Designing Grit: Discovering Features Towards Supporting Novice Programmer DevOps Integration

Tyrone Justin Sta Maria
De La Salle University
Manila, Philippines
tyrone_stamaria@dlsu.edu.ph

Gavin Raine Dizon
De La Salle University
Manila, Philippines
gavin_dizon@dlsu.edu.ph

Vince Anthony Esquivel
De La Salle University
Manila, Philippines
vince_esquivel@dlsu.edu.ph

Jordan Aiko Deja*
De La Salle University
Manila, Philippines
jordan.deja@dlsu.edu.ph

Unisse Chua
De La Salle University
Manila, Philippines
unisse.chua@dlsu.edu.ph

ABSTRACT

DevOps is usually an industry approach that is practiced by seasoned and experienced programmers and developers. In most university settings especially in the Philippine context, DevOps is not usually part of the curriculum and in some cases are only introduced to learner programmers as an elective or as bonus material. We refer to these students in computing degree programs starting out in learning programming, as novice programmers. Upon graduation, these developers transition into industry roles where they are expected to be familiar with DevOps practices [18]. In most cases, they are not prepared, and fortunately, a great number of them are given training before fully transitioning into their hired roles. In this paper, we attempt to discover and design an intervention mechanism that can assist and prepare novice programmers to easily learn DevOps at an early stage. We gathered data and insights from novice programmers and inquired into their pains and struggles in learning and practicing DevOps. To help them in this process, we propose *Grit*, a prototype tool to support novice programmers in integrating DevOps. Initial insights provided affordances and design elements for a version control prototype with targetted intervention features. In the long run we intend to discover more insights involving the other stages in DevOps beyond version control.

Author Keywords

DevOps; novice programmers; programmer support

*Also affiliated with University of Primorska, Koper, Slovenia

AsianCHI '20, April 25, 2020, Honolulu, HI, USA

 ISBN 978-1-4503-8768-2/20/04…$15.00
DOI: https://doi.org/10.1145/3391203.3391214

CCS Concepts

•**Human-centered computing** → **Collaborative and social computing systems and tools;** •**Social and professional topics** → *Student assessment;*

INTRODUCTION AND RELATED WORK

DevOps is the evolution of the software development life cycle where development and operations are integrated together for faster delivery [4, 13]. Initially, [12] suggested DevOps to follow four dimensions: *Culture*, *Automation*, *Measurement* and *Sharing*. Later on, a new dimension *Lean* was included to the approach renaming it to CALMS [17]. These four dimensions agree with the four pillars of DevOps defined by [6] which is CATS - *Collaboration*, *Affinity*, *Tools* and *Scaling*.

According to the 2019 Accelerate State of DevOps Report [7], businesses have been shifting to DevOps with roughly 26% of over 31,000 respondents working in a DevOps team. However, DevOps is usually not included in the course coverage of software engineering courses in higher education [2] despite recent studies that push for introducing DevOps into the curriculum [5, 14]. Introductory programming courses for novice programmers tend to focus on an individual's logic formulation skills and understanding of basic programming concepts first. In terms of outputs and expectations, novice programmers are expected to produce deliverables done either individually or in pairs. However, when working on bigger projects, novice programmers often struggle on dividing tasks into small segments. They tend to look at the overall task as a whole and solve different tasks simultaneously. Aside from this, they also suffer from a range of struggles that are either cognitive, social development, external commitments and cultural perceptions, which have led these novice learners to poor performance in programming [20]. Collaborative programming and culture is usually introduced later in the curriculum [2]. However, to effectively implement DevOps, establishing the *culture* is critical. Since programming is introduced early on as an individualistic and competitive environment [19], changing this view later on can pose as a challenge.

