CD4AI: Regression Testing for AI-Enabled Systems

João Paulo Iasco[1][0009−0009−0199−9315],
Renato Cordeiro Ferreira[1,2,3,4][0000−0001−7296−7091], and
Alfredo Goldman[1][0000−0001−5746−4154]

[1] Institute of Mathematics, Statistics and Computer Science (IME), São Paulo, Brazil
[2] Jheronimus Academy of Data Science (JADS), 's-Hertogenbosch, The Netherlands
[3] Eindhoven University of Technology (TUe), Eindhoven, The Netherlands
[4] Tilburg University (TiU), Tilburg, The Netherlands
jppaulo11@usp.br
{renatocf,gold}@ime.usp.br

Abstract. This paper describes the *Continuous Delivery for Artificial Intelligence* (CD4AI) pattern, a strategy to balance deployment velocity and rigorous validation in AI-enabled systems. The pattern establishes a three-stage lifecycle for these systems: (1) *testing*: a Continuous Integration / Delivery (CI/CD) step combining automated tests and probabilistic evaluations, allowing new versions into production if they succeed at a minimum rate (e.g., 95% success rate); (2) *monitoring*: a monitoring step that collects real-world interactions for later analysis, flagging potential anomalies via automated heuristics; and (3) *curation*: a feedback loop step to gather candidate interactions that can be converted into new tests, incrementing the regression suite. By leveraging production observability as a source of truth for test generation, this pattern enables teams to maintain high deployment velocity while automatically evolving their test suites to match the complexity of real-world usage. This ensures both the testability and robustness of the system over time.

Keywords: AI-Enabled Systems · SE4AI · Software Engineering for Artificial Intelligence · CI · Continuous Integration · CD · Continuous Delivery · Software Testing · Large Language Models.

1 Intent

The primary intent of this pattern is to reconcile deployment velocity with system reliability in agentic AI. It establishes a feedback loop that accelerates the release of new features while utilizing production observability to rapidly identify failures. By systematically converting real-world errors into automated regression tests, the pattern ensures continuous system improvement and prevents the recurrence of known issues.

[2] [6] [1] [3] [4] [5]

2 Motivation

In high-velocity agile environments, the imperative to "ship fast" conflicts with the stochastic nature of AI agents. Unlike traditional software, agentic behaviors—governed by volatile prompts and tools—can shift drastically within a single sprint, rendering static validation strategies obsolete.

This creates a fundamental friction between traditional Machine Learning workflows and Agile Engineering:

- **Traditional ML:** relies on large, pre-curated datasets. This approach is incompatible with Agentic AI, where features are implemented faster than high-quality datasets can be constructed. Waiting to build a "Gold Standard" dataset for every new capability introduces unacceptable latency to the release cycle.
- **The Prediction Trap:** Attempting to preemptively create test cases for all possible agent interactions is inherently speculative. Synthetic data often fails to capture the "unknown unknowns" of user behavior, leading to a false sense of security before deployment.

Consequently, the challenge shifts from attempting to predict every failure mode to establishing a mechanism for rapid feedback and organic test evolution. Engineering teams require a pattern that accepts pre-deployment uncertainty and leverages production observability as the primary engine for constructing regression suites. This ensures that the test coverage evolves organically and at the same pace as the product itself.

3 Problem

How can an engineering team validate rapidly evolving AI agent behaviors when reliable evaluation datasets cannot be constructed prior to deployment?

4 Forces

The application of this pattern requires balancing the following conflicting forces:

- **Testability:** Traditional software relies on predefined inputs and deterministic outputs. In Agentic AI, predicting all valid user interaction paths pre-deployment is impractical due to the vast state space of LLMs. The architecture must enable verification mechanisms that do not rely solely on manually curated datasets, bridging the gap between synthetic testing scenarios and complex real-world usage.
- **Robustness:** AI systems are inherently stochastic. A key challenge is distinguishing between transient LLM noise and genuine logic regressions. The solution must provide a mechanism to "lock in" fixes for known failure modes, ensuring that once a specific edge case is identified and resolved in production, it becomes a permanent constraint that prevents future regressions.

