Studying Software Developer Expertise and Contributions in Stack Overflow and GitHub

Sri Lakshmi Vadlamani
School of Computer Science
Carleton University
Ottawa, Canada
Sri.Vadlamani@carleton.ca

Olga Baysal
School of Computer Science
Carleton University
Ottawa, Canada
Olga.Baysal@carleton.ca

***Abstract*—Knowledge and experience are touted as both the necessary and sufficient conditions to make a person an expert. This paper attempts to investigate this issue in the context of software development by studying software developer's expertise based on their activity and experience on GitHub and Stack Overflow platforms. We study how developers themselves define the notion of an "expert", as well as why or why not developers contribute to online collaborative platforms. We conducted an exploratory survey with 73 software developers and applied a mixed methods approach to analyze the survey results. The results provided deeper insights into how an expert in the field could be defined. Further, the study provides a better understanding of the underlying factors that drive developers to contribute to GitHub and Stack Overflow, and the challenges they face when participating on either platform.**

The quantitative analysis showed that JavaScript remains a popular language, while knowledge and experience are the key factors driving expertise. On the other hand, qualitative analysis showed that soft skills such as effective and clear communication, analytical thinking are key factors defining an expert. We found that both knowledge and experience are only necessary but not sufficient conditions for a developer to become an expert, and an expert would necessarily have to possess adequate soft skills. Lastly, an expert's contribution to GitHub seems to be driven by personal factors, while contribution to Stack Overflow is motivated more by professional drivers (i.e., skills and expertise). Moreover, developers seem to prefer contributing to GitHub as they face greater challenges while contributing to Stack Overflow.

***Index Terms*—Software developer expertise, developer contributions, developer profiles, GitHub, Stack Overflow, qualitative study, survey, developer perception.**

I. Introduction

Wikipedia defines an expert as "…someone who has a broad and deep competence in terms of knowledge, skill, and experience through practice and education in a particular field". In other words, an expert is an individual who has demonstrated their special skills or knowledge in solving problems. An expert can have these abilities innate in them or they could have possibly inculcated them over time and through conscious deliberate practice. While there is some correlation between amount of experience and level of expertise, it is rather weak [1]. For example, one may spend years or decades trying to learn how to play the piano, and still achieve a rather low level of proficiency.

In some fields it is easy to identify experts and to measure and rank their expertise because there are objective measures to identify and measure expertise. For example, in sports, there are numerous objective measures (e.g., the time to run a marathon) that can be used to reliably rank experts. However, in software engineering, it is much more difficult to find reliable and objective metrics to measure expertise, and this has been previously reported in the software engineering literature [2], [3].

Traditionally, software development is considered to be a collection of many tasks: design, coding, testing, maintenance. However, software development field has evolved quite rapidly since the early 21st century and has created various avenues for specialization and expertise, such as expertise in programming languages (e.g., JavaScript, C/C++, Python, JavaScript, SQL), frameworks and libraries (e.g., Node.js, Angular.js, Spring, ASP, Django, JSF, Cocoa, React, Hadoop, Spark), technologies (e.g., IoT, deep learning, ML, computer vision, blockchain, quantum computing, cloud computing, augmented/VR) and databases (e.g., MySQL, PostgreSQL, Oracle, Redis, Cassandra).

This paper attempts to understand whether knowledge and experience are both necessary and sufficient conditions to become an expert. We are also interested in understanding developer behaviour and decisions on why they contribute to different online collaborative platforms. While software research community offers a plethora of studies on how developers participate, contribute to and interact with each other on various collaborative platforms [4]–[10], there is a lack of qualitative research studying developer motivation and behaviour that drives their contributions to these platforms. Therefore, our research is organized around an exploratory survey that we design based on the state-of-the-art qualitative research [4], [11], [12] and our observation on how developers interact and participate on Stack Overflow and GitHub. To analyze the survey data, we applied a mixed methods research approach that involves both quantitative and qualitative analyses.

Although, there are numerous ways to identify experts and their expertise, this paper however, focuses only on two platforms to help measure the expertise of a software developer: GitHub (GH) and Stack Overflow (SO). GitHub is the most

popular social coding platform that allows massive worldwide collaboration for software developers on open source projects [6], [7]. On the other hand, Stack Overflow is the most popular question answering website for software developers, providing a large number of code snippets and text on a wide variety of topics [13], [14]. SO helps programmers and software developers to discuss their (technical) day-to-day problems with others within the software development community and help solve their problems. At a fundamental level, both platforms offer very different set of features. However, both transcend physical and language boundaries and serve as a valuable resource to modern day developers across the globe in their quest to become an expert software developer. This is evident from the current usage of these platforms worldwide by developers. In a most recent data dump from December 9th 2018[1], Stack Overflow lists over 42 million discussion posts from almost 10 million registered users, while GitHub's search results report over 45.5 million users as of May 28, 2020[2].

This research study could benefit multiple stakeholders. Employers could benefit from knowing trends in specialization and understanding characteristics of an expert as it would help them hire the right expert at lower search-costs. Researchers, on the other hand, could benefit by designing studies and developing theories related to multi-layered expertise and possibly expand the number of platforms to included developer profiles and contributions from Twitter and LinkedIn. Software developers themselves could possibly benefit by learning the various factors that lead to becoming an expert software developer and the various clusters of specialization that they ought to focus on. Knowing the trends in specialization would help developers formulate their goals and acquire the right set of tools to be successful at the job. Also, understanding the motivations that drive developers to contribute and participate to different online platforms can help the platform's UX and user success teams to further improve platforms by eliminating some of the current challenges that developers face.

This paper makes the following contributions:

- A qualitative study with the developers active on both SO and GH who participated in our survey on the topic of developer expertise and their motivation for contributing to each platform.
- A survey analysis that offers insights into the developer perception of what constitutes to the notion of an expert developer, as well as what factors drive their contribution on both platforms.
- A publicly available replication package [15].

II. Related Work

Software development is divided into a collection of tasks, of which the important ones are: programming/coding, testing and debugging. Bergernsen et al. [16] proposed an instrument to measure the first aspect (programming), based on a fixed set of tasks pertaining to it. However, this measure might not help in identifying expertise in software development, as modern-day software developers have diversified their expertise across multi-specialties and the importance of soft skills has increased over time, i.e., this measure has no objective manner to measure their expertise in these multiple facets of software development.

Baltes and Diehl [2] have presented the first conceptual theory of software development expertise that is grounded in data from mixed methods survey of 335 software developers, literature review on expertise and focused surveys to come up with a comprehensive conceptual theory of expertise. They note that experts' performance may be inversely proportional to time and that experience and expertise may not necessarily be correlated. However, their study only focused on expertise in Java programming language.

