained their preference of a video as "I find it easier to follow along; I feel like I can process information faster and I also have the option sometimes to just speed up the video faster if it's going slowly." In another search session, P4 was specifically looking for blog articles on the website Medium *because* "I find it a good place for quick reads that aren't very deep into the area". Thus, Beliefs do not

guarantee that preferences will always remain the same, because participants may change their preferences between or within a search session.

We recorded 14 instances of Beliefs in our data set. In all cases, the Belief is based on prior experience of the resource, and/or a general notion of what that resource would provide. For example, P2 searched for 'roadmap.sh': "There's a really good resource I already know about called roadmap.sh. It's updated frequently." [P2].

6 Resource-Oriented Components

Resources, Cues, and Impression Factors are resource-oriented components because they are anchored to the resources that programmers access during their search.

6.1 Resources

Table 4 shows the types of resources that we observed in our study. Each instance of a resource access is characterized across two distinct aspects, i.e. structure (seven categories), and source (three categories).

With respect to the structural aspect, we observed that 45 resources accessed are traditional types of software documentation like ***Reference*** documentation and ***Tutorials***, yet the majority of instances are unconventional learning resources. A total of 51 instances of resource accesses were to ***Forums***, 41 of which are Stack Overflow posts. The popularity of Stack Overflow can be attributed to two reasons. First, Stack Overflow is often placed prominently in the first page of the search results page when searching for documentation by Google (Treude and Aniche, 2018), biasing users to click on this resource: "Whenever I search for any results in the domain software development, I tend to see more of [particular] resources, for example, Stackoverflow comes first." [P10] Second, Stack Overflow acts as a hub for programmers: "As a practice, I always tend to open the first search result and most of the time it happens to be from Stack Overflow, and as we know, Stack Overflow is the go-to place for us [programmers]." [P6].

Thirty-four of the resource accesses were to ***Indexes***, which are directory pages that contain links to other useful resources. Of these, there are four cases where the participants found the information that they needed within the search engine results page, usually because of some keyword present in the result snippet. "Simple problem, didn't even click on a link. The minute I saw to_string I knew how to use it because I've used it a few times before." [P5] In eleven cases, participants used the search results to identify that they were not going to find the information they needed, and consequently either refined or aborted their search.

Articles are the resources that do not follow the structure of a tutorial, a reference documentation or a discussion forum, e.g. a Wikipedia or textbook page. We found 33 instances of *Article* accesses in our study.

P8 made the most accesses to the ***Project Management Webpage*** type of Resources with 15 accesses to Github repository pages. This was because

Table 4: Resource Categories. The last column corresponds to the number of instances of each category in our data set.

Category	Description	Example	#
Structure-based			
Forum	A post on Stack Overflow or another discussion forum	stackoverflow.com/questions/9695329/c-how-to-round-a-double-to-an-int	51
Tutorial	An instructive document that generally provides steps to follow to achieve a particular task	medium.com/@deependra.ariyadewa/envoy-in-kubernetes-373d5621e243	37
Indexes	A directory, registry or set of search results	virusu.github.io/3D_kibana_charts_vis/	35
Article	A Wikipedia entry, arbitrary article, or textbook page.	`en.wikipedia.org/wiki/Genetic_improvement_(computer_science)`	33
Project Management Webpage	A page in a project management (e.g. Github) repository, including an issue discussion thread	github.com/dzharii/awesome-elasticsearch	29
Video	A video (typically found via Youtube search)	youtu.be/6P1ivCvofuk	11
Reference	API reference documentation	intel.github.io/libva/group__api__core.html	8
Misc.	A resource that can not clearly be differentiated as one of the above categories	`www.cs.swarthmore.edu/~kwebb/cs31/s14/stackframe.pdf`	11
Source-based			
Official	Resource hosted on a technology's official website or by the company that created or is managing the technology	logz.io/blog/elk-stack-raspberry-pi/	82
Third-party	Resource is created or hosted by a single party that is not the creator	www.geeksforgeeks.org/this-pointer-in-c/	65
Crowd-Sourced	Resource contains content that is crowd-sourced	lua-users.org/wiki/TablesTutorial	52
Misc.	The source of the resource is unknown or multiple (in case of search results)	*Search Results*	16

the technology that the participant was learning, *torch*, as well as its documentation, are both hosted on the project management platform. Despite making only five Github accesses, most of P9's searches were spent on Github. In this case, the technology's documentation was mainly accessible via source code comments. As P9 described, "it was discouraging to see that upstream calls auto-generated documentation from the source their 'official documentation', [...] the actual source comments in the files for which documentation is *not* auto-generated seem to have quite a bit of information." [P9] We leave the study of source code comments to future work because it is stored in resources that are not primarily in a human speech language. We also observed participants accessing ***Videos***: "It [watching videos] is in general in my learning process. Even if it's learning a new programming language sometimes, I like watching a brief video." [P4]

In the source-based categorization, the majority (81) of resources are from ***Official*** sources, i.e. from the developers of the technology themselves or associated companies. Accesses to ***Third-party*** (65) and ***Crowd-sourced*** (52) documentation are nearly the same in our data set.

6.2 Cues

We elicit five types of Cues in our data set (see Table 5). The most frequent type with 66 instances is ***Recommended*** which indicates some implicit or explicit endorsement by other users, resources, or the search engine that this resource is useful. P7 explained "I clicked on this [first link] first because it was recommended by Google, so they had a little box with an excerpt of the information. This made me think the link would likely have the correct information". P6 echoes this by simply stating "I have realized that the first result tends to meet my expectations". We also observed that some participants opened multiple resources in different tabs and briefly look for cues to determine which resource to access. For example, the number of upvotes a resource has is usually not displayed in the search results, but within the resource page. P4 stated "I find the high like/dislike ratio a good indicator of the video being good and giving information correctly".