Teague et. al. [20] evaluated collaborative learning between novice programmers and found that it enhances their perception of programming's difficulty, their enjoyment, and their confidence. This leads to higher retention, deeper learning, and enhanced confidence when learning programming concepts. These perceptions are also influenced by different aspects such as gender, domain knowledge, experience, and stereotypes. This is supported by Tissenbaum [21] when they found users shift from unproductive to productive states using the *Divergent Collaboration Learning Mechanisms* (DCLM) framework. The study also found out that interactions with others played an important role in transitioning to productive states of novices. In this paper, we focus on the understanding and evaluating the perception of DevOps of novice programmers in the Philippines. First, we attempt to measure and assess what initial knowledge novice programmers have about DevOps. We do this investigation in reference to the DevOps CALMS approach [9, 17] as aligned with the CATS pillars by [6]. We then attempt to analyze the different opinions of novice programmers with regards to integrating DevOps in their processes. We do this through affinity and scenario mapping in order to discover what possible interventions we can introduce to help them in their pains and struggles. Lastly, we reflect on how we can design and evaluate a tool that will support novice programmers in the different phases of the DevOps practice.

METHOD

The general methodology of this research follows the approach by [16] where we gathered initial user data and performed affinity analysis on the results. The process of extracting insights and deriving guidelines for novices follows the work of [3, 10, 8] where they investigated novice programmers and came up with design guidelines that lead to coming up with prototype features as well.

Participants

We recruited 50 undergraduate students enrolled in a computing degree program in several Philippine universities through convenience and snowball sampling. Participants should only be starting their first formal programming course in their degree program. Inclusion criteria of a novice programmer was based on the guidelines by [19]. Majority (N=35) of the participants were male and the average age is 19 years old.

Study Protocol and Data Analysis

The questionnaire was sent out using an online Google Form. In addition to their demographics (i.e. age, sex, domicile), we also inquired about their programming experience during their junior and senior high school years and how many years they have been programming. The main questionnaire composed of a mix of Likert-style and open-ended questions that aim to (1) perform a diagnostic, measuring the initial knowledge of novice programmers regarding DevOps, (2) assess their openness towards learning DevOps and (3) record their insights on learning about DevOps. The questionnaire was developed following the CALMS approach by [17] and was further verified and organized following the CATS pillars of DevOps by [6]. The Likert-style questions are measured using 4 levels

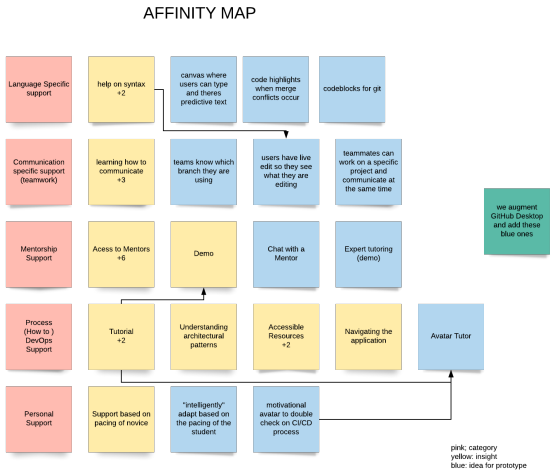

Figure 1. Affinity diagram of open-ended questions

with 1 for strongly disagree and 4 for strongly agree. The list of questions can be found in the project website[1]. Both quantitative and qualitative answers were then processed to derive guidelines and insights that will be used to design an initial software prototype and its features. We wanted to investigate how open novice programmers are to the practice of DevOps and on how much they do and they do not know about it. For this, we looked into their answers in the survey and did a comparative analysis of their responses from the questions.

Affinity and Scenario Mapping

Some of the questions required qualitative insights in the form of essays and long texts which need to be analyzed properly. We used an affinity map to categorize the different types of support needed following the steps defined by [10, 3]. There were at least two rounds of affinity mapping done by two members of the research team. We grouped their insights into three color codes for easy referencing: pink for the support category, yellow for the sample support insight, and blue for an implementation idea for the prototype (See 1). From the affinity map we proceeded to doing scenario mapping where we derived five types of support categories that the respondents needed.