Montandon et al. [17] have studied the question of identifying experts in software development by focusing their efforts on identifying experts in usage of libraries and frameworks among GH users. Supervised and unsupervised ML classifiers were used to identify experts in the three most popular JavaScript libraries (facebook/react; mongodb/nobe-mogodb; & socketio/ socket.io). A ground truth of expertise was built on a self-reported survey of 575 developers on these libraries. The paper tackles two research questions: 1) identifying the accuracy of the ML classifiers in identifying experts in libraries, and 2) the features that distinguish an expert from a novice in the studied libraries. The ML classifiers F-measure turned out to be low and the authors interpreted this performance might be due to usage of GitHub as a full proxy for expertise and they note that experts rarely contribute to public GitHub projects. On the second question, they rely on clustering to identify experts with similar feature values. Their results suggest that GitHub could be a partial proxy for expertise in frameworks and libraries. This paper differs from Montandon et al. [17] by allowing the respondents to self-select themselves and by choosing two platforms (GH and SO) to help cross-validate. Xiong et al. [5] studied developer behavior across GH and SO by mining accounts that were linked, using a CART decision tree approach. Their results found high levels of correlation between activity on the two platforms.

GH and SO have also been empirically studied for a variety of topics. For example, some research focused on analyzing topics that developers asked about in SO [18]–[23]. Similarly, an extensive research has analyzed programming languages that developers used in GH and their relationships to GH contributions [4], [8], [24]–[29]. There are also studies characterizing social network properties of GH and SO [22], [30]–[33].

Many researchers have studied expertise in GH and SO including (not limited to) Chen et al. [34] who recommend experts based on an analysis of correlations between search keywords in Google and SO. Silvestri et al. [35] conducts a comparative study across SO, GH and Twitter to investigate correlations between interactions. Yan et al [36] use heterogeneous information network analysis across various online communities to draw a comprehensive picture of a

[1]https://zenodo.org/record/2273117

[2]https://github.com/search?q=type:user&type=Users

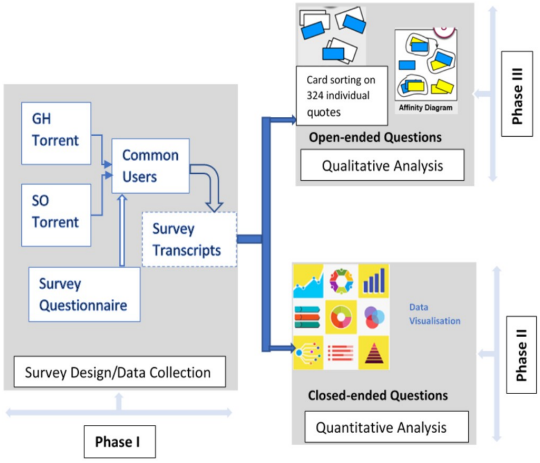

Fig. 1. Mixed methods research approach.

software developer and their expertise. Yu et al [37] using feature matching developed a cross-domain developer recommendation. Venkataramani et al. [38] built a recommendation system on SO based on data mined from GH. Constantinou et al. [39] measured developers' commit activity on GH and validated that by checking on their activity on SO. Huang et al. [40] provides skill scoring by correlating developer activity on GH and SO. Zhang et al. [41] proposed DevRec, a recommendation system on GH and SO, based on association matrix is proposed. However, the system does not consider user attributes and weighting mechanism.

III. Methodology

We conducted an exploratory qualitative study that involved data collection through a survey with SO and GH users. This section describes the survey design, the participants, and the survey analysis in detail.

A. Research Design

The research was designed in three phases as follows. During Phase 1 we conduct an online survey which was sent to a random sample of developers that are active on both GitHub and Stack Overflow. In Phase 2, we perform a quantitative analysis on the survey data, while in Phase 3, we applied an open coding approach. Our goal is to better understand how developers define, build and maintain their expertise via contributions to two different collaborative platforms. Figure 1 summarizes the overall mixed methods research approach.

B. Research Questions

Prior to conducting an open coding approach, we had no predefined notions nor specific research questions. Rather, our survey design was focused on developers' opinions about how they may define an expert in the field, and what behaviour and motivation drive their SO and GH contributions. Thus, our research questions originated after the four concept categories have emerged as a result of the open coding. Therefore, our four research questions are reflective of the concept categories (by choice) to be able to better discuss and discover these key categories. The research questions are as follows:

- RQ1: How do developers define an expert?
- RQ2: What factors drive developers to contribute to GH?
- RQ3: What factors drive developers to contribute to SO?
- RQ4: What challenges do developers face when contributing to each platform?

C. Survey Design (Phase 1)

The paper started with no preconceived notions about the possible results (i.e., open-minded), and therefore the survey mainly included open-ended questions for the purpose of data collection. Furthermore, since this paper aims at understanding expertise from a multi-specialty point of view, we did not define any specific selection criteria (i.e., programming language), instead we offered multiple choice to the respondents for providing their input.

The survey questionnaire was divided into four sections:

1) Background information — Ten closed-ended questions (main role, age, gender, education level, years of development experience, etc.) were included in this section. In particular, we were interested in asking the following questions:
   a) purpose of using GH,
   b) purpose of using SO.
2) Expertise levels — Five closed-ended questions were asked in this section (participants to rate on a 1 to 5 scale, with 5 being an expert):
   a) expertise in programming languages,
   b) expertise in frameworks and libraries,
   c) expertise in technologies,
   d) expertise in databases, and
   e) consideration(s) given by respondents when rating their expertise.
3) Expertise discussion — Two open-ended questions were posed to respondents:
   a) list 3 to 5 important skills of an expert software developer,
   b) asked if the respondents regularly monitored their skills, if so how?
4) Contribution to GH and SO — Three open-ended questions were posed to respondents in this section:
   a) how they decide to contribute on GH,
   b) how they decide to participate on SO,
   c) top 5 tags on SO they participated and whether these tags reflected their expertise.

D. Survey Participants

The survey questionnaire was sent to a sample of software developers. The survey sample for Phase 1 was randomly drawn from users active on both GH and SO for the last three years. An "active user" for the purpose of this study was defined as the person who made 10 or more commits in GH and similarly 3 or more comments on Stack Overflow over the last three years ("Active Users"). From the GHTorrent database a query was run to select the Active Users on GH.

This query resulted in 30,775 users qualifying as active users. Similarly, a query was run on SO to collate a set of active users from the publicly available dataset. This query on SO resulted in 676 qualifying as active users. Thereafter, the users' common on both platforms were matched, with their email addresses, following the approach of Vasilescu et al. [9]. The email addresses of SO users are only available from an old data dump released on September 10, 2013; thereafter, they stopped publishing the email addresses of the contributors. From these two sources, a sample of 423 active users that were common among both the lists was derived.

