Replication project

Source code properties of defective infrastructure as code scripts

Nicolas Legros

Montréal, Canada
nicoals.legros@polytml.ca

Thomas Trépanier

Montréal, Canada
thomas.trépanier@polytml.ca

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Problem statement

Continuous delivery or continuous deployement (CD) is the act of releasing new software versions to the end-users as frequently as possible. This practice has been on the rise in the last decade, and with it, the need for tools to automate the deployment process. Infrastructure as code (IaC) is a practice that aims to automate the deployment of infrastructure by using code. IaC scripts are used to describe the desired state of the infrastructure, and the tools will then deploy the infrastructure to match the desired state.

The IaC scripts are usually stored in a version control system (VCS) such as Git. This allows the developers to collaborate on the scripts and to keep track of the changes made to the scripts. The VCS also allows the developers to review the changes made to the scripts before merging them into the main branch, just as they would do with regular code.

However, few other mechanisms exist to ensure the quality of the scripts. This can lead to faulty configuration being deployed to the infrastructure, which is problematic because it can lead to downtime, security breaches, and other issues. As mentionned in the original paper, in 2017, Wikimedia Commons executed a defective IaC script which led to the deletion the home directory of around 270 users.

The replicated paper aimed at introducing a new gating mechanism for IaC scripts by identifying the source code properties of defective scripts.

II. Research questions

This paper aims at answering two of the three research questions in the original paper, *RQ1* and *RQ3*, which are the following:

A. RQ1: What source code properties characterize defective infrastructure as code scripts?

For RQ1, we identified the code properties that characterize defective IaC scripts using *Mann-Whitney U* test and *Cliff's Delta*. We didn't compute the feature importance using a Random Forest since it was only mentionned that we had to do the methods above in the replication guide. Unfortunately, our computers missed computing power to complete the analysis for the Openstack dataset.

B. RQ3: How can we construct defect prediction models for infrastructure as code scripts using the identified source code properties?

For RQ3, we replicated *3.5.1 Principle component analysis*, *3.5.2 Statistical learners* and *3.5.3 Evaluation methods*. It's good to know that we didn't replicate *3.5.4 Comparison model construction* since we only did the analysis by properties and not by bag-of-words.

These questions will be answered using the approach described in Section III.

III. Approach

3) Methodologies for answering each RQ (How to mine the repositories, the tools employed, and ML models they've used).

A. Repository mining

B. RQ1: What source code properties characterize defective infrastructure as code scripts?

For this research question we used the reported data from the paper[1]. We used the *Mann-Whitney U* test with the Scikit Learn package to evaluate which properties had the biggest influence on defective files. The null hypothesis is that the property is not different between defective and neutral files, and the alternative hypothesis is that the property is larger for defective than neutral files. As in the paper, we consider a significance level of 95% which means we reject the null hypothesis when $p - value < 0.05$.

We also used *Cliff's Delta* by calculating it with Neilernst's package[2] to measure how large the difference between the distribution of each characteristics for defective and neutral files is.

[1] https://figshare.com/s/ad26e370c833e8aa9712
[2] https://github.com/neilernst/cliffsDelta

C. RQ3: How can we construct defect prediction models for infrastructure as code scripts using the identified source code properties?

Before using statistical learners, we completed a PCA analysis to determine what properties should be used. We only used the principle components that accounted for at least 95% of the total variance as the input for the statistical learners. We can see in Table I that only one or two principle components account for 95% of the total variance depending on the dataset.

With the component created, we than used it as the input for the different statistical learners. Like the paper, we used Scikit Learn packages to construct the models. The learners that were used are Classification Tree (CART), K Nearest Neighbor (KNN), Logistic Regression (LR), Naive Bayer (NB) and Random Forest (RF).

To evaluate the performance of the different classification models, we used the same metrics as the paper (i.e. precision, recall, AUC, F-measure).

TABLE I
NUMBER OF PRINCIPLE COMPONENTS FOR THE MODELS

Dataset	Number of components
Mirantis	1
Mozilla	1
Openstack	2
Wikimedia	2

TABLE II
VALIDATION OF IDENTIFIED SOURCE CODE PROPERTIES FOR MIRANTIS

Property	$p - value$	$delta$
Attribute	< 0.001	0.47
Command	0.005	0.24
Comment	< 0.001	0.37
Ensure	< 0.001	0.38
File	< 0.001	0.36
File mode	< 0.001	0.41
Hard coded string	< 0.001	0.55
Include	< 0.001	0.33
Lines of code	< 0.001	0.45
Require	< 0.001	0.36
SSH KEY	< 0.001	0.39
URL	0.009	0.22

