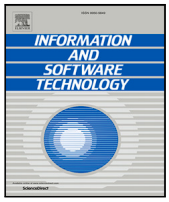
Information and Software Technology 138 (2021) 106633

Contents lists available at ScienceDirect

Information and Software Technology

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



Visual Resume: Exploring developers' online contributions for hiring

Sandeep Kaur Kuttal [a,*], Xiaofan Chen [b], Zhendong Wang [c], Sogol Balali [d], Anita Sarma [d]

[a] University of Tulsa, OK, USA
[b] Massey University, New Zealand
[c] University of California, Irvine, USA
[d] Oregon State University, OR, USA

ARTICLE INFO

Keywords:
Aggregators
Contribution profile
Online communities

ABSTRACT

Context: Recruiters and practitioners are increasingly relying on online activities of developers to find a suitable candidate. Past empirical studies have identified technical and soft skills that managers use in online peer production sites when making hiring decisions. However, finding candidates with relevant skills is a labor-intensive task for managers, due to the sheer amount of information online peer production sites contain.
Objective: We designed a profile aggregation tool—Visual Resume—that aggregates contribution information across two types of peer production sites: a code hosting site (GitHub) and a technical Q&A forum (Stack Overflow). Visual Resume displays summaries of developers' contributions and allows easy access to their contribution details. It also facilitates pairwise comparisons of candidates through a card-based design. We present the motivation for such a design and design guidelines for creating such recruitment tool.
Methods: We performed a scenario-based evaluation to identify how participants use developers' online contributions in peer production sites as well as how they used Visual Resume when making hiring decisions.
Results: Our analysis helped in identifying the technical and soft skill cues that were most useful to our participants when making hiring decisions in online production sites. We also identified the information features that participants used and the ways the participants accessed that information to select a candidate.
Conclusions: Our results suggest that Visual Resume helps in participants evaluate cues for technical and soft skills more efficiently as it presents an aggregated view of candidate's contributions, allows drill down to details about contributions, and allows easy comparison of candidates via movable cards that could be arranged to match participants' needs.

1. Introduction

"*When it comes to hiring, I'll take a GitHub commit log over a resume any day*" [1] tweeted John Resig, the creator of jQuery. Assessing online contributions has become increasingly popular with the growing popularity of peer-production sites such as GitHub and Stack Overflow. Potential employers, as well as recruiters, are increasingly mining the history of public contributions to locate suitable candidates, filter through applicants for a position, or inform interview interactions [2]. For example, Barney Pell, the founder of Powerset said, "*online open-source communities like GitHub bring large numbers of developers together and are thus a natural place for recruiting*" [3]. Several research studies have also confirmed the use of GitHub to recruit developers [2,4–9].

A key reason behind the popularity of these peer production sites as hiring aids is the level of transparency afforded by them. For example, sites like GitHub, allows access to in-depth details about developers' activities: lines of code committed, issues resolved, code reviews performed, and interactions and discussions around the code that the developer participated in. Research has shown that managers and developers use such information to form impressions of new employees or their colleagues when evaluating them [2,4] or their code [10]. For instance, the contribution history allows reconstructing what is the developer working on, what their code looks like, their frequency and speed of work, and how they work and interact [5]. Non-technical skills such as developer's motivation/passion can also be inferred based on the projects they own, have contributed to, or the diversity of projects or languages in which they are involved [2,4]. Similarly, collaboration skills can be assessed through discussions or code review comments, as these reveal how developers talk about their work or negotiate changes to their project.

However, assessing the online contributions of potential job candidates is not easy. Such evaluations require hirers to process a massive amount of information that is often fragmented across multiple projects or even different types of archives (e.g., code hosting sites vs. technical

* Corresponding author.
E-mail address: sandeep-kuttal@utulsa.edu (S.K. Kuttal).

https://doi.org/10.1016/j.infsof.2021.106633
Received 23 June 2020; Received in revised form 13 April 2021; Accepted 11 May 2021
Available online 27 May 2021

question and answer (Q&A) forums). Hence, creating the problem of information overload and increased cognitive load because of frequent context switching between the archives. Potential employers, who have limited time and many applicants to review, are unlikely to spend significant amounts of time searching online archives. Marlow and Dabbish [4] found that employers assessed those online activities that were "low-effort". In the majority of cases, employers did not investigate the contribution or interaction details, and instead focused only on the aggregate amounts of activity, despite identifying interaction style and type of contributions as important factors for hiring decisions [4].

To alleviate the problem of excessive information load, researchers and practitioners have developed various tools for quickly evaluating developers' online activities, e.g., CVExplorer [6], Statocat [11], and MasterBranch [12]. These tools leverage developers' online activity traces, but still do not provide an integrated view of activities across various platforms (see Section 8). This is a problem since developers tend to be active on multiple technical platforms.

To address these challenges, we designed a profile aggregation tool, Visual Resume, that aggregates the activity traces of developers across different types of contributions and repositories into a single developer profile. Visual Resume goes well beyond the current state-of-the-art aggregator sites by: (1) aggregating data across two different types of peer-production sites–GitHub and Stack Overflow, (2) creating profiles that not only provide overviews of activities, but also allow deeper exploration of contributions that are contextualized and easy to access, (3) extracting and visualizing quality attributes of contributions, and (4) allowing side-by-side comparison of contributors and different types of contributions.

We designed and evaluated an initial prototype of Visual Resume to study how employers (those experienced with hiring) can use aggregators and formatively evaluated it in [13]. Based on this evaluation and participants' feedback, we extended the prototype to include additional features for assessing the quality of contributions. In this paper, we present: (1) the set of design guidelines for creating aggregators, (2) an extended version of Visual Resume and its implementation details, (3) a user study comparing evaluation of "job candidates" via GitHub and Stack Overflow, and with Visual Resume, and the set of cues that participants used when making their evaluation.

2. Background

A developer's technical and social skills are crucial for their effectiveness in a team, especially in a global setting [14–16], where differences in organizational culture can make it challenging to build a working (trust) relationship [17]. Numerous studies have found that trust among team members allows them to work effectively [18,19]. Past interviews of respondents in global development settings have identified the characteristics that people look for when evaluating the fit of a new team member and if they can be trusted, namely: technical competency, collaboration and communication proficiency, and how passionate they are about the project [14,18]. Similarly, research on recruitment in online communities has found technical and social skills play an important role in making hiring decisions [2,4,20,21]. Here, we summarize how different types of skill sets are used for evaluating job candidates by reviewing the literature (see Table 1).

Technical Skills These are developers' skills related to writing code and the quality of the code. Technical skills can be further divided into two areas.

Coding Competency, the primary qualifications of any software developer include their programming knowledge and coding ability. The level and amount of past activities (in a project or programming language) indicate an individual's experience level [5,18]. Managers seek the following cues from an online environment: (1) owned and forked projects, (2) frequent contributions to projects, such as providing commits or answering questions, and (3) the number of languages in which a candidate is proficient.

Table 1
Candidate qualities and activity traces.

Quality inferred		Quality cues
Tech skills	Coding competency	Level and amount of past visible activity – Number of projects owned or forked – Number of commits/issues/comments – Frequency of commits/issues/comments – Programming language used
	Quality of work	Content of the contribution Community acceptance of the work – Number of accepted commits/answers Test case inclusion
Soft skills	Collaboration proficiency	Visible communication activity – Number of comments/answers/questions – Types of comments/answers/questions Endorsement of contributions – Number of followers – Reputation scores
	Project management	Number of projects owned
	Motivation	Recency and volume of commits/issues/comments – Number of commits/issues/comments – Recency of commits/issues/comments Number of non-work-related side projects Diversity of skills – Number of programming language – Number of contributed projects

Quality of Work, the quantity of an individual's contributions must be understood alongside their quality, which can signal a candidate's competence and skill level [4]. Quality is a subjective measure, and its signals can range from code reviews to test coverage metrics. While managers most frequently use the actual lines-of-code produced (or the post), other criteria exist. For example, contributions that include test cases can indicate a well thought out contribution [2,22]. Similarly, information about whether the community accepted a candidate's work (e.g. commits) can also indicate quality [4,21].

Soft Skills These are the non-technical skills related to motivation, project management skills and collaboration proficiency of developers. Soft skills can be divided into three categories.

Collaboration Proficiency, when deciding to collaborate or trust others, key factors include whether the person is polite or arrogant, and whether they are willing to help others and provide sufficient context to make their solution useful [14]. An individual's interaction histories can indicate whether she helps others and what she is like to work with [5,23,24]. Developer activities that serve as cues for (positive) interactions include: (1) comments regarding issues, (2) answers or questions submitted in Q&A forums, and (3) details about the nature of these activities, such as whether developers provide polite, articulate, and helpful answers. Endorsements also can be used as a proxy to assess collaboration ability [2].

Project Management Ability, managers prefer candidates who have some management skills [25]. When someone owns a project, they need to set the projects overall design, manage incoming contributions, and interact with potential contributors [4].

Motivation/passion, an important trait of volunteer contributors is their passion. Studies [18,26,27] show that motivation and performance are deeply connected: highly motivated individuals are more likely to perform better and influence future engagement. It has also been found that developers are likely to trust colleagues who are passionate about their work. A developer's motivation can be indicated by: (1) the recency and volume of activities (e.g. commits, issues, comments) across projects, (2) the number of owned or forked projects, which are not directly related to the developer's own work, but are done as a hobby or for fun [4], (3) the diversity in languages that a developer is comfortable with and the diversity in projects they take on (e.g., different technologies and programming languages) [2]. Research

has found that employers and developers have started using online project hosting sites to evaluate job candidates [4] or to assess the performance of their colleagues [2,22]. Since contributions in online peer production sites are archived and maintained by a third party, they are seen as assessment signals that are hard to fake [26], and managers prefer them over static resumes or out-of-context code samples [2,4].

However, the amount of effort that is required to reliably assess the skills of a developer using online activities is non-trivial. To evaluate the lines of commit in GitHub, one must first identify the projects to which a developer has contributed from their profile and then navigate to specific projects. Once on the project page, one must scroll through all the commits in the project to find the developer's commits, which can be located by recognizing their user-id. Clicking on the commit-id takes the user to a page where the lines of code changed are listed. If there are multiple commits, one has to keep scrolling through the list to identify the commits from a particular developer.

