



Jung Sungwoo

Posted on Dec 10, 2025 • Edited on Dec 12, 2025

# Manifesto: A Semantic World Model Interface for AI-Operated Software Systems

#ai #agents #architecture #webdev

You can try it right now.

[playable demo](#)

[github](#)

## 1-Page Technical Concept Summary

### 1. Problem

Current AI agents control software systems (SaaS, internal tools, workflows) through **vision, DOM heuristics, or textual reasoning**.

These methods fail because modern applications contain:

- Hidden state
- Distributed business logic
- Non-deterministic UI behavior
- Implicit constraints & policies
- Asynchronous dependencies
- Multi-step workflows with conditional transitions

From an AI perspective:

**The environment is not observable, not stable, and not structurally encoded.**

→ No agent can reliably build a functional world model of such systems.

Thus, AI cannot *plan, explain, or take actions safely*.

This is **not** a model limitation—it is a **representation problem**.



## 2. Core Idea

**Manifesto provides a formal, declarative world-model interface for software systems.**

It exposes the *semantics*, *state transitions*, and *action space* of an application in a deterministic, machine-interpretable structure.

Instead of forcing the model to infer business rules from pixels, DOM, or natural language, Manifesto makes those rules **explicit**:

Domain Semantics → Snapshot → Expression-based Rules → Action

In other words:

**Manifesto transforms software from a black-box UI into a white-box, symbolic environment.**

This is the missing substrate required for reliable AI agents to operate real-world software.

## 3. Representation Model

Manifesto formalizes a domain into four machine-interpretable namespaces:

### 3.1. `data.*`

Mutable user-level inputs.

### 3.2. `state.*`

System-level or async state (e.g., loading, error, fetched lists).

### 3.3. `derived.*`

Deterministic values computed via a **pure Expression DSL**:

- Comparable to Mapbox-GL expressions
- JSON-serializable
- Static dependency graph
- No side-effects
- Fully analyzable

### 3.4. `actions.*`

Side-effectful behaviors executed through structured **Effect graphs**:



- ApiCall
- SetValue / SetState
- Parallel / Sequence
- Conditional / Catch
- EmitEvent

Action **preconditions** represent domain policies (i.e., semantic constraints).

## 4. Deterministic Runtime

Manifesto's core runtime:

1. Builds a **Dependency DAG** from all expressions.
2. Computes a **Semantic Snapshot** of the domain:

```
{ data, state, derived, validity, timestamp, version }
```

1. Executes effects deterministically.
2. Regenerates snapshots after each mutation.
3. Emits **explainable causal traces** ("why is this action blocked?").

This creates a **stable, inspectable, reproducible environment**—a property no UI or DOM-based system has today.

## 5. Agent Interface

Manifesto exposes a unified agent-facing representation:

- Current world state (Snapshot)
- Action space with preconditions
- Policy violations and their explanations
- Expected outcomes of each action
- Type-safe input schemas
- Semantic metadata attached to every path

This enables:

- Planning
- Explanation
- Counterfactual reasoning



+

...

- Safety checks
- Repair strategies
- Autonomy under constraints

No inference from UI is needed; the agent receives a **structural world model** similar to RL environments in research—but directly connected to real software.

## 6. Why This Matters for AI

### 6.1. Symbolic x Neural Integration

Manifesto provides the symbolic substrate AI systems have lacked:

- Structured state
- Deterministic transitions
- Explicit rules
- Finite action space

LLMs reason over these structures much more reliably than raw UI observations.

### 6.2. The Missing Layer Between LLMs and Applications

Existing AI stacks:

LLM  $\leftrightarrow$  (DOM / Vision / Heuristics)  $\leftrightarrow$  Application

Manifesto replaces the brittle middle layer with a **formal, semantic interface**:

LLM  $\leftrightarrow$  Manifesto World Model  $\leftrightarrow$  Application

### 6.3. Enables True Agent Autonomy

Because the agent knows:

- what it *can* do (action space)
- what it *should not* do (preconditions)
- what *will happen* if it acts (effect semantics)
- why something *failed* (explain tree)

It gains an unprecedented level of controllability and safety.

## 6.4. Enables Real-World Generalization

Every SaaS domain becomes a standardized environment:

- Agents can transfer patterns across domains.
- A universal semantic layer emerges.

---

## 7. Key Insight

AI does not require an LLM to *infer* the structure of software systems.

Software systems already **have** structure—it simply isn't exposed.

**Manifesto exposes that structure.**

By doing so, it provides:

- A computable world
- A declarative logic layer
- A deterministic transition model
- A machine-consumable ontology
- A safe action interface

This is the missing link that allows AI to **act**, not just **predict**.

---

## 8. Summary Sentence

Manifesto is a formal semantic interface that transforms real software systems into deterministic, explainable world models —enabling safe and generalizable AI agents to operate them.

### Top comments (2) ⚡

[Code of Conduct](#) • [Report abuse](#)

 Sentry PROMOTED

...

The screenshot shows the Sentry interface for monitoring application performance. The main area displays a trace for a project creation request. The trace tree shows various components and their execution times, such as middleware.django, db queries, and http.client calls. A detailed transaction view is open, showing the SDK configuration (sentry\_javascript.cloudflare) and trace data. Below the trace, tags and logs are listed.

