Which Factors Influence Practitioners' Usage of Build Automation Tools?

Akond Rahman
North Carolina State University
Raleigh, NC, USA
aarahman@ncsu.edu

Asif Partho
NestedApps
Dhaka, Bangladesh
asif@nestedapps.com

David Meder
Red Hat
Raleigh, NC, USA
dmeder@redhat.com

Laurie Williams
North Carolina State University
Raleigh, NC, USA
williams@csc.ncsu.edu

***Abstract*—Even though build automation tools help to reduce errors and rapid releases of software changes, use of build automation tools is not widespread amongst software practitioners. Software practitioners perceive build automation tools as complex, which can hinder the adoption of these tools. How well founded such perception is, can be determined by systematic exploration of adoption factors that influence usage of build automation tools. *The goal of this paper is to aid software practitioners in increasing their usage of build automation tools by identifying the adoption factors that influence usage of these tools.* We conducted a survey to empirically identify the adoption factors that influence usage of build automation tools. We obtained survey responses from 268 software professionals who work at NestedApps, Red Hat, as well as contribute to open source software. We observe that adoption factors related to complexity do not have the strongest influence on usage of build automation tools. Instead, we observe compatibility-related adoption factors, such as adjustment with existing tools, and adjustment with practitioner's existing workflow, to have influence on usage of build automation tools with greater importance. Findings from our paper suggest that usage of build automation tools might increase if: build automation tools fit well with practitioners' existing workflow and tool usage; and usage of build automation tools are made more visible among practitioners' peers.**

Keywords-build automation tools; survey; practitioners

I. INTRODUCTION

Use of non-automated build tools can have negative consequences on delivering software changes to end-users. For example, prior to 2012, LinkedIn experienced failures in releasing software changes using non-automated build tools and techniques [1]. In 2012, Knight Capital Group, a financial firm worth $460 million, went bankrupt in 45 minutes by performing 4 million erroneous transactions [2], [3]. This failure was attributed to their use of non-automated tools and techniques for software build and delivery [2], [3], [4]. Despite evidence of such negative consequences of using non-automated tools and techniques for building software, software professionals are hesitant to adopt build automation tools [5]. Some professionals have attributed this hesitancy to the complex nature of build automation tools [6]. Understanding whether or not this attribution is well founded, requires systematic identification of adoption factors that influence usage of build automation tools.

The goal of this paper is to aid software practitioners in increasing their usage of build automation tools by identifying the adoption factors that influence usage of these tools.

In our research study, we collected and analyzed survey responses from software practitioners to identify the adoption factors. Identified adoption factors from our research study might help software practitioners to take appropriate actions to increase the usage of build automation tools. As a hypothetical example, if empirical analysis from our research study provides evidence of compatibility being an influential adoption factor for usage of build automation tools then toolsmiths can design build automation tools such that the tools fit well with the existing workflow of software practitioners.

Our research study focuses on tools related to build automation, and we consider four types of tools that constitute a build automation process [7]. We refer to these tools as build automation tools. These tools are: build (B) tools, such as Ant or Maven; continuous integration (CI) tools, such as Bamboo or Jenkins; infrastructure as code (IaC) tools, such as Chef or Puppet; and version control (VC) tools, such as Git or Mercurial. Each of these tool types has different purposes, necessitating individual analysis for each type of tool. For each type of build automation tool, we investigate what adoption factors influence the usage of that particular tool. We state the research questions as following:

RQ-1: Which adoption factors influence usage of B, CI, IaC, and VC tools?

RQ-2: How can we prioritize the identified influencing adoption factors for B, CI, IaC, and VC tools?

In our research study, we used adoption factors that belong to six factor groups: advantages, compatibility, complexity, education, observability, and trialability. Prior work has used these adoption factors to explain adoption of technologies in multiple domains, including software engineering [8], [9]. We conducted a survey to collect quantitative data from software practitioners working in NestedApps, Red Hat, and open source contributors. We analyzed the collected survey responses from 268 software professionals, to answer the two research questions.