Marlow and Dabbish [4] found that evaluators did not assess the actual lines of code changed, but instead used their perception of the reputation of the project as a proxy for the quality of a developer's contributions. This was especially true if a contribution to a high-status project did not appear in the top recent activities of the project, since this would entail scrolling through hundreds of commits to identify the commit from the developer. Assessing developer interactions requires even more effort. To evaluate the discussions around a code snippet, one must first identify the pull request associated with the commit (and the lines of code) and then read the discussion around it. In fact, most evaluators did not assess information regarding developer interactions when forming hiring impressions in GitHub [4].

3. Guidelines for creating Visual Resume

Designing a tool that allows exploration and assessment of developers' skills from online contributions requires answering the following questions:

Question 1: What information should we display? Past studies (Table 1) have found that both technical and soft skills are important in a candidate; and developers contribute to multiple projects and different types of content. While a few initial aggregator sites exist [2], they only provide activity overviews: projects and programming languages that a developer has contributed to or owns, and overall commits. To assess contribution details, one still must exert the manual effort to navigate to the developer's profile or the project page.

Question 2: How should we present information that is contextualized to the project, easy to access, and allows comparison? Typically, employers first screen developers based on the quantity of their contributions, and then perform a detailed comparison on a subset of developers [28]. This indicates that profiles not only need to present a high-level overview, but also allow easy access to details about contributions in a manner that facilitates comparison across candidates.

3.1. What to present?

The following guidelines are derived from the cues that managers and hires are known to use when making their decisions, as aggregated in Table 1.

DG 1: Aggregate cues across projects and sites. Aggregating individuals' public activities across online communities can help build more accurate profiles [5]. There are two reasons for this. First, aggregating activities across communities overcomes the problem of fragmentation. That is, developers may be active in one community, but inactive in others. Such a developer's profile will be inaccurate if only one community (or one type of skill) is considered. Second, aggregating data from different sites into one site helps solve the problem of disparate design across sites. That is, project managers do not have to navigate through different site designs and can therefore be more efficient. But, more importantly, developer profiles will be consistent, thereby avoiding the formation of biased impressions because of differences in how individual sites highlight specific contribution types.

DG 2: Provide cues for both technical and soft skills. Project managers make use of activities involving both the technical practices and the social communication (soft skills) when evaluating contributions [22]. A developer's complete profile should include activities that signal both types of skills (see Table 1) to project managers.

DG 3: Provide cues for quality. In addition to quantity of contributions, it is also important to reflect the quality of a developer's work [4]. Table 1 lists a set of cues that can be used to assess quality. For example, the quality for a developer's answers in Stack Overflow can be measured by whether their answers were accepted or up-voted; the quality for commits in GitHub can be signaled through whether their pull requests (or commits) were accepted (or merged), and whether the contributions contained test cases.

DG 4: Present social standing in the community. Many sites include badges or reputation points to motivate their users to participate. Stack Overflow users accrue reputation points when their contributions are accepted. Similarly, GitHub developers collect followers if their projects are interesting or perceived as high-quality work [22]. These endorsements serve as proxies for the candidate's overall amount and quality of work. However, it is best to provide community-derived reputation values, as they are likely to be more trustworthy [28].

3.2. How to present?

We recommend that contribution information be displayed through the following mechanisms.

DG 5: Summarize activity. Activities are archived over time, but large volumes of archived activity can overwhelm a user. Concisely summarizing expertise based on types of activities ameliorates this issue and reduces the burden of investigating profiles to assess developers' expertise and contributions. For example, summaries of languages used, projects contributed to, and commits can help project managers assess developers' technical skills. Summaries of comments, questions and answers can help project managers examine developers' soft skills.

DG 6: Visualize summaries. Project managers favor cues that take less effort to verify [4]. Visually summarizing activities can help project managers quickly assess developers' quality and allow them to view activities over time. For example, a visual summary of accepted versus unaccepted commits can give project managers an overview of the quality of a developer's contributions. Similarly, developers' soft skills can be judged by viewing a visual summary of answers versus questions or forked versus owned projects.

DG 7: Allow drill down. Detailed activity information can shape the evaluation outcomes for complex contributions [29]. However, since project managers favor cues that take less effort [4], activity summaries can decrease the effort to access information indicating work quality. For example, in order to assess a developer's coding ability, project managers may be interested in looking at not only the summary of their commits, but also the source code related to all or some of their commits.

3.3. How to compare?

We recommend the following interaction mechanisms to effectively and efficiently evaluate candidates.

DG 8: Allow quick and detailed assessment. Project managers often first filtering candidates by assessing summaries of technical activities and then taking a more detailed look at activities to assess the experience and social skills of a candidate [2]. In other words, they initially skim through candidates and then perform a more detailed pass.

DG 9: Allow pairwise comparison. Past studies have shown that pairwise comparison is a key recruitment strategy [28,30]. Project managers typically develop a list of desired knowledge, skills, and abilities, and then use pairwise comparisons to reveal the relative priority of potential candidates.

4. Visual resume

Visual Resume provides concise summary information about developers' activities to allow a quick assessment of their skills (DG 8) and facilitate comparison across candidates (DG 9). The source code for Visual Resume can be found at [31]. It uses a card-based design that summarizes contribution histories from two types of online peer production sites (code-based and Q&A) (DG 5)and allows quick drill down to specific contributions (DG 7). In this section, we present the design process behind Visual Resume, the information that it uses to create developer profiles, and its user interface.

4.1. Design of visual resume

We first identified the information that should be part of the developer profile by leveraging the cues identified from the literature and summarized in Table 1. Additionally, we conducted interviews with two industry contact team leads with extensive hiring experience. We asked them about current practices when making hiring decisions and how they evaluated online contributions. We then asked them to select the most relevant cues from Table 1. Before implementing the tool, we conducted two rounds of paper prototyping by creating wire-frame mock-ups for the tool. The paper prototype evaluation was done with other researchers in the group.

We then implemented the first version of Visual Resume, which was formatively evaluated by nine industry partners (P1–P9); the details of this evaluation can found in our prior work [13]. The results of this evaluation showed that (all nine) participants used the summarized views to create overall impressions and then drilled down for a deeper view of the contributions' contents. Participants recommended including contribution quality metrics as part of the developer profile. For example, one of our participants (P4) mentioned, *"...[while looking at the activity] this is meaningless without quality...how can he have fixed anything if there is no code in the commit"*.

We therefore extended Visual Resume to include metrics for quality of contribution. Additionally, we synthesized the set of design guidelines from the literature, our experiences in building the initial version of Visual Resume, and the feedback from participants. We used these design guidelines when extending Visual Resume. In the rest of this section, we describe the design and implementation of the second version of Visual Resume.

4.2. Contributor profile

Visual Resume summarizes the following pieces of information as part of the developer profiles.

4.2.1. Historical activities

Developers' activity histories gather activity traces in terms of *issues, comments, commits, questions, answers,* and *join period* (DG 2). These activities are obtained by mining issue tracking systems and version control systems (DG 1). Questions and answers also can be collected from online Q&A forums.

Issues: issues refer to bugs and contributions of new code that are submitted by developers and stored in issue tracking systems. For each developer, we collect the total number of issues they submitted and the total number of issues they closed before a specific timestamp.

Comments: comments are discussions created in a project's issue tracker. These discussions often focus on resolving specific issues and are technical in nature. We collect the total number of comments created by each developer before a specific timestamp (comments).

Commits: Commits refer to changes of source code performed to resolve related issues and improve features. They are saved in version control systems. We collect the total number of commits that are submitted to the version control system by each developer and the total number of commits that are committed by each developer before a specific timestamp.

Questions: Developers pose questions to seek help from others, often in regard to programming. We collect the total number of questions asked by each developer before a specific timestamp.

Answers: Developers provide detailed answers to questions to earn reputation. Accepted or voted up answers can earn a higher reputation score. We collect the total number of answers that are submitted to the Q&A forum by each developer before a specific timestamp.

Join Period: we collect this information both from the version control system and the Q&A forum.

4.2.2. Quality of work

The quality metrics that we use is derived from cues for quality of work as described in Table 3. We measure the quality of developers' work from the following five metrics: centrality, passed tests, closed issues, merged pull requests, and accepted answers. (DG 3)

Centrality: centrality is to measure whether developers make core contributions i.e., contributions that span multiple files in the code base and can be of high impact [32,33]. We are interested in identifying the centrality of each commit made to the project's source code repository. Source code can be thought of as forming a network of different files that are connected to each other. One common metric for computing the centrality of each file in the project is eigenvector centrality [34]. Then the source code file level centrality needs to be translated into commit centrality. As commits made by developers often touch multiple code files, we define each commit's centrality as the mean of the centrality of each of the code files that are related to the commit. From this we generate a centrality score for each commit. A commit with high centrality scores deals with files that are closer to the core of the project than those with low centrality scores.

Passed tests: passed tests indicate whether commits made by the developer are successfully compiled and pass the tests. We use the following procedure to verify every commit:

1. Check a commit: we use git commands – "git reset –hard, git clean –xdf, git checkout" – to return to a specific commit and check it.
2. Compile and run all tests: this step is to verify the commit by calling out compiling and testing systems (e.g. Maven, Ant, rails-dev-box).
3. Process the results: finally, we retrieve the past-test status of the commit by checking the output result, allowing us to see whether the build and test for the commit succeeded.

Closed issues: issues are used to keep track of bugs, enhancements, ideas, or other requests. This metric reveals whether issues submitted by the developer are closed or still open. Once the bug is fixed, the new contribution can be merged, or once the request is accomplished, the related issue is closed.

Merged pull requests: pull requests are a type of issue. However, pull requests involve changes the developer has pushed to a repository. Once a pull request is opened, the changes are fully discussed and reviewed by collaborators and other contributors before they are merged into the repository.

Accepted answers: an accepted answer refers to the answer that is marked as such by the person who asked it.

4.2.3. Other cues

Endorsement: It is the metrics for trust of the community members and can be unique to socio-technical platforms. We collect the number of followers from the version control system and the reputation score from the Q&A forum. (DG 4)

Projects/languages: To assess the breadth of a developer's experience, we collect the number of owned or forked projects from version control system. The programming languages/tags are also gathered to assess the diversity of projects from the version control system and Q&A forum.

