Addressing the Testing Gap in PL-IaC: A Case Study on Pulumi .NET

Authors: Albin Rönnkvist & Piran Amedi | {alrn1700, roam2200}@student.miun.se

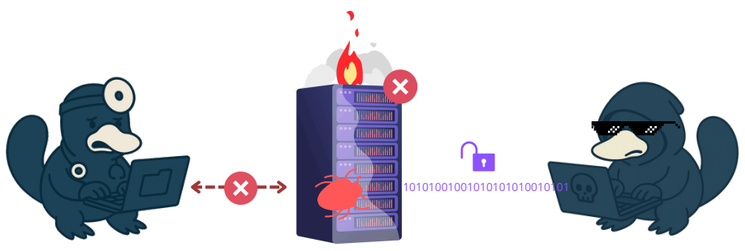
Department of Communication, Quality Management and Information Systems
Mid Sweden University, Östersund, Sweden

Infrastructure as Code (IaC) brings software engineering practices to infrastructure. However, testing remains challenging and underutilized, especially in Programming Language-based IaC (PL-IaC). Low adoption rates threaten system reliability and highlight the need for better testing support. This thesis investigates barriers to unit testing in PL-IaC and explores how tailored tooling can improve adoption.

01 Introduction

Context

PL-IaC has aligned infrastructure management with Software Engineering practices by enabling the use of programming languages. However, testing practices remain limited and challenging, with only **25% adoption across the ecosystem**. This gap threatens reliability, as undetected defects can lead to security vulnerabilities and outages, impacting the stability of our increasingly digitalized society.



Contributions

This thesis investigates the barriers to unit testing in PL-IaC through a case study of Pulumi .NET, a widely used tool with the **lowest testing adoption rate at only 1%**. It examines developer-reported challenges, presents a tool designed to address them, and evaluates its effectiveness compared to existing solutions.

Research Questions

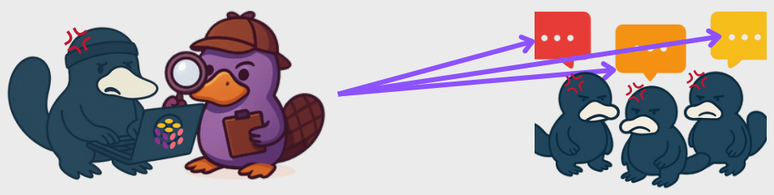
The research questions are framed around the case, while also offering insights that extend to the broader PL-IaC ecosystem:

- **RQ1**: What are the common challenges developers face when unit testing Pulumi .NET programs?
- **RQ2**: In what ways can Pulumock mitigate common challenges when unit testing Pulumi .NET programs?
- **RQ3**: How does Pulumock compare relative to Pulumi's default testing framework in terms of mitigating common challenges when unit testing Pulumi .NET programs?

02 Methodology

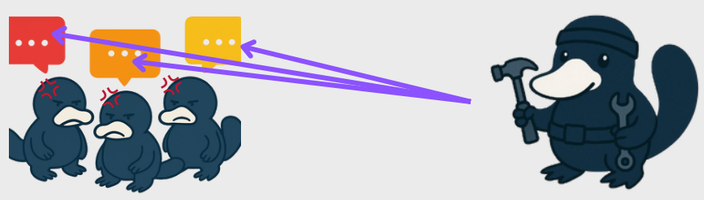
Dataset Construction

In response to RQ1, we developed Pulumissues to identify common unit testing challenges reported in public forums.



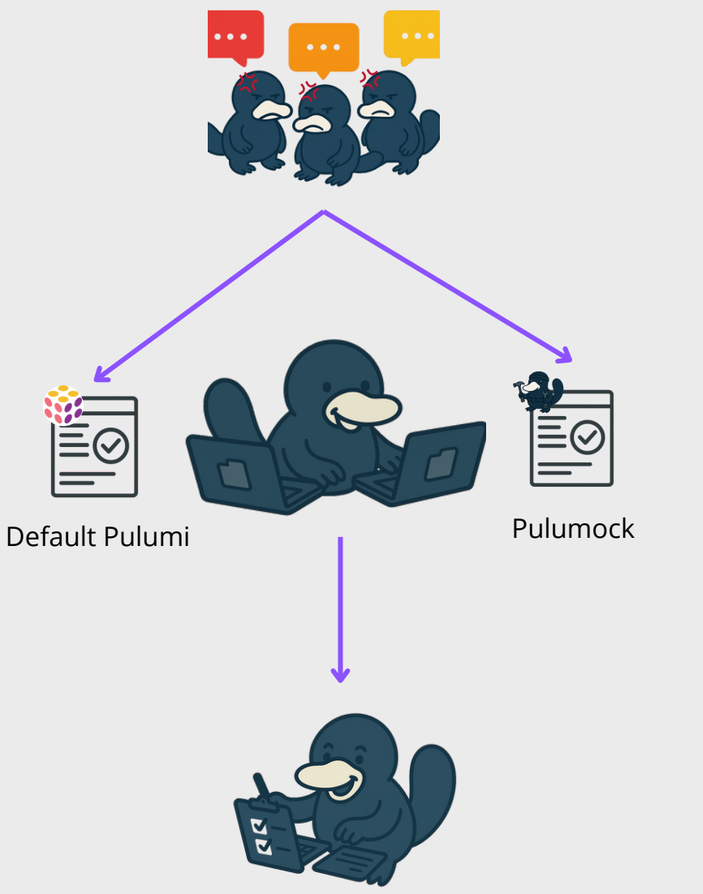
Tool Construction

As for RQ2, we built Pulumock, a tool designed to mitigate the previously identified challenges.