The university's Ethics Review Board approval for the entire experiment was obtained prior to sending out the survey to the random sample of 423 active users. An online survey platform (Qualtrics [42]) was used to conduct the survey and data collection.

The first round of survey questionnaire was sent in November 2019 and was available to the respondents for two weeks. A total of 423 emails were sent out inviting the participants; of these, 30 respondents opted out of this survey and 8 emails bounced back. Thus, leaving a final sample of 385 respondents. Of these, 73 respondents started working on the survey (a response rate of 18.9%). Finally, 41 of the 73 respondents completed and submitted the survey, i.e., a submission rate of 56%. In this study, we only consider the fully completed survey responses, even though many of the incomplete ones provided input to many of the questions.

The beginning of the survey consisted of background-related questions. We summarize the characteristics of our population of participants as follows:

1) Main role: The majority of the respondents have identified themselves as software developers (33 out of 41, i.e., 80% of the respondents). The other categories of respondents include: software architects (2), team lead (2), consultant (1), researchers (1) and others (2).
2) Age of respondents: Most of the respondents are in the age group of 30 to 40 (23, 56.1%), followed by 40+ (14, 34.1%) and the remaining are between the age of between 20 and 30 (4, 9.8%).
3) Gender: 39 of the 41 respondents are male and the remaining 2 have self-identified themselves as "other".
4) Geographic location: 38 of the 41 respondents have provided their geographic location; 44.7% (17) of the respondents who answered are from Europe (including UK), 36.8% (14) respondents are from North America, 13.2% (5) are from Australia and 5.3% (2) are from Asia.

E. Survey Data Analysis (Phase 2 and Phase 3)

We applied quantitative analysis on the multiple-choice responses of the survey (Phase 2), while an open coding approach was used to analyze the the open-ended survey responses (Phase 3). Since we had no predefined groups or categories, we used an open coding approach [43] to build the theory. As we analyzed the quotes, themes and categories emerged and evolved during the open coding process.

The first author created the "cards", splitting 41 survey responses into 324 individual quotes; these generally corresponded to individual cohesive statements. In further analysis, first and second authors acted as coders to group cards into themes, merging themes into categories. For each open-ended question, we proceeded with this analysis in three steps:

1) The two coders independently performed card sorting on the 20% of the cards extracted from the survey responses to identify initial card groups. The coders then met to compare and discuss their identified groups.
2) The two coders performed another independent round, sorting another 20% of the quotes into the groups that were agreed-upon in the previous step. We then calculated and report the inter-coder reliability to ensure the integrity of the card sort process. We selected two of the most popular reliability coefficients for nominal data: percent agreement and Cohen's Kappa. Coder reliability is a measure of agreement among multiple coders for how they apply codes to text data. To calculate agreement, we counted the number of cards for each emerged group for both coders and used ReCal2 [44] for calculations. The coders achieved a substantial degree of agreement; on average two coders agreed on the coding of the content in 97% of the time (the average percent agreement varies across the questions and is within the range of 86–100%; while the average Cohen's Kappa score is 0.6835).
3) The rest of the card sort for each open-ended question (i.e., 60% of the quotes) was performed by both coders together.

IV. Results

In this section, we first present the results of our open coding approach by presenting the overview of the concept categories that emerged (in Section IV-A) and then offer results and findings for each of our research question.

A. Overview of Concept Categories

In summary, we applied an open coding technique on the 324 individual statements. During this process, 46 main categories emerged. Each identified category consisted of between one and forty-four comments or quotes. Further, these categories were broadly grouped into four concept categories: 1) skills of an expert, 2) contribution to GH, 3) contribution to SO, and 4) challenges faced by developers in contributing to collaborative platforms. Table I presents these categories in detail reporting the number of quotes, the number of respondents, and the totals. Each of these concept categories consists of a minimum of five to a maximum twenty three categories.

The four concept categories that have emerged during the card sort do not necessarily directly correspond to the tasks performed by experts on the online collaborative platforms. Instead, these categories are a combination of actions (i.e., behavior) and mental activities (e.g., motivation) performed by experts contributing to the online collaborative platforms under consideration (i.e., GH and SO).

TABLE I
OVERVIEW OF CONCEPT CATEGORIES.

Concept Category	*Participants*	*Quotes*	*Categories*
Experts' skills	41	176	23
Contribution in GH	41	75	9
Contribution in SO	41	50	9
Challenges	20	23	5

1) *Defining features of an expert*. This concept category identifies the essential skills and competencies of an expert, as well as the characteristics and behavior that an expert might possess.
2) *Factors driving developers to contribute to GH*. This category offers insights into why developers decide to contribute to GitHub and what factors facilitate their activity on GitHub.
3) *Factors driving experts to contribute to SO*. This category highlights the details on the developer motivation behind their contribution to Stack Overflow.
4) *Challenges faced by experts in contributing to online collaborative platforms*. This category addresses the challenges developers face when contributing to each platform. Such challenges can be addressed by improving current platforms to increase developer motivation and engagement on these platforms. While all 41 participants answered a "yes/no" question related to challenges, only 20 of them provided a rationale for their answer (as reflected in Table I for "Challenges" category).

B. RQ1: How do developers define an expert?

Since we used both quantitative and qualitative analysis methods on the survey data, we next present the results for each of the method, respectively.

1) Quantitative Analysis: The software development experts were asked four closed-ended questions that asked them to rank their expertise in various areas of software development (programming language, frameworks & libraries, technologies and databases). The respondents were given a list of options in each of these modules and were asked to self-select their expertise level on a scale of 1 to 5 (with 1 being "no knowledge" and 5 being an "expert"). Also, there was one closed-ended question that asked the considerations given by the respondents in rating their expertise in the areas of software development. The respondents were given five options and were also given a freedom to respond outside of these choices. These questions aimed at uncovering whether the respondents would consider their professional experience, their depth of knowledge, the size of the projects for measuring their performance against other developers. The respondents could choose more than one option on this question.

The first question probed on expertise in programming languages. Out of all the programming languages, JavaScript turned out to be most popular among respondents and more than 70% have self-identified themselves as expert (level 5) or proficient (level 4). This was followed by SQL (50% combined for both level 5 and level 4), C/C++ (49%), and then Python (44%). It was somewhat surprising to see C/C++ to be towards the top of this table. It could be also possible that this was a popular language a few years ago and given that about 90% of our respondents are above 30 years, it is likely that these respondents might have previously acquired expertise in these languages [45].