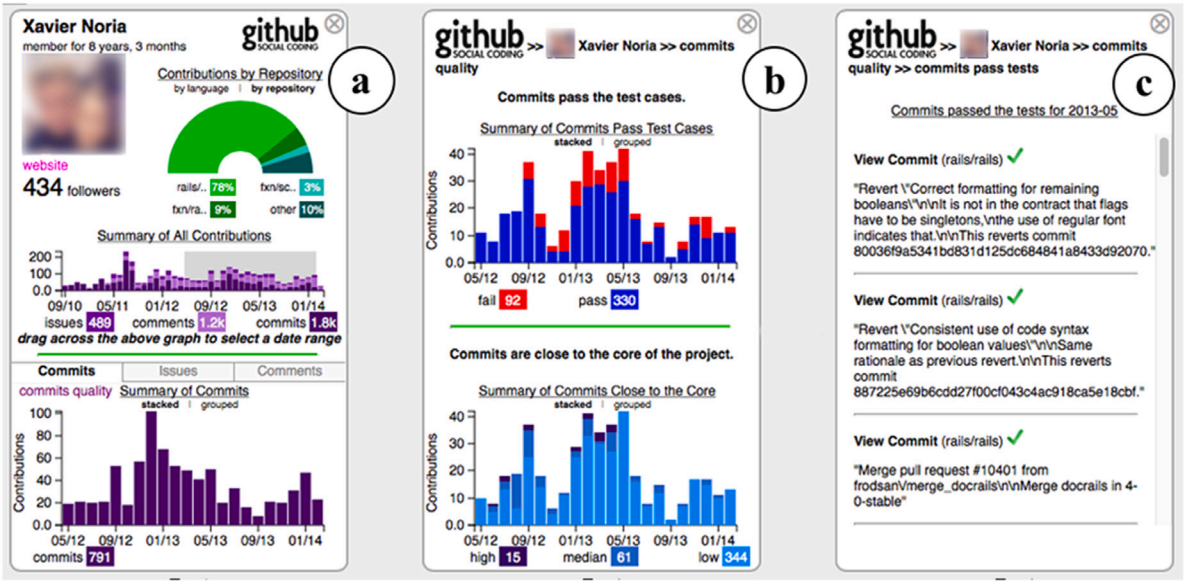


Fig. 1. Drill-down functionality for (a) GitHub card when commits tab selected, (b) the quality of commits and (c) details of the commits.

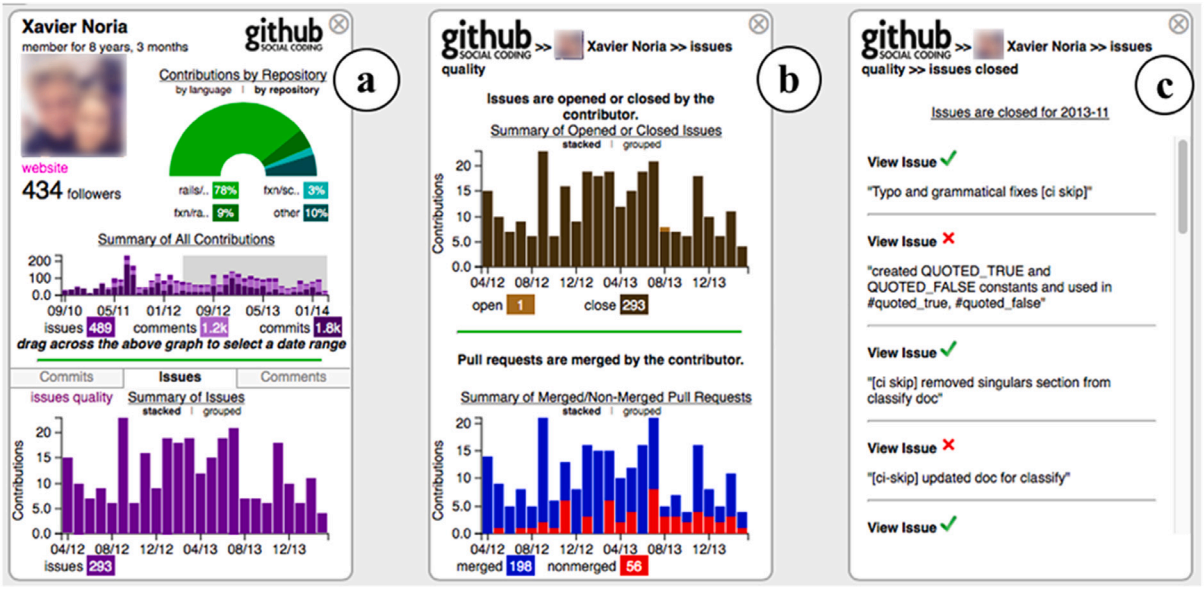


Fig. 2. Drill-down functionality for (a) GitHub card when issue tab selected, (b) the quality of issues and (c) details of the issues.

4.3. User interface

Visual Resume presents a candidate's information through "cards" –small cards that are 300 x 500 pixels. These cards evoke the notion of business cards, and allow quick side- by-side comparison of developers (DG 8, DG 9)– an important recruitment strategy [28,30]. Fig. 1(a) shows a candidate's GitHub (GH) card, displaying his activity summary. The left top of the card shows his profile information GitHub ID, picture, tenure in the site, personal website(s) or blogs, and number of followers (DG 4). A user can click and navigate to each item in the profile. In the rest of the card, contributions are summarized and visually presented (DG 5, DG 6) as project managers favor information that takes less effort to verify [4]. While details of contributions are the final criteria that shape evaluation outcomes [5], they are harder to access. For example, to assess coding ability, an employer may want to look at the source code and its style, but doing so may require scrolling through numerous pages of commit history in a (highly active) project to locate to the specific contribution. Such a detailed assessment of all of a developer's contributions or for all candidates is infeasible.

To overcome this problem, Visual Resume allows easy access to contribution details. At the right top of the card, a radial chart provides a breakdown of the repositories in which the person is most involved, based on the number of their commits, comments, and issues in the repository. Hovering over a slice of the chart presents the repository name, the main programming language of the repository, and the number of watchers. A user can click on a slice and drill down to examine the contributions of the developer unique to that specific repository (DG 7). Clicking on a slice in the radial chart opens a new card, which is similar to the GH card (and therefore not shown in Fig. 1, but shows information pertaining to the selected repository. If contributions by language is selected, the radial chart changes to show the breakdown of the programming languages of the candidate's contributions. In this case, the slices in the chart aggregate activities (commits, comments, and issues) across all projects that use the same programming language. In the lower half of the GitHub card, the bar charts summarize a

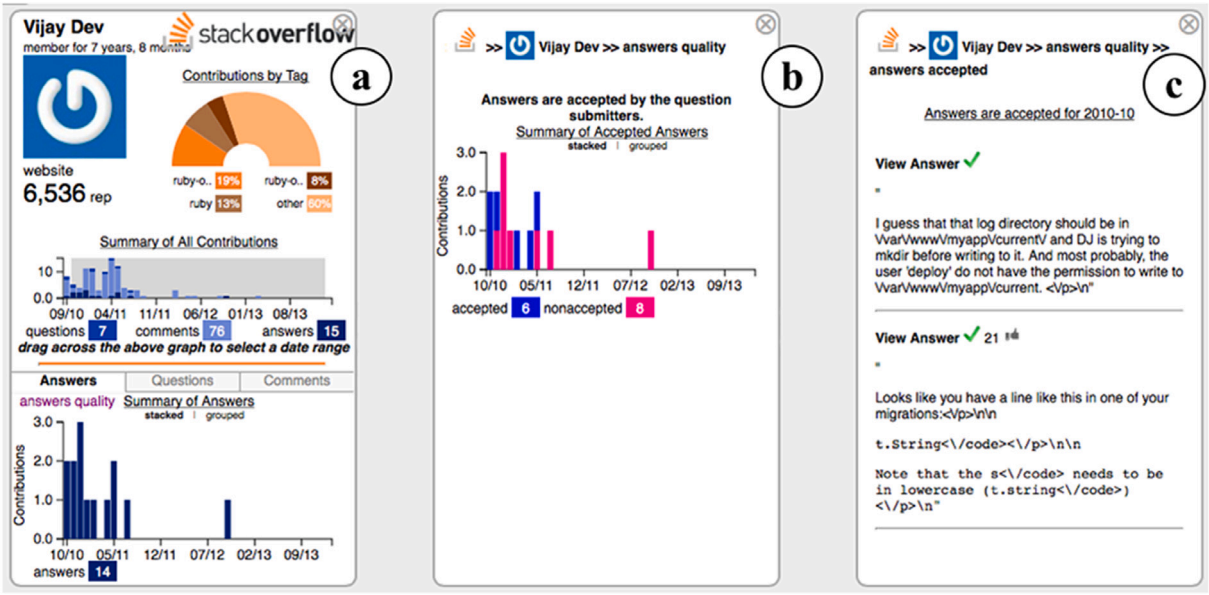


Fig. 3. Drill-down functionality for (a) Stack Overflow card, (b) the quality of answers and (c) details of the answers.

user's contributions: commits, issues, and comments across projects on a monthly basis (DG 6). The chart in the middle of the card presents the entire history of the candidate, from which a user can select a specific date range.

Our prior study [13] signaled that assessing the quality of contributions was important in the decision-making process. Visual Resume is extended to include more explicit quality metrics that allow easy assessment of commits and issues (DG 3). At the bottom of the GH card, clicking the commit's quality opens a new card, which provides two quality metrics (see Fig. 1(b)). The first metric for measuring commit quality checks whether commits passed a test case or not, which is presented in the top bar chart. The bottom chart also provides a summary of commit centrality, allowing viewers to identify if commits deal with core files. The card (see Fig. 2(b)) on quality opens by clicking "issue quality" under the Issues tab (see Fig. 2(a)) at the bottom of the GH card. The first chart displays the summary of opened or closed issues, which indicates how enthusiastic the person is about participating in the development of the project. The second chart shows the information on whether pull requests are merged or not, which can signal the quality of the work based on the acceptance by the community.

Contributions are displayed either as a stacked bar chart or a grouped bar chart (DG 6). In the former, all types of contributions are stacked into a single bar, whereas in the latter, each contribution type is represented by its own bar. These charts can help portray contribution patterns or trajectories. For example, if a developer has become more active in a project, the stacked bar chart easily shows this increase, regardless of the type of contribution. However, if one wants to track the activity levels of a specific type of contribution (e.g., commits), then the grouped bar chart is a better option. Hovering over a segment in the chart displays the number of contributions for the month in a pop-up. Clicking a bar (segment) in the bar chart opens a new (drill-down) card that displays detailed information on the type of contribution. For example, if the segment of the 05/13 bar in the (stacked) bar chart that shows the commits that passed the test case is clicked (see Fig. 1(b)), a new card opens listing commits and an excerpt of the commit comment (see Fig. 1(c)). We opted to display the commit message instead of the commit-id since it can provide some information about the commit. The commits include an annotation about whether it passed the test case. A user can click the View Commit link to further investigate a contribution. Doing so takes the user to the GitHub page, where they can view the commit and the lines of changed code.

