Development of a Metric for Assessing Full-Stack Developers’ Expertise

Jordan Shropshire  
Professor  
Department of IST  
University of South Alabama  
jshropshire@southalabama.edu

Jeffrey P. Landry  
Professor  
Department of IST  
University of South Alabama  
jlandry@southalabama.edu

Steven S. Presley  
Doctoral Student  
Department of Information Systems and Technology (IST)  
University of South Alabama  
ssp1521@jagmail.southalabama.edu

Abstract

Full-stack development is a broadly-defined concept. Full-stack developers have a solid breadth of knowledge across a given tech stack and deeper experience in one or two particular areas. Organizations often seek full-stack developers with expertise in a specific stack layer. However, it is difficult to communicate skillset specifics between parties. This results in mismatches between developers and projects. Hence, this research introduces a metric for improving skillset communication between full-stack developers and project managers. The metric allows full-stack developers to allocate a fixed quantum of expertise points across stack layers. It forces them to distinguish their areas of expertise from areas in which they are less knowledgeable. A test was conducted to determine if the metric results in improved skillset matches. Subjects matched full-stack developers with systems development projects in a series of tests. The results indicate the metric results in better skillset matches than resumes alone.

Keywords

Full-stack development, developer, recruiting, assessment, hiring, self-assessment

Introduction

Full-stack development is an emerging approach to systems development (Cartwright 2015; Chen 2013). This concept focuses on developers whose expertise spans an entire tech stack (Shandling 2014; Westberg 2015; Yonatan 2016). Full-stack developers have a big-picture appreciation of the problem space. They have a solid understanding of each layer in a tech stack and deeper expertise in one or two specific areas. Full-stack developers interpret business proposals to design, build, and implement full-stack solutions (Bueno 2010). They can help launch minimum-viable implementations of new concepts to validate them.

Organizations tend to prize full-stack developers, not only because they ensure cross-layer functionality but also because they can execute the work of multiple specialists (Gellert 2012). As a result, full-stack developers typically command above-market salaries and benefits. Seizing on the opportunity to secure positions with better pay and more in-job flexibility, many developers are billing themselves as full-stack. Although many of these individuals have a broad range of knowledge across a stack, they have different
strengths and weaknesses. This makes it difficult to predict which skills a full-stack developer can contribute.

Much of the confusion surrounding full-stack development stems from its broad definition. Two individuals calling themselves full-stack developers may have significantly different skillsets and depths of mastery (Gellert 2012; Rothwell 2017). For a given stack, many individuals will have the same general understanding of the foundational technologies but different areas of deeper expertise (Westberg 2015). For instance, some full-stack developers are more familiar with front end tools while others gravitate toward the business logic layer. This makes it difficult to predict full-stack developers’ breadth and depth of stack expertise without any background knowledge (Wiggins 2017).

Likewise, various full-stack development projects may have different focal areas. By definition, full-stack projects are distributed across a stack. However, some projects tend to focus more on certain layers over others (Cartwright 2015). For instance, one full-stack development project may be data-centric while another focuses on user experience. Both projects can be classified as full-stack development projects, even though they have different emphasis areas. This makes it difficult to determine if a full-stack developer possesses the expertise profile needed for a particular project (Loukides 2014). This results in mismatches between full-stack developers and software development projects (Shiotsu 2016).

An interviewer could ask a job seeker to delineate his or her strengths and weaknesses across the stack. However, the desire to present oneself as the best choice for a position would introduce self-rating biases (Dunning et al. 2004). Interviewers want to highlight their strengths and downplay their weaknesses. The lack of candor contributes to the problem space. Furthermore, the organization is apt to describe the ideal candidate as someone with deep expertise in all stack layers, even though they may only need a developer with specific strengths. This discourages openness on the part of the candidate and further complicates the problem.

The purpose of this research is to develop and validate a new metric for improving skillset communication between full-stack developers and hiring organizations. The metric allows full-stack developers to rate their own expertise at each layer of a given stack. It provides a fixed quantum of expertise points across stack layers. This means it is not possible to give oneself the highest rating for every stack layer. Because there is a limited number of points developers are forced distinguish their areas of expertise from areas in which they are less knowledgeable. In doing so, it circumvents personal biases and provides more clarity into full-stack developers’ skillsets. This provides a relative comparison of developers’ skills. The metric will facilitate skillset communication between full-stack developers and hiring companies.

The remainder of this research is organized as follows: the following section is the background. It introduces the concept of full-stack development and describes the theory of relative assessment. The following section is the conceptual development. It introduces the proposed metric and describes its use cases. The next section describes the initial evaluation of the metric. It outlines the subjects, procedure, and measures. The following section is the results section. The results and implications of testing are noted. The final section provides concluding comments and directions for future research.