RESULTS AND FINDINGS

We were able to analyze and visualize the insights of our participants.The individual results can be found as listed in our project website[2] More than half of the respondents are confident in working regardless of whether they are alone, with a partner, or with a team. However, most respondents prefer having more people to work with, than having to work alone. In contrast to this, the confidence levels of novice programmers are higher when asked about their capabilities in problem-solving. 54% find it more difficult to solve problems when working alone. Concerning this, majority of the respondents find it easier to work with teammates. From the study we also looked at the familiarity of novice programmers

[1] http://comet.dlsu.edu.ph/grit/questions
[2] http://comet.dlsu.edu.ph/grit/results

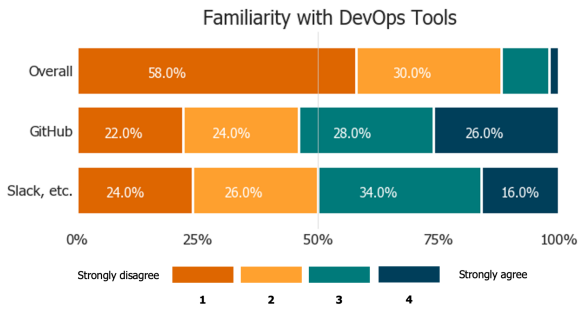

Figure 2. Participants' familiarity with DevOps tools based on a 4-point Likert-style question

with the different DevOps tools as seen in Figure 2. Majority (58%) are not familiar with the various DevOps tools in general. However, more than half were familiar with GitHub. This shows that novice programmers are unaware that version control is essential to the culture of DevOps [6]. In terms of communication, half of the respondents tend to use use professional communication software such as *Slack* and *Google Hangouts* when working with peers. If we wish to further analyze the corresponding DevOps culture, we can see that novice programmers think that DevOps is important (96%) and should be taught in school (92%). However, on the question of whether it is important to learn DevOps to become better in programming, almost half (44%) were unsure whether it was relevant to become a better developer. With regards to the collaboration culture of DevOps, we observe that majority of the respondents (57.4%) strongly perceived it as a platform for knowledge sharing and trust and respect. This is an interesting take as it sparks the discussion that even as novices, they understand wholly that trust and respect are key elements in collaborative activities such as DevOps. Additionally, the respondents perceived DevOps as a collaborative process that involves pair programming and working in teams.

DISCUSSION

Proposed Prototype Features

Using the insights from the affinity maps, we identified support categories that the novice programmers believe would be helpful to them in learning DevOps. For now, we only intend to cover the Version Control (VC) phase in DevOps since VC is one of the initial concepts introduced to a developer when starting out in DevOps[6, 11, 15]. These support categories and their equivalent design features can be seen in Table 1. From these support categories we came up with design features and translated them into a VC prototype. We call this proposed system as *Grit*. It will have the derived features seen in table 1. We have designed a mock-up that can be seen in our project website[3]. The *version control module* is based on GitHub desktop is intended to run git commands such as *pull*, *push*, and *commit*. The Virtual Assistant shall serve as a guided walkthrough for the git commands and its functions. Additionally, the Command-Line feature shall allow the users to execute git commands with predictive text. Likewise, the Coachmarks shall enable first time users to navigate through the system easily. On the other hand, the practice with Codeblocks feature shall help users to practice and understand the basic git commands. Lastly, the Find-a-Coach feature shall be an online mentoring search tool that allows novices to find a mentor that guiding them through their DevOps process real-time. With these proposed features, we seek to investigate whether these novices can be accustomed to the Version Control processes in DevOps but these would still have to be verified through further tests and experiments.

[3] http://comet.dlsu.edu.ph/grit/screenshots

Support Category	Derived Features
Language-Specific	Command Line with Predictive Text
Communication-Specific	Live-Edit, Team chat
Mentorship Support	Find-a-coach
Process (How to)	Virtual Assistant
Personal Support	Coachmarks, Practice with Codeblocks

Table 1. Identified Support Categories and their Corresponding Features based on the Affinity Maps