Fig. 3(a) shows a Stack Overflow (SO) card, which is very similar to the GitHub card. The top left of the SO card shows the tenure and reputation score in Stack Overflow. The radial chart gives a breakdown of the various tags (programming languages, concepts, etc.) with which the person is most involved based on their number of questions, comments, and answers. Bar charts show the individuals contributions (in terms of questions, answers, and comments) on a monthly basis. Clicking "answer quality" under Answers tab at the bottom of the SO card opens a new card (see Fig. 3(b)), which shows whether answers were accepted by the question submitter. This metric indicates the quality of answers.

Clicking on a specific activity bar opens a new card, Fig. 3(c)) that lists the contributions (in this case a list of answers). The answers include annotations about the number of up-votes, whether it was accepted (thumbs up sign), and the number of comments associated with that answer. Users can drill down to view the full answer, its associated question, and comments by clicking on the View Answer link, which takes them to the Stack Over page.

Each person's activity is shown in a card, which can be closed or rearranged by simply selecting and moving a card across the screen (DG 8). The cards allow easy pairwise comparison. Viewers can compare contributions between multiple users or for the same user in different contexts (DG 9). For example, users can compare the GitHub contributions between two developers or the contributions across different sites (Figs. 1 and 2), projects, or programming languages.

4.4. Implementation

Since Visual Resume is designed as a web application, it does not need to be installed on the client site. It follows a 4-step approach: collect, process, filter, and visualize (see Fig. 4). The former two steps are performed on the server side, and require a wrapper for each repository; the latter two are part of a rich web client that uses a model-view-controller architecture.

Collect: Visual Resume can collect data across different repositories. Each data source requires a specific extractor that collects and stores the data in our database. Currently, we have implemented extractors for two different types of peer production sites: GitHub and Stack Overflow [35]. Extractors for other sites can be easily designed. We need site-specific extractors, since data from each site are accessible in different ways. For example, the extractor for GitHub uses the GitHub

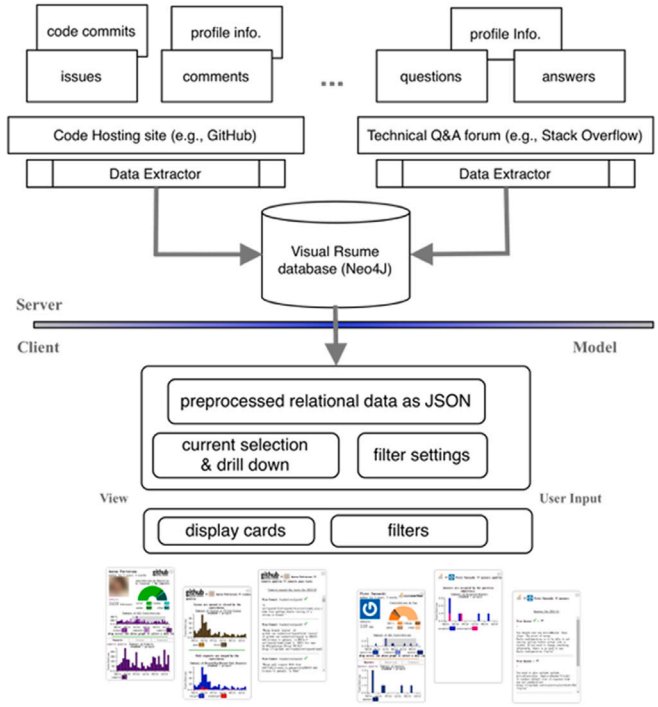


Fig. 4. Architecture of Visual Resume.

API. While the GitHub API only allows 5000 requests per hour when using basic authentication, requiring us to periodically extract the data and incrementally update the database, Stack Overflow provides periodic data dumps of the entire history. The extractor needs to identify the new data from the dump to update the database.

Process: This step has three functionalities. First, it transforms data collected in different formats into a uniform format (we use Neo4j a graph database [36]). Second, it creates a data model designed to generalize across different types of project hosting and Q&A sites. New sources of information (discussions in mailing lists vs. comments on an issue) can be easily added to the schema. The data is linked such that aggregations and queries can be performed per user, per repository, per language, per tag, etc. Other pertinent information (personal page, blogs) available from profiles in GitHub or Stack Overflow are also linked. This model is then encoded as a JSON file.

Visualize: The visualization is created by using the d3.js framework [37]. Currently, our card template uses a top-down layout. It uses label, radial chart, and bar chart widgets to display aggregated data. Different templates that use other layout or widgets can be easily implemented and incorporated.

Filter: Different filters can be used to adjust the amount of information presented to the user. A basic filter that we have currently implemented is time period selection. We can easily create other filters that adjust other kinds of information (e.g., the amounts or types of contributions).

5. User study

We conducted a scenario-based, task analysis study with ten participants to: (1) understand what information participants seek out when they have to choose from a set of potential job candidates, and (2) investigate how participants collect information to make a hiring decision in a peer production sites such as GitHub and Stack Overflow vs. the information provided by Visual Resume. Note, that we were not seeking (or expecting) that all participants would converge on the same candidate, since this is a subjective decision; instead, we observed how participants arrived at their decisions.

Table 2
Background of study participants.

Code	Background
P11	Vice president of a large organization. Experiencing in hiring for over 12 software related employees as the hiring manager.
P12	Data scientist at a small startup with experience in hiring colleagues.
P13	Research scientist at a government software contractor for 5 years. Experience in recruiting and hiring interns as well as employees.
P14	Director of software product development at large corporation.
P15	Research computer scientist with experience in hiring.
P16	Software engineer at a large corporation. Experience in interviewing several team members.
P17	Software system analyst at medium size corporation.
P18	CEO of a software startup for 5 years.
P19	Research lab director at large corporation. Experiencing in hiring over 20 software related positions.
P20	Computing coordinator for 16 years. Experiencing in hiring 6 developers.

5.1. Study participants

We recruited ten participants for our study. These participants were selected to represent individuals who had experience in the hiring process. We also recruited participants to obtain both corporate and small software company participants. This was done because the software engineering practices used in established corporations vary from those used in lean startup operations, where agile methods are more prevalent. Further, we included participants who hire for their teams as well as those who interview their peer developers. Differences in typical development practices may in turn favor different cues for evaluating job candidates. Table 2 summarizes our study participants' backgrounds.

5.2. Study design

Participants were told that they were the technical lead of a project and had to assess a set of candidates in response to a job advertisement. We created the job description to represent a typical posting for web development positions. To do so, we reviewed postings on popular job posting sites, such as LinkedIn, Careers 2.0 [38], and CareerBuilder [39]. We included aspects that typically appeared across many postings, including: position description, job responsibilities, required qualifications, and benefits.

The user study was within-subjects and comprised of selecting top two candidates from a given set of five candidates for each treatment. The evaluation tasks were designed based on our formative study [13] and current industry practices [2]. We interviewed two industry contacts who were team leads with extensive experience in hiring. We asked them about current practices when making hiring decisions and how they evaluated online contributions.

The tasks were divided into two sets (Task 1 and Task 2): In Task 1, participants were asked to use GitHub and Stack Overflow, or any other online resources they wished to evaluate a set of five candidates. In Task 2, participants were given Visual Resume, and were free to again use any other online resources (e.g., blogs, personal website, Google search etc.) that they wished to evaluate another set of five candidates. The goal of Task 1 and Task 2 was to evaluate and compare the GitHub and Stack Overflow websites with Visual Resume when making hiring decisions to select top two candidates. Task 1 always preceded Task 2 (i.e., we did not counterbalance tasks) as we did not want to bias

participants' information-seeking behavior based on the cues provided by Visual Resume.

We asked participants to think aloud during the study, verbalizing their actions and the intention behind them [25,40]. We screen-recorded the participant actions and collected their feedback through an exit interview. To analyze the data from our study, we transcribed participants' verbalizations and actions from observations, notes, and think-aloud data. We used the code set related to technical and soft skills (Table 1) for analyzing the transcripts. Additional cues related to quality of work and social competency were identified, which we added to our (cue) code set. Two researchers collaborated on the coding until 80% agreement was reached on about 20% of the data.

5.3. Job candidate selection

We selected ten potential job candidates for the study. These candidates were selected to represent typical GitHub and Stack Overflow users with some expertise in the topic areas indicated in the job description (Ruby and Java Script). To identify these candidates, we first extracted a dataset of GitHub participants with at least one commit to the Ruby on Rails project on GitHub. Next, we queried Stack Overflow for these users to identify a subset of users with profiles and activity on the site. From this sample of 2300 common users in both communities, we then identified candidates who had monthly activity on both sites (240 users). From this set, we randomly selected the ten candidates for the study, and divided them into two groups. We counterbalanced the groups of candidates used in Task 1 and Task 2. That is, half the participants used the first group of candidates for Task 1 and the rest used that same group of candidates for Task 2.

5.4. Limitations of study design

Visual Resume currently only collects activities from two sources: GitHub and Stack Overflow, and our results might not generalize to cues found in other sources. Moreover, these sites may lack contributions from underrepresented populations in software, because of which aggregating contributions from them can perpetuate a self-reinforcing behavior of exclusion. Future work should extend Visual Resume to include other types of contribution sites also. Our participants were limited to a small subset of employers who volunteered and therefore might be biased towards assessing online contributions. There might be learning effects since we did not counterbalance the treatments, but this was necessary since we did not want to bias the type of information (and sources) that participants in the Control condition would look for based on data provided by Visual Resume. Finally, we evaluated Visual Resume by using only ten subject candidates. Our results regarding strategies and cues used might not hold true for a larger candidate pool.

6. Results

This Section presents the cues that participants used to evaluate candidates' technical and soft skills (Section 6.1), followed by a discussion of the key information features that were used for the evaluations (Section 6.2) and how the information was accessed (Section 6.3).

6.1. What cues were utilized for selecting candidates?

Participants used a variety of cues when selecting the top two candidates from the set of given five in Task 1 and Task 2. Table 3 summarizes the information sources, the cues, and the associated (tool) features that participants used in the Control and Experimental conditions.

