File 2603.25723, filed 26 March 2026, section: Agents & harnesses

Natural-Language Agent Harnesses

Linyue Pan, Lexiao Zou, Shuo Guo, Jingchen Ni, Hai-Tao Zheng

In brief

Represents an agent harness as an editable natural-language document, run by a shared runtime, and shows it matches code-based harnesses on coding, terminal and computer-use benchmarks.

Why builders should care

If harness logic can live in a readable document, you can inspect, compare and swap it like a config file.

The paper, in plain English

Agent performance depends heavily on the harness, the execution system around the model that organizes a task run. That logic is usually buried in tightly coupled controller code, which makes harnesses hard to inspect, compare and transfer.

The paper introduces Natural-Language Agent Harnesses (NLAHs): editable documents that describe run-level harness policy. A shared Intelligent Harness Runtime (IHR) interprets them into agent calls, handoffs, state updates, validation gates and artifact contracts.

Across coding, terminal-use and computer-use benchmarks, harnesses written this way reach comparable task outcomes to code and prompted versions, with much shorter static policies. Module ablations show explicit harness modules can be analyzed one at a time.

How to apply it

1. Move run-level policy (stages, handoffs, checks) out of controller code into a versioned document.
2. Make validation gates and artifact contracts explicit, so each stage knows what it must hand over.
3. Ablate one harness module at a time to see what actually moves your results.

Key results

Quoted from the abstract

> *Across coding, terminal-use, and computer-use benchmarks, IHR-executed NLAHs achieve comparable task outcomes to code and prompted realizations, while exposing much shorter static harness policies.*

> *Module ablations further show that explicit harness modules are analyzable.*

Summary written from the arXiv abstract. Read the original: https://arxiv.org/abs/2603.25723

Curated by Atharv Pawar