We list our contributions in this paper as following:

- A list of adoption factors that influence usage of B, CI, IaC, and VC tools; and
- A rank of the identified adoption factors that influence usage of B, CI, IaC, and VC tools

We organize rest of the paper as following: in Section II we provide background concepts and related work. Section III illustrates our methodology in details. We use Section IV to present our findings. In Section V we discuss our findings.

978-1-5386-0428-1/17 $31.00 © 2017 IEEE
DOI 10.1109/RCoSE.2017..8

We describe the limitations of the research study in Section VI. Finally we conclude the paper in Section VII.

II. Background and Related Work

We use this section to briefly describe the related academic work as well as the background concepts needed for this research study.

A. Related Work

Our research study is closely related to two types of prior academic studies: prior work that focuses on build automation as a technology, and prior work that discusses usage of tools related to software engineering. We briefly discuss these prior academic works as below:

Humble and Farley stated use of build automation technology as an integral pillar to achieve continuous delivery and continuous deployment [7]. McIntosh et al. [10] studied the evolution of Java build systems in terms of complexity and coverage. In a recent work, Rahman et al. [11], through qualitative analysis of Internet artifacts, discussed how practitioners are using automation practices to achieve continuous deployment, and listed a set of tools and techniques to realize those practices. Our research study focuses on the adoption factors that influence usage of build automation tools.

Prior academic studies have discussed the adoption of software engineering-related tools. Murphy-Hill and Murphy [12] explored how practitioners in a software team are influenced in their discovery and use of software technologies by their peers. Johnson et al. [13] investigated reasons about why developers do not want to use static analysis tools through systematic investigation of developer interviews. Meyerovic and Rabkin [9] surveyed open source programmers to identify the factors that contribute to programmers' selection and usage of a programming language. Mosley [14] proposed six categories of questions to assess the usage of software engineering tools amongst software practitioners. Witschey et al. [8] performed a quantitative study using the Diffusion of Innovations (DOI) theory to investigate the factors that contribute to programmers' security tool usage. Raghavan and Chand [15] stated the importance of applying the DOI theory in software engineering, and used it to propose and evaluate multiple models for technology adoption in the domain of software engineering.

In this research project, we consider factors that belong to the theory of DOI, and can influence practitioners' usage of build automation tools. We selected DOI because of its usage in software engineering to explain adoption of technologies [8], [9].

B. Background

Humble and Farley defined build automation as the technology which automatically compiles and tests software changes, packages the software changes into a binary, and prepare the created binary for deployment [7]. In Section I we have stated the four types of build automation tools that we consider in our research study. We briefly define these four types of tools as following:

- Build (B) tools compile software changes into executable binaries. Example: Ant and Maven.
- Continuous integration (CI) tools integrate software changes into the shared mainline of the software. Example: Jenkins and Travis-CI.
- Infrastructure as Code (IaC) tools manage configuration options of the software as well as configure the deployment environment. Example: Chef and Puppet.
- Version control (VC) tools manage all the changes of artifacts related to the software of interest. Example: Git and Subversion (SVN).

The DOI theory explains why technologies and tools are adopted amongst end-users [16]. The DOI theory has five factor groups that are dependent upon the technology or tool of interest [16]. We describe these five factor groups as following:

- Relative Advantages: This factor group corresponds to the relative advantages that a certain tool can have over the other. For brevity, we refer to relative advantages as 'advantages' throughout the paper.
- Compatibility: This factor group accounts for the level of how a new technology is consistent with existing experiences and practices.
- Complexity: This factor group presents the level of difficulty of learning, using, and understanding a certain technology of interest.
- Observability: This factor group indicates how the results of a new tools' usage are visible to other tool users.
- Trialability: This factor group corresponds to the degree of experimentation that tool users can do within a limited basis.

As prior work [12], [9], [17], [8] in software engineering has provided evidence on how the learning process of end-users influences the discovery and usage of tools, we included the factor group of education. Education refers to how end-users learn a tool or technology [8].

Altogether, we consider six factor groups in our research study. Each of these six factor groups can be subdivided into a number of adoption factors [16]. For example, the factor group of complexity can be subdivided into the depth of knowledge required, the mental effort required to use that particular tool [8], or other adoption factors.