In both the Control and Experimental conditions, participants started by getting an overview of the candidates' profile and then details on personal interests. Ten participants started with profile pages of GitHub and Stack Overflow in both conditions, making it the most investigated feature. Participants also often visited candidate's personal websites (seven participants in Control, and six participants in Experimental) to get an overview of candidates' external contribution and personal interests.

After which participants (nine participants in both conditions) focused on cues related to the amount and type of candidates' contributions to get a thorough understanding of candidates' technical skills. This evaluation pattern is not surprising as participants' primary goal was to "hire" a suitable software developer, which likely prompted them to focus on candidate's technical skills; P12 reflected, "*I also didn't use the Stack Overflow at all in the first task, because I believe reputation in Stack Overflow does not depend on how well someone codes*".

Only after evaluating technical skills they focused on soft skills, so as to hire a candidate with a comprehensive skill set. P14 commented: "*I checked Stack Overflow to be able to choose among them (candidate short list) better*".

6.1.1. Cues for technical skills

Technical skills were evaluated primarily through the perspective of Coding Competency and Quality of Work (Table 1)

Coding Competency: In both Control and Experimental conditions, participants reviewed candidates' commit histories and the number of projects candidates contributed to by either using the "GitHub repository list" or the "Visual Resume repository cards". They largely relied on the visualizations provided by the tools. In the Control condition participants reviewed the GitHub "activity graph" and "recent activity", which includes commits, opening or closing issues and pull requests. Only one participant evaluated the details of technical contributions by either drilling down to the commit page or the issue list.

In the Experimental condition participants reviewed the "Summary of all Contributions", which also includes contributions about issues, comments and commits (see Fig. 1a). Some participants further drilled down into the technical contributions: commit page (three participants) and issue list (five participants).

Quality of Work: Participants leveraged different types of cues to decipher the quality of candidates' contributions as follows:

(a) ***Commit Details***: The "content of the contribution" is a key quality evaluation criteria in Table 1. However, few participants in the Control condition sought to assess the quality of the contribution (only one participant viewed the committed source code). Instead they relied on the amounts of contribution. This could be because there is no direct metric defined in GitHub to evaluate the quality of the committed code. Additionally, evaluating source code would be difficult for users who were not familiar with the project or its programming language. For example, P17 explained: "*Since I am not a Ruby or JavaScript expert...maybe if I knew about the language itself I would try and check the commits more deeply. I just based [my decision] on the core commits and seeing the steady flow of the contributions in the last couple of years mainly*".
Six participants in the Experimental conditions used the metrics provided by Visual Resume to assess the quality of the GitHub contribution (features of centrality and test passes, see Fig. 1(b)). For example, P11 stated: "*[candidate] is a core contributor to several large frameworks*". P20 mentioned, "*[I chose this candidate] based on the number of commits, especially the commits that are close to the core*". Similarly, P19 also commented, "*[I chose this candidate because of] total commit but fewest fails*"..

(b) ***Community Up-votes:*** Up-votes[1] and Stars[2] are features in Stack Overflow and GitHub that represent the appreciation and support from community users. Three participants (Experimental

[1] https://stackoverflow.com/help/privileges/vote-up.
[2] https://help.GitHub.com/articles/about-stars/.

Table 3
Cues and feature used by numbers of participants in selecting candidates across both Control and Experimental conditions (task 1 and task 2).

Data source	Cues	Features in Ctrl. group	Features in Exp. group
GitHub	GitHub Contribution Overview	GitHub profile pages (10) GitHub recent activity (9) GitHub activity graph (4)	GitHub profile card (10)
	Programming Language Expertise	Not able to display	Language radial chart (7)
	Commit Overview	Commit history list (5)	Commit history list (6)
	Commit Detail	Commit page (1)	Commit page (3)
	Repository Overview	Repository list (5)	Repository list (7)
	Repository Detail	Repository information (7)	Repository card (8)
	Issue Overview	Issue list (1)	Issue list (5)
Stack Overflow	Stack Overflow Contribution Overview	Stack Overflow Profile (9)	Stack Overflow Profile (8)
	Answer Overview	Answer list (3)	Answer list (3)
	Answer Detail	Answer page (4)	Answer page (1)
	Question Overview	Question list (3)	Question list (2)
	Question Detail	Question page (1)	Question page (1)
	Personal Website linked to Stack Overflow	Personal website (7)	Personal website (6)

condition) verbalized using the number of up-votes and stars as indicators of the quality of the answers or code artifacts. For instance, P13 commented, "*number of votes in answers in Stack Overflow is a good indicator of someone's understanding about an issue*". Similarly, P12 commented on leveraging the number of stars to decipher quality, "*[I choose her since] one of the projects that she created has 4000 stars and it's very well used in rails community*". Further, P15 also picked a candidate as her final choice due to community up-votes, "*he has answered some questions, which are pretty high ranked…and his response get[s] good feedback*".

(c) ***Association with Popular Projects***: Participants assessed whether a candidate owned or contributed to popular projects. Two participants in the Control condition and three participants in the Experimental condition considered *owning* a project as important. Similarly, two and one participants in Control and Experimental conditions, respectively considered contributing to popular projects as a factor to evaluate whether a candidate is a suitable choice for a job position. For example, P20 commented "*[Jeremy] seems to have contributed to the early Rails and* MYSQL2-GEM*…and has some original projects on Rails*". In addition, P12 chose Aaron and Yahuda as his top two candidates and said, "*because both founded famous projects. They were self-motivated to see those projects come to life*".

(d) **Number of Followers**: Participants considered number of followers as a factor to determine the quality of work on GitHub—three participants in the Control condition and four participants in the Experimental condition. In both conditions, participants perceived a higher number of followers to relate to a higher quality of work. For instance, one of the main reasons P20 chose Jose (a candidate) was "*he has over 3.1K followers*".

(e) **Reputation Points**: Participants considered the reputation of the candidate as a factor for quality of work (six participants in both conditions). For example, P11 chose Jeremy as one of his top two candidates in the Control condition mainly because "*he is on Ruby on Rails' contributor page. He is number 2*". However, reputation points in Stack Overflow were not considered by our participants while making hiring decisions, as P12 commented, "*reputation in stack overflow does not depend on how well someone codes. Sometimes poor coders just want to help out*".

6.1.2. Cues for soft skills

Soft skills were evaluated primarily from the perspective of Collaboration Proficiency, Project Management skills, and Motivation (similar to cues in Table 1).

Collaboration Proficiency:

(a) ***Interaction traces***: Participants assessed communication skills based on candidates' communication traces: the amounts of Stack Overflow contribution as well as whether the contribution was accepted or up-voted by the user community. Some participants also reviewed the "answer page" in the Control condition (four participants), and "answer list" in the Experiment condition (one participant) to get detailed overview of the contributions. Only a few participants, across both conditions, drilled down to review the details of the contributions such as, the content of "questions" (one participant in both conditions) and "answers" (four and one participants in the Control and Experimental conditions, respectively). Three participants, during the feedback session, emphasized that they reviewed how the candidates phrase their questions/answers and their engagement with the community. For example, P15 commented, "*by looking at some of his responses in Stack Overflow, he is able to communicate, and his responses get good feedback*". In addition, P20 commented, "*[his] Stack Overflow presence is very active. [He has] good communication skill and willingness to help others*".

(b) **Endorsements:** Participants used endorsements to evaluate candidates. Six in Control and four in Experimental condition considered endorsements such as, number of followers, the GitHub contributor list, and reputation points to assess social competency. For example, P17 said "*I choose Aaron based on the number of followers, he seems to be a more important part of the ruby community. Which might indicate he is being a better collaborator*".She added that the number of followers reveals candidate's interest in the software: "*he [Aaron] seems to have quite a lot of followers, which seems to demonstrate he's been really known in the community and is the most enthusiastic*".

Management skills: Four participants mentioned that past project experience was the most significant evidence (which is inline with [13]). P11 commented, "*he [Yahuda] is a core contributor to several large frameworks and he is a key member of a startup*". P13 chose the same candidate for the same reason, commenting: "*he is running his own project Wycats*". However, the remaining six participants mentioned that the cues contained in archives such as GitHub and Stack Overflow were not enough to judge management skills. For example, P12 commented: "*the fact that they led their projects to fame makes me think that Yehuda and Aaron are the best project managers but it's scanty evidence. I would want to talk to them*". Similarly, P14 also mentioned, "*it [project management]…cannot be judged based on numbers and data alone*".

Motivation: A variety of cues were used to infer passion for learning. Most participants considered a high volume of activity in GitHub and Stack Overflow as cues for interest and motivation. One participant in the Control condition and three participants in the Experimental condition evaluated contributions on Stack Overflow to evaluate passion for learning. For instance, P11 commented: "*he [Yahuda] has more answers and comments on Stack Overflow showing that he is motivated to help others learn about the topic [Ruby]*".P15 considered asking questions on

Stack Overflow as an indicator of passion for learning, "*he [Ryan] asked interesting questions, that shows he wants to learn*". Participants used the number of up-votes to determine candidates' enthusiasm to help the community. Participants also used candidates' personal website as a source for evaluating passion. For example, P17 believed that personal websites can indicate developers' motivation toward their project saying, "*he's been having the most contribution and is being consistent. His website shows a lot of passion for the issues related to software*".

6.2. What information features were used?

In addition to the cues discussed above, participants also used cues from the following information features:

Candidate details: Participants started their exploration with an overview of the candidate; nine participants viewed the profile pages in Control condition and eight participants used the profile cards in the Experimental condition. From there participants (seven in Control and six in Experimental conditions) accessed candidates' information from external webpages (e.g., personal websites and blogs) to get a better understanding about the candidate. For example, P11 commented: *"I like to see what he writes in [his] page, what parts [of contribution] is he proud of"*. Participants reported that with Visual Resume it was convenient to access candidate's personal websites. For example, P18 said: *"VR tool provides links to candidates website, which is nice and I like it... just under the profile picture...which makes the external website more visible"*.

Amount of contribution: Participants in both Control (six) and Experimental (seven) conditions considered the amounts of contribution as important when selecting candidates. P17 mentioned the following factors helped determine her top two choices: *"(1) volume of work, (2) contributing to a lot of different projects including rails, . . . , (3) being very active in GitHub, (4) having a lot of recent activity, (5) participating a lot in the community, answering lots of questions about rails, (6) having lots of repositories linked to rails"*. Similarly, P11 used the popularity (number of stars) of a project that the candidate had started as a criterion for selection: "*...one of the projects that she [Akira] created has 4000 stars and it's very well used in rails community*". On the other hand, participants considered a low number of contributions as a criterion for elimination; P20 selected candidates based on their lack of activity or presence, as he stated "*he [Jeremy] has minimal Stack Overflow presence*". Similarly, P11 selected two candidates because *"they have less contributions and fewer commits compared to the top two [candidates]*".

