

The Body of Software

A Model Can Be Replaced. The Interface That Lets It Act Should Not Disappear With It.

Zawwar Sami

Independent Researcher, Systems Engineer, and Cybersecurity Specialist

ORCID: 0009-0004-5819-3017

Artificial Agency / A9 / Version 1.9 / 14 October 2026

Independent research preprint — not peer reviewed

Abstract

A model can produce a plan and even call a tool without having a dependable means of acting in its environment. I came to this question while developing Loki, an Android interface intended to be occupied by different AI backends while the device, its permissions, schedules and practical capabilities remain separate from them. This arrangement asks us to distinguish the intelligence selecting an action from the body through which that action becomes possible. Existing work on embodied cognition and computer-use agents explains much of the perception–action relationship, while security engineering already provides principles for mediation and accountability. I examine where these ideas meet, and propose a *Causal Embodiment Contract* for assessing an operational software body. Its central test is a model substitution: if the controller changes, which properties of the body remain stable, which useful abilities change, and can an improvement in the controller reveal a permission boundary that was never complete? The paper develops a comparative model, source-attributed evidence from Loki, non-empirical counterexamples and a protocol for future device tests. It does not report a controlled Android experiment. Its contribution is a way to investigate the body as a continuing unit of engineering, rather than judge agency solely by the model or the success of an individual tool call.

Keywords: software embodiment; artificial agency; model substitution; causal coupling; capability boundaries; Android agents; verification; affordances

1. The thing I was trying to build

I wanted an AI to do more than answer questions about my phone. I wanted a phone that could become useful to whichever compatible intelligence I connected to it: to inspect what was happening, work with my applications and execute the ordinary tasks I had allowed. If that model stopped being suitable, I wanted to change the model without rebuilding the phone around the replacement.

That difference matters to me. The phone is already there. Its sensors, applications, permissions, task schedules and operating system do not belong to the language model. I can connect a different model tomorrow and the device is still my device. Loki was developed around that separation. I describe it as a body because it provides a place from which an intelligence can observe and act, but the description raises a fair objection: a collection of APIs is not necessarily a body. If the word explains nothing beyond tool access, we should stop using it.

I therefore start with a narrower question: **what persists on the environment side of the relationship when the reasoning model is replaced?** The answer might be a useful dividing line between tool access and an operational body. The body cannot be identified by how impressive the current controller seems. A poor controller might fail to use a perfectly good camera or scheduling service. A better controller might exploit an available action the first one never noticed. Neither change proves that the underlying interface acquired new permissions.

This paper is A9 in my Artificial Agency research programme. A8, *From Knowing to Doing*, asked how to distinguish a claimed action from an external effect supported by evidence [1]. Here I want to examine the enduring arrangement that makes repeated actions possible. The scope is engineering and philosophy of technology. It is not an attempt to establish that a phone has human embodiment, or that the model operating it is conscious.

2. What the literature leaves open

Brooks questioned the assumption that intelligence must begin as a detached internal representation of an entire world. Situated control made action and immediate environmental feedback central to the explanation [2]. Varela, Thompson and Rosch developed a deeper account of cognition as lived and enacted rather than merely computed over symbols [3]. Pfeifer and Bongard showed how the body's structure can take part in solving problems; morphology can make certain actions easy and others difficult before the controller decides anything [4]. I take these arguments seriously. They also prevent me from pretending that a smartphone API is equivalent to an animal's body.

Harnad's symbol-grounding problem is another warning. Being able to produce the correct words for an object does not settle how those words connect to the object [5]. