Comparative Study: Infrastructure as Code Using Prompt Engineering Techniques.

Fergal Connolly
School of Enterprise Computing and Digital Transformation, TU Dublin, Ireland
X00218711@mytudublin.ie

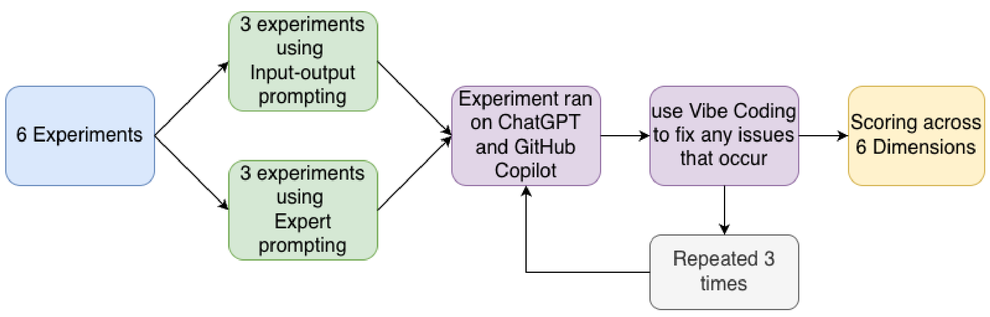
Introduction

Infrastructure as Code (IaC) has transformed the design, deployment, and management of cloud environments. Since 2024, the emergence of Large Language Models (LLMs) such as ChatGPT and GitHub Copilot has transformed software development practices. These platforms have made advanced generative AI technologies readily accessible. This study examines whether LLMs can generate IaC that is functionally correct, secure, and comparable to human-written Terraform, and whether prompt engineering can improve the quality. To investigate this, six experiments were carried out using ChatGPT and GitHub Copilot, with each introducing additional architectural complexity. The overall results were compared against a manually authored baseline, with evaluation focused on maintainability, compliance, and presence of hallucinations.

Research Question

RQ1: Can prompt-engineered IaC generated by an LLM match or outperform manually written Terraform configurations.
RQ2: Can non-experts effectively use LLMs to produce functional, high-quality IaC?

Method

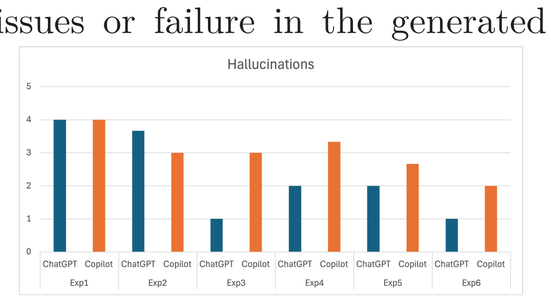


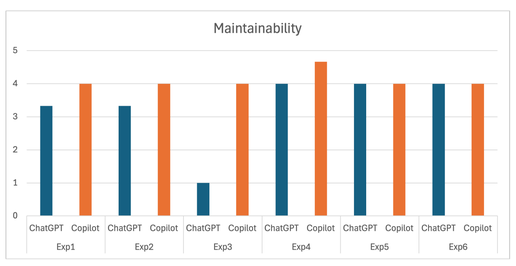
Experiments were conducted using an incremental infrastructure complexity model and executed three times per LLM to ensure result consistency and to mitigate the effects of output generation. IaC artifacts were produced using structured natural language prompts based on Input–Output Prompting and Expert Prompting techniques. A total of six prompts were designed to provision small, medium, and large Azure Kubernetes Service (AKS) clusters, with each introducing additional architectural elements as complexity increased. The analysis focuses on quality dimensions relevant to IaC: **Infrastructure Deployment**, **Hallucinations**, **Linting and Formatting**, **Monitoring Deployment**, **Maintainability**, and **Compliance**. This approach enabled the evaluation of how LLMs perform as the scenarios increased in complexity.

