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
University of Primorska
Koper, Slovenia
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 learners as novice programmers as defined by [5]. Upon graduation, these developers transition into industry roles where they are expected to be familiar with DevOps practices [10]. 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. Features and insights provided affordances and coming up with a set of guidelines that cater to the needs of novice programmers.

Author Keywords

devops; novice programmers; programmer support

CCS Concepts

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

Asian CHI Symposium '20, April 25–30, 2020, Honolulu, HI, USA

ISBN 978-1-4503-6708-0/20/04…$15.00

DOI: https://doi.org/10.1145/3313831.XXXXXXX

INTRODUCTION AND RELATED WORK

Novice programmers are individuals that are starting to learn how to program. They tackle lessons on basic programming concepts such as data types and structures and algorithms. In practice, they approach terms in algorithms in a concrete manner rather than abstract [5]. As such, the learning process becomes more process-centric rather than something high-level [11]. Novice programmers are mostly found in university classrooms attending introductory programming courses. These courses tend to focus on an individual's logic formulation skills and understanding basic programming concepts. The problems they tackle usually require the proper application of these basic programming concepts taught in class. In general, computer programming has been traditionally taught and practiced as an individual activity even though software development tasks are done through collaboration [6].

In terms of outputs and expectations, novice programmers are expected to produce deliverables done either individually or in pairs. 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. We want to find out if they have innate practices in managing files and code versions among multiple collaborators. 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 [12]. On an additional note, students are introduced to Software Engineering usually in their early or late junior years and most of the time, DevOps is not included in the course coverage [2]. However, there are already recent studies that push for introducing DevOps into the curriculum [3, 4].

Teague et. al. [12] 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.

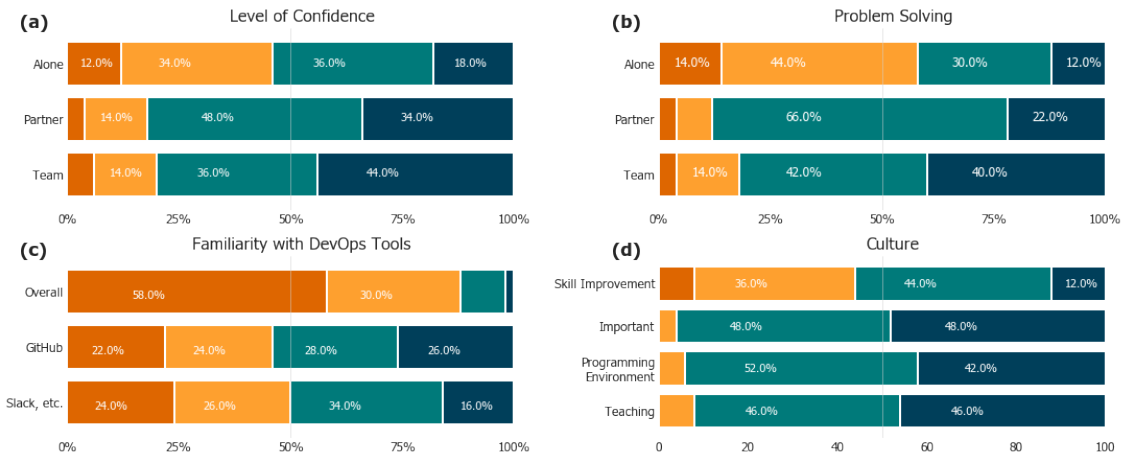

Figure 1. Likert Scale Visualizations of Survey Results following the CALMS Model

These perceptions are also influenced by different aspects such as gender, domain knowledge, experience, and stereotypes. This is supported by Tissenbaum [13] when they found users shift from unproductive to productive states using the 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: (1) attempt to Measure and Assess what initial knowledge novice programmers have about DevOps. We do this investigation in reference to the CALMS model as introduced by [9]. We also attempt to (2) 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 (3) reflect on how we can design and evaluate a tool that will support novice programmers towards adopting DevOps in their practice.

METHOD

The general methodology of this research follows the Design Science approach by [8]. The process of extracting insights and deriving guidelines for novices follows the work of [7].