III. Methodology

We describe the steps of our research study as following:

A. Survey Design

We used a survey to collect practitioners responses related to adoption of build automation tools. An empty version of the deployed survey is available online[1]. The introduction of the survey included a brief explanation of the four types of tools, with appropriate examples. The survey

[1] https://ncsu.qualtrics.com/jfe/form/SV_dgMKDlk62hH3RMp

consisted of four parts. Each of the four parts had the same set of questions but focused on one of the types of tools namely B tools, CI tools, IaC tools, and VC tools. Practitioners might be familiar with a subset of the four tool types, and hence we provided options to select the tool type that the practitioners are familiar with. Based on their selection, the survey respondents were redirected to the corresponding survey page. For example, if a practitioner was only familiar with VC tools then he/she was redirected to the survey questionnaire related to VC tools. If a practitioner was familiar with all of the four types of tools then he/she was redirected to the survey questionnaire page that corresponds to all the four types of tools.

For each type of tool, we first asked the survey participants about their duration of experience in industry, as well as the duration of how long they have been using that particular tool. Then, we asked survey participants about their usage and the adoption factors that influenced their usage of that tool. We asked survey participants their usage of the tool in the following manner:

Which of the following statement describes the best for you?

- I frequently use <x> tools

- I occasionally use <x> tools

- I almost never use <x> tools

In the above question, <x> corresponds to the type of the tools for example, 'version control' to represent VC tools, and 'continuous integration' to represent CI tools. We chose a three-scale point to avoid scale points that might be ambiguous to the survey respondents, such as 'often', and 'most often' [18]. We refer to the above-mentioned question as 'usage question' for the rest of the paper.

Initially, we started with the complete set of 74 adoption factors stated in Witschey et al. [8]'s work, to identify the adoption factors that might influence usage of build automation tools. From the set of 74 factors we identified a specific subset of factors based on the following criteria:

- the factor must belong to any of the six factor groups used in our study: advantages, compatibility, complexity, education, observability, and trialability;
- the factor must have applicability for B tools;
- the factor must have applicability for CI tools;
- the factor must have applicability for IaC tools; and
- the factor must have applicability for VC tools

In our selection process of adoption factors, applicability refers to whether or not a certain factor can influence tool usage from a practitioner perspective. For example, the factor '*Use of IaC tools is a good use of my time*' may or may not be an influencing adoption factor to a software practitioner for IaC tools. Furthermore, two individuals can differ in determining what adoption factor is applicable or not. In our research study we consider these issues. The first two authors of the paper, who are familiar with the four types of tools with professional experience, individually determined the factors that can be applied for this particular study. We included 26 factors that were agreed upon by the first two authors. Of these 26 factors, 24 were used in Witschey et al.'s aforementioned study [8], and two of them were not included in Witschey et al.'s paper, and were identified by the first two authors of this paper, based on their professional experience.

Each of the 26 factors of interest was presented to survey participants as a five-point Likert-scale question. Considering the importance of a midpoint in Likert scale items [19] we used a five-point scale: 'Strongly Disagree', 'Disagree', 'Neutral', 'Agree', and 'Strongly Agree'. We required the survey participants to answer all questions provided in the survey to ensure complete responses.

B. Survey Deployment

We deployed the survey to software professionals who work at NestedApps, Red Hat, and contribute to open source software projects via emails. For NestedApps, the second author forwarded the survey to all software practitioners working in NestedApps. In the case of Red Hat, a Red Hat representative provided us the email addresses. In the case of open source contributors, we first asked approval of the open source mailing list administrators. Then upon approval we collected the emails of software practitioners who belong to those mailing lists. We deployed our survey to 1255 software professionals in the two companies and 1445 software professionals listed in four mailing lists. Following Smith et al. [20]'s observations on software engineering survey incentives, we offered a drawing of six Amazon gift cards as an incentive for participation. The survey was available to the targeted software practitioners from June 23, 2016 to September 05, 2016. As per agreements with the company representatives and the mailing list administrators, we do not disclose any information in the paper that can map survey responses to the software professionals.