**Background**

*Technology Stack*

A “tech stack” or technology stack may also be referred to as a software stack or simply called a stack (Charland et al. 2011). Traditionally, most technology stacks consist of a server operating system, web server, database server, and programming language (Shandling 2014). They may be expanded to include additional components such as development frameworks and client side software (Bryksin 2017) (see Figure 1). It should be noted that not all stack’ components will be vertically integrated. Some may interconnect laterally. The purpose of a tech stack is to render a networked service to geographically distributed clients (Edelman 2015). The service is usually delivered via some combination of applications which provide dynamic output to clients (Gregor et al. 2013). A web server and a database are considered integral to most tech stacks. The bulk of the data processing may be performed on the server-side, the client-side, or distributed evenly between both points.
Some organizations may rely on a single tech stack to deliver a variety of enterprise services (Charland et al. 2011). Others implement multiple stacks, using them to deliver different services. Although a number of standard tech stacks exist, it is not necessary to follow any conventions when selecting the software for each layer (Maake 2016). It is possible to pick and choose software based on expected needs, in-house expertise, and client input (Dingsøy et al. 2012). In fact, some organizations create tech stacks which are highly customized to meet their own needs. The most popular software for each layer varies with time. Figure 2 (below) depicts trending software for various tech stack layers. This data is summarized from a ranking study conducted by stackshare.io. describes some of the most common tech stacks. Each layer of these stacks builds on the features below it, creating a vertical conglomeration. Most include a development framework. These frameworks consist of tools which provide developers with vetted implementations of common web and server-side application features like user authentication and data access (Ghiucu 2014). This saves developers from having to re-recreate common, low-value features and allows them to focus on developing a minimum viable implementation of a new service or product.

Relative assessments

People are notoriously bad judges of their own ability (Dunning et al. 2004). Exaggerated perceptions of mastery are common attributes of normal people (Harris et al. 1988). Recent research has shown that people are systematically flawed judges of their own standing. They tend to rate their skills too low when a task seems difficult and too high when a task seems easy. One consequence of this phenomenon is that people will incorrectly report their abilities on job applications and interviews (Epley et al. 2000). This leads
to a misfit between the work at hand and the worker’s ability. In order to address this known deficiency, it may be necessary to restate the task as a series of relative comparisons.

A relative assessment is “unit-less” analysis. This concept is also known as relative sizing. It is based on comparisons with equivalents to get a sense of scale. In other terms, it is a way of sizing values relative to each other. This concept is widely used and adopted in the agile software development methodologies for a number of estimates (Radigan 2016). For instance, it is used to assess scope, estimate time requirements, gauge backlog, estimate difficulty, and determine required effort. This approach has a number of benefits over traditional, unit-based estimates (Radigan 2018).

The benefits are significant for assessing developer capabilities. Relative assessments are quick. They allow developers to make estimations when there is still much uncertainty and not much time. Further, they are more accurate in some respects. Precise metrics are difficult to quantify if performed in isolation. There is a chance the estimate will be off because there is no scale. Additionally, relative assessments get better with time. The more an individual makes relative assessments, the better they become. With added practice they get better at interpreting data and making comparisons. Finally relative assessments are context specific. Generalizing estimations to a specific context lends credence and velocity (Balbous 2017).

Outside of agile software development, relative assessment is used in marketing to gauge consumer sentiment. A long-held marketing truism is that consumers evaluate products not just on their own inherent properties, but also on where they stand relative to other products (Burson 2007). Consumers rely on relative assessments to compare the features of products against each other. For instance, prices might be compared. Or size, weight, volume, color, warranty, speed, or efficiency might be compared. These comparisons allow buyers to determine which product in a group has the best feature or features (Alba et al. 2000). Relative comparisons are also used in polling. Pollsters often ask respondents to indicate their satisfaction with a political office holder on a given subject compared to previous holders of the same office (Erikson et al. 2008). For instance, a likely voter may be asked to indicate if the current president is doing more to help the economy than the previous president. Such as comparison is necessary because there is no natural unit of measure for political satisfaction.

**Conceptual development**

As previously indicated, self-appraisals of skill inventories are unreliable. People are also subject to self-assessment biases (Harris et al. 1988). A self-assessment bias is the tendency to see oneself as superior to others. This phenomenon often occurs beyond the periphery of awareness. The forces that influence social behavior are complex. People often lack all the information needed to make accurate self-judgments. And even when people do have sufficient information to render an accurate self-assessment, they often neglect the information. This leads them to make worse assessments than they are capable of (Dunning et al. 2004). Finally, many job seekers overrate themselves to stand out among their peers. This makes it difficult to match candidates’ skillsets with project needs.

