

AI Development Harness

2026-09-21
SimpleModeling

AI development is reshaping how software is developed. To realize its potential in business applications, we need to evaluate AI-generated programs against business rules and quality requirements so that people can decide whether to adopt them.

The key is the concept of an AI Development Harness and the technologies that implement it.

This is the first installment of a ten-part AI Development Harness series. The series examines various harnesses that make the results of software engineering available to AI, and how those harnesses work together. This installment presents the background, challenges, shared context, and overall structure of the series.

1 How AI Changes Software Development

AI development offers several advantages:

- Requirements and specifications can be expressed in natural language, making ideas and business knowledge a starting point for development.
- AI can infer and fill in what a concise specification leaves unstated, bringing assumptions and alternatives into view.
- AI can explore existing systems and technical information, then apply relevant knowledge to design and implementation candidates.
- Programs can be generated at remarkable speed and scale.
- As a result, programs can be repeatedly rebuilt in a short time to try different designs and implementations.
- This opens the way to new applications that respond to ambiguous information and changing circumstances.

These advantages may be realized early in screen-centered Web applications whose business rules and state management are relatively simple.

2 Requirements and Challenges for Business Applications

Core business applications require consistency in system-wide rules and quality as well as in individual features.

Business applications must consistently satisfy business rules and invariants, authorization, state transitions, transactions, and auditability. They must also honor contracts with external

systems and maintain responsiveness, reliability, availability, security, and observability appropriate to their usage. These conditions extend across the system and its operation.

It is difficult to meet these conditions consistently through natural-language prompts and AI attention alone, leaving substantial challenges for applying AI development to core business applications.

When AI fills in a concise specification during development, three concerns need to be examined separately.

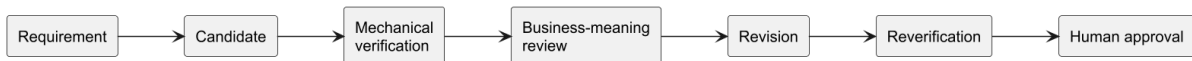
- Development efficiency: Can the requested change be completed with limited rework?
- Product quality: Does a change, including judgments supplied by AI, satisfy business correctness and the required quality attributes?
- AI compute cost: How much can context reconstruction, exploration, regeneration, and repeated review be reduced?

Faster candidate generation can increase review and correction work if it produces more candidates based on unsuitable assumptions or technologies. Relying on ever larger contexts for quality also requires substantial computation to reconstruct and reason over them each time.

To assess AI-supplied judgments and the technologies it applies against system-wide requirements, the assumptions and verification conditions must be made explicit.

If AI-supplied judgments are not retained as specifications or designs, and development relies only on natural-language exchanges and AI attention, a change in conversation history or reference information can lead to different implementations of the same requirement. Later installments will examine the design problems that arise when AI must also reconstruct procedures and exceptional behavior each time.

AI development should be evaluated by the time, computation, decisions, and rework required to move from generating a candidate to approving it as an adoptable change. Here, the total is called Cost to Convergence.



The task is to make AI-supplied content and verification conditions explicit, distinguish unsuitable candidates, and limit rechecking after a correction to what is necessary.

3 The AI Development Harness Approach

An AI Development Harness structures the meaning given to AI, its exploration scope, artifact formats, boundaries for state changes, verification methods, evidence, and approval. It makes judgments supplied through AI inference reviewable and guides exploration toward candidates that satisfy business rules.

Retaining requirements and AI-supplied judgments as models or Typed Intent allows candidates to be checked mechanically against the same criteria and reviewed for their business

meaning. After a correction, the changed and affected areas are verified again, and the results inform the approval decision. Reusing the criteria reduces review rework; defining the scope of reverification reduces AI computation spent on unnecessary exploration and regeneration.

A harness provides constraints that make the results of software engineering—models, architecture, types, DSLs, programs, runtimes, tests, reviews, and approvals—usable in AI-driven development. Within those constraints, AI can focus on its strengths in interpreting intent, exploring, forming hypotheses, generating candidates, and explaining, as well as on applications that only AI can enable.

Assurance at the scale of business applications requires determinism, reproducibility, and traceability. Each harness in the series gives concrete form to software engineering results as inputs, boundaries, artifacts, and verification.

A set of skills, tools, instructions, and guardrails for generative AI is one form of harness. This series uses the term more broadly. Harnesses take different forms where models are created, architectural responsibilities are assigned, programs are written, runtimes execute them, results are verified, and knowledge is consulted.

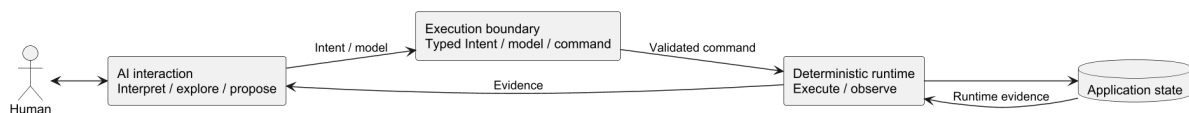
Combining harnesses across development connects AI interpretation, inference, and exploration to business rules, deterministic execution, evidence, and human approval. Together they provide a foundation for developing business-application-scale programs with AI.

The series has ten installments. This first installment sets out the advantages, the obstacles to applying them, the broader harness approach, shared vocabulary, and the series outline. The remaining nine installments examine harnesses for models, architecture, programming, runtimes, verification, and knowledge, and propose how to combine them in a feedback loop of verification and improvement.

Two principles are central:

- Prompts express intent; models are programs.
- Think nondeterministically; execute deterministically.

AI’s nondeterminism is useful for understanding intent, asking questions, exploring, forming hypotheses, generating candidates, and explaining. State changes, business rules, validation, authorization, persistence, workflows, retries, idempotency, concurrency, and compensation are instead entrusted to programs and runtimes that execute explicit meaning.



The distinction is between the stage in which AI interprets, explores, and proposes, and the stage in which state changes follow approved meaning. It clarifies which responsibilities allow nondeterminism and where deterministic execution becomes necessary.

4 Technologies and Products Used in the Series

The series uses the following technologies and products to show how harnesses work in practice.

- SimpleModeling: a development methodology that connects design, implementation, and execution around models.
- CML: the language used to describe models.
- Cozy: a tool that transforms models into programs, configuration, and documents.
- Textus: the software execution foundation.

AI is used for implementation choices that transformation rules alone do not determine.

5 Series Outline

The series comprises ten installments in four parts.

- Part I, Problem Definition (1): Why is an AI Development Harness needed?
- Part II, Constraining the Model Space (2-4): What should be modeled, and which structures and behaviors should be allowed?
- Part III, Constraining the Realization Space (5-8): How should responsibilities, knowledge, implementation, and execution be constrained?
- Part IV, Returning from Execution to the Authority (9-10): How should evidence be evaluated and returned for human approval?

The series that follows will address development process definition using the Essence framework.

6 Summary

- AI can use inference to complete natural-language specifications and find and apply technical information. Whether its judgments fit the requirements still needs separate verification.
- Explicit rules, verification, evidence, and human approval support the correctness and quality of business applications.
- Across ten installments, the series proposes various harnesses that make software engineering results usable by AI and shows how to combine them.

7 Next Installment

The second installment, "Model Harness," turns natural-language intent into models, Typed Intent, and commands that people and programs can examine.