Participants

We recruited 50 novice programmers as participants through convenience and snowball sampling. We operationalize the definition of a novice programmer based on the guidelines by [11]. Since we wanted to understand this phenomenon specifically in the Philippine context, we limited our recruitment to those who are currently studying programming in the Philippines or in any Philippine university. Filipinos comprise an underrepresented population in literature who may largely benefit from technology-empowered developer practices. We aim to see common behaviors and factors considered despite the difference in university and programming culture. Each participant provided their personal details (i.e. age, sex, domicile, prior programming training) which allows for an examination of possible prior knowledge about software engineering and DevOps. We also inquired about their programming experience (both learning and practicing) and the number of years they have been programming.

Study Protocol and Data Analysis

The questionnaire was sent out using an online Google Form. A semi-structured quantitative approach took place in order to: (1) perform a diagnostic measuring the initial knowledge of novice programmers, (2) assess their openness towards learning DevOps and (3) record their insights on learning about DevOps through the experiment. We processed a combination of qualitative and quantitative answers containing the insights of the respondents. We wanted to investigate how receptive novice programmers are to the practice of DevOps and also 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. The questionnaire was developed following the CALMS model by [9]. The questions were organized based on each of the factors described. We can easily pull and categorize their insights based on the different factors with the help of the CALMS model. We applied affinity mapping to be able to sort and rank the problems that we can identify to ultimately support that prototype that is described in the latter part of this paper. From this, we derived ideas and compared these with existing DevOps tools. After a brainstorming and mapping session, we designed features into a high fidelity mock-up that will be tested and evaluated iteratively by its target users.

RESULTS AND FINDINGS

Based on Fig.1(a), 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 plummet when asked about their capabilities in problem-solving. As seen in Fig.1(b), 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 with the different DevOps Tools as seen in Fig.1(c). Majority (58%) are not familiar with the various DevOps tools in general. However, more than half

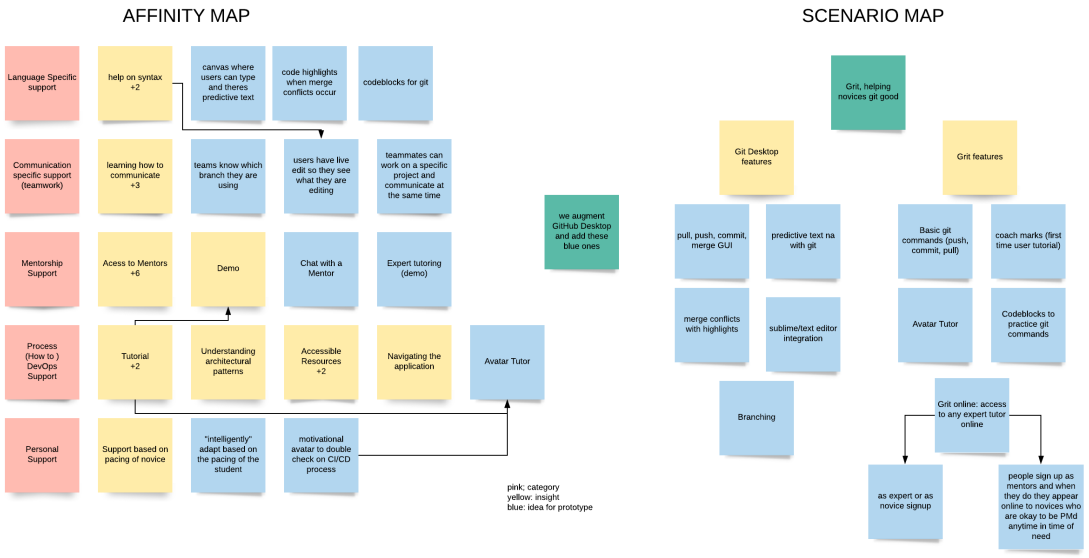

Figure 2. Affinity Map and Scenario Map containing insights and ideas for Grit

were familiar with GitHub though it is believed that not all who are not familiar with DevOps tools may be familiar with Github as a DevOps tool. Moreover, it can be hypothesized that people still opt to do face-to-face communication than to use communication/social applications. The results also show that half of the respondents use professional communication software such as *Slack*, *Google Hangouts*, etc. If we wish to further analyze the corresponding DevOps culture in Fig.1(d), 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 & 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. Looking back at Fig.1(a), we observed that there was a distinct decrease in confidence of the respondents when compared to the general level of confidence in problem-solving scenarios. Moreover, we noticed that most respondents prefer having more people to work with, than being entirely alone. There is an obvious decrease in respondents agreeing about their confidence in working with others as compared to working alone.

