Poster: Defect Prediction Metrics for Infrastructure as Code Scripts in DevOps

Akond Rahman, Jonathan Stallings, and Laurie Williams
North Carolina State University
aarahman@ncsu.edu,jwstalli@ncsu.edu,andlawilli3@ncsu.edu

ABSTRACT

Use of infrastructure as code (IaC) scripts helps software teams manage their configuration and infrastructure automatically. Information technology (IT) organizations use IaC scripts to create and manage automated deployment pipelines to deliver services rapidly. IaC scripts can be defective, resulting in dire consequences, such as creating wide-scale service outages for end-users. Prediction of defective IaC scripts can help teams to mitigate defects in these scripts by prioritizing their inspection efforts. *The goal of this paper is to help software practitioners in prioritizing their inspection efforts for infrastructure as code (IaC) scripts by proposing defect prediction model-related metrics.* IaC scripts use domain specific languages (DSL) that are fundamentally different from object-oriented programming (OOP) languages. Hence, the OOP-based metrics that researchers used in defect prediction might not be applicable for IaC scripts. We apply Constructivist Grounded Theory (CGT) on defect-related commits mined from version control systems to identify metrics suitable for IaC scripts. By applying CGT, we identify 18 metrics. Of these metrics, 13 are related to IaC, for example, count of string occurrences in a script. Four of the identified metrics are related to churn, and one metric is lines of code.

CCS CONCEPTS

• **Software and its engineering → Software defect analysis**;

KEYWORDS

Continuous Deployment, DevOps, Infrastructure as Code, Metrics

ACM Reference format:
Akond Rahman, Jonathan Stallings, and Laurie Williams. 2018. Poster: Defect Prediction Metrics for Infrastructure as Code Scripts in DevOps. In *Proceedings of 40th International Conference on Software Engineering Companion, Gothenburg, Sweden, May 27-June 3, 2018 (ICSE '18 Companion),* 2 pages.
https://doi.org/10.1145/3183440.3195034

1 INTRODUCTION

Information technology (IT) organizations are increasingly adopting DevOps practices [1]. DevOps organizations i.e. IT organizations that adopt DevOps, have strong collaboration between software development and operations teams to deliver software rapidly [2] [7]. DevOps organizations use technologies to automate repetitive work and increase transparency between software development and operations teams [5]. One technology that these organizations consider essential to implement DevOps is the use of infrastructure as code (IaC) scripts [7] [11]. DevOps organizations use IaC scripts such as Puppet [1] scripts, to automatically manage their configurations and operations infrastructure [6] [7].

IaC scripts help to create and manage automated deployment pipelines, and deploy software rapidly [7]. But similar to software source code, IaC scripts churn frequently [9] [10], and can contain defects [10]. Defects in IaC scripts can have dire consequences: for example, Github experienced a DNS outage caused by a defect in an IaC script [3]. Defect prediction of software modules helps software teams to prioritize inspection efforts [12] [4]. Prediction of defective IaC scripts can help IT organizations make informed decisions about allocating inspection efforts to those IaC scripts that are likely to be defective.

The goal of this paper is to help software practitioners in prioritizing their inspection efforts for infrastructure as code (IaC) scripts by proposing defect prediction model-related metrics.

We examine the following research question: **RQ: *What metrics can be used to characterize defective infrastructure as code (IaC) scripts?***

2 METHODOLOGY

We first provide definitions, then we describe our methodology.

- **Defect**: An imperfection in an IaC script that needs to be replaced or repaired. We follow the IEEE definition of defects [8].
- **Defect-related commit**: A commit whose message indicates that an action was taken related to a defect.
- **Defective script**: An IaC script which is listed in a defect-related commit.

We use the following steps to derive necessary metrics:

Repository Collection: We mine open source version control repositories from three organizations: Mozilla, Openstack, and Wikimedia.
Commit Message Processing: We extract messages from commits, where at least one IaC script is modified.
Determining Defect-related Commits: We apply qualitative analysis with the help of multiple raters to determine defect-related commits.
Constructivist Grounded Theory (CGT): We apply CGT on defective commit messages to derive metrics.

[1] https://puppet.com/

Table 1: Metrics that Characterize Defective IaC Scripts

Metric	**Measurement Technique**
Lines of Code (LOC) [GE]	Total lines of code
Churn Count (CC) [CH]	Count of times a script was churned
Churn Deleted (CD) [CH]	Total lines deleted/Total lines in script
Churn Total (CT) [CH]	Total lines in the script added or modified
Churn per LOC (CT_PER_LOC) [CH]	$CT\_PER\_LOC = CT/LOC$
Comment (CMT) [CO]	Total count of comments
Command Execution (CMD) [CO]	Count of 'cmd' syntax occurrences
Ensure [CO]	Count of 'ensure' syntax occurrences
FILE [CO]	Count of 'file' syntax occurrences
File Mode (FM) [CO]	Count of 'mode' syntax occurrences
Include (INCL) [CO]	Count of 'include' syntax occurrences
URL [CO]	Count of URL occurrences
Location (LOCA) [CO]	$LOCA = FILE + URL$
Require (REQ) [CO]	Count of 'require' syntax occurrences
SSH Authorized Key (SSH_KEY) [CO]	Count of 'ssh_authorized_key' syntax occurrences
Strings (STR) [CO]	Count of string occurrences
Strings per LOC (STR_PER_LOC) [CO]	$STR\_PER\_LOC = STR/LOC$
Value Assignment (VA) [CO]	Total count of '=>' usages

