Jung Sungwoo
Posted on Jan 3

❤️‍🔥 1

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

#agents #ai #automation #llm

You know that feeling when your AI agent confidently declares a task complete, but when you check the UI... nothing changed?

The button's still there. Unclicked.

The form's still empty. Unsubmitted.

The page hasn't moved.

But your logs show:

```
✓ Task completed successfully
✓ All steps executed
✓ Goal achieved
```

If you've built UI automation with LLMs, you've seen this. Probably more times than you'd like to admit.

And here's the thing everyone gets wrong at first:

It's not a model problem.

The Real Problem: We're Asking Agents to Decide What's True

Most agent frameworks do something like this:

```
# Agent decides when it's done
if agent.thinks_task_is_complete():
    return "done"
```

Seems reasonable, right?

But here's what actually happens in the real world:

- Buttons can be disabled
- Forms can fail validation
- Network requests can be rejected
- DOM can change mid-action
- Timing can break everything

Executing an action ≠ Achieving a state

Yet we keep writing code that treats them the same.

The Bug That Keeps Coming Back

I've debugged this pattern in multiple production systems now:

```
// ❌ This looks fine but fails in practice
async function completeTask(agent, ui) {
  const steps = agent.planSteps();

  for (const step of steps) {
    await ui.execute(step);
  }

  return { status: "complete" }; // 🚨 Lies
}
```

The agent executed all its steps.

But that doesn't mean the UI actually changed.

Maybe the click event didn't fire.

Maybe the API call got rate-limited.

Maybe JavaScript just... didn't.

The agent declared victory. Reality disagreed.

Why Bigger Models Won't Save You

When this happens, the instinct is:

- "Let's add more CoT reasoning"
- "We need a better model"

- "Maybe fine-tuning will help"

I've tried all of these. They don't fix it.

Because this isn't an intelligence failure.

It's an **authority failure**.

We gave the agent the power to say "this is true" about a world it can't fully observe.

That's like letting a process decide it successfully wrote to disk without checking the file system.

The Fix: World-Verified Completion

Instead of this:

```
// Agent declares done
if (agent.saysTaskComplete()) {
  return "done";
}
```

Do this:

```
// World confirms done
if (
  agent.intent === "submit_form" &&
  ui.state.formSubmitted === true &&
  ui.state.validationPassed === true
) {
  return "done";
}
```

The difference is subtle but critical:

- **Before**: Agent decides when reality is true
- **After**: Agent proposes, world verifies

This is how operating systems work (user/kernel separation).

This is how databases work (ACID guarantees).

This is how distributed systems work (consensus protocols).

Why wouldn't AI agents work the same way?

A Real Example

I built this into TaskFlow, a demo app where agents manage a task list.

The key architectural shift:

```
// Agent proposes intent
const intent = agent.decide(snapshot);

// Core validates and computes next state
const result = compute(schema, snapshot, intent);

// Only then does state actually change
if (result.valid) {
  snapshot = result.nextSnapshot;
}
```

The agent never directly mutates state.

It only proposes changes.

The world decides what's actually true.

Result: **Invalid actions dropped from 80%+ to 0%** in testing.

Not because I used a better model.

Because I stopped asking the model to be the source of truth.

What This Means for Your Code

Three practical changes:

1. Separate intent from execution

```
// ❌ Don't mix these
async function clickButton() {
  await button.click();
  return true; // who verified this?
}

// ✅ Separate clearly
async function clickButton() {
  const intent = { type: "click", target: "submit" };
  const result = await execute(intent);
  return result.verified; // world checked
}
```

2. Make state observable

```
// ❌ Hidden state
let formSubmitted = false; // where? can agent see this?

// ✅ Explicit snapshot
const state = {
  form: { submitted: false, validated: false },
  ui: { currentPage: "form", loading: false }
};
```

3. Define completion as predicates

```
// ❌ Implicit
if (noMoreSteps) return "done";

// ✅ Explicit
const isDone =
  state.form.submitted &&
  state.ui.currentPage === "success";
```

The Deeper Pattern

This isn't just about UI agents.

It's about **who has authority over truth**.

In reliable systems:

- Components propose
- The world verifies
- State is explicit
- Changes are traced

This applies to:

- Tool-using agents (did the API call succeed?)
- Multi-agent systems (who decides what happened?)
- Human-in-loop (when do we need verification?)

Try It Yourself

If you're building UI agents, ask yourself:

1. Can the agent declare "done" without world verification?
2. Is there hidden state the agent can't observe?
3. Can actions fail silently?

If any answer is "yes", you're vulnerable to this bug.

The good news: it's fixable with architecture, not better prompts.

I'm working on Manifesto – a framework built around these principles. The core idea is simple: **explicit state, verified transitions, traced execution**.

Early specs and demos are live. Feedback welcome.

What's the worst "agent said done but nothing happened" bug you've hit? Drop a comment – I'm collecting war stories for the next post.

Top comments (0)

Code of Conduct • Report abuse

Jung Sungwoo

UX/UI Software Engineer

LOCATION
Korea, Seoul

JOINED
Sep 4, 2024

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

World-Centric Agent Architecture: Why Your AI Agent Keeps Failing (And It's Not the Model's Fault)
#ai #architecture #llm #agents