DISCUSSION

With the help of the CALMS model, specifically the factors Culture, Automation and Sharing, we were able to process these insights and conduct affinity and scenario mapping. We used an affinity map to categorize the different types of support needed. 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. From the affinity map, we derived five types of support categories that the respondents needed. These categories will serve as the design guidelines that may potentially support the affordances of our novice programmers. They are: (1) Language Specific Support, (2) Communication Specific Support, (3) Mentorship Support, (4) Process (How to) DevOps Support, and (5) Personal Support. This was followed by a scenario mapping session where we brainstormed on features from existing DevOps tools and other features that can be augmented to provide novice support based on the guidelines. See Fig.2 for the affinity and scenario map used.

Proposed Prototype Features

Upon analyzing the data gathered, we designed a prototype that is intended to help novice programmers in integrating DevOps in their projects and processes. We call this proposed system as *Grit*, which shall serve as a platform to guide novice programmers starting with DevOps. It will have the regular features from Git but will be augmented with six main features namely: (1) *Version Control module*, (2) Virtual Assistant, (3) Command-Line with Predictive Text, (4) Coachmarks, (5) Practice with Codeblocks and a feature accessed only online called (6) Find-a-Coach. Fig.3 shows a mock-up of these features. The *version control module* is based on GitHub desktop which can run git commands such as pull, push, and commit. The Virtual Assistant serves as a guided walkthrough for the git commands and its functions. Additionally, the Command-Line feature allows the users to call git commands with predictive text. Likewise, the Coachmark enables first time users to navigate through the system easily. On the other hand, the practice with Codeblocks feature helps users to practice and understand the basic git commands. Lastly, the Find-a-Coach feature is an online mentoring tool that allows novices to find a mentor that will guide them through their

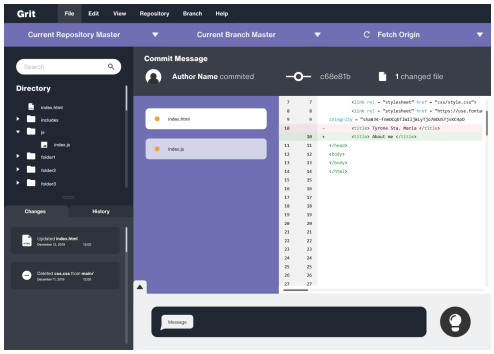

Figure 3. Preview of Grit and some of its features

DevOps process real-time. With these proposed features, we seek to investigate whether these novices can be accustomed to the DevOps process but these would still have to be verified through further tests and experiments.

CONCLUSION AND FUTURE WORK

In this study, we were able provide an initial understanding on the readiness of novice programmers to do DevOps. The collected data still needs further analysis. We were able to inquire into their insights considering the *Culture, Automation* and *Sharing* aspects of the CALMS model. We synthesized and extracted these insights into factors that may potentially support affordances of novice programmers in integrating DevOps in their practice. This allowed us to initially 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* to augment Novice DevOps experiences. Since the study focused on novices, we believe it would be an interesting approach to gather and understand the insights of adept and more experienced programmers as well. 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 **FURPS** model in practice [1].

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

[4] 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.

[5] Diane P McCarthy and Rob Oliver. The Game's the Thing: Levelling up from Novice Status. (????).

[6] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The effects of pair-programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*. 38–42.

[7] Giselle Nodalo, Jose Ma Santiago III, Jolene Valenzuela, and Jordan Aiko Deja. 2019. On Building Design Guidelines for An Interactive Machine Learning Sandbox Application. In *Proceedings of the 5th International ACM In-Cooperation HCI and UX Conference*. 70–77.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.