

# CASE: A Computational Autonomous Systems Environment

Jeff Boakye  
Independent Researcher

## Abstract

Modern computing systems require users to interact through low-level, syntax-driven interfaces that are difficult to learn, error-prone, and potentially unsafe. Despite advances in programming languages and development environments, fundamental issues such as unsafe privilege escalation, fragmented tooling, and operating system dependence remain unresolved.

This paper introduces CASE (Computational Autonomous Systems Environment), a human-language, language-agnostic execution environment that operates as a secure computational layer above traditional operating systems. CASE allows users to express computational intent in natural language, which is translated into deterministic execution plans, automatically mapped to appropriate programming languages, and executed within isolated, capability-based sandboxes.

The system supports autonomous testing, security auditing, and performance benchmarking by default, and optionally enables decentralized execution across verifiable compute nodes. CASE does not replace existing operating systems, but abstracts their complexity, enabling safer, more accessible, and more reproducible computation across platforms.

## 1 Introduction

Computing systems have historically required humans to adapt to machine-oriented interfaces. Command-line terminals, programming languages, and privilege systems such as sudo impose cognitive and operational burdens that limit accessibility and introduce security risks.

CASE proposes a reversal of this paradigm.

## 2 Background and Motivation

Since early time-sharing systems of the 20th century, computation has relied on symbolic instruction languages. While powerful, these systems assume technical expertise and expose users to dangerous failure modes.

## 3 Problem Statement

Current systems suffer from:

- Syntax-heavy interfaces
- Unsafe privilege escalation
- Fragmented development environments
- OS-dependent tooling

## **4 System Overview**

CASE introduces an intent-driven execution layer that sits above traditional operating systems and abstracts their complexity.

## **5 Architecture**

CASE consists of:

- Intent Parser
- Execution Planner
- Language Router
- Secure Runtime (WASM + Containers)
- Verification Layer

## **6 Security Model**

CASE replaces root privileges with explicit capability grants, limiting access by default and enabling auditable execution.

## **7 Decentralized Execution**

CASE optionally supports execution across distributed nodes with verifiable and reproducible results.

## **8 Use Cases**

CASE applies to software development, security testing, education, AI tooling, and enterprise systems.

## **9 Comparison to Existing Systems**

CASE complements rather than replaces operating systems and development environments.

## **10 Limitations and Open Problems**

Challenges include natural language ambiguity, performance overhead, and governance of decentralized execution.

## **11 Roadmap**

Development proceeds in phases: MVP, multi-language support, decentralized execution.

## 12 Conclusion

CASE represents a shift toward human-centric, secure, and reproducible computation.