The category ***Requirements*** indicates that the resource seems to fulfil the participant's criteria. For example, after entering a query looking specifically for Medium blog articles, P4 "clicked on the second link that appeared, as I saw it was from medium.com". ***Familiarity*** with a resource is also a Cue that the participants used. In most cases, the participants recalled accessing the resource or similar resources and used their prior experience to determine if the resource could be useful. For example, "I clicked on this because I have watched videos by this creator before and liked his teaching style." [P7] In two cases, P8 clicked on the resource simply because they had clicked on it before, even though they did not remember their previous experience. P8 said "The second result, I opened anyway since I saw that I had visited this page before and I was curious to know if there was something interesting there."

Participants clicked on resources because they came from an ***Authoritative*** source. P2 explained: "Since it is recommended by the elastic devs themselves, it is hard to go wrong." The comments from P3 and P8 echo the same point: "Since

Table 5: Cue categories. The last column corresponds to the number of instances of each category in our data set.

Category	Description	Example	#
Recommended	The resource is chosen because it is among one of the top four search results, is featured by the search engine, has high number of upvotes, claps, likes, etc., is explicitly recommended by users, mentioned in another resource, or is generally popular or well-known.	*Google expands the blurb so must be relevant*	66
Requirements	The resource is the exact or similar resource wanted, or contains the exact characteristics needed. For e.g., if the resource is up to date, is in the correct or related domain, is the correct level of granularity, or its information is presented, styled, or formatted in a manner that is needed or preferred. If it is mentioned that this resource is accessed *because* it was useful in the past, it should be categorized as ***Familiarity*** instead.	*[Resource] Provides a formal introduction with historic information (wanted)*	53
Familiarity	The participant has some familiarity with the resource or the content that it contains, generally prefers it, or has used the resource in the past and has had a positive impression.	*Is a Stack Overflow link (preferred for technical questions)*	44
Authoritative	The source of the resource seems to be a credible or reputed authority. This includes cases where the resource is from the developers of the technology themselves.	*Resource is a reputed training website*	35
Keywords	The search result title/snippet or resource contains keywords that were present in the query, or are relevant to their question.	*Resource title seems closest to the error*	31
Misc.	Cues that can not be exclusively grouped into one of the other categories, or does not contain enough context to identify the appropriate category.	*Similar to what the participant intended to search for next*	17

the url is plotly.com, it's probably credible and good information" [P3], "I chose the official doc because I thought this would be more reliable." [P8] However, reliability is not the only reason to choose a resource by an authoritative source. P2 explained "[...] I like to get started on the software page to make sure I see updates to API" indicating such resources are generally up-to-date.

The ***Keyword*** category describes when participants assessed words in the search results to determine whether it could be a useful resource. When searching for the difference between the softmax and softmin functions, P8 clicked on the third search result because it "looked more promising since the title and description contained both words softmax and softmin." [P8] In one instance of searching how two technologies interact, P2 used the search term "kibana react visualization".

Table 6: Impression Factor categories. The last column corresponds to the number of instances of each category in our data set.

Category	Description	Example	#
Content	Comments related to the nature or quality of the information contained	*Information in resource is not friendly to Windows-users*	64
Pertinence	Comments that are about the specific context, such as a target domain, use-case or question	*Too early in learning [process] for this resource to be useful*	35
Structure	Comments related to the *organization* of the resource	*Contains good demo code*	18

Of the search results, they said: "I would have expected an actual kibana-react app to show up higher in the search results...". For P6, this expectation is so high that when clicking on the Stack Overflow post that is the first result, they said: "I do not usually read the question because I trust Google to give me the exact answer. I think it takes practice for us to get the right result as well from having to type the keywords." Participants felt they need not consciously look for matching keywords in search results.

We noted cases where a participant used multiple cues to determine whether to follow a link: "I clicked on the second one because I hope that since it is listed second, it might be related [to the query] and also since it is from official Pytorch doc." [P7] In this example, the participant considered that the resource is both *Recommended* and *Authoritative*. Sometimes, participants were unsure which resources may prove useful, because of the lack of clear Cues. P10 explained their strategy in one such case: "I open many links [from the search results page] in all new tabs. I do it for four or five links... I usually start from the first link." [P10]

6.3 Impression Factors

We grouped Impression Factors into three categories based on what the participants used to form their impression about the resource (see Table 6).

In the majority (64 of 117) of cases, participants evaluated the *Content* of the resource, making comments about its quality such as whether the information was sufficiently detailed, beginner-friendly, or up-to-date. For example, P2 noted about a resource that "this was from 2017, so specs may have changed with more recent Raspberry Pis." Participants also mentioned that resources were jargon-heavy or too advanced. P8 found that the Envoy official documentation "would require [readers to have] intermediate/advance knowledge of DevOps in order to quickly grasp the information". The comments on content can be useful for resource creators to improve the information presented in it and make it more accessible to readers.

Many participants made comments about the *Pertinence* of the resource to their needs, including whether they found the answer to their specific question. P9 was looking for information about how a particular encoder works when

they landed on a seemingly relevant project "ffvademo". However, they said "I read the README on the repository and saw that ffvademo is actually a decoder, not an encoder". Essentially, this type of Impression Factor is about the alignment with the programmers' information need. When trying to find out how to determine whether a word embedding contains a particular word, P8 accessed a Stack Overflow resource and commented about it: "The answers talk about how embedding layers work, which I am not interested in".