– **Evolvability:** User interaction patterns with AI agents are emergent and often diverge from initial design assumptions. A static test suite inevitably develops a coverage gap over time as usage behavior drifts. The testing infrastructure must be capable of **organic growth**, enabling the continuous ingestion of production failure scenarios into the regression suite. This ensures that the system's quality assurance measures evolve in lockstep with the complexity of real-world usage.

5 Solution

The CD4AI pattern is designed to address two distinct categories of failure: (1) regressions in established behaviors, and (2) unanticipated failures emerging from real-world usage. Crucially, the pattern unifies these by treating the resolution of unanticipated failures as a form of regression prevention: once a unique failure mode is detected in production, it is codified into the test suite to ensure it never recurs.

The solution establishes a continuous feedback loop divided into three distinct stages:

5.1 Testing (The Cumulative Gate)

This stage occurs pre-deployment within the Continuous Integration (CI) pipeline. Its primary objective is regression prevention.

– **Mechanism:** The system executes a regression suite composed of both synthetic cases (created for new features) and historical cases derived from prior production failures.
– **Objective:** To strictly enforce the system's "known constraints." Deployment is conditional on meeting a strict success threshold (e.g., $> 95\%$ pass rate), ensuring that neither new changes nor edge cases discovered in the past compromise the system's stability.

5.2 Monitoring (The Discovery Net)

Once deployed, the focus shifts to identifying the "unknown." The objective is to identify potential anomalies in production interactions.

– **Mechanism:** The system observes production traffic, applying broad heuristics to flag "candidate failures." Examples include explicit negative user feedback, excessive tool usage loops, or guardrail violations.
– **Metric Strategy:** This stage prioritizes **High Recall**. The goal is to capture a wide range of candidate failures, accepting a higher rate of false positives to minimize the risk of missing silent defects.

5.3 Curation (The Regression Factory)

This stage bridges production operations and engineering. It filters candidate failures into actionable regression tests.

- **Mechanism:** A high-precision evaluation layer (e.g., a specialized LLM-as-a-Judge or human review) analyzes candidate failures to separate noise from genuine defects.
- **Objective:** To transform "unknown unknowns" into permanent regression assets. Validated failures are sanitized and converted into automated test cases that feed back into Stage 1. This ensures the system's test coverage evolves organically to match the complexity of real-world usage.

To maintain agility, the internal logic of the *Monitoring* and *Curation* stages must remain relatively stable. These stages should serve as consistent baselines for system health rather than requiring alteration with every single feature release.

6 Dynamics

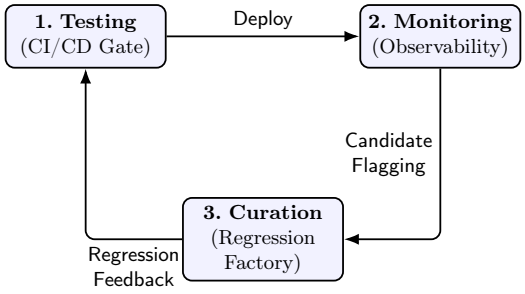

Fig. 1. Dynamics of the CD4AI pattern

References

1. Forsgren, N., Humble, J., Kim, G.: Accelerate. IT Revolution, 1st edn. (3 2018)
2. Fowler, M.: Refactoring. Addison-Wesley Professional, 2nd edn. (11 2018)
3. Humble, J., Farley, D.: Continuous Delivery. Addison-Wesley Professional, 1st edn. (7 2010)
4. Huyen, C.: AI Engineering. O'Reilly, 1st edn. (1 2025)
5. Lakshmanan, V., Hapke, H.M.: Generative AI Design Patterns. O'Reilly Media, 1st edn. (11 2025)
6. Sato, D., Wider, A., Windheuser, C.: Continuous Delivery for Machine Learning (9 2019), https://martinfowler.com/articles/cd4ml.html