In short, resumes are not always the best way to gauge a technical workers’ skills, experience, and knowledge. Other, newer approaches to capturing and summarizing expertise are needed. Therefore, we propose a new metric for self-assessment of technical expertise.

This metric will improve and standardize communication between full-stack developers and hiring organizations. The metric presents a complimentary set of skill domains. It encourages individuals to make relative comparisons of their skills. For the present study, the skill domains are layers in a given tech stack. For instance, the layers in a LAMP tech stack include Linux, Apache Server, MySQL, and PHP. The layers are integral components of the tech stack. Each is necessary to rendering a dynamic, data-driven website.

For this metric, each domain becomes an axis radiating from a central node in a hub-and-spoke graph. Individuals rate their strength in domain on a scale from 1 to 10. A 1 corresponds with little or no expertise. A 10 corresponds with deepest level of expertise. The metric allows full-stack developers to rate their own expertise at each layer of a given stack. In doing so, it circumvents personal biases and provides more clarity into full-stack developers’ skillsets. It forces them determine which domains are their strengths and weaknesses. This provides a relative comparison of developers’ skills. An example metric is shown in Figure 3 (below).
To force respondents to make relative comparisons of their skills, a fixed quantum of expertise points is provided. These points must be allocated across stack layers. The number of point is significantly less than $10 \times$ the number of axis. It is not possible to give oneself the highest rating for every stack layer. Because there is a limited number of points, developers are forced distinguish their areas of expertise from areas in which they are less knowledgeable.

![Expertise Metric for LAMP Stack](image)

**Figure 3: Expertise Metric for LAMP Stack**

In order to select an appropriate number, two issues must be reconciled. First, it can be observed that the lower the number of expertise points, the greater the distinction between strengths and weaknesses. However, it should be noted that an excessively low number of points may encourage people to give themselves an equal number of points across the board. A number between 50% and 70% of the total points across axes is recommended. The number of points should also not be cleanly divisible by the number of axes. This will prevent respondents from evenly allocating points across the board. Hence, the following hypothesis is proposed:

H1: *Use of the proposed metric will result in better matches than use of resumes*

Using the proposed metric, it will be easier to assess a developer’s expertise across a given tech stack. This metric can also be used by organizations require full stack developers with strengths in a given stack layer. Furthermore, this metric is language neutral. It can easily be translated across borders and allow for level comparisons.

**Methods**

**Procedure**

A test was conducted to evaluate the hypothesis. The purpose of the test is to determine if use of the proposed metric results in better matches between open positions in full stack development projects and job seekers than use of resumes. This study focused on full stack development positions for the MEAN tech stack. The MEAN tech stack consists of MongoDB, Express, Angular, and NodeJS.

For this test, half of the participants were asked to match developers with open positions using resumes and job descriptions. The other half was asked to use completed expertise metrics instead of resumes. For each position, the participants were asked to give the name of the most appropriate candidate. Responses were scored on basis of whether they suggested the right candidate for a position. Individuals who came up with a higher percentage of matches were said to have done better than those with lower match rates.
Subjects

Subjects participated in the study via Qualtrics. Subjects were recruited using Amazon Mechanical Turk (MTurk). Respondents were recruited based on being currently employed in the computing field and having managed at least 3 employees for one year. In total, we recruited 150 technology workers to participate in the study. After following previously validated data cleansing guidelines for MTurk participants (Crump et al., 2013; Steelman et al., 2014), we achieved a final sample size of 102. This sample was randomly divided into two groups. Group 1 used the traditional data (resumes and job descriptions) while Group 2 used the job descriptions and expertise metrics. Table 1, below, provides a demographic summary of the respondents.

<table>
<thead>
<tr>
<th>Total number of respondents</th>
<th>102 persons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td>73% male; 27% female</td>
</tr>
<tr>
<td>Average Age</td>
<td>27.5 years old</td>
</tr>
<tr>
<td>Years Management Experience</td>
<td>3.1 years</td>
</tr>
<tr>
<td>Number of Employees Hired</td>
<td>2 employees</td>
</tr>
<tr>
<td>Number of Employees under Management</td>
<td>4 employees</td>
</tr>
</tbody>
</table>

Table 1: Demographics

Measures

Resumes of full-stack developers were obtained from public LinkedIn profiles. The developers were all specialists in the MEAN stack. The developers were contacted and asked to evaluate themselves. A web-based version of the metric was created in JavaScript for each tech stack. Some 16 developers complied with the request and completed the metric. Interestingly, none of the completed expertise metrics were identical. Each developer submitted slightly different point allocations. Additionally, 10 job postings for full-stack developers were identified on the same social media platform. The job postings were all for companies that are hiring MEAN stack developers. These job descriptions were carefully selected to ensure that they each require a different strength. The companies were contacted and asked if the hiring manager would evaluate the position’s requirements along the axes of the proposed metric. Some six companies replied.