CONCLUSION AND FUTURE WORK

In this study, we were able provide an early understanding on the readiness of novice programmers to do DevOps. We were able to inquire into their insights considering the *CALMS* approach and *CATS* pillars. We extracted and analyzed respondent insights into support categories that may potentially support novice programmers in the Version Control phases of DevOps. This allowed us to understand their pains and struggles but this is still subject to further inquiry and validation. Lastly, we were able to create a mock-up design of *Grit* these novice devops programmers. Since the study focused on novices and on the Version Control phase of DevOps only, we believe it would be an interesting approach to gather and understand the insights of adept and more experienced programmers as well in the several stages of the DevOps process. This way, we could do a retrospective approach on the pains and struggles that we intend to investigate. The same focus can also be done on expert DevOps practitioners. The study can take another approach in inquiring into their perceived effectiveness, level of engagement, and other factors that can be attributed to their readiness to do DevOps. A participatory study can also be done to review and evaluate the proposed features of Grit. Since DevOps involves Software Industry practices, we can also do an investigation and cross-comparison of novice programmer insights with experienced industry practitioners with the help of the *Functionality, Usability, Supportability, Reliability, Performance* (FURPS) model in practice [1]. The study mainly focused on the novice programmers' familiarity of DevOps practices and VC tools such as Git. We believe that we should also take into account the respondents' familiarity with other DevOps phases like build servers, deployment platforms etc.

REFERENCES

[1] Rafa E Al-Qutaish. 2010. Quality models in software engineering literature: an analytical and comparative

study. *Journal of American Science* 6, 3 (2010), 166–175.

[2] Matthew Bass. 2016. Software Engineering Education in the New World: What Needs to Change?. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*. IEEE, 213–221.

[3] Hugh Beyer and Karen Holtzblatt. 1999. Contextual design. *interactions* 6, 1 (1999), 32–42.

[4] Evgeny Bobrov, Antonio Bucchiarone, Alfredo Capozucca, Nicolas Guelfi, Manuel Mazzara, and Sergey Masyagin. 2019. Teaching DevOps in academia and industry: reflections and vision. *arXiv preprint arXiv:1903.07468* (2019).

[5] Jean-Michel Bruel, Manuel Mazzara, and Bertrand Meyer. 2019. *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer.

[6] Jennifer Davis and Ryn Daniels. 2016. *Effective DevOps: building a culture of collaboration, affinity, and tooling at scale*. " O'Reilly Media, Inc.".

[7] Nicole Forsgren, Dustin Smith, Jez Humble, and Jessie Frazelle. 2019. *2019 Accelerate State of DevOps Report*. https://cloud.google.com/blog/products/devops-sre/the-2019-accelerate-state-of-devops-elite-performance-productivity-and-scaling

[8] Judith Good and Kate Howland. 2017. Programming language, natural language? Supporting the diverse computational activities of novice programmers. *Journal of Visual Languages & Computing* 39 (2017), 78–92.

[9] Joonas Hamunen and others. 2016. Challenges in adopting a Devops approach to software development and operations. (2016).

[10] Karen Holtzblatt, Jessamyn Burns Wendell, and Shelley Wood. 2005. Rapid contextual design: a how-to guide to key techniques for user-centered design. *Ubiquity* 2005, March (2005), 3–3.

[11] Jez Humble and Gene Kim. 2018. *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution.

[12] Jez Humble and Joanne Molesky. 2011. Why enterprises must adopt devops to enable continuous delivery. *Cutter IT Journal* 24, 8 (2011), 6.

[13] Michael Hüttermann. 2012. *DevOps for developers*. Apress.

[14] Christopher Jones. 2018. A Proposal for Integrating DevOps into Software Engineering Curricula. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*. Springer, 33–47.

[15] Gene Kim, Jez Humble, Patrick Debois, and John Willis. 2016. *The DevOps Handbook:: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution.

[16] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. 2007. A design science research methodology for information systems research. *Journal of management information systems* 24, 3 (2007), 45–77.

[17] C Riley. 2014. How to Keep CALMS and Release More! Logentries blog. Online. (2014).

[18] Christopher Scaffidi. 2018. Employers needs for computer science, information technology and software engineering skills among new graduates. *International Journal of Computer Science, Engineering and Information Technology* 8, 1 (2018), 1–12.

[19] Donna Teague and Raymond Lister. 2014. Longitudinal think aloud study of a novice programmer. In *Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148*. Australian Computer Society, Inc., 41–50.

[20] Donna Teague and Paul Roe. 2008. Collaborative learning: towards a solution for novice programmers. In *Proceedings of the tenth conference on Australasian computing education-Volume 78*. Australian Computer Society, Inc., 147–153.

[21] Mike Tissenbaum. 2020. I see what you did there! Divergent collaboration and learner transitions from unproductive to productive states in open-ended inquiry. *Computers & Education* 145 (2020), 103739.