C. Filter Responses

Surveys can have incomplete responses. They also can be completed using bots or computer programs, and therefore needs to be filtered before data analysis. In our research study we applied the following two filtering steps to filter survey responses collected for B, CI, IaC, and VC tools:

- First, we removed incomplete survey responses for each type of tool. We considered a survey response as incomplete, if the responses were missing for any one of the 26 Likert-scale questions.
- Second, we remove survey responses that are completed programmatically or with negligence by using the duration required completing the survey. We use a *duration threshold* and determine the duration threshold by taking the 25^{th} percentile of survey completion time of all survey responses. We measure duration threshold in seconds.

After applying the two filtering steps we get a set of survey responses that we use to answer the two research questions for each type of build automation tools. We refer to these survey responses as valid responses for the rest of this paper.

D. Survey Data Analysis

In this section, we describe the steps to answer each of the two research questions.

D.1. RQ-1: Which adoption factors influence usage of B, CI, IaC, and VC tools?

For each type of build automation tool, we applied logistic regression to identify the adoption factors that can influence usage. For each of the 26 adoption factors, we created 26 individual logistic regression models. In each of these 26 individual regression models, the dependent variable was the usage question, and the independent variable was the adoption factor. We treated the responses to each of the 26 factors as ordinal variables. For determining if an adoption factor is influential, we used an adjusted p-value for each type of tool to control false discovery rate [21]. We determined the adjusted p-value by applying a Benjamini-Hochberg correction to control the false discovery rate [21]. We implemented logistic regression using the MASS package available for R v.3.1.2 [22].

From the aforementioned regression analysis we record the p-values for each adoption factor, in case of each type of build automation tool. We determine an adoption factor as influential if the corresponding p-value is smaller than the adjusted p-value.

D.2. RQ-2: How can we prioritize the identified influencing adoption factors for B, CI, IaC, and VC tools?

Answers from RQ-1 provide the adoption factors that influence usage of build automation tools. Practitioners might further benefit from findings related to RQ-1 if we can systematically prioritize the influencing adoption factors. We answer RQ-2 using Akaike Information Criterion (AIC) [23], an analysis technique that compares the relative quality of the models created to answer RQ-1, and is not susceptible to model under-fitting and over-fitting. AIC measures the quality of candidate regression models by estimating the loss of information (LOI) for the regression model of interest [24]. A relatively lower AIC score of a regression model indicates relatively better quality considering LOI [24].

For each of the identified influential adoption factors of B, CI, IaC, and VC tools, we applied the following steps:

- We identified the minimum of all AIC scores for the influencing adoption factors of interest, which we label as AIC_{min}. The corresponding factor is labeled as F_{min}. Following Burnham and Anderson's approach [24], we identify F_{min} as the best factor because F_{min} has the lowest AIC score indicating lowest LOI, amongst all the influential adoption factors.
- We used minimization of information loss (MIL) metric that computes how likely one model can minimize information loss compared to other models [24]. A higher MIL of a factor indicates smaller LOI of the model [24]. Except for F_{min}, we computed the MIL for each of the identified influential adoption factors using the formula provided by Burnham and Anderson [24]:

$$MIL(F_i) = \exp((AIC_{min} - AIC_i)/2) \quad (1)$$

In equation 1, $MIL(F_i)$ corresponds to MIL of factor F_i, and AIC_i corresponds to AIC of factor F_i.

We illustrate the MIL approach using a hypothetical example. Let us assume the three influential adoption factors that influence usage of VC tools are F1, F2, and F3. The AIC scores of F1, F2, and F3 respectively are 92, 90, and 99. According to our methodology, F2 is the best factor, and becomes F_{min}, and AIC_{min} is 90. Using Equation 1, we compute MIL for F1 and F3, which are respectively 0.368, and 0.011. According to our methodology, F1 has better LOI compared to F3. The rank of the three adoption factors with respect to LOI can be expressed as F2 > F1 > F3. From this hypothetical example we conclude that considering adoption of VC tools, practitioners might prefer F2 over F1 and F3, as the quality of created regression model using F2, is better than that of F1 and F3.