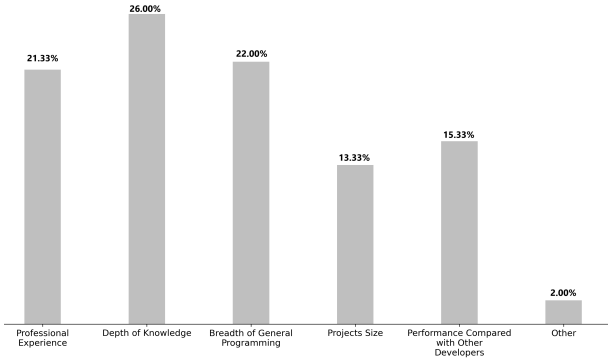

Fig. 2. Expertise determinants.

The second question pertained to expertise in frameworks and libraries. Surprisingly, the response rate was low and also, they have mainly marked as beginner (level 2) or intermediate (level 3). When the responses for level 4 (proficient) and level 5 (expert) were combined, we found that about 29% of the respondents have stated they are experts in React (Facebook) and NodeJS.

The third question was about expertise in technologies. The response was even lower here. Only 2 respondents noted that they were experts in blockchain. Thereafter, about 6 respondents have noted that they are proficient in IoT. Moreover, 2 of the 41 respondents claimed that they were proficient in deep learning, 4 claimed proficiency in machine learning and 3 in computer vision. This is indeed a surprising result, especially given that ML, deep learning and computer vision have been recently gaining a momentum by becoming the hot trend in expertise demand.

The fourth question was on database expertise. It was observed that about 29% of our respondents have claimed expertise in PostgreSQL, followed by 22% claiming expertise in MySQL and SQLite, each.

The last question asked the respondents to answer the factors they considered to be important to them when they were answering the above questions about expertise. Depth of knowledge (as supported by 26% of the responses) is the top factor that developers believe to influence their answers to the above four questions on expertise. The other factors influencing their opinions are: breadth of general programming (22%), professional experience (21%), performance compared to others (15%) and project size (14%). Figure 2 presents the overall summary of these results.

2) Qualitative Analysis: We now probe the developers themselves to find the necessary and sufficient conditions that define an "expert". More specifically, the aim is to better understand if knowledge and experience alone are sufficient or if a requisite amount of soft skills are needed for a software developer to become an expert. Moreover, the order

of importance of these three factors was further investigated to understand if there is any normative difference between the technical and social skills.

Technical skills (number of total participants (P): 21; number of total quotes (Q): 44 as shown in Table II), broad knowledge (P:7; Q:8) and domain knowledge (P:2; Q:2) are perceived to be the features that characterize *expertise*. On the other hand, *experience* could be captured by features such as planning & organization (P:9; Q:11), experience (P:2; Q:4), and vision (P:1; Q:1). Lastly, communication skills (P:15; Q:15), analytical thinking (P:24; Q:26), behavior which includes tenacity and perseverance (P:9; Q:17) are some of the features that characterize *soft skills* of an expert. Effective and clear communication is a key skill for an expert software developer and was emphasized by the respondents. One respondent added the need for clarity and "clear communication" (P3Q21P4, where P3Q21P4 is the "code" we assign to each individual statement meaning Participant 3, Question 21, Phrase 4), while another noted the need for "soft communication skills" (P31Q21P4). The other key consideration is analytical thinking, which is a soft skill that could be honed with experience. Analytical thinking not only requires an expert to "think beyond technology" (P11Q21P3), but also requires that they think abstractly about complex problems (P38Q21P2) and also have an ability of "...breaking down complex problems into simpler problems" (P40Q21P1). Behavior or attitude has been identified as another critical factor in identifying an expert. Some of the features identified are, being stubborn (P5Q21P3), patience (P13Q21P3, P27Q21P3, P38Q21P4), tenacity (P13Q21P4, P27Q21P4), curiosity (P27Q21P3) and "being empathetic about the needs of other people" (P41Q21P1). Furthermore, based on the detailed responses, Table II reports the features, emerged categories and their response rate (i.e., number of participants and total number of quotes) for each of these three factors.

Moreover, the survey asked respondents to list the important factors in an order of importance. It is noted that even in that ordering mostly the soft skills appeared quite prominently in the beginning of a response, i.e., they were more important features of an expert, as per the respondents. Furthermore, from Table II, it is evident that the most important skills that a software development expert ought to possess are perceived to be soft skills, skills that are both necessary and sufficient conditions to become an expert.

Within this same broad area, the respondents were asked about the ways they *monitor their own activities* on a regular basis (using software metrics or other statistics) and the specific mechanism that they use to measure their software development activities. The majority of the respondents (61%) have noted that they do not monitor their activities and only 34% have responded affirmatively on this question. Three main features that emerged from this category are self tracking (P:3; Q:3), testing (P:7; Q:6), online collaborative tools (P:4; Q:5), time tracking tools (P:3; Q:3), and dashboards (P:5; Q:7).

In summary, the results for this research question indicate that:

TABLE II
FEATURES OF SOFTWARE DEVELOPER EXPERTISE.

Expertise	*Participants*	*Quotes*
Technical	21	44
Domain knowledge	2	2
Broad knowledge	7	8
Total	30	54
Experience		
Experience	2	4
Critical thinking	3	3
Vision	1	1
Requirements	6	7
Teamwork	3	3
Teaching skills	2	2
Total	17	20
Soft Skills		
Planning & organization	9	11
Analytical thinking	24	26
Creativity	3	3
Understanding	4	4
Behavior	9	17
Communication skills	15	15
Searching & seeking help	2	2
Total	66	78
Monitoring Skills		
Self tracking	3	3
Dashboards	5	7
Testing tools	5	6
Online tools	4	5
Time tracking tools	3	3
Total	20	24

- JavaScript is perceived to be the leading programming language within a software development expertise, in line with the findings of Georgiou et al. [46];
- Surprisingly, the expertise diversification is lower than what one would expect;
- Knowledge (48%, calculated as the sum of depth of knowledge (26%) and breadth of general programming (22%)) and experience (50%) are considered to be the key "expert" characteristics. This finding is in line with the high-level concept (relationships) of the expertise theory by Baltes and Diehl [2].

Answer to RQ1: Soft skills together with adequate knowledge and experience are the key characteristics that make a software developer an expert. In other words, soft skills are as important as knowledge and experience for software developers to have and develop to be considered an expert in the field.

C. RQ2: What factors drive developers to contribute to GH?

GitHub allows developers to contribute to projects in multiple ways: submitting a bug report, a pull request, engaging in the discussion on project's features, etc. This research question aims to understand how developers participate in projects (private vs. public contributions), as well as what motivates developers to contribute to GitHub. In other words, this question seeks to provide insights into the possible drivers behind developers' contributions to GitHub.