Participants also assessed the *Structure* of a resource, noting the presentation of content it contained and the way it was organized: "This website was very helpful- it had different sub topics on the left and there were many examples that helped me understand the concepts well" [P10]. Structure-related comments can be useful for resource creators to reflect upon and improve the organization based design of the resource.

7 Relations Between Components

We identified nine unique relations among the six components in the resource-seeking model. Five of these relations are infrequent: *explains*, *inspires*, *is used to refine*, *is used to specify*, and *develops*. We discuss these relations qualitatively in Section 7.4. For the four frequent relations, i.e. *expands*, *is accessed for*, *is used to select*, and *is evaluated through*, we noted that some instances were more commonly occurring than others. For example, participants often referred to *Forums* to answer their *HowTo* Questions. To quantitatively analyze the coincidence of the relations we adopted the Fisher's Exact Test.

Fisher's Exact Test is performed on categorical variables where the frequencies of co-occurrence between categories may be below five, and was originally proposed for a 2x2 matrix. Because our contingency tables are larger than 2x2, i.e. the number of categories of some components are more than two, we approximated the p-value using 200000 Monte Carlo simulations (Mehta and Patel, 2011).

Since three of the four relations to be statistically tested involve the Resource component, we performed two tests per relations, i.e. one for each Resource aspect: *Structure* and *Source*. Thus, we performed a total of seven statistical tests to determine if there is a significant association among the components in our model, one for each of the relations shown via dark grey arrows in Figure 4. To mitigate the Type-I error during multiple comparison tests, we applied the Bonferroni correction to the p-values (i.e. multiplying each p-value by 7, one for each of the statistical tests performed) calculated via the Fisher's Exact Test (Abdi et al., 2007).

Table 7 shows the Bonferroni-corrected p-values (henceforth referred to as p-values) of this analysis. The two tests for the *is accessed for* relation between Resource Structure and Question, and Resource Source and Question have a p-value lesser than α of 0.05. Similarly, the two tests for the *is used to select* relation between between Resource Structure and Cue, and Resource Source and Cue, and the *is evaluated through* relation between Resource Source

Table 7: Bonferroni-adjusted p-values calculated by Fisher's Exact test using 200000 Monte Carlo simulations of connections between each pair of model components. The numbers in bold indicate statistically significant relations.

	Question	Cue	Impression
Preference	1.0472	-	-
Resource Structure	**3.5*e*-5**	**3.5*e*-5**	0.4064
Resource Source	**3.5*e*-5**	**3.5*e*-5**	0.0022

		Question				
		Conceptual	Document Type	HowTo	Debug	Misc	Total
Preference	Type	7	13	5	2	0	**27**
	Style	12	5	6	1	1	**25**
	Total	**19**	**18**	**11**	**3**	**1**	**52**

Fig. 5: Contingency table of *expands* relation between Preferences and Questions

and Impressions also result in a p-value lower than 0.05. Hence for these five tests, we reject the null hypothesis that the distribution of values between the component pair under test is due to chance, and we thus refer to the participating two components as "associated". For the remaining two tests, i.e. *expands* relation between Preferences and Questions and the *is evaluated through* relation between Resource Structure and Impression, we cannot reject the null hypothesis. Figures 5 and 8 show the contingency table for these two relations.

For each identified pair of associated components, we computed the adjusted standardized residuals (henceforth referred to as *residuals*) for each cell in the contingency tables (Sharpe, 2015). Residuals are the normalized difference between the expected and observed frequencies of the relations, and reflect the effect size for the relation between two components in our model. Following common practice (Sharpe, 2015), if a standardized residual is greater than +2, we consider the effect to be meaningful and the associated relation to be "favored" by our participants. For example, we say that participants favored *Forums* to answer *HowTo* Questions based on the residuals, to indicate that the relationship "*Forums is accessed for HowTos*" occurs more than expected by chance in our data set. This does not necessarily mean that participants consciously expressed a favoritism for using *Forums* to answer *HowTo*s.

When a residual is lesser than -2, we interpret the corresponding association as being "disfavored". We examine the favored residuals (stated in bold) and contingency tables of associated components in more detail below.

7.1 Resource $\xrightarrow{is\ accessed\ for}$ Question

Figure 6 illustrates which Resource *is accessed for* what kind of Question.

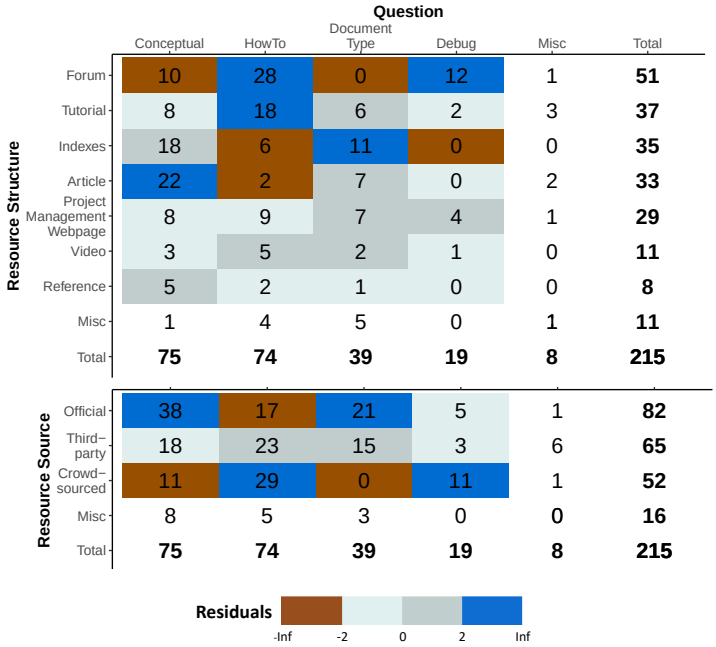

Fig. 6: Adjusted Standardized Residuals and Contingency table for the *is accessed for* relation between Resources and Questions. The values in each cell represent the frequency of connections between the pair of categories. Non-colored cells indicate that they were not included in the statistical analysis.