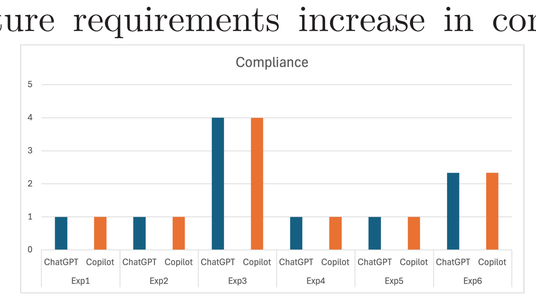
OpenAI ChatGPT and GitHub Copilot were both evaluated, and applying a technique called vibe coding, whereby the LLMs were prompted iteratively in response to validation and deployment errors. IaC quality was assessed using Terraform validation to verify syntax and structural correctness, while Terrascan was used to evaluate security posture and policy compliance. Following successful infrastructure provisioning, a lightweight Grafana monitoring stack was deployed to confirm operational readiness. A manually authored IaC implementation served as the baseline for comparison, allowing for the assessment of technical accuracy and consistency across workflows.

Results

Large Language Models can rapidly generate functional IaC for low-complexity deployments, making them effective tools for scaffolding and boilerplate generation. However, as architectural complexity increases, reliability, contextual consistency, and compliance degrade significantly, with recurring security and governance failures across all experiments. Sample results, evaluated on a five-point scale (where 5 denotes no issues and 1 denotes serious issues or failure in the generated code), demonstrate a decline in LLM effectiveness as infrastructure requirements increase in complexity.

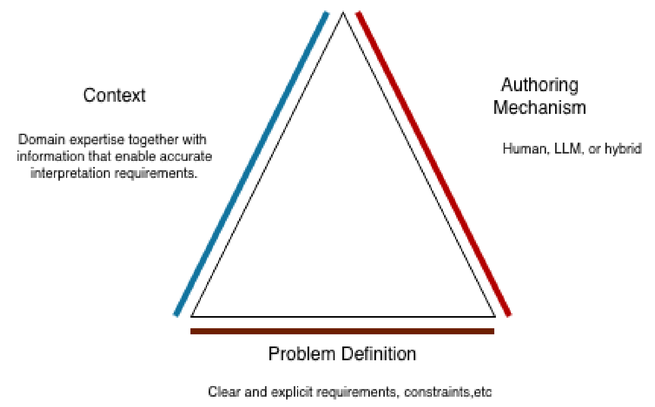






While AI-generated IaC accelerates development, and prompt engineering techniques enhance the structure and clarity of generated code, it also requires sufficient domain knowledge to craft prompts that effectively leverage LLM capabilities. LLMs can lower the barrier to entry, enabling non-experts to participate more readily in IaC development. However, expert domain oversight remains essential. Without it, the risk of insecure, non-compliant, or unstable infrastructure increases significantly, especially in more complex environments.

To synthesise these findings, this study proposes the IaC Dependency Triangle, a conceptual model that explains Infrastructure as Code quality as the interaction of three factors: problem definition, contextual knowledge, and the authoring mechanism. When requirements are clearly defined, context is stable and well understood, and the code creation method is reliable, IaC outcomes are significantly more consistent and secure. Conversely, weaknesses in any one dimension, such as ambiguous requirements, missing platform context, or unreliable AI generation, can lead to hallucinations, architectural drift, or security gaps. The model provides a practical framework for reasoning about manual, AI-assisted, and automated IaC workflows, while highlighting the continued need for expert oversight in complex environments.



Conclusions and Future Work

This study demonstrates that LLMs can generate deployable Terraform code for simple architectures when requirements are clearly defined and prompts are well-structured, but hallucination and issues increase as architectural complexity increases. Compared to AI-generated code, manually authored Terraform remains more stable, secure, and predictable, although it is more time-intensive. The proposed IaC Dependency Triangle demonstrates that IaC quality depends on problem definition, context, and the authoring mechanism. Overall, LLMs are valuable assistive tools, but expert oversight remains essential for complex infrastructure.

Future research should examine how developers with different levels of experience interact with LLM-generated IaC and evaluate performance. The use of AI-generated code also raises ethical concerns, particularly in academic and professional settings, which require further study. In addition, validation of the IaC Dependency Triangle is needed.

QR Code for Recording