IV. RESULTS

We start this section by providing a brief summary of the collected survey data. Altogether, 268 software professionals responded to the survey, yielding a response rate of 9.9%. Following our methodology, first we removed 86 of the 268 survey respondents who provided incomplete survey responses. After applying this filtering step we were left with 182 survey responses. Of the 182 survey respondents who provided complete responses, 18, 36, 75, and 61, respectively, answered for all four, three, two, and only one type of build automation tools. Next we applied the duration threshold, which gave us 54, 76, 56, and 112 valid survey responses for B, CI, IaC, and VC tools, respectively. The duration threshold used in filtering survey responses was 120, 100, 180, and 120 seconds, respectively for B, CI, IaC, and VC tools.

We use Figures 1 and 2, respectively, to summarize the industry experience, and familiarity with a specific tool type of the survey respondents. In both Figures 1 and 2, the x-axis corresponds to the four types of tools. In Figure 1, the y-axis quantifies the experience of survey respondents. For example, according to Figure 1, 55.5% of the valid respondents who were familiar with B tools had industry experience of more than 10 years.

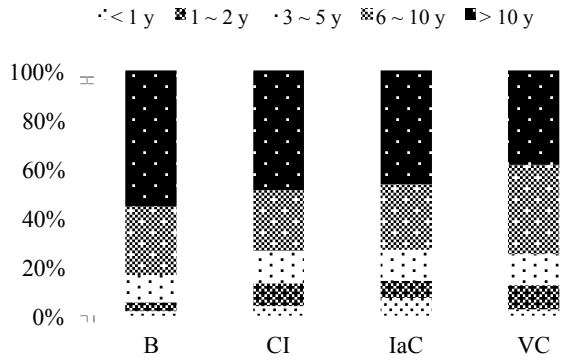

Figure 1: Professional experience summary of the survey respondents.

We use Figure 2 to summarize the level of familiarity with a specific tool, of valid survey respondents. For example in Figure 2, we observe that 18.5% survey respondents that were familiar with B tools, have been familiar with B tools for more than ten years.

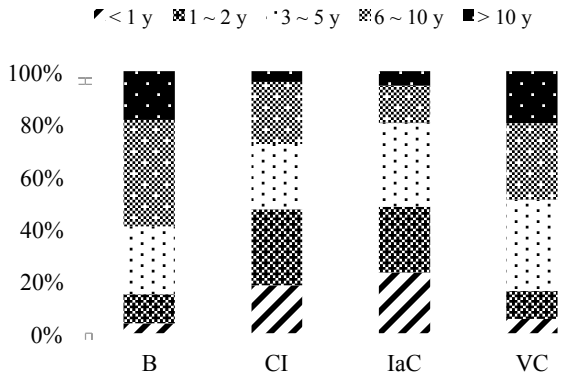

Figure 2: Summary of tool usage duration amongst survey participants.

A. Answer to RQ-1

We present our findings related to RQ-1 in Table I. In Table I the 'Factor' column presents the adoption factors followed by the index of the factor. The 'Group' column presents the factor group to which the adoption factor belongs. The 'Tool' column presents the tool type(s) for which the adoption factor was determined as influential, along with the AIC score and p-values recorded for that particular factor. The p-values are represented in symbols decoded in the table footer, whereas, the AIC score is enclosed within squared brackets. If an adoption factor is not influential for any of the four tool types then the corresponding row in Table I is marked in italic.

Let us use the adoption factor 'Use of <TOOL X> improves the quality of work I do (AD1)' as an example for interpreting Table I. The index of this factor is AD1, and is identified as an influential adoption factor for B tools. The corresponding AIC for this factor is 88.0 for B tools. The recorded p-value for AD1 is < 0.001. This adoption factor belongs to the factor group 'Advantages'.

Recall from Section III-D that we applied adjusted p-value using Benjamini-Hochberg correction. We use these p-values to determine which of the 26 adoption factors influence usage of B, CI, IaC, and VC tools. The adjusted p-value for B, CI, IaC, and VC tools were respectively, 0.025, 0.017, 0.009 and 0.030. As shown in Table I, 24 of the 26 adoption factors are influential for at least one type of build automation tool.