The 16 resumes, six job listings, and all of the metrics were supplied to an expert panel of three reviewers. Each reviewer had an average of seven years’ experience in IT management and had hired and managed several IT workers in the recent past. They were also familiar with the MEAN technology stack. The review panel was asked to identify the most suitable candidate for each position based on the available information. The panel conferred and named the best candidates for each of the six positions. This was used as a score card to ascertain how well participants matched full-stack developers with open positions.

Results

After the results of the 102 subjects were collected an analysis was performed. A one-tailed t-test of significant differences was performed to determine if use of the expertise metric improves candidate selection. For this t-test, the null hypothesis is that there is no difference in the number of matches. The alternative hypothesis is that use of the expertise metric results in more matches than use of traditional resumes. Group 1 is the control group. It uses the default method of reviewing developers’ self-reported skill sets. Group 2 is the treatment group. It contains the subjects who used the expertise metric to match developers with positions. As indicated above, there is a sufficiently large population to assume a normal distribution. The mean scores for each group were calculated. The mean score for Group 1 is 3.2 with a standard deviation of .3. The mean score for Group 2 is 4.6 and its standard deviation is .6. See Table 2 (below) for descriptive statistics.

<table>
<thead>
<tr>
<th></th>
<th>N</th>
<th>Mean</th>
<th>St Deviation</th>
<th>SE mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>51</td>
<td>3.2</td>
<td>0.3</td>
<td>.084</td>
</tr>
<tr>
<td>Group 2</td>
<td>51</td>
<td>4.6</td>
<td>0.6</td>
<td>.042</td>
</tr>
</tbody>
</table>

Table 2: Descriptive Statistics
The differences were compared using a t-statistic. Differences are inferred at a .05 level of significance. The t statistic was 14.9054. The P value was .000236, which indicates a significant difference at the .0001 level of confidence. This indicates that there is enough evidence to reject the null hypothesis. It is confirmed that use of the expertise metric results in significantly better matches than simply perusing resumes.

The implication of this finding is that self-assessment of expertise give more valuable insight in developer strengths and weaknesses than resumes. It is expected that most organizations would choose to look at both data sources, which might prove optimal in a future study.

This test indicates that the proposed metric is better than simply perusing resumes. However, it does not evaluate the efficacy of the metric against less naive approaches to candidate evaluation. In future studies, it makes sense to compare to the proposed method (relative assessments) against methods which use absolute assessments with unlimited points to allocate. This might underscore the self-assessment bias which drives candidates to overstate their skills across the stack and further validate the proposed approach. It would also be useful to pinpoint the value of the limited-points features by comparing it against a method which allows candidates to rank-order their strengths and weaknesses.

**Conclusion**

Full-stack development is a broad concept. This leads to confusion regarding software developer skillset. For a given tech stack, two full-stack developers may have the same basic knowledge of stack components but different areas of expertise within the stack. Likewise, two full-stack development projects may cover the same tech stack but emphasize different stack layers. This leads to mismatches between full-stack developers and software development projects. This research addressed this issue by proposing a new metric to facilitate skillset communication between full-stack developers and hiring organizations. The proposed metric forces developers to indicate their relative strengths and weaknesses within a stack. The proposed metric was evaluated using a series of simulations. The results indicate the metric improves the skillset-project fit. It is expected to increase hiring managers' confidence and simplify the decision-making process. The proposed metric was developed in consideration of the full-stack development field. However, it lays valid claim to any area in which it is necessary to differentiate candidates based on their portfolio of strengths and weaknesses.

Although the proposed metric was validated, it should be noted the metric alone is not suitable for making hiring decisions. There is much more to being a good software developer than having technical familiarity given technologies. Thus, it is recommended that the metric be treated as one tool for screening candidate prior to interviewing them. The next step is to assess the utility of the metric in actual hiring scenarios. Another possible research direction is the automation of the proposed metric: could the metric be populated using techniques such as natural language processing and the data from candidate resumes? Future research should address these areas.

**REFERENCES**