To start with, our closed-ended question enquired how our respondents used GH, i.e., either contributing to public or private projects. The majority of respondents (28 out of 41

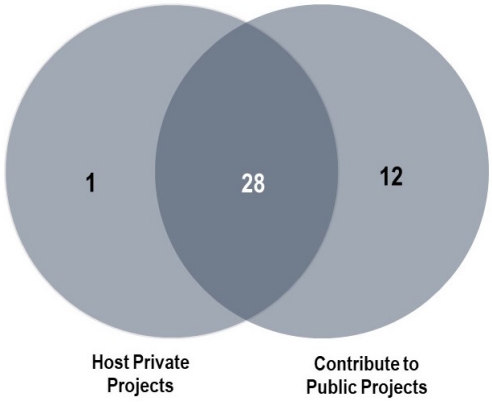

Fig. 3. Contribution visibility on GitHub.

or 68% of the participants) noted that they use it for both private and public projects; while 29.7% contributed only to public projects, and only 2.4% used GitHub for hosting private projects (as shown in Figure 3).

Upon establishing how developers use GH, we identify the drivers that motivate developers to contribute on GitHub (Table III). We classified the drivers into: (1) personal drivers and (2) professional drivers. The *personal drivers* included contributing for hobby/fun, catering to self needs, and providing inputs due to altruistic tendencies of developers. On the other hand, the *professional drivers* were more related to job-related contributions to GitHub, including organizational culture, job, expertise and contributions to open source community.

We observe that experts contribute to GitHub more due to personal rather than professional motivation. Among the personal drivers, self needs is determined to be the key driver behind developers' contributions to GitHub. And, most of the participants mention that they started contributing because they were trying to solve a problem or a challenge they were going to resolve; some responses on contribution included, "if there's a bug I ran into" (P1Q25P2), "contribute bug fixes to software I use" (P3Q25P2), "I need to fix something broken that's blocking me" (P10Q25P2) and "I scratch my own itches" (P22Q25P1). Self needs, however, is not a prime driver for contribution to Stack Overflow, because of fundamental difference between the two collaborative platforms, GitHub allows for active participation, while Stack Overflow is more passive contribution based on specific questions and answers.

The other important personal drivers that motivate developers to contribute to GitHub, include, hobby/fun and helping. Some of the quotes of the former include, "...some of it is hobby though" (P4Q25P2) and "personal interest, if I spend my time for free, I want to enjoy it" (P11Q25P1). And, on helping others, the responses were as follows: "I contribute to new projects I think would be useful to others" (P6Q25P2), "I contribute because I care about what's being made" (P10Q25P1), while another developer noted that "they look like they might be useful" (P18Q25P1).

On the other hand, among the professional drivers we found that "expertise" is the most critical driver. One developer mentions that "mostly contribute to libraries used in work-related projects" (P36Q25P2) as a response about their nature of contribution in GitHub. Others contribute because it is their job (P19Q25P1, P4Q25P1) or to rectify existing bugs (P34Q25P3, P32Q25P1, P10Q25P3) or it is compatible with their skills (P7Q25P1). The other professional drivers include organizational culture (P7Q25P4, P7Q25P3) and experts working on B2B solutions; for example one respondent noted, "I work on B2B solutions or dev oriented solutions, so the core of these solutions can be open-sourced so other devs can contribute to it." (P2Q25P2).

TABLE III
CONTRIBUTION TO GITHUB.

Personal Drivers	*Participants*	*Quotes*
Hobby/fun	7	8
Self needs	27	34
Helping	5	5
Total	39	47
Professional Drivers		
Organizational culture	1	2
Job	1	1
Expertise	10	10
Open source community	1	1
Total	13	14
Challenges		
Issue complexity	6	6
Lack of time	8	8
Total	14	14

Answer to RQ2: Majority of experts participate in both hosting private projects, as well as contributing to public projects. Our findings show that developers believe that personal drivers are more critical than professional factors in motivating them to contribute to GitHub. Among the personal drivers, developers are mainly driven by self needs when contributing to GitHub.

D. RQ3: What factors drive developers to contribute to SO?

Stack Overflow, an online Q&A community for developers, allows software developers to freely ask and answer questions on any aspect of coding and hence, share their knowledge and advance their careers. Moreover, Stack Overflow also helps programmers and software developers by providing a platform to discuss technical problems with others within the software development community. The search functionality also enables others with similar coding problems to benefit from the posted solutions. Participants within this community usually participate for three specific reasons, i.e., participation trends: (i) view discussions to get help, (ii) post questions on a particular problem to seek help and (iii) offer answers and participate in their discussions. This research question explores developers' participation trends in Stack Overflow and then investigates the factors that motivate developers to contribute to Stack Overflow. In summary, this research question aims at understanding the possible drivers behind developers' contributions in Stack Overflow.

We start with a closed-ended question to understand the participation trends within Stack Overflow. The majority of respondents (37 out of 41 or 90%) noted that they use SO for

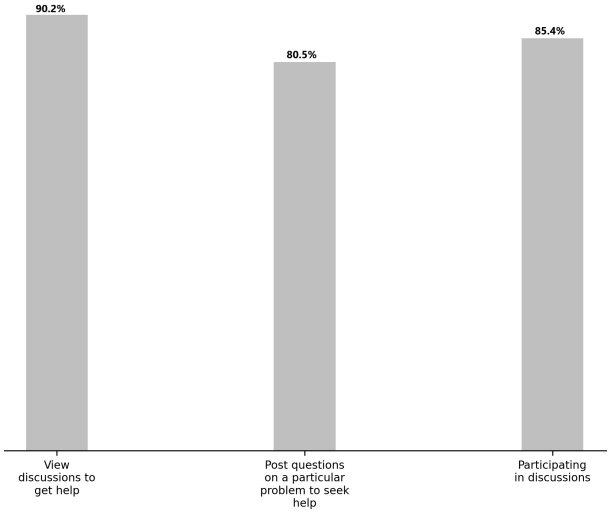

Fig. 4. Purpose of using Stack Overflow.

viewing discussions to get help; while 80% (33 out of the 41) of the respondents said that they post questions on a particular problem to seek help; and, 85% (35 out of 41) of respondents noted that they also participate in answer discussions. Figure 4 presents these results.

We found that unlike for GitHub results (RQ2), skills & expertise are the key drivers behind contribution in Stack Overflow (Table IV). The respondents mention that they contribute only in areas that they are confident in and which they believe they are qualified. More specifically, *professional factors* are the primary drivers behind contribution in Stack Overflow, while *the personal drivers* seem to take the back seat. Moreover, an "expert" bias can be observed, especially when it comes to contributing via posting questions as one respondent mentions that "...the SO answerer community can be quite cruel to askers; this has made me less inclined to participate recently" (P25Q25P4). And another respondent noted "I participate on discussion I can contribute. I avoid discussion in which I can't" (P37Q26P1). This is in stark contrast with participation in GitHub, and this may be because a certain level of expertise is expected by the SO community while answering or asking questions, whereas GitHub community is more open contributions from a greater community, i.e., any developer is free to contribute and the project owners can either accept or decline a contribution (e.g., a pull request).