Traces > [be3a75067857](#)

**default**  
mcptool/create\_project

Issues | Search in trace | Jump to: Tags Log

DEV | Explore | Dashboards | Insights | Settings

Search in trace: 2a0d98c0:3600:103 3729334 cloudflare cloudflare

1 Trace: 2a0d98c0:3600:103 3729334 | cloudflare | cloudflare

1. [mcp.tool/create\\_project](#) | Autogrouped | 100ms

2. [http.client - POST https://sentry.io/api/0/teams/mcprun-test/](#) | Autogrouped | 796.00ms

3. [http.server - /api/0/teams/\(organization\\_id|or|slug\)/\(tz|\)](#) | Autogrouped | 729.7ms

4. [db - SELECT `sentry\\_apikenn`.\\*`sentry\\_apikenn`](#) | Autogrouped | 726.25ms

5. [db - SELECT `select\\_array\\_aggregation`.\\*`select\\_array\\_aggregation`](#) | Autogrouped | 0.94ms

6. [db - SELECT `select\\_array\\_aggregation`.\\*`select\\_array\\_aggregation`](#) | Autogrouped | 2.38ms

7. [db - SELECT `sentry\\_appinstallation`.\\*`sentry\\_appinstallation`](#) | Autogrouped | 2.17ms

8. [db - SELECT `sentry\\_appinstallation`.\\*`sentry\\_appinstallation`](#) | Autogrouped | 1.87ms

9. [db - SELECT `sentry\\_appinstallation`.\\*`sentry\\_appinstallation`](#) | Autogrouped | 2.04ms

10. [middleware.django - process\\_request](#) | Autogrouped | 711.36ms

11. [middleware.locale - process\\_request](#) | Autogrouped | 0.08ms

12. [middleware.django - sentry.middleware.ratelimit.\\_call](#) | Autogrouped | 710.77ms

13. [rate limiting.\\_call - bf8f21cf5fa2e](#) | Autogrouped | 710.74ms

14. [middleware.django - sentry.middleware.ratelimit.\\_process](#) | Autogrouped | 710.65ms

15. [middleware.django - sentry.middleware.ratelimit.\\_process](#) | Autogrouped | 0.01ms

16. [middleware.django - sentry.middleware.ratelimit.\\_process](#) | Autogrouped | 0.01ms

17. [db - SELECT `sentry\\_organization`.\\*`sentry\\_organization`](#) | Autogrouped | 2.05ms

18. [sentrygate.proxy.\\_request\\_durs](#) | Autogrouped | 706.72ms

19. [http.client - POST http://0.0.0.0:9000/projects/mcprun-test/](#) | Autogrouped | 704.84ms

20. [middleware.ratelimit.\\_process](#) | Autogrouped | 0.00ms

21. [http.client - POST https://sentry.io/api/0/projects/mcprun-test/](#) | Autogrouped | 248.00ms

x Close | Reset Zoom

**Transaction**  
ID: 3501d5f52c244ed9bd25d7eb4e3374 ⚡

**SDK**

- SDK Name: [sentry\\_javascript.cloudflare](#)
- SDK Version: 9.16.1
- Cache Avg Value Size: 0.0 B

**Trace Data**

```
mcpt.param_0 {"organizationSlug": "mcprun-testing", "teamSlug": "mcprun-testing", "name": "test-project", "platform": "javascript"}  
mcpt.param_1 {"signal": {}, "requestId": 2}  
sentry.environment manual  
sentry.sample_rate 1  
sentry.source custom
```

**Tags**

| Tag            | Value                          |
|----------------|--------------------------------|
| durable_object | True                           |
| environment    | cloudflare                     |
| level          | info                           |
| mcp            | info                           |
| server_version | 0.7.1                          |
| organization   | <a href="#">mcprun-testing</a> |
| slug           | mcprun-testing                 |

**Logs**

Search logs for this event

```
> May 9, 11:53:23 AM [sentryApi] POST https://sentry.io/api/0/projects/mcprun-testing/test-project/keys/  
> May 9, 11:53:22 AM [sentryApi] POST https://sentry.io/api/0/teams/mcprun-testing/mcprun-testing/projects/
```

# Monitoring your MCP Server in Production (with Sentry).

A walkthrough mcp error handling, navigating Cloudflare setup quirks with Sentry, and tracing to connect all the dots.

[Read more →](#)



# Jung Sungwoo

UX/UI Software Engineer

**More from Jung Sungwoo**

The Mind Protocol: Why Your AI Agent Needs a World Before It Can Think

#ai #architecture #typescript #agents

## Buttons Are Not Functions



#ai #frontend #architecture #agents

Why Your AI Agent Says "Done" But Nothing Actually Happened

#agents #ai #automation #llm

Sentry PROMOTED

## Smarter debugging with Sentry MCP and Cursor

No more copying and pasting error messages, logs, or trying to describe your distributed tracing setup or stack traces in chat. MCP can investigate real issues, understand their impact, and suggest fixes based on the actual production context.

[Read more →](#)