3 FINDINGS

We answer RQ in this section. In Section 2, we discussed our process for using CGT to identify metrics that characterize defective IaC scripts. Through this process, we identify 18 metrics. Each of these 18 metrics correspond to a characteristic of defective IaC scripts. We present these 18 metrics with measurement techniques and rationales in Table 1. In Table 1, the 'Metric' column presents each metric, followed by the category of the metric i.e. churn-related metric (CH), IaC-related code metric (CO) or generic metric (GE) included in square brackets.

Jiang and Adams [9] hypothesized that churn can potentially make IaC scripts defect-prone. Our findings from CGT provide empirical support to their hypothesis. We observe IaC-related code metrics to characterize defective scripts as well, e.g. assignment of configuration and attribute values, file paths, and URLs.

4 LIMITATIONS

We discuss the limitations of our paper as following:

Metrics: We used defect-related commits to identify the 18 metrics. We acknowledge that our selection of metrics is not comprehensive. In the future, we plan to investigate other metrics, such as process metrics, for predicting defective IaC scripts.

Datasets: We use three datasets to evaluate our methodology. We acknowledge that more datasets can help generalizing our findings. Also, the datasets do not include temporal information i.e. we do not account for presence or absence of defects across time. We plan to include more datasets in future that will also account for the temporal information for defects in IaC scripts.

5 CONCLUSION

IaC is one of the fundamental pillars to implement DevOps. But similar to software source code, IaC scripts are susceptible to defects. Defect prediction models for IaC can help software teams to prioritize inspection efforts. Using Constructivist Grounded Theory technique, we identified one generic metric (lines of code), 13 IaC code-related metrics, and four churn metrics that characterize defective IaC scripts.

REFERENCES

[1] N. F. Alanna Brown, Jez Humble, Nigel Kersten, and Gene Kim. 2017. 2016 State of DevOps Report. https://puppet.com/resources/whitepaper/2016-state-of-devops-report. (2017). [Online; accessed 15-August-2017].

[2] Jürgen Cito, Philipp Leitner, Thomas Fritz, and Harald C. Gall. 2015. The Making of Cloud Applications: An Empirical Study on Software Development for the Cloud. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015).* ACM, New York, NY, USA, 393–403. https://doi.org/10.1145/2786805.2786826

[3] James Fryman. 2014. DNS Outage Post Mortem-Github. https://github.com/blog/1759-dns-outage-post-mortem. (2014). [Online; accessed 24-August-2017].

[4] Baljinder Ghotra, Shane McIntosh, and Ahmed E. Hassan. 2015. Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1 (ICSE '15).* IEEE Press, Piscataway, NJ, USA, 789–800. http://dl.acm.org/citation.cfm?id=2818754.2818850

[5] Viral Gupta, P.K. Kapur, and Deepak Kumar. 2017. Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling. *Information and Software Technology* (2017). https://doi.org/10.1016/j.infsof.2017.07.010

[6] Oliver Hanappi, Waldemar Hummer, and Schahram Dustdar. 2016. Asserting Reliable Convergence for Configuration Management Scripts. *SIGPLAN Not.* 51, 10 (Oct. 2016), 328–343. https://doi.org/10.1145/3022671.2984000

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

[8] IEEE. 2010. IEEE Standard Classification for Software Anomalies. *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)* (Jan 2010), 1–23. https://doi.org/10.1109/IEEESTD.2010.5399061

[9] Yujuan Jiang and Bram Adams. 2015. Co-evolution of Infrastructure and Source Code: An Empirical Study. In *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15).* IEEE Press, Piscataway, NJ, USA, 45–55. http://dl.acm.org/citation.cfm?id=2820518.2820527

[10] C. Parnin, E. Helms, C. Atlee, H. Boughton, M. Ghattas, A. Glover, J. Holman, J. Micco, B. Murphy, T. Savor, M. Stumm, S. Whitaker, and L. Williams. 2017. The Top 10 Adages in Continuous Deployment. *IEEE Software* 34, 3 (May 2017), 86–95. https://doi.org/10.1109/MS.2017.86

[11] D. Spinellis. 2012. Don't Install Software by Hand. *IEEE Software* 29, 4 (July 2012), 86–87. https://doi.org/10.1109/MS.2012.85

[12] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, A. Ihara, and K. Matsumoto. 2015. The Impact of Mislabelling on the Performance and Interpretation of Defect Prediction Models. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. 812–823. https://doi.org/10.1109/ICSE.2015.93