Notwithstanding the "expert bias" in Stack Overflow, it is ironic to note that some of the experts who responded report that they do not participate in Stack Overflow anymore because of lack of interesting questions (P10Q26P1) and that the reward system is skewed towards easier questions, as noted by one expert, "Questions that I can answer, particularly if technically challenging. Unfortunately SO rewards contributing to easy topics over hard." (P34Q26P1).

There is however, one interesting case for using Stack Overflow for the purpose of documenting expertise (or knowledge base) of a software developer, as one response highlights: "...due to time constraints: I recently use Stack Overflow

TABLE IV
CONTRIBUTION TO STACK OVERFLOW.

Personal Drivers	*Participants*	*Quotes*
Helping	3	3
Hobby/fun	5	5
Self-learning	3	3
Own needs	6	6
Total	17	17
Professional Drivers		
Skills & expertise	19	20
Better answer	3	3
Alternative solutions	1	1
Total	23	24
Challenges		
Rewarding system	1	1
Tagging system	7	8
Total	8	9

primarily as a way of documenting my hard-won findings on a topic (e.g., posting and answering a question, then linking to it in the code), but am happy to get other answers and opinions added" (P16Q26P1). Another respondent discusses this further saying that it "...needs to be a niche area where there isn't much expert competition. I can't compete with ppl writing a comprehensive answer..." (P20Q26P1).

Some of the similarities between the factors driving contributions across both platforms are related to participants being motivated by "doing it for fun" (for GH, P:7; Q:8; while in SO, P:5; Q:5) and out of personal interest rather than as a chore; some responses include "personal interest. If I spend time for free, I want to enjoy it" (P11Q25P1), "has to be interesting" (P37Q25P1), and personal interest (P39Q25P2, P21Q26P2, P9Q26P1).

Answer to RQ3: Contribution to SO is motivated more by professional drivers (i.e., skills and expertise). There seems to be a bias towards experts, yet developers do not seem to be keen to participate on Stack Overflow as they believe the questions are either no longer interesting or they are too easy.

E. RQ4: What challenges do developers face when contributing to each platform?

This research question identifies key challenges developers face when contributing to Stack Overflow or GitHub. In a way, this question provides insights into why developers do not contribute to the platform or contribute in a limited capacity. We identified two categories of challenges for GitHub: one related to *issue complexity* and one due to *lack of time*. While for Stack Overflow, challenges are related to its *tagging system* and *rewarding system*.

For our open-ended question on why developers do not contribute to GitHub, majority of the respondents mention lack of time as the reason, including " I do not contribute more to other people's projects due to time constraints" (P16Q25P3), " I rarely contribute because I don't have enough spare time" (P23Q25P1). It seems that developers are already contributing to their open source projects in GH and simply have no time to contribute to other projects as explained by P13Q25P2: "I

have my own projects to work on on the side, so I don't really have the time or interest to contribute to other open source projects". Keeping GitHub projects small enough so others can understand and build them, making issues small and interesting are our key suggestions for GitHub project owners for encouraging better contributions and engaging more contributors on the projects. Another reason of why developers may decide to contribute or not to the project is the project's or issue's complexity. One developer (P8Q25P1) mentions that "it mostly depends on the complexity of the project", while another respondent says that "when I try to contribute, I generally choose small intriguing issues" (P23Q25P2). We also observed that developers' contributions are driven based on whether they can build the project locally on their machines as supported by the following quote "whether I can build the project locally on my Macbook" (P26Q25P3).

Based on the responses we found that developers prefer to contribute to GH rather than participate on SO. Some of the issues cited were related to the difficulty finding interesting questions to answer: "I mostly don't participate in Stack Overflow any more. It's hard to find interesting questions to answer, and my time to create answers has gone down, too" (P10Q26P1). Another respondent (P34Q26P1) pointed out that SO is rewarding answers to easy topics rather than more challenging ones: "Unfortunately, SO rewards contributing to easy topics over hard". While some developers felt that SO platform is less relevant to more specialized topics and thus they do not feel it's rewarding to contribute to the discussions, "I do a lot more than I write on SO. SO is a waste of time outside specialized niches" (P33Q29P1).

Furthermore, about 50% of the participants mentioned that their poor contribution to SO is related to the current tagging system. Since tags are automatically determined and assigned by the gamification mechanisms within SO, participants found that their "collected" tags (tags assigned to a developer's profile) may no longer reflect their current expertise or interests. This response was consistent and reflects the fact that individual interests and expertise are dynamic in nature and they do evolve over time. As one respondent notes that "my SO profile [is my] expertise during PhD" (P34Q29P2) and thus does not reflect her current skills.

Answer to RQ4: In GitHub, developers mainly face technical challenges such as project/issue complexity, as well as personal challenges such as lack of time. While on Stack Overflow, developers' lack of contribution and participation is perceived to be related to its tagging system (mismatch of SO tags with the developers' expertise) and outdated/demotivating rewarding system.

V. Discussion

We now discuss several research directions that have emerged from this work, as well as implications that can help researchers to plan their next research projects and developers to make use of our findings.

Theory of software developer expertise. Baltes and Diehl [2] have recently proposed a conceptual theory of software development expertise. While their theory is grounded in the online survey responses of 355 software developers, the targeted population was focused primarily on Java experts. Our study further investigates the concepts of "expert" and "expertise" within the field of software development and discovers additional findings on why developers may contribute to some social platform but not the other. While our study results are grounded in the survey responses of fewer developers, we targeted a different population of developers who are active on both platforms in order to better understand how they build and maintain their expertise across multiple platforms. Our next steps would be to extend the theory developed by Baltes and Diehl by considering broader scope of "expertise" definition, in particular we are interested in building a theory of cross-platform expertise that focuses on the specific properties and characteristics of the experts who are contributing to multiple platforms. We are interested in investigating what skills and behaviour are supportive of developers' success in becoming better experts.

Knowledge transfer across multiple platforms. Our findings demonstrate that developers do share their knowledge on multiple platforms. One of the interesting direction would be to conduct studies to further understand what technical knowledge and skills are more transferable across different platforms. Such insights can help developers better plan how they contribute to different platforms. If some platforms are more favourable for specific knowledge transfer, developers can, for example, increase their participation on that platform.

Implications for developers. The insights from this work can help developers to better understand the platforms and increase their awareness on some of the critical aspects that may discourage their expertise gaining (e.g., Stack Overflow's tagging system and reward mechanisms) or, on the contrary, further support their learning and expertise development (e.g., contributing to open source community in GH or sharing knowledge on SO). Furthermore, developers often monitor their activities and progress. However, modern tools are too complex and create an informational overload on software developers. Thus, in order to reduce burden of information overload it is critical to offer personalized dashboards [47], [48], [49], as some participants have mentioned, that improves developer situational awareness [50], [51] and support self-monitoring activities.