TABLE I: ADOPTION FACTORS THAT INFLUENCE USAGE

Factor	Group	Tool
Use of <TOOL X> improves the quality of work I do (AD1)	Advantages	B** [88.0]
Use of <TOOL X> make my job easier (AD2)	Advantages	B** [72.4]
Use of <TOOL X> improve my performance (AD3)	Advantages	B** [78.2]
Use of <TOOL X> is cost-effective (AD4)	*Advantages*	*B' [77.3]* *CI‡ [147.3]*
		IaC' [106.1] *VC' [127.3]*
I think that the use of <TOOL X> fits well with the way I work (CP1)	Compatibility	B** [89.9] VC** [110.7]
<TOOL X> is highly configurable (CP2)	Compatibility	B** [63.6] VC** [126.8]
I had to adjust my workflow to use <TOOL X> (CP3)	Compatibility	B** [78.0] CI** [151.1] IaC** [103.1]
<TOOL X> is compatible with the technologies that I use (CP4)	Compatibility	CI** [139.1] IaC** [99.8] VC‡ [118.7]
My use of <TOOL X> require a lot of mental effort (CX1)	Complexity	B** [78.9] VC** [116.0]
Use of <TOOL X> requires deep knowledge of <TOOL X> (CX2)	Complexity	B** [74.6] VC** [128.2]
The internal workings of <TOOL X> are complex (CX3)	Complexity	B** [86.1] VC** [125.5]
<TOOL X> present their analysis in understandable ways (CX4)	Complexity	CI** [148.0] VC** [120.3]
My organization holds frequent training on <TOOL X> (ED1)	Education	VC** [127.2]
I learned about <TOOL X> as part of my university courses (ED2)	Education	CI** [148.1]
I prefer to learn about <TOOL X> from online tutorials (ED3)	Education	B** [89.1] CI** [145.5] VC** [125.3]
I prefer to learn about <TOOL X> from their manual (ED4)	Education	B** [88.9] VC** [131.5]
I learn about <TOOL X> as I perform my professional duties (ED5)	Education	B** [81.8] CIᵖ [140.7] VC** [121.5]
I prefer to learn about <TOOL X> from my colleagues (ED6)	*Education*	*B' [85.7]* *CI' [139.7]* *IaC' [106.6]* *VC' [131.1]*
Use of <TOOL X> is visible within the community of <TOOL X> users (OB1)	Observability	B** [77.5] VC** [122.6]
I have seen how my colleagues use <TOOL X> (OB2)	Observability	VC** [122.5]

Use of <TOOL X> is not very visible in my organization (OB3)	Observability	IaC** [99.1]
I can easily observe my colleagues' use of <TOOL X> in my organization (OB4)	Observability	CI** [146.2] IaC** [105.3] VC** [127.7]
I know how I can satisfactorily try out variation of the use of <TOOL X> (TR1)	Trialability	IaC** [105.2] VC** [124.8]
<TOOL X> is available for me to adequately try or not (TR2)	Trialability	VC** [127.8]
I experiment with <TOOL X> whenever necessary (TR3)	Trialability	VC** [127.4]
I did not have to extend very much effort to try out <TOOL X> (TR4)	Trialability	CI** [142.0]

Ŧ indicates a p-value of 0.020 **ρ** indicates a p-value of 0.010
** indicates a p-value of < 0.001 **!** indicates a p-value of > 0.191

B. Answer to RQ-2

In our study, RQ-2 focuses on prioritizing the identified adoption factors that influence usage of B, CI, IaC, and VC tools. We use Table II to present our findings related to RQ-2. In Table II the 'Prioritization' column lists the adoption factors that are influential for B, CI, IaC, and VC tools, and sorted according to their MIL scores. We used the AIC scores presented in Table I to compute the MIL scores.