Recent activity: We define recent activity as the amount of candidates' recent contributions in GitHub. It was one of the most used information feature. Five and six participants in the Control and Experimental conditions, respectively, used the commit history activity as their primary cue. P16 mentioned, *"Xavier has a lot of rails contribution recently"*. In contrast, a lack of activity was used to dismiss candidates, as P11 commented, *"[Candidate's] activity has died out in the past 2 years. He is not as active as other candidates"*. Most participants resorted to using recent activities (in addition to the amounts of contribution) as a key selection criteria: P19 commented, *"commits are enough for finding top [candidates]"*.

Commit Aspects: Commit aspects included the change content of the commit (diffs and modified files) as well as the commit message describing the change (Fig. 1c). This cue was used more in the Experimental condition (eight participants evaluated commit aspects), but only one participant did this in the Control condition. This was likely because Visual Resume made it easier to access the commit aspects. P12 commented, *"I made a decision based on candidates' commits, and whether they have related contribution to Rails and also the projects and plugins [which] the candidates created for Rails and how much of those were followed [by other GitHub users]"*.

Commits vs. Issues: While evaluating candidates, participants used the data on commits more often than on issues. In the Control condition, five participants viewed commits, but only one viewed the issue list; whereas in the Experimental condition eight participants viewed commits and five viewed the issue list. The reason for this could be that commits provide more details about the history (both success and failure) of the candidate, while the list of issues opened by the candidate only provides information about failures identified by contributors. For instance, P19 commented, *"commits show everything, tell a lot from commits, how someone is active"*. And later he commented, *"...don't need to look at issues, if they do not find a lot of failures, there are not lots of issues"*.

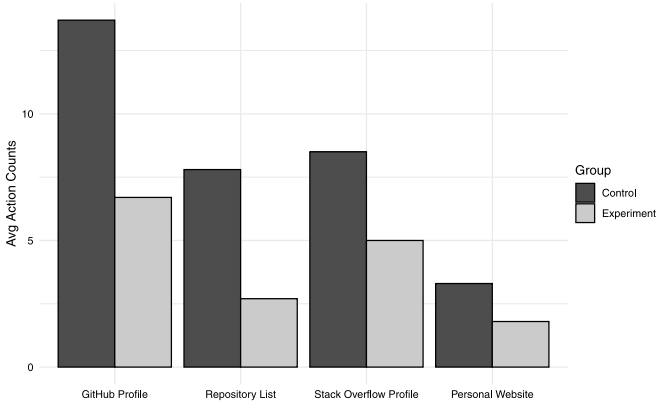


Fig. 5. Average participant actions (number of clicks) for various features in Control and Experimental conditions.

Type of projects: Participants used the repository list in order to identify the type of projects to which a candidates contributed (five participants in Control and seven in Experimental). They used the GitHub repository list and radial charts in the Control and Experimental conditions, respectively. For example, in the Control condition P12 selected Akira as his second top candidate as she had high reputation score and was working on a popular project while, rejecting Jose (candidate) commented *"I didn't see any rails on the first glance"*. Similarly, in the Experimental condition, P12 selected Aaron (candidate) and commented, *"Because he is the author of Rack, which is another hugely influential project. It's much more ruby oriented"*., while P17 rejected Vijay (candidate) and commented, *"He doesn't have lot of core commits [in a large project], he has a lot of small projects"*.

In summary, participants evaluated cues from a variety of information features to make their candidate selection. Although the overall time completion is similar for the Control and Experimental conditions, participants tended to use more information features when using Visual Resume.

6.3. How information was accessed?

Information Density: "Information density is the compactness of an interface in terms of the amount of information" [41]; Interfaces with higher information density require less "moving around" to access information and there is a higher likelihood that users will see the information they are foraging for more quickly [42]. Visual Resume aggregates information from multiple sources and has a higher information density than the individual pages in GitHub or Stack Overflow. As a result, participants when using Visual Resume took about half the actions (clicks) than when accessing the same types of information in the Control condition. Fig. 5 shows the average participant actions for various features in the Control and Experimental conditions.

Additionally, participants in the Experimental condition accessed more information than those in the Control condition. All ten participants "drilled down" into the cards to get a deeper view of a commit, issue, answer or question. For instance, P20 reviewed candidates' (ruby) programming language experiences across multiple repositories

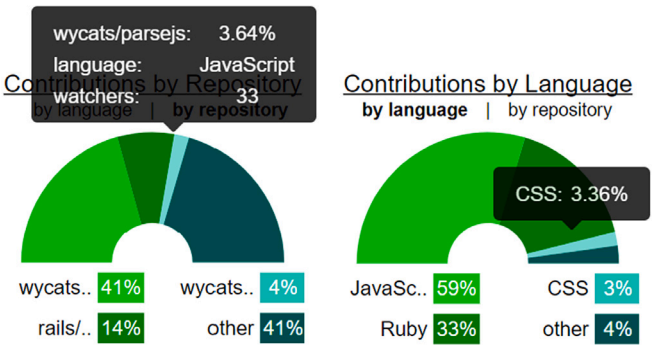


Fig. 6. Comparing candidates' contribution experience by programming language or repository through the radial chart feature of Visual Resume. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

by using the "repository" percentage diagram (the green radial chart in the profile card in Fig. 6). He viewed all five candidates' profile cards and then hovered over the repository percentages to identify (1) the number of repositories to which the candidate contributed and (2) the major programming language for each repository. In the Control condition, participants rarely investigated deeper into contributions. Only one participant manually drilled down to investigate commit details.

Side-by-side comparison: When evaluating candidates, participants kept track of the different candidates either mentally (Control condition) or by using the cards feature in Visual Resume. In the Control condition, five participants had to keep a mental record of all the differences between multiple candidate profiles while comparing them by switching between the candidates profiles (in GitHub site). In the Experimental condition, all ten participants kept open multiple candidate profiles (Visual Resume profile cards) and "closed" a card when they rejected a candidate. Keeping multiple profile cards allowed for easy comparisons across candidates. Participants found Visual Resume useful for making such comparisons, as P12 said, *"The first thing I need to determine is what the candidate contributed to and whether they were Rails oriented. Visual Resume is nice for comparison purposes".*

7. Discussions

Participants when using Visual Resume accessed more information sources related to candidates' technical and social skills, and did so in a shorter period of time. Our results indicate that Visual Resume's success arose from its ability to: (1) present both technical and soft skills through the same portal, (2) provide aggregated views of candidate's contributions, (3) allow drilling down to details about each type contribution, and (4) allow easy comparison of candidates via movable cards that could be arranged to match participants' needs. The main distinction of Visual Resume was that it allowed participants to efficiently scrutinize information through the cards, whereas in the Control condition participants had to "work harder" by scrolling through a long list of projects, switching between portals and multiple tabs (or windows), to evaluate who owns which projects, what programming languages candidates' worked in, the quality of the contributions, and so forth—a time-consuming and inefficient process.

What about contribution quality? A key feedback from the work in [13] was the need for a mechanism to better understand candidates' contributions. Therefore, the current version of Visual Resume provided explicit quality attributes for candidates' GitHub contributions such as, commits that passed test cases and their closeness to the core (Fig. 1), summary of closed issues and the status of pull requests (Fig. 2), as well as Stack Overflow contributions details such as summary of accepted answers (Fig. 3). All ten participants in the Experimental conditions drilled down into the cards to get a deeper understanding of the contributions. Very few participants in the Control condition did so. Instead participants mainly relied on the amounts and recency of contributions.

Six out of ten participants in the Experimental condition explicitly mentioned that the quality details of the commits (passing test cases and centrality of code commits) helped them select their top candidates. A few participants, both in Control and Experimental conditions, used (what we term as) community indicators—up-votes (three participants in each condition), number of followers (two and three participants in Control and Experimental, respectively), association with popular project (three and four in Control and Experimental, respectively), and reputation points (six in each condition)—as proxies to evaluate the quality of candidates' contributions. This shows that participants in addition to making their own assessments, also depended on the community's evaluation of the candidate based on their social standing.

Finally, when evaluating candidates participants also factored in "availability" along with quality. For example, some participants avoided selecting candidates who they felt would be overqualified. For example, P12 commented *"He is a creator of a language and his email address is a company name. He probably owns that company. I doubt that Jeremy is available. So I don't want to spend energy".*

What about interviews? Although participants felt reviewing the activity traces in GitHub and Stack Overflow was important to determine candidates' expertise, they mentioned that evaluating contribution traces in these communities cannot replace interviews. For example, P12 after selecting the top two candidates commented, *"I prefer to talk to them [candidates] next".*

Participants were comfortable evaluating coding competency and motivation from the online traces, but wanted to talk with the candidates to evaluate collaboration and management skills. For example P18 commented, *"I cannot judge [collaboration skills] from here".* and P17 mentioned, *"I don't know how can GitHub or Stack Overflow data help on [assessing] management skills".* This is inline with prior work that found interviews help assess qualities such as integrity, personality, emotional intelligence, capacity as a team player, empathy, and cross-cultural awareness [43–48]. These qualities can rarely be captured via data from online communities.

The role of Visual Resume, therefore, is not to be a substitute for interviews. Instead it aims to help interviewers easily digest information currently fragmented across different sites and types of data to make an initial assessment of candidates. As P19 commented: *"This tool is great for filtering out unwanted candidates and find top candidates who can continue to the interview step: a screen through for interview".* Therefore, its important that interviewers recognize this fact and not soleley depend on aggregator tools such as Visual Resume.

What about behavioral adaptation? Visual Resume aggregates and presents contribution data from two different types of online technical communities. As with any dashboard it highlights a handful of metrics (contributions and proxies of their quality) from these communities. But highlighting metrics can lead to behavioral adaptations [49]. First, individuals can self-monitor their behavior or compare their contributions with others', which might foster competition and in worst cases feelings of inadequacies. The latter may especially be a problem for newcomers or those with lower self-efficacy.