Suggestions for Stack Overflow decision makers. The results of our study highlight the challenges developers face when participating on Stack Overflow. More specifically, we found that developers feel that current SO tagging system needs major upgrade as tags assigned to the developers' profiles do not accurately reflect their current level of expertise. We recommend Stack Overflow as a platform to add a tag updating mechanism so developers can decide which tags are still relevant to their expertise. Additionally, in the interest of nurturing future talent Stack Overflow needs to be more welcoming to new and novice contributors having their moderators be aware

of and better manage subtle and overt "bullying" that these developers might be facing.

Suggestions for GitHub decision makers. Our results also highlight the challenges developers face when participating on GitHub. We observed that developers face project and/or issue complexity on GitHub. Moreover, some developers lack adequate computing resources when accessing and working on GitHub projects. Our recommendation to GitHub would be to consider integrating cloud computing resources to alleviate this challenge.

Implications for researchers. Some of the findings of our work can form the basis for further comprehensive multi-layered research by expanding the number of platforms and social media sources to draw a complete picture of on "expert" and "expertise". One of the observations from our study is that each developer brings his/her own opinion on the definition of the expertise. And therefore, the targeted population of the developers who participate in such expertise-related surveys should be diverse and inclusive. In addition to conducting surveys on the topic of expertise, we believe more robust qualitative research methods such as interviews with developers would be beneficial for collecting in-depth information about developers' opinions, experiences and behaviour.

VI. Threats and Limitations

Our work is subject to several threats and limitations. One of the limitation is related to our findings from the qualitative study. While we carefully designed our survey questions to ensure their clarity, as with any exploratory study, there is a chance we may have introduced the researcher bias when applying an open coding approach. We tried to minimize this bias by coding the 20% of the card sorts extracted from each question independently, measuring the coder reliability on the next 20% and reporting these values in the paper.

As with any survey method, to control for sampling bias can be challenging. We targeted users of Stack Overflow and GitHub who actively participate on both platforms by contributing to Stack Overflow discussions and being active developers and contributors on various GitHub projects. While our findings might not generalize outside of the selected population of users, we believe that developers who contribute to both platforms may share and experience similar behaviour and patterns.

We realize that our definition of "active users" is somewhat ambiguous. While we obtained developer emails from the 2013 GHTorrent dataset (due to the GDPR being adopted in 2016, email addresses are no longer published), our "active users" are determined to be active on these platforms based on their activity and contributions as of December 2019. However, our data certainly lacks active users who have joined SO and GH after 2013.

We are also aware that our randomly selected sample of active users on SO and GH lacks gender diversity. While challenging, inclusion and equity are becoming more critical parameters when recruiting participants for qualitative studies like this one. The dataset does not provide gender information on users (which can be classified as personal information and thus violate privacy protection laws such as GDPR), while we could have inferred the gender by applying gender detection techniques [52], [53] in order to strive for a gender balanced sample. Our next steps would be to include such techniques and approaches for targeting a more diverse population of participants.

However, further research studies are needed to be able to provide greater insight into developer contributions to various online social and collaborative platforms, beyond Stack Overflow and GitHub, to be able to develop an empirical body of knowledge on this topic. To encourage replication of our study, we documented our survey questions and open coding results in a technical report which together with anonymized survey responses is made available online [15].

VII. Conclusion

In this paper, we presented a qualitative study of software developer expertise with an attempt to understand the reasons why or why not developers contribute to GitHub and Stack Overflow. Also, we tried to understand the ways in which experts monitor their performance and the challenges they face when contributing to either collaborative platform. To understand these aspects better, we designed a survey with both open-ended and closed-ended questions. The survey was sent to a sample of experts that were active on both Stack Overflow and GitHub. We conducted an open coding to the 324 individual quotes obtained from the open-ended questions. As a result of the open coding, 46 main categories together with four concept categories have emerged. Through our research questions, we were able to gain insights into developers' opinions and perception on key characteristics of an expert, factors that drive developers to contribute to GitHub and Stack Overflow, as well as the challenges developers face in the process.

Our findings suggest that JavaScript and C/C++ are the leading programming language skills and surprisingly the extent of skill diversification among experts was lower than expected. Moreover, a software development expert would have both technical and soft skills. Furthermore, users who contribute to both platforms seem to favour GitHub more often as it's less biased towards experts. Also, developers use GitHub for personal reasons, while Stack Overflow is more catered to professional purposes. Lastly, we found that developers on GitHub face technical challenges such as project complexity, as well as personal challenges such as lack of time. On the other hand, developers on Stack Overflow faced a more negative reinforcements demotivating their contributions.

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). We thank all developers who participated in our study for their time, input, and opinion.

REFERENCES

[1] K. A. Ericsson, R. T. Krampe, and C. Tesch-Römer, "The role of deliberate practice in the acquisition of expert performance." *Psychological Review*, vol. 100, pp. 363–406, 1993.

[2] S. Baltes and S. Diehl, "Towards a theory of software development expertise," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 2018, pp. 187–200.

[3] R. Robbes and D. Röthlisberger, "Using developer interaction data to compare expertise metrics," in *Proc. of Working Conf. on Mining Software Repositories*, 2013, pp. 297–300.

[4] G. Gousios, M.-A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: The contributor's perspective," in *Proc. of Int. Conf. on Soft. Engineering*, 2016, pp. 285–296.

[5] Y. Xiong, Z. Meng, B. Shen, and W. Yin, "Mining developer behavior across github and stackoverflow," in *Conference: The 29th International Conference on Software Engineering and Knowledge Engineering.*, 2017.

[6] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proceedings of the 36th International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2014, p. 356–366.

[7] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Conference: CSCW '12 Computer Supported Cooperative Work, Seattle, WA, USA, February 11-15, 2012*. ACM, 2012.

[8] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.

[9] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stack Overflow and GitHub: Associations between software development and crowdsourced knowledge," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 188–195.

[10] J. Liao, G. Yang, D. Kavaler, V. Filkov, and P. Devanbu, "Status, identity, and language: A study of issue discussions in github," *PloS one*, vol. 14, no. 6, p. e0215059, 2019.

[11] A. Begel and T. Zimmermann, "Analyze this! 145 questions for data scientists in software engineering," in *Proceedings of the 36th International Conference on Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2014, p. 12–23.

[12] O. Kononenko, O. Baysal, and M. W. Godfrey, "Code review quality: How developers see it," in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 2016, pp. 1028–1038.

[13] D. Posnett, E. Warburg, P. T. Devanbu, and V. Filkov, "Mining stack exchange: Expertise is evident from earliest interactions," in *Social Informatics*, 2012.