TABLE III
VALIDATION OF IDENTIFIED SOURCE CODE PROPERTIES FOR MOZILLA

Property	$p - value$	$delta$
Attribute	< 0.001	0.40
Command	< 0.001	0.18
Comment	0.58	0.03
Ensure	< 0.001	0.09
File	< 0.001	0.18
File mode	< 0.001	0.24
Hard coded string	< 0.001	0.40
Include	< 0.001	0.31
Lines of code	< 0.001	0.50
Require	< 0.001	0.19
SSH KEY	< 0.001	0.24
URL	0.081	0.08

IV. RESULTS AND DISCUSSION

4) Results and discussion, including a comparison between the results of the replication study and the original study and a list of the limitations encountered during the replication.

A. RQ1: What source code properties characterize defective infrastructure as code scripts?

After completing the *Mann-Whitney U* test and *Cliff's Delta* test we can identify which properties have a $p-value < 0.05$ for all datasets (i.e. Attribute, Command, Ensure, File, File mode, Hard coded string, Include, Lines of code, Require and SSH Key). Surprisingly, we don't quite get the same results as the paper. In our case, the *Comment* property doesn't have a $p-value < 0.05$. Actually, in the paper either but it's in bold in Table 8. Maybe it's a simple mistake. You can see these results in Table **??** for the Mirantis dataset, in Table III for the Mozilla dataset and in Table IV for the Wikimedia dataset.

B. RQ3: How can we construct defect prediction models for infrastructure as code scripts using the identified source code properties?

As mentionned in the previous section, we only used the principle components that accounted for at least 95% of the total variance. We can see in Table I that only one or two principle components account for 95% of the total variance depending on the dataset. The number of principle components for each dataset corresponds to the ones in the paper.

Since the paper doesn't specify the different parameters for the models, it would be difficult to obtain exactly the same results. Nevertheless, we obtained results that are very similar that bring to the same conclusions. The results from the cross-validation for each model can be found in Table V for the Mirantis dataset, in Table VI for the Mozilla dataset, in Table VII for the Openstack dataset and in Table VIII for the Wikimedia dataset. We haven't included the results from the actual paper, since it's publicly available.

V. CONCLUSION

5) conclusion

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, "Title of paper if known," unpublished.

[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

TABLE IV
VALIDATION OF IDENTIFIED SOURCE CODE PROPERTIES FOR WIKIMEDIA

Property	$p-value$	$delta$
Attribute	$<$ 0.001	0.47
Command	0.008	0.18
Comment	$<$ 0.001	0.22
Ensure	$<$ 0.001	0.29
File	$<$ 0.001	0.31
File mode	$<$ 0.001	0.24
Hard coded string	$<$ 0.001	0.55
Include	$<$ 0.001	0.37
Lines of code	$<$ 0.001	0.51
Require	$<$ 0.001	0.32
SSH KEY	$<$ 0.001	0.24
URL	0.011	0.17

TABLE V
CROSS-VALIDATION RESULTS FOR MIRANTIS

	RF	NB	LR	KNN	CART
AUC	0.701661	0.714252	0.750981	0.693334	0.659597
Recall	0.707425	0.407360	0.649691	0.672546	0.707425
Precision	0.701199	0.846909	0.798322	0.667389	0.701199
F1-measure	0.695896	0.541781	0.708236	0.663964	0.698448

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.

TABLE VI
CROSS-VALIDATION RESULTS FOR MOZILLA

	RF	NB	LR	KNN	CART
AUC	0.731664	0.699599	0.756323	0.713161	0.691230
Recall	0.649017	0.392519	0.565923	0.619417	0.627106
Precision	0.642651	0.831862	0.706600	0.604170	0.642550
F1-measure	0.645764	0.532261	0.626990	0.608864	0.633393

TABLE VII
CROSS-VALIDATION RESULTS FOR OPENSTACK

	RF	NB	LR	KNN	CART
AUC	0.647741	0.694343	0.659972	0.659195	0.574832
Recall	0.667616	0.368902	0.731321	0.687022	0.660176
Precision	0.653112	0.847009	0.643218	0.661449	0.655685
F1-measure	0.660440	0.512676	0.682243	0.673287	0.657360

TABLE VIII
CROSS-VALIDATION RESULTS FOR WIKIMEDIA

	RF	NB	LR	KNN	CART
AUC	0.664721	0.709438	0.736270	0.699140	0.583164
Recall	0.591171	0.366128	0.586200	0.627349	0.587493
Precision	0.664007	0.885945	0.774041	0.732415	0.666620
F1-measure	0.628276	0.515651	0.663515	0.673132	0.623400