Second, the metrics being showcased may prompt users to change their contributions to better suit what the system can reliably track. For example, Visual Resume does not currently track code reviews, which may prompt people to not participate in this really important activity. Similarly, editing questions in Stack Overflow is another important activity that is not currently tracked. Of course, in future versions of Visual Resume, we can add these as metrics, but there still will remain other difficult to track, but nevertheless important activities such as mentoring or managing a project.

Finally, individuals may optimize for the tracked parameter rather than the underlying concept, leading to a kind of 'cheating'. For example, since the number of commits is a tracked metric, contributors may make small, but numerous commits. This kind of gamification might already be a challenge with sites like GitHub [50,51] that display contribution charts or Stack Overflow that tracks reputation points [52].

Challenges with behavioral adaptations are problems that afflict dashboards and other mining related approaches. Overcoming these challenges requires individuals (developers, managers and hirers) to recognize the dark side of metrics—that metrics present only certain facets of an individual, can be gamified, and does not alone define a person (or their contributions).

What about excluding underrepresented populations? Visual Resume aggregates and presents the contribution traces (metrics, tags, and statistics summary) captured in peer production sites, which are often used by developers [10] and hirers [2]. However, the information in such peer production sites has intrinsic societal biases such as, low gender and non-English speaking diversity. Research has found that women: (1) newcomers (who have gendered profiles) have low acceptance of pull requests in OSS projects [53], (2) are more active when they have peer-parity [54], and (3) abstain from using gamification and gamified elements (such as badges or reputation points) [55,56]. Similarly, past work has found that non-English speaking developers are often inactive on platforms like Stack Overflow while being active on their native language platforms equivalent of Stack Overflow [57].

Therefore, peer production sites, as well as aggregator tools like Visual Resume, need to pay particular attention in finding ways to minimize reinforcing the harmful effects of under representation of these populations. Visual Resume seeks to reduce unconscious biases by focusing on transparency in decision making as well as aggregating multiple types of contributions and highlighting contribution quality in addition to quantity. Future work on Visual Resume can do more to highlight community building activities such as, (1) code review efforts and review quality by extracting pull request comments, (2) maintenance activities in GitHub such as issue closing/editing, (3) maintenance activities in Stack Overflow such as cleaning questions or answers, and (4) community building activities such as OSS advocacy or event creation. Visual Resume can be easily extended to tap into other peer-production sites (e.g., GitLab or Stack Overflow sites in other languages) by adding a new "Data Extractor" component (Fig. 4). If the concept of Visual Resume gets widely adopted, we can envision individuals creating their own Visual Resume profiles that import their own traceable contribution histories (e.g., private code repositories). Another interesting future work is to investigate the impact of the demographics data embedded in the contributor profiles (picture, name, followers) on hiring decision making. Recent works have investigated the biases in how humans review code as a function of its apparent author [58] or social signals embedded in profiles [10]. A similar multi-factor user study on hiring decisions that controls the different profile elements can identify the impact of each of these factors and their interaction effects, which can then inform the design of future aggregator tools.

7.1. Implications

Tools like Visual Resume can be used by managers and software developers to hire prospective candidates in their teams and company. Further, it can be used by individual programmers for self-improvement by observing their progress and comparing with peers. Additionally, team members can track the online collaboration and communication traces as Visual Resume can be extended to monitor and then visualize these traces within the software teams. It can be also used by project Gatekeepers – individuals who are familiar with the team knowledge repository – and guide the information seekers to desired experts.

8. Related work

We identified six popular and freely available aggregator tools and sites. CVExplorer [9], Open Source Resume [8] and Statocat [11] create developers' profiles based on their activities in GitHub. Further, Statocat provides statistics of the programming languages used on GitHub. MasterBranch [12] and CoderWall [59] collect activities across several code hosting sites (e.g. GitHub and BitBucket). CoderWall awards achievement badges to developers, such as when the developer has a number of original repositories in a programming language. OpenHub generates developer profiles based on activities collected by their own code search engine (OpenHub Code search), and also awards achievement badges based on amount of activities. Here, we compare these aggregator sites with our tool based on the design guidelines listed in Section 3 (see Table 4).

DG1 — Provide cues for technical and soft skills: The majority of aggregator tools provide overviews of contributions that are typically inferred as technical skills. CVExplorer, MasterBranch, OSR, Statocat, and OpenHub [60] provide information about the numbers of commits, projects contributed to, and programming languages, but they do not display interaction histories for inferring soft skills (e.g. comments, answers, or questions). CoderWall, on the other hand, lists programming languages and project names in which the developer is interested in, but it provides no information on code artifacts.

DG2 — Provide cues for quality: Most tools provide links to project pages from the code hosting site but do not provide direct cues of quality. The users can manually investigate activities in the project to identify the source code they committed. Statocat and OSR simply link to the GitHub developer profile, where a user can investigate project contributions on their own.

DG3 — Present social standing in the community: CoderWall and OpenHub award achievement badges to developers who meet (site-specific) criteria based on the number of their contributions. Statocat and OSR display the number of followers to suggest their social standing. None of the tools provide social standing in the Q&A communities.

DG4 — Aggregate cues across projects and sites: CVExplorer, OSR and Statocat focus only on contributions in GitHub, whereas MasterBranch, CoderWall, and OpenHub create developer profiles that are generated by aggregating activities across multiple code sharing sites (See DG4 in Table 4). None of these tools aggregate provide information about contribution on both code hosting sites and Q&A forums.

DG5 — Summarize activity: CVExplorer lists the developer's skills based on the type of repositories that they contributed to. MasterBranch presents the total lines-of-code of contributions categorized per programming language. Similarly, OpenHub, OSR and Statocat provide statistics about the number of commits to a repository and also its programming language. Particularly, OSR lists the most recent user activity according to GitHub event log. Thus, these aggregators (as well as GitHub) are on par in summarizing code contributions.

DG6 — Visualize summaries: OpenHub displays commit summaries as bar charts grouped by projects, and stacked line charts grouped by languages. OSR and Statocat use pie charts to visualize developer contributions based on programming language. CVExplorer applies a wordcloud-like visualization to display the skillset of developer. However, CoderWall and MasterBranch do not provide any visualizations.

DG7 — Allow drill down: CoderWall, Statocat and OSR link to either developer profiles or repository pages on the code hosting sites, where users can further find out information details manually. Similar to Visual Resume OpenHub provide detailed statistics about the commits and repositories within its site. However, CVExplorer and MasterBranch do not provide the drilling down feature.

DG8 — Allow pairwise comparison: Majority of current profile aggregators do not provide any comparison functionality. OpenHub is the only tool that provides functionality to allow comparisons between projects, programming languages and repositories, but it is not geared for comparison of developers.

Table 4
Table of aggregator sites.

Project	Developer/ Project centered	DG1		DG2		DG3	DG4	DG5	DG6	DG7	DG8
		Tech skills	Soft skills	Code	Q&A						
Visual Resume	Developer	Yes	Yes	Direct	Direct	#GitHub followers, reputation points	GitHub, Stack Overflow	#Commits, #Issues, #Comments, #Q&A, Languages, Repositories	Language Commit, Issue, Project, Q&A	GitHub Profile, Stack Overflow Profile, Repository, Commit, Post	Yes
OpenHub	Developer/ Project	Yes	No	Indirect	None	OpenHub Badges	OpenHub Code Search	#Commits, #Projects, Languages	Commit Language	Repository, Commit	Yes
CoderWall	Developer	Yes	Yes	Indirect	Indirect	KARMA Point	GitHub, Stack Overflow, BitBucket, Codeplex	CoderWall-badges	None	Repository	No
CVExplorer	Developer/ Project	Yes	Yes	None	None	None	GitHub	Predefined Skillset	None	None	No
OSR	Developer	Yes	Yes	Indirect	None	#GitHub followers	GitHub	Languages, Repositories, GitHub Event	Commit, Language	Repository	No
Statocat	Developer	Yes	Yes	Indirect	None	#GitHub followers	GitHub	Languages, Repositories #Starred #Forked	Language	GitHub Profile	No
Master Branch	Developer	Yes	No	None	None	Rankings, Developer Score	Google Code, SourceForage, GitHub, Apache, Codeplex, BerliOS, Java.net	Languages	None	None	No

9. Conclusions

This paper presents Visual Resume, a contribution aggregator tool, designed based on the emerging needs of hiring practices. Visual Resume is built using a set of nine design guidelines spanning about what information to present and how to present it.

A scenario-based user study evaluators assessing job candidates revealed three key findings.

Cues Utilized for Assessing Candidates: Participants in both treatments focused on the amounts and type of contributions to first understand candidates' technical skills and then focus on soft skills. Cues to assess technical skills included candidates' coding competency and their contribution quality (e.g., commit details, the number of up-votes and stars, the number of popular projects associated or owned, the number of followers, reputation points). The metrics for soft skills were communication skills (e.g., amounts of contributions, community acceptance of contribution), social competency (e.g., endorsements), management skills (e.g., past project experience, the number of owned projects), and motivation (e.g., volume of activity, the number of up-votes and personal website).

Information Features Used: The information features used with and without Visual Resume were amounts of contributions, contribution history, commit aspects, and type of projects. When using Visual Resume participants had an easy access to candidate's profiles and related external webpages. Further, with Visual Resume they tended to use more information features and drill down into specifics of a contribution.

Information Accessed: Participants when using Visual Resume took half the number of actions to access information and utilized its built-in features to conduct side-by-side comparisons of candidates.

Our results suggest that an aggregator built using these design guidelines is effective beyond conventional approaches and requires less cognitive effort. Further, Visual Resume can help developers, recruiters, and managers evaluate developers' skills and contributions through multiple cues before embarking on interviews.

CRediT authorship contribution statement

Sandeep Kaur Kuttal: Conceptualization, Formal analysis, Validation, Investigation, Writing - original draft, Writing - review & editing. **Xiaofan Chen:** Methodology, Software, Conceptualization. **Zhendong Wang:** Data curation, Visualization, Investigation, Writing - original draft. **Sogol Balali:** Data curation, Writing - original draft. **Anita Sarma:** Supervision, Conceptualization, Writing - review & editing.

Declaration of competing interest

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

Acknowledgments

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-21-1-0108 and National Science Foundation 1815486 and 2008089. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the Air Force Office of Scientific Research and National Science Foundation.

References

[1] J. Resig, 2011, https://twitter.com/jeresig/status/33968704983138304/ (Retrieved May-2017).

[2] L. Singer, F. Figueira Filho, B. Cleary, C. Treude, M.-A. Storey, K. Schneider, Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators, in: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, ACM, 2013, pp. 103–116.