Our analysis reveals that **participants favored *Articles* for *Conceptual* Questions. For *HowTos*, participants favored *Forums* and *Tutorials*.** *Forums* provide flexibility to users to search for information within the exact context needed: "I feel I trust this (Stack Overflow) website, so I hope to find the answer [here]. If I don't find the answer, there is an option for me to ask people for help. That gives me more leverage." [P10] Dondio and Shaheen (2019) showed that Stack Overflow can be as effective as a traditional textbook and course-based instruction for students to gain practical knowledge.

Participants also favored *Forums* for *Debugging* Questions. P8 said of their general search behavior: "If it's a bug or an issue that's not specifically working, I tend to go to Stack Overflow.".

That **participants favored training resources like *Tutorials* to answer task-oriented *HowTo* questions**, seems an intuitive result. Training materials have evolved to prioritize *procedural* information (Carroll, 1990), i.e. information that directly supports actions, based on prior work that found software users use prior knowledge and procedures stated in text to perform tasks (Mack et al., 1983).

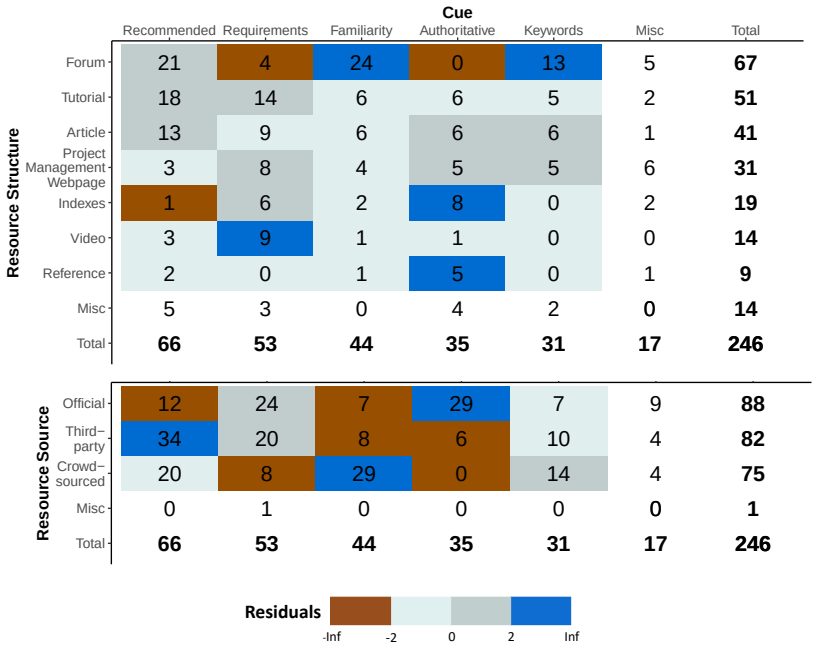

Fig. 7: Adjusted Standardized Residuals and Contingency table for *is used to select* relation between Resources and Cues. The values in each cell represent the frequency of connections between the pair of categories. Non-colored cells indicate that they were not included in the statistical analysis.

Participants favored *Indexes* for *Document Type* Questions. Ten of these eleven cases were to either home pages of a website (e.g. `freedesktop.org`) or directory pages of a particular topic on the website (e.g. `www.elastic.co/demos`). When participants used search results as a resource, it was useful to assess the pertinence of search results. For example, P2 looked at the search results and found that none were dated post-2017, and so they did not continue searching further. In another such case, P9 determined they would not easily find the resources containing the information they needed because "All the results looked auto generated (they had paths for page names)", and so aborted their search soon after.

Across the source dimension, ***Official* Resources were favored by participants for *Conceptual* and *Document Type* Questions**. Whereas, **participants favored *Crowd-sourced* Resources similarly to *Forums***.

7.2 Cue $\xrightarrow{\textit{is used to select}}$ Resource

The reason *why* a particular resource is accessed varies for different resources (see Figure 7). Participants favored the ***Requirements* Cue when selecting *Videos***. This can be as broad as believing that watching a video would be better than reading text: "I figured I would find a youtube video on it more easy to follow than a textual post that might contain more jargon I don't follow." [P5], or as specific as that the video is short: "Clicked on this because its short length compared to others." [P7].

***Familiarity* was favored when selecting *Forums* and similar *Crowd-sourced* Resources**. P10 points out: "It has come over time that I have the bias that these [W3Resource, Stack Overflow] websites are good, because I find many answers in them."

Participants favored *Keywords* when accessing *Forums*: "I clicked on this first because it had the same wording as what I was looking for. So it made me think that it would be a good place to find the answer to my question." [P7]

The *Authoritative* Cue was favored for selecting *Reference* documentation and *Indexes* by participants. The former link is intuitive since reference documentation generally accompanies the technology as *official* documentation. However the latter link is less obvious and is likely because when accessing a home page of a technology or documentation, participants consider the credibility of the source of the web sites they are clicking on.