Tool Evaluation

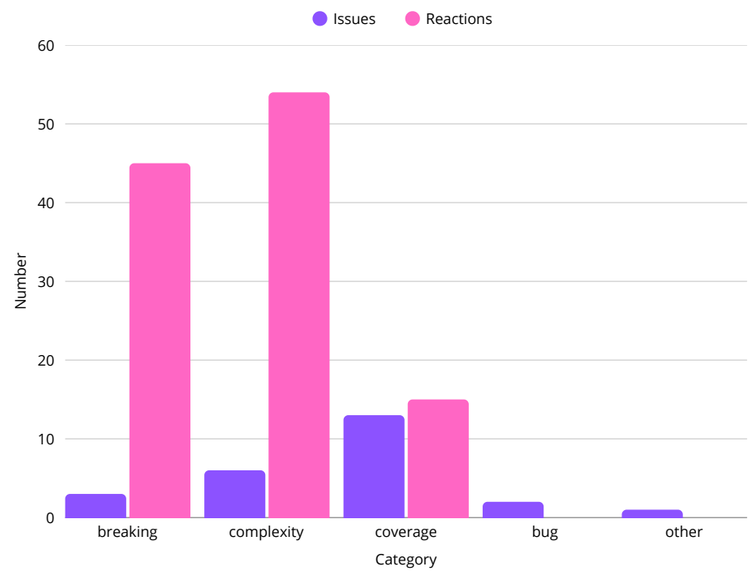
To answer RQ3, we conducted the Puluvaluation within-subjects study, comparing Pulumock to Pulumi's default testing framework. Participants completed tasks and questionnaires to evaluate whether and how Pulumock mitigated the identified challenges and impacted the overall testing experience.



03 Results

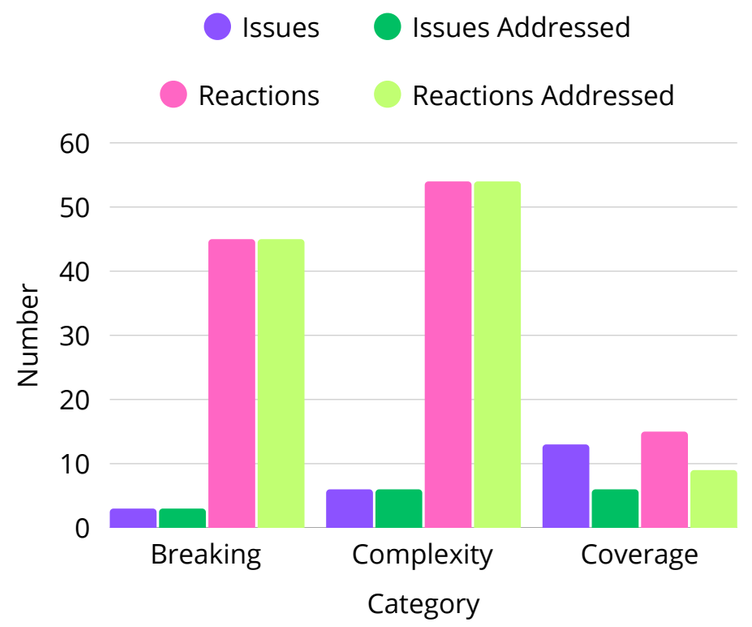
RQ1

Based on 20 relevant issues, our analysis showed that the most common challenges involve the complexity and lack of coverage when mocking and accessing test data. These challenges often arise from incomplete or unclear testing APIs.



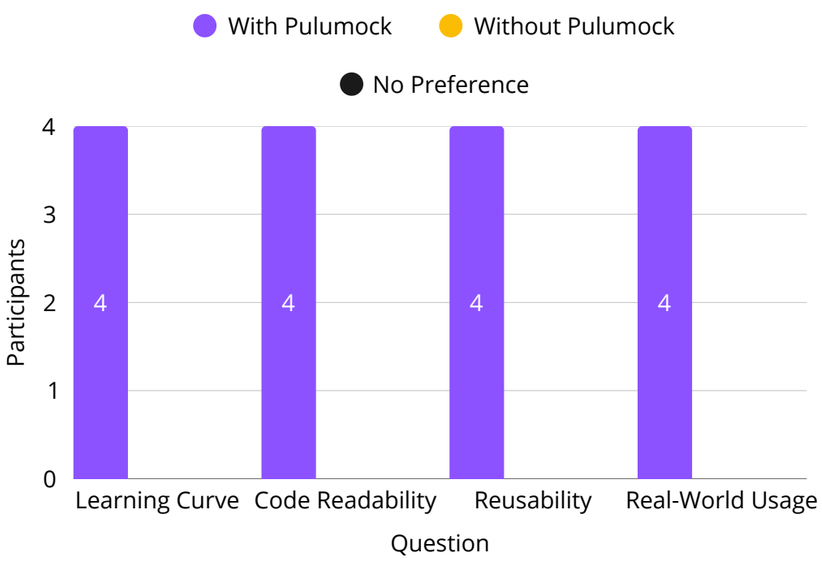
RQ2

Pulumock targeted several challenges spanning multiple issue categories. All reported issues in the two prioritized categories, *breaking* and *complexity*, and 6 out of 13 of issues in the less prioritized *coverage* category.



RQ3

All participants preferred Pulumock over Pulumi's default testing framework when implementing test cases involving common challenges. Tasks were completed 65% faster on average, and positive questionnaire feedback indicated a more efficient, adoptable, and maintainable testing experience with Pulumock.



04 Conclusion

Reported issues highlighted challenges with mocking and accessing test data due to inadequate APIs. To address this, we developed Pulumock, which participants consistently preferred over Pulumi's default testing framework. This suggests that targeted tooling can reduce barriers and support greater testing adoption in the PL-IaC community.

Future work should examine more PL-IaC tools, the impact of program design on testability, and the role of unit tests in infrastructure testing.