TABLE II: PRIORITY OF ADOPTION FACTORS

Tool	Prioritization
B	CP2 > AD2 > CX2 > OB1 > CP3 > AD3 > CX1 > ED5 > CX3 > AD1 > ED4 > ED3 > CP1
CI	CP4 > ED5 > TR4 > ED3 > OB4 > CX4 > ED2 > CP3
IaC	OB3 > CP4 > CP3 > TR1 > OB4
VC	CP1 > CX1 > CP4 > CX4 > ED5 > OB2 > OB1 > TR1 > ED3 > CX3 > CP2 > ED1 >TR3 > OB4 > TR2 > CX2 > ED4

We summarize our findings related to RQ-2 as following:

- For **B** tools, the highest priority adoption factor is '<TOOL X> is highly configurable' (**CP2**). Overall, Education related adoption factors have lower priority.
- For **CI** tools, the highest priority factor is '<TOOL X> is compatible with the technologies that I use' (**CP4**).
- For **IaC** tools, the highest priority factor is 'Use of <TOOL X> is not very visible in my organization' (**OB3**). After OB3, the adoption factor **CP4** has higher priority than other adoption factors.
- For **VC** tools, the highest priority factor is 'I think that the use of <TOOL X> fits well with the way I work' (**CP1**).

V. DISCUSSION

In this section we discuss our findings and possible implications for practice.

From our analysis presented in Section IV, we have observed that the ability to customize B tools influence their usage. Furthermore, for CI and VC tools, usage is influenced by how well the tools fit with practitioners' style of work. This finding also indicates that practitioners do not want to change their usual style of work, and might prefer tools that are easy to integrate with their usual style of work. Usage of build automation tools might increase if they are customizable and if they do not hinder practitioners' usual style of work.

▫ *To increase usage of build automation tools, teams can select build automation tools that fit well with the usual work style of the team members and that can be easily customized to the needs of the team members.*

Findings from Section IV indicate that for B tools usage is influenced by how well B tools are used within the practitioners' community. For CI, IaC, and VC tools usage is dependent on how these tools are used within practitioners' peers and the organizations they work for. These findings imply that generally speaking, practitioners' are more likely to adopt build automation tools that are used by their peers or by their community. Blog posts, conferences, and live demonstrations might help in increased usage of build automation tools.

▫ *Practitioner-led demonstrations of build automation tools at company events and public events, such as conferences and meetups, might help in increasing the usage of build automation tools.*

Unlike adoption of security tools [8], from our analysis of build automation tools, we observe practitioners' preference of tools that are customizable and can easily be used without hindering their natural style of work. We consider this particular observation as unexpected, yet explainable. Build automation tools are often applied to achieve continuous deployment (CD). One of the core practices of CD is 'shepherding your own changes' that implies a practitioner who makes software changes is responsible to fix the errors induced by those software changes, all the way from development, through testing, finally to deployment [11]. The practice of shepherding software changes implicitly recommends CD practitioners to be familiar with tools that is related to every phase of software deployment such as Git, Jenkins, Maven, and Puppet. These tools have different purposes, yet compatible with each other to facilitate CD [7]. Practitioners might be more willing to use those build automation tools that fits their existing workflow, or for which they have to adjust

their workflow with minimum effort. Making build automation tools open source might also help in this regard.

▫ *Toolsmiths might influence usage of build automation tools by considering the existing workflow of software practitioners' and designing related tools accordingly*.

VI. LIMITATIONS

We present the limitations of this paper as following:

Survey response rate: Our overall survey response rate was 9.9%, which is not ideal. Low response rate however is not uncommon in the field of software engineering. Buse and Zimmermann [25] reported a response rate of 6% in their research study.

Factors used in the survey: We include 26 adoption factors in our survey. These factors belong to the five DOI innovation factors and education. We acknowledge the list of adoption factors is not comprehensive.

Factor selection process: Our selection process of identifying relevant factors depends on the judgment of two individuals. We acknowledge that this part of our research methodology is subjective.

VII. CONCLUSION

In this research study, we use a quantitative survey analysis to identify the adoption factors that influence usages of build automation tools. We collected responses from practitioners, and conducted analysis on the collected survey responses using logistic regression. Our analysis indicates that compatibility and observability-related factors have relatively more influence on build automation tool usage. We hope that findings from this study will help practitioners in executing the appropriate steps necessary to increase usage of build automation tools.