Participants favored the *Recommended* Cue for *Third-party* Resources. A majority of such connections occur because the Resource was featured by the search engine. This shows that participants strongly trust a search engine's ranking algorithm to suggest a pertinent Resource.

7.3 Resource $\xrightarrow{\textit{is evaluated through}}$ Impression Factor

Figure 8 shows the contingency tables for the *is evaluated through* relation between Resources and Impression Factors. While Resource Source is associated with Impression Factor, Resource Structure is not.

Participants favored evaluating *Official* Resources based on their *Content*. This may be because a certain level of quality is expected of a resource if it is from an authoritative source, especially if the original technology is well presented: "MITRE is a pretty detailed framework so I wasn't surprised that their info [in the official documentation] was in-depth and dense" [P2].

Evaluating *Pertinence* to participants' context was favored for *Crowd-sourced* Resources. While in some cases, participants were able to find the information they needed, in others, they faced various issues while looking through *Crowd-sourced* resources. This included not finding posts that ask the same questions they have and not finding any answers to a post. Additionally, despite a lot of information available online, participants found it difficult to find the answer to their exact questions. P5 explained after unsuccessfully looking through three resources: "I felt like I was looking for an answer that was obvious, but I was only seeing questions that I wasn't asking".

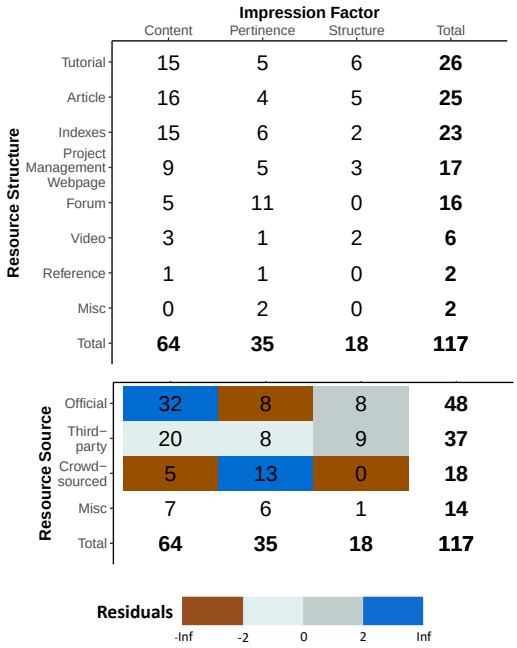

Fig. 8: Contingency table of *is evaluated through* relation between Resources and Impression Factors

7.4 Infrequent Relations

We observed an additional five relations that each occurred less than fifteen times. Except for the relation, Belief *explains* Preference, the other four relations involve a change in thought process or action based on the impression of a resource.

Belief $\xrightarrow{explains}$ Preference

In fourteen cases, we observed participants describing the reason for their Preference based on an existing Belief. The Belief is usually developed from prior search experiences. In a majority of cases, a Belief was used to explain a Preference on the *Resource Type*, either describing the specific website or a format. P4 explained their Preference for articles hosted on the website 'Medium': "I have found some very useful articles in Medium before, when learning about a certain area, and thought it would be a good resource again to learn something new. It usually has a lot of visual examples and explains things quite well." In two cases, the Belief explained a Preference on the *Content Style*, once while looking for an "easy explanation" and the other when looking for "code examples".

[Impression Factor] —*inspires*→ [Cue]

We observed six instances in which an Impression Factor explicitly *inspired* a new Cue. When a participant initially browses through resources, they may use some Cue to assess the pertinence of a resource. After clicking on a resource and evaluating it, they may realize that there is an additional criterion they need, and employ it as a future Cue when searching for more resources. This relation occurs between accesses of multiple resources for the same question within a single search session.

When searching for how to deploy Envoy in Kubernetes, P11 landed on the first tutorial in the search results. However, upon reading through it, they discovered that "the author skipped a lot of basic information ... because he assumes people who read it will have advanced knowledge [of] and familiarity with Kubernetes". The participant proceeded to another resource, a blog hosted on 'Medium', stating that "Usually Medium contains a lot of entry-level tutorials for any technology". Thus their new Cue of looking for a beginner tutorial was inspired by their impression of the previous resource. Two other instances of the relation involved finding a resource too advanced, and thus using a new Cue to find a resource that is better adapted to beginners.

[Impression Factor] —*is used to refine*→ [Question]

In eight instances, participants refined their search query as a direct consequence of assessing a Resource as not suitable to their needs. This relation occurs within a single search session but between two iterations of searching for similar information.

Most instances of this relation resulted from the unsatisfactory outcome using a first search query. In two cases, participants made the assessment on the first few search results: "None of the first links looked like it might provide an answer, so I searched again." [P5] In these cases, the original query changed only by one or two terms. In other cases, participants realized the resources did not have the exact information they needed. After finding a Stack Overflow link with a question that did not exactly match the P5's Question about a particular line in a "valgrind" error summary, the participant realized their query may be too specific. They reframed their search query to look for general information about the error message, instead. Query refinement may also involve moving search platforms, as in one instance where the participant switched from searching on Google to on Youtube directly.

[Impression Factor] —*is used to specify*→ [Preference]

In addition to refining a question, in three instances, participants specified a new Preference based on their impression of a Resource. In two of these cases, the motivation to refine the query was an incorrect *Resource type* of the resources. P4 scrolled through search results after searching for "Genetic

programming Automatic Software Repair" but did not find a web page that was not a scientific paper. They refined their query, and additionally stated the specific resource they would be looking for going forward: "As I didn't want to read a scientific study as an introduction, I tried my luck on finding it on Medium, as I find it a good place to have quick reads that aren't very deep into the area." [P4]

In the third instance of this relation, a participant introduced a new *Content Style* Preference, after reading a Stack Overflow post that did not provide a "precise understanding" [P5] of "shallow copying". P5 subsequently stated that they were looking for "a nice and easy explanation of a concept".

Impression Factor $\xrightarrow{develops}$ Belief

An Impression Factor can also play an integral role in influencing future searches for information. For example, a positive impression of a resource can result in a searcher returning to this resource during future searches. In one search session, P2 came across a guide hosted on https://logz.io, and found it to be an "exceptional" tutorial. They said, "the high quality of Logz.io search results, especially for this query, will probably make me look them up first for future learning in DevOps" [P5]. Although, we did not observe the conversion of the Impression Factor into a Belief within our limited study days (and thus the connection does not exist in our data set), P2's statement suggests the potential for an Impression Factor to form the foundation for a new Belief.

8 Implications

Our resource-seeking model captures the different factors that play a role in the thought process of programmers as they navigate to a resource that could answer their question. Our observations of the types of Preferences, Cues and Impression Factors and how they relate to the Questions and Resources involved in a search session provide insight to resource creators on how to improve the appeal of a resource for their target audience. Our results also have implications for the design of search tools and can help programmers improve their on-line search techniques: "I noticed some patterns that I usually follow and I thought about how I can improve them." [P8] We discuss important observations and their implications from our study below.

Preferences, potentially backed by pre-existing Beliefs, are used to elaborate criteria for searching for resources to answer Questions. We observed that participants sometimes had expectations of the resources, prior to their search, and justified these expectations by their prior experiences. For example, P2 searched for "awesome ELK stack github", explaining that "My experience with the awesome lists is that they're both open source and up to date". In our model, we formalize this behavior with the two relations: Belief *explains* Preference which *expands* Question. *Equipped with the knowledge that programmers could have preferences during resource seeking, resource creators can study the behavior of target audiences to gain insight about their expectations, and thus*

make conscious decisions about whether to satisfy identified expectations and how to do so. P2 and P11 mentioned that resources did not provide sufficient hands-on learning material such as practice questions, and their content was not well supported by diagrams, respectively. Upon identifying this preference of *Content Style*, resource creators can consider the trade-offs of adding these types of content in the resources to increase appeal (Arya et al., 2021). *Search tools can be enhanced by allowing programmers to customize their needs based on their criteria.* P4 suggested a filtering mechanism that would allow programmers to specify the types of resources to search among, so that they would not have to wade through long results pages of non-preferred resource types. Such a filter would allow programmers to specify their *Document Type* Preference while searching.

Our participants' search behavior illustrates that **they accessed different Resources to answer different types of Questions** when learning a new technology. All participants in our study accessed more than one type of resource through the study. Thus, searchers are forced to access multiple resources to satisfy their information needs, despite prior work discovering that there is some correspondence in information between different documentation types (Arya et al., 2020). Furthermore, our analysis of the *is accessed for* behavior reveals that some resources are favored when answering particular types of questions. Whether this is because of the participants implicit thinking process, or because of the nature of the resources themselves, requires further investigation. *Resource creators can be informed by this access behavior to tune particular types of resources to answer particular types of questions, allowing search tools to efficiently direct searchers to appropriate resources.* Meanwhile, *a centralized page indexing all the pertinent resources for learning a new technology would be useful for programmers to navigate the resource space*. Two participants mentioned they would prefer documentation to be standardized for ease of learning. P3 quoted the neatness of Java API documentation, explaining that other programming language documentation should follow suit. In the questionnaire, the other participant went as far as suggesting: "It would be nice if all software distributed complete man pages and info pages with documentation." [P9]

There exist visible and non-visible hints, or Cues, related to the quality and familiarity of a resource, to determine whether it is pertinent to information needs. Our model reifies the "scents" that information seekers use to find the information they need, according to information foraging theory (Pirolli and Card, 1999), as Cues. We observed that these Cues can be either *objective* or *subjective* to programmers' wants and needs. For example, while the *Recommended*, *Authoritative*, and *Keywords* Cues are relatively objective, the *Requirements* and *Familiarity* Cue are influenced by the search context and the programmers' mindset. Prior work has focused largely on the *objective* Cues (see Section 2.2). However our observation that participants use different Cues to select different Resources suggests that *it is also important to incorporate search customization to support programmers in their use of subjective* Cues.

Different Impression Factors may be used to evaluate different Resources. We observed that participants use different criteria to evaluate the quality and usefulness of a resource, especially depending upon the source of the resource. Particularly, some participants mentioned that learning resources should be easy to understand for beginners. P5 explained why they avoided a certain C++ *Forum*: "[...] the people who answer questions there get into a lot of detail using words that I don't follow." They explained that the case is different with Stack Overflow answers, where predominantly lesser technical jargon is used making answers easier to understand, thereby motivating their use of the web site. P11 also stated that searching is especially tougher for beginners as it is expected that they are aware of technical jargon - an interesting paradox since beginners are still in the learning phase of technical terms and details. *Our model provides the dimensions of a resource that programmers would use to evaluate it, and thus, the aspects of quality that resource creators must consider*.

A feedback loop can be formed using Impression Factors during the search process. We observed that participants used Impression Factors to refine their questions and clarify their criteria of resource type and content style. For example, while looking for general information about what graph objects are in the plotly library, P3 found a resource that contained too much text. After accessing this resource, they explained that their Preference was resources with precise information, such as a list of methods and how to use them. Only after evaluating a Resource, did they consider the Preference more seriously and use the criterion as a Cue for the subsequent resource access. Thus, Impression Factors *encourage programmers to reflect on their criteria and context during search.* Furthermore, Impression Factors *can be used to inform the creation of tools that assist in query refinement during the resource seeking process.* (Lu and Hsiao, 2017)

Threats to Validity

As part of our study, we required participants to self-report the steps they take in their search. This poses a threat to internal validity because the steps may not be reported exactly as they are performed. To verify the accuracy of reported steps, we performed an interview at the end of the diary study in which we asked the participants to recreate two search sessions. This way, we were able to determine that the reported entries are correct. In only three entries participants made corrections to their diary entries, and in no case did the corrections significantly impact our findings.

The analysis methodology involves manual annotation which are subjective. To alleviate annotator bias and measure the subjectivity of the task, three annotators were involved in categorizing our data set. We measured the agreement scores between them and found that they had substantial agreement (see Section 4.2). To validate our coding guides, we asked two external annotators who had little to no context about our study to annotate our data set using the coding guides. They achieved agreement scores of 0.73 for Questions,

0.74 for Preferences, 0.73 for Cues, and 0.63 for Impression Factors, indicating substantial agreement (Landis and Koch, 1977) for all four components.

We also face the threat to external validity, i.e. the generalizability of our observations to other programmers who are learning a new technology. Our sampling of ten participants does not allow a generalization from sample to population. This is inevitable for diary studies, and our sample size is consistent with the norm for this research method. The implication is that our observations may be limited to the behaviour of our participants. However, our goal is not to make claims about general population behavior, but to theorize the factors programmers think about as they access resources. Our findings are also supported by robust statistics of the relations between components in our data set. Thus, we refrain from making assumptions and comments about general behaviour of programmers. Furthermore, our proposed model represents the possible aspects of a search session. The current set of components may not be exhaustive, and can be augmented with additional components that may be observed in future work.

9 Conclusion

Based on a diary and interview study of ten programmers learning a new technology, we propose a resource seeking model that captures systematically how programmers make decisions when navigating between online learning resources for software technology. Specifically, we found that programmers may have Preferences of the kind of Resources they are looking for, which may be backed by Beliefs from previous experiences. These two components provide context for the Questions that programmers are looking to solve. Furthermore, programmers may use various Cues to select Resources to click on during their search, and evaluate them based on certain Impression Factors. We elicit nine relations between the six components of our model, and investigate the nature of these relations. We found, for example, that participants depended on the Cue *Familiarity* to select Resources that are *Forums* more than expected.

Learning resource creators can leverage the model components to estimate how target readership would perceive the usefulness of resources. As a result, the resource creation process would consider both perspectives - the goal of the resource, as well as typical reader behavior to arrive at the resource. Our observations can also inform search tools to alleviate time and effort spent in search. Furthermore, our model provides a means for resource seekers to reflect upon their search process, and optimize for efficient searching of pertinent resources.

This work lays the foundation for further investigation into the behavior of programmers searching for online learning resources, and subsequent stages of the foraging process including information sensemaking. Future work can leverage our model to study the resource seeking behaviour of a wider population of programmers. Our model could also be applied to other online resource seeking contexts. For example, our observation of infrequent relations points to

other emerging relations between components. The model remains adaptable to additional components, relations, and timelines. Our work also has broader implications for software documentation research, providing insight to support programmers in determining the quality and pertinence of documentation at different stages of their resource seeking process, including before and after they access a resource.

Declarations

Funding: This work has been funded by the Natural Sciences and Engineering Research Council (NSERC) Discovery Grant.
Conflicts of Interests: The authors have no conflicts of interests to declare that are relevant to the content of this article.
Competing Interests: The authors have no competing interests to declare that are relevant to the content of this article.

References

Abdi H, et al. (2007) Bonferroni and šidák corrections for multiple comparisons. Encyclopedia of measurement and statistics 3:103–107

Arya DM, Guo JLC, Robillard MP (2020) Information correspondence between types of documentation for APIs. Empirical Software Engineering 25(5):4069–4096

Arya DM, Nassif M, Robillard MP (2021) A data-centric study of software tutorial design. IEEE Software

Bai GR, Kayani J, Stolee KT (2020) How graduate computing students search when using an unfamiliar programming language. In: Proceedings of the 28th International Conference on Program Comprehension, Association for Computing Machinery, ICPC, p 160–171

Brandt J, Guo PJ, Lewenstein J, Dontcheva M, Klemmer SR (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, Association for Computing Machinery, CHI '09, p 1589–1598

Carroll J (1990) An overview of minimalist instruction. In: Annual Hawaii International Conference on System Sciences, IEEE Computer Society, vol 4, pp 210–219

Chattopadhyay S, Nelson N, Au A, Morales N, Sanchez C, Pandita R, Sarma A (2020) A tale from the trenches: Cognitive biases and software development. In: Proceedings of the 42nd ACM/IEEE International Conference on Software Engineering, Association for Computing Machinery, New York, NY, USA, ICSE '20, p 654–665

Dondio P, Shaheen S (2019) Is Stack Overflow an effective complement to gaining practical knowledge compared to traditional computer science learning?