[3] D. Terdiman, Forget linkedin: Companies turn to github to find tech talent, 2012, https://www.cnet.com/news/forget-linkedin-companies-turn-to-github-to-find-tech-talent/ (Retrieved May-2017).

[4] J. Marlow, L. Dabbish, Activity traces and signals in software developer recruitment and hiring, in: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, ACM, 2013, pp. 145–156.

[5] J. Marlow, L. Dabbish, J. Herbsleb, Impression formation in online peer production: activity traces and personal profiles in github, in: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, ACM, 2013, pp. 117–128.

[6] G.J. Greene, B. Fischer, Cvexplorer: Identifying candidate developers by mining and exploring their open source contributions, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, in: ASE 2016, Association for Computing Machinery, New York, NY, USA, 2016, pp. 804–809, http://dx.doi.org/10.1145/2970276.2970285.

[7] C. Zhou, S.K. Kuttal, I. Ahmed, What makes a good developer? An empirical study of developers' technical and social competencies, in: 2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 2018, pp. 319–321.

[8] T. Jaruchotrattanasakul, X. Yang, E. Makihara, K. Fujiwara, H. Iida, Open source resume (OSR): A visualization tool for presenting oss biographies of developers, in: 2016 7th International Workshop on Empirical Software Engineering in Practice (IWESEP), 2016, pp. 57–62.

[9] Z. Wang, H. Sun, Y. Fu, L. Ye, Recommending crowdsourced software developers in consideration of skill improvement, in: 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), 2017, pp. 717–722.

[10] D. Ford, M. Behroozi, A. Serebrenik, C. Parnin, Beyond the code itself: how programmers *really* look at pull requests, in: R. Kazman, L. Pasquale (Eds.), Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Society, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019, ACM, 2019, pp. 51–60, http://dx.doi.org/10.1109/ICSE-SEIS.2019.00014.

[11] Statocat, 2017, http://www.upsingapore.com/ideas/statocat/ (Retrieved May-2017).

[12] MasterBranch, 2017, https://masterbranch.com (Retrieved May-2017).

[13] A. Sarma, X. Chen, S. Kuttal, L. Dabbish, Z. Wang, Hiring in the global stage: Profiles of online contributions, in: Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on, IEEE, 2016, pp. 1–10.

[14] B. Al-Ani, D. Redmiles, In strangers we trust? Findings of an empirical study of distributed teams, in: 2009 Fourth IEEE International Conference on Global Software Engineering, 2009, pp. 121–130.

[15] F.Q. da Silva, C. Costa, A.C.C. Franca, R. Prikladinicki, Challenges and solutions in distributed software development project management: A systematic literature review, in: Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on, IEEE, 2010, pp. 87–96.

[16] J. Noll, S. Beecham, I. Richardson, Global software development and collaboration: barriers and solutions, ACM Inroads 1 (3) (2010) 66–78.

[17] L. Dabbish, C. Stuart, J. Tsay, J. Herbsleb, Social coding in github: transparency and collaboration in an open software repository, in: Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, ACM, 2012, pp. 1277–1286.

[18] B. Al-Ani, M.J. Bietz, Y. Wang, E. Trainer, B. Koehne, S. Marczak, D. Redmiles, R. Prikladnicki, Globally distributed system developers: their trust expectations and processes, in: Proceedings of the 2013 Conference on Computer Supported Cooperative Work, ACM, 2013, pp. 563–574.

[19] R. Sabherwal, The role of trust in outsourced IS development projects, Commun. ACM 42 (2) (1999) 80–86.

[20] J. Long, Open source software development experiences on the students' resumes: Do they count?-insights from the employers' perspectives, J. Inf. Technol. Educ.: Res. 8 (1) (2009) 229–242.

[21] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, 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, ACM, 2013, pp. 886–893.

[22] J. Tsay, L. Dabbish, J. Herbsleb, Influence of social and technical factors for evaluating contribution in github, in: Proceedings of the 36th International Conference on Software Engineering, ACM, 2014, pp. 356–366.

[23] C. Gutwin, R. Penner, K. Schneider, Group awareness in distributed software development, in: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, ACM, 2004, pp. 72–81.

[24] M. Zhou, A. Mockus, What make long term contributors: Willingness and opportunity in oss community, in: Software Engineering (ICSE), 2012 34th International Conference on, IEEE, 2012, pp. 518–528.

[25] Upper Rio Grande, What skills do employers want - Workforce solutions, 2017, https://www.yumpu.com/en/document/view/51546223/what-skills-do-employers-want-workforce-solutions-upper-rio-/6 (Retrieved May-2017).

[26] J. Tsay, L. Dabbish, J.D. Herbsleb, Social media in transparent work environments, in: Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on, IEEE, 2013, pp. 65–72.

[27] C.-G. Wu, J.H. Gerlach, C.E. Young, An empirical analysis of open source software developers' motivations and continuance intentions, Inf. Manage. 44 (3) (2007) 253–262.

[28] G. Stephan, A. Pahnke, A pairwise comparison of the effectiveness of selected active labour market programmes in Germany, 2008.

[29] Gitto, 2017, https://gitto.io (Retrieved May-2017).

[30] K.J. Jenne, M. Henderson, Hiring a director for a nonprofit agency: A step-by-step guide, Pop. Gov. (2000) 25.

[31] Visual-resume, 2020, https://github.com/asarma/Visual-Resume (Retrieved May-2020).

[32] N. Ducheneaut, Socialization in an open source software community: A socio-technical analysis, Comput. Support. Coop. Work (CSCW) 14 (4) (2005) 323–368.

[33] C. Jergensen, A. Sarma, P. Wagstrom, The onion patch: migration in open source ecosystems, in: Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, 2011, pp. 70–80.

[34] P. Bonacich, Power and centrality: A family of measures, Amer. J. Sociol. 92 (5) (1987) 1170–1182.

[35] B. Vasilescu, A. Serebrenik, P. Devanbu, V. Filkov, How social q&a sites are changing knowledge sharing in open source software communities, in: Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, ACM, 2014, pp. 342–354.

[36] Neo4j, 2017, http://www.neo4j.org (Retrieved May-2017).

[37] D3.js, 2017, http://d3js.org (Retrieved May-2017).

[38] Careers 2.0, 2017, https://careers.stackoverflow.com/ (Retrieved May-2017).

[39] CareerBuilder, 2017, http://www.careerbuilder.com/ (Retrieved May-2017).

[40] D. Damian, L. Izquierdo, J. Singer, I. Kwan, Awareness in the wild: Why communication breakdowns occur, in: Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on, IEEE, 2007, pp. 81–90.

[41] E. Kandogan, Hierarchical multi-window management with elastic layout dynamics, 1999.

[42] J. Nielsen, Utilize available screen space, 2019, https://www.nngroup.com/articles/utilize-available-screen-space/ (Retrieved Nov-2019).

[43] M.C. Blackman, D.C. Funder, Effective interview practices for accurately assessing counterproductive traits, Int. J. Sel. Assess. 10 (1–2) (2002) 109–116.

[44] J.F. Salgado, S. Moscoso, Comprehensive meta-analysis of the construct validity of the employment interview, Eur. J. Work Organ. Psychol. 11 (3) (2002) 299–324.

[45] C. Sue-Chan, G.P. Latham, The relative effectiveness of external, peer, and self-coaches, Appl. Psychol. 53 (2) (2004) 260–278.

[46] T.D. Allen, J.D. Facteau, C.L. Facteau, Structured interviewing for OCB: Construct validity, faking, and the effects of question type, Hum. Perform. 17 (1) (2004) 1–24.

[47] C. Cliffordson, Interviewer agreement in the judgement of empathy in selection interviews, Int. J. Sel. Assess. 10 (3) (2002) 198–205.

[48] F. Lievens, S. Highhouse, The relation of instrumental and symbolic attributes to a company's attractiveness as an employer, Pers. Psychol. 56 (1) (2003) 75–102.

[49] E.T. van Dijk, F. Beute, J.H. Westerink, W.A. Ijsselsteijn, Unintended effects of self-tracking, in: CHI'15, April 18–April 23, 2015, Seoul, South-Korea. Workshop on 'beyond Personal Informatics: Designing for Experiences of Data', 2015, p. 5.

[50] L. Moldon, M. Strohmaier, J. Wachs, How gamification affects software developers: Cautionary evidence from a quasi-experiment on github, 2020, arXiv:2006.02371.

[51] L. Singer, K. Schneider, It was a bit of a race: Gamification of version control, in: Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques (GAS), 2012, pp. 5–8.

[52] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, B. Hartmann, Design lessons from the fastest q&a site in the west, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, in: CHI '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 2857–2866.

[53] J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, J. Stallings, Gender differences and bias in open source: Pull request acceptance of women versus men, PeerJ Prepr. 4 (2016) e1733v2, http://dx.doi.org/10.7287/peerj.preprints.1733v2.

[54] D. Ford, A. Harkins, C. Parnin, Someone like me: How does peer parity influence participation of women on stack overflow? in: A.Z. Henley, P. Rogers, A. Sarma (Eds.), 2017 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2017, Raleigh, NC, USA, October 11-14, 2017, IEEE Computer Society, 2017, pp. 239–243, http://dx.doi.org/10.1109/VLHCC.2017.8103473.

[55] A. May, J. Wachs, A. Hannák, Gender differences in participation and reward on stack overflow, Empir. Softw. Eng. (2019) 1–23.

[56] B. Vasilescu, A. Capiluppi, A. Serebrenik, Gender, representation and online participation: A quantitative study of stackoverflow, in: Proceedings of the 2012 ASE International Conference on Social Informatics, SocialInformatics 2012, 2012, http://dx.doi.org/10.1109/SocialInformatics.2012.81.

[57] N. Oliveira, M. Muller, N. Andrade, K. Reinecke, The exchange in stackexchange: Divergences between stack overflow and its culturally diverse participants, Proc. ACM Hum.-Comput. Interact. 2 (2018) 1–22, http://dx.doi.org/10.1145/3274399.

[58] Y. Huang, K. Leach, Z. Sharafi, N. McKay, T. Santander, W. Weimer, Biases and differences in code review using medical imaging and eye-tracking: Genders, humans, and machines, in: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020, 2020, pp. 456–468.

[59] CoderWall, 2017, https://coderwall.com/ (Retrieved May-2017).

[60] OpenHub, 2017, https://www.openhub.net (Retrieved May-2017).