ACKNOWLEDGMENT

We express our gratitude to all survey participants. We also thank members of the Realsearch group for their feedback. The opinions expressed in this publication are those of the authors and have no association with NestedApps, Red Hat or the open source communities. The first author of this paper conducted part of this study during his internship at Red Hat.

REFERENCES

[1] Y. Brikman, *Hello, Startup: A Programmer's Guide to Building Products, Technologies, and Teams*, 1st ed. O'Reilly Media, Inc., 2015.

[2] S. and E. Commission, "Knight Capital LLC - 34:70694," 16-Oct-2013.

[3] M. Philips, "Knight shows how to lose 440 million in 30 minutes," *Bloomberg News*, 02-Aug-2012.

[4] D. Seven, "Knightmare: A DevOps Cautionary Tale," *Knightmare: A DevOps Cautionary Tale*, 17-Apr-2014. .

[5] G. Research, "Trends in DevOps, Continuous Delivery and Application Release Automation," Gatepoint Research, May 2016.

[6] O. Deploy, "The Benefits of Deployment Automation," Octopus Deploy.

[7] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*, 1st ed. Addison-Wesley Professional, 2010.

[8] J. Witschey, O. Zielinska, A. Welk, E. Murphy-Hill, C. Mayhorn, and T. Zimmermann, "Quantifying developers' adoption of security tools," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, Bergamo, Italy, 2015, pp. 260–271.

[9] L. A. Meyerovich and A. S. Rabkin, "Empirical Analysis of Programming Language Adoption," in *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications*, New York, NY, USA, 2013, pp. 1–18.

[10] S. Mcintosh, B. Adams, and A. E. Hassan, "The Evolution of Java Build Systems," *Empir. Softw Engg*, vol. 17, no. 4–5, pp. 578–608, Aug. 2012.

[11] A. A. U. Rahman, E. Helms, L. Williams, and C. Parnin, "Synthesizing continuous deployment practices used in software development," in *Agile Conference (AGILE), 2015*, 2015, pp. 1–10.

[12] E. Murphy-Hill and G. C. Murphy, "Peer Interaction Effectively, Yet Infrequently, Enables Programmers to Discover New Tools," in *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, New York, NY, USA, 2011, pp. 405–414.

[13] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 672–681.

[14] V. Mosley, "How to assess tools efficiently and quantitatively," *IEEE Softw.*, vol. 9, no. 3, pp. 29–32, 1992.

[15] S. A. Raghavan and D. R. Chand, "Diffusing software-engineering methods," *IEEE Softw.*, vol. 6, no. 4, pp. 81–90, 1989.

[16] E. M. Rogers, *Diffusion of innovations*. Simon and Schuster, 2010.

[17] C. Timothy, "What knowledge is important to a software professional?," *IEEE Softw.*, pp. 44–50, 2000.

[18] P. V. Marsden and J. D. Wright, Eds., *Handbook of Survey Research. Second Edition*. Bingley, UK: Emerald Group Publishing, 2010.

[19] R. Garland, "The mid-point on a rating scale: Is it desirable," *Mark. Bull.*, pp. 66–70, 1991.

[20] E. Smith, R. Loftin, E. Murphy-Hill, C. Bird, and T. Zimmermann, "Improving developer participation rates in surveys," in *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on*, 2013, pp. 89–92.

[21] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *J. R. Stat. Soc. Ser. B Methodol.*, pp. 289–300, 1995.

[22] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, Fourth. New York: Springer, 2002.

[23] T. R. B. David Posada, "Model Selection and Model Averaging in Phylogenetics: Advantages of Akaike Information Criterion and Bayesian Approaches over Likelihood Ratio Tests," *Syst. Biol.*, vol. 53, no. 5, pp. 793–808, 2004.

[24] K. P. Burnham and D. R. Anderson, *Model Selection and Inference A Practical Information-Theoretic Approach*, 2nd ed. New York, NY: Springer New York, 2002.

[25] R. P. L. Buse and T. Zimmermann, "Information Needs for Software Development Analytics," in *Proceedings of the 34th International Conference on Software Engineering*, Piscataway, NJ, USA, 2012, pp. 987–996.