[14] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, and C. Faloutsos, "Analysis of the reputation system and user contributions on a question answering website: Stackoverflow," in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. New York, NY, USA: Association for Computing Machinery, 2013, p. 886–893.

[15] S. Vadlamani and O. Baysal, "Software developer expertise: replication package," https://github.com/Sri-Vadlamani/Software-Developer-Expertise, 2020.

[16] G. Bergernsen, D. Sjoberg, and T. Dyba, "Construction and validation of an instrument for measuring programming skill," *IEEE Transactions on Software Engineering*, vol. 40, no. 12, pp. 1163–1184, 2014.

[17] J. E. Montandon, L. L. Silva, and M. T. Valente, "Identifying experts in software libraries and frameworks among github users," in *Proceedings of the 16th International Conference on Mining Software Repositories*. IEEE Press, 2019, pp. 276–287.

[18] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Softw. Engg.*, vol. 19, no. 3, pp. 619–654, Jun. 2014.

[19] X. Yang, D. Lo, X. Xia, Z. Wan, and J. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, pp. 910–924, 2016.

[20] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Softw. Engg.*, vol. 21, no. 3, pp. 1192–1223, Jun. 2016.

[21] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, and X. Zhang, "Which non-functional requirements do developers focus on? an empirical study on stack overflow using topic analysis," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 446–449.

[22] S. Wang, D. Lo, and L. Jiang, "An empirical study on developer interactions in stackoverflow," in *Proc. of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1019–1024.

[23] H. Alharthi, D. Outioua, and O. Baysal, "Predicting Questions' Scores on Stack Overflow," in *Int. Workshop on Crowd Sourcing in Software Engineering*, 2016, pp. 1–7.

[24] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Proc. of Conf. on Computer Supported Cooperative Work*, 2012, pp. 1277–1286.

[25] M. M. Rahman and C. K. Roy, "An insight into the pull requests of github," in *Proc. of Working Conference on Mining Software Repositories*, 2014, pp. 364–367.

[26] B. Ray, D. Posnett, P. Devanbu, and V. Filkov, "A large-scale study of programming languages and code quality in github," *Commun. ACM*, vol. 60, no. 10, pp. 91–100, Sep. 2017.

[27] J. Jiang, D. Lo, J. He, X. Xia, P. S. Kochhar, and L. Zhang, "Why and how developers fork what from whom in github," *Empirical Softw. Engg.*, vol. 22, no. 1, pp. 547–578, Feb. 2017.

[28] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, and J. Ell, "Understanding "watchers" on github," in *Proc. of Working Conference on Mining Software Repositories*, 2014, pp. 336–339.

[29] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proc. of Int. Conf. on Software Engineering*, 2014, pp. 345–355.

[30] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang, "Network structure of social coding in github," in *European Conf. on Software Maintenance and Reengineering*, 2013, pp. 323–326.

[31] B. Vasilescu, A. Serebrenik, and V. Filkov, "A Data Set for Social Diversity Studies of GitHub Teams," in *Proc. of Working Conference on Mining Software Repositories*, 2015, pp. 514–517.

[32] B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge," in *2013 Int. Conference on Social Computing*, 2013, pp. 188–195.

[33] A. Begel, J. Bosch, and M.-A. Storey, "Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder," *IEEE Softw.*, vol. 30, no. 1, pp. 52–66, Jan. 2013.

[34] C. Chen and Z. Xing, "Towards correlating search on google and asking on stack overflow," *IEEE 40th Annual Computer Software and Applications Conference*, 2016.

[35] G. Silvestri, J. Yang, A. Bozzon, and A. Tagarelli, "Linking accounts across social networks: the case of stackoverflow, github and twitter." in *KDWeb*, 2015, pp. 41–52.

[36] J. Yan, H. Sun, X. Wang, X. Liu, and X. Song, "Profiling developer expertise across software communities with heterogeneous information network analysis," in *Internetware '18*, 2018.

[37] X. Yu, Y. He, Y. Fu, Y. Xin, J. Du, and W. Ni, "Cross-domain developer recommendation algorithm based on feature matching," in *ChineseCSCW*, 2019.

[38] R. Venkataramani, A. Gupta, A. Asadullah, B. Muddu, and V. Bhat, "Discovery of technical expertise from open source code repositories," in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 97–98.

[39] E. Constantinou and G. M. Kapitsaki, "Identifying developers' expertise in social coding platforms," in *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2016, pp. 63–67.

[40] W. Huang, W. Mo, B. Shen, Y. Yang, and N. Li, "Cpdscorer: Modeling and evaluating developer programming ability across software communities." in *SEKE*, 2016, pp. 87–92.

[41] X. Zhang, T. Wang, G. Yin, C. Yang, Y. Yu, and H. Wang, "Devrec: a developer recommendation system for open source repositories," in *International Conference on Software Reuse*. Springer, 2017, pp. 3–11.

[42] Qualtrics. Qualtrics software. [Online]. Available: https://www.qualtrics.com

[43] M. Miles and A. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook*. SAGE Publications, 1994.

[44] "Recal2: Reliability for 2 coders," December 2019. [Online]. Available: http://dfreelon.org/utils/recalfront/recal2/

[45] P. Sumanth and K. Rajeshwari, "Discovering top experts for trending domains on stack overflow," in *Discovering Top Experts for Trending Domains on Stack Overflow*, vol. 143. Elsevier BV, 2018, pp. 333–340.
[46] K. Georgiou, M. Papoutsoglou, A. Vakali, and L. Angelis, "Software technologies skills: A graph-based study to capture their associations and dynamics," in *BCI'19*, 2019.
[47] M. Kersten and G. C. Murphy, "Using task context to improve programmer productivity," in *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2006, p. 1–11.
[48] J. T. Biehl, M. Czerwinski, G. Smith, and G. G. Robertson, "Fastdash: A visual dashboard for fostering awareness in software teams," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2007, p. 1313–1322.
[49] O. Kononenko, O. Baysal, R. Holmes, and M. Godfrey, "Dashboards: Enhancing developer situational awareness," in *Proceedings of the ICSE 2014, May 31 - June 7, 2014, Hyderabad, India*. ACM, 2014.
[50] C. Treude and M.-A. Storey, "Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, 2010, p. 365–374.
[51] O. Baysal, R. Holmes, and M. Godfrey, "Situational awareness personalizing issue tracking systems," in *Proceedings of the 2013 International Conference on Software Engineering*, 2013, p. 1185–1188.
[52] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study of stackoverflow," in *2012 International Conference on Social Informatics*, 2012, pp. 332–338.
[53] B. Lin and A. Serebrenik, "Recognizing gender of stack overflow users," in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, 2016, pp. 425–429.