In: Proceedings of the International Conference on Education Technology and Computers (ICETC), p 132–138
Duala-Ekoko E, Robillard MP (2012) Asking and answering questions about unfamiliar apis: An exploratory study. In: 2012 34th International Conference on Software Engineering (ICSE), IEEE, pp 266–276
Earle RH, Rosso MA, Alexander KE (2015) User preferences of software documentation genres. In: Proceedings of the 33rd Annual International Conference on the Design of Communication, Association for Computing Machinery, SIGDOC '15
Erdem A, Marsella S, Johnson W (1998) Task oriented software understanding. In: Proceedings of International Conference on Automated Software Engineering, IEEE Computer Society, p 230
Erdos K, Sneed HM (1998) Partial comprehension of complex programs (enough to perform maintenance). In: Proceedings. 6th International Workshop on Program Comprehension. IWPC'98, pp 98–105
Escobar-Avila J, Venuti D, Di Penta M, Haiduc S (2019) A survey on online learning preferences for computer science and programming. In: Proceedings of International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), pp 170–181
Gallardo-Valencia RE, Sim SE (2011) What kinds of development problems can be solved by searching the web?: A field study. In: Proceedings - International Conference on Software Engineering, pp 41–44
Jääskeläinen R (2010) Think-aloud protocol. Handbook of translation studies 1:371–374
Ko AJ, DeLine R, Venolia G (2007) Information needs in collocated software development teams. In: 29th International Conference on Software Engineering, pp 344–353
Landis JR, Koch G (1977) The measurement of observer agreement for categorical data. Biometrics 33:159–174
Lawrance J, Bellamy R, Burnett M, Rector K (2008) Using information scent to model the dynamic foraging behavior of programmers in maintenance tasks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, CHI '08, p 1323–1332
Lazar J, Feng JH, Hochheiser H (2017a) Chapter 11 - Analyzing Qualitative Data. In: Lazar J, Feng JH, Hochheiser H (eds) Research Methods in Human Computer Interaction (Second Edition), second edition edn, Morgan Kaufmann, Boston, pp 299–327
Lazar J, Feng JH, Hochheiser H (2017b) Chapter 6 - Diaries. In: Lazar J, Feng JH, Hochheiser H (eds) Research Methods in Human Computer Interaction (Second Edition), second edition edn, Morgan Kaufmann, Boston, pp 135–152
Liu MX, Kittur A, Myers BA (2021) To reuse or not to reuse? a framework and system for evaluating summarized knowledge. Proceedings of the ACM on Human-Computer Interaction (CSCW1)

Lu Y, Hsiao IH (2017) Personalized information seeking assistant (pisa): From programming information seeking to learning. Information Retrieval 20:433–455

Mack RL, Lewis CH, Carroll JM (1983) Learning to use word processors: Problems and prospects. ACM Transactions on Information Systems 1(3):254–271

Marques A, Bradley NC, Murphy GC (2020) Characterizing task-relevant information in natural language software artifacts. IEEE International Conference on Software Maintenance and Evolution (ICSME) pp 476–487

Mehta CR, Patel NR (2011) IBM SPSS Exact Tests. Armonk, NY: IBM Corporation

Meng M, Steinhardt S, Schubert A (2019) How developers use API documentation: An observation study. In: Communication Design Quarterly Review, vol 7, pp 40–49

Nadi S, Treude C (2020) Essential sentences for navigating stack overflow answers. In: International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, pp 229–239

Piorkowski D, Fleming SD, Scaffidi C, Burnett M, Kwan I, Henley AZ, Macbeth J, Hill C, Horvath A (2015) To fix or to learn? how production bias affects developers' information foraging during debugging. In: IEEE International Conference on Software Maintenance and Evolution (ICSME), pp 11–20

Pirolli P, Card S (1999) Information Foraging. In: Psychological Review, vol 106, pp 643–675

Pirolli P, Fu Wt (2003) SNIF-ACT: A model of information foraging on the World Wide Web

Rao N, Bansal C, Zimmermann T, Awadallah AH, Nagappan N (2019) Analyzing web search behavior for software engineering tasks. URL `http://arxiv.org/abs/1912.09519`, `1912.09519`

Robillard MP, Marcus A, Treude C, Bavota G, Chaparro O, Ernst N, Gerosa MA, Godfrey M, Lanza M, Linares-Vásquez M, Murphy GC, Moreno L, Shepherd D, Wong E (2017) On-demand Developer Documentation. In: International Conference on Software Maintenance and Evolution (ICSME), IEEE, pp 479–483

Sadowski C, Stolee KT, Elbaum S (2015) How developers search for code: A case study. In: Proceedings of the 10th Joint Meeting on Foundations of Software Engineering, Association for Computing Machinery, ESEC/FSE, p 191–201

Sharpe D (2015) Chi-square test is statistically significant: Now what? Practical Assessment, Research, and Evaluation 20(1):8

Sillito J, Murphy GC, De Volder K (2006) Questions programmers ask during software evolution tasks. In: Proceedings of the SIGSOFT International Symposium on Foundations of Software Engineering (FSE), Association for Computing Machinery, p 23–34

Sprent P (2011) Fisher Exact Test, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 524–525

Srinivasa Ragavan S, Kuttal SK, Hill C, Sarma A, Piorkowski D, Burnett M (2016) Foraging among an overabundance of similar variants. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, Association for Computing Machinery, New York, NY, USA, CHI '16, p 3509–3521

Teevan J, Alvarado C, Ackerman MS, Karger DR (2004) The perfect search engine is not enough: A study of orienteering behavior in directed search. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, p 415–422

Treude C, Aniche M (2018) Where does google find API documentation? In: In Proceedings of International Conference on Software Engineering, ACM, pp 23–26

Xia X, Bao L, Lo D, Kochhar PS, Hassan A, Xing Z (2017) What do developers search for on the web? Empirical Software Engineering 22:3149–3185

Xie I, Joo S (2012) Factors affecting the selection of search tactics: Tasks, knowledge, process, and systems. Information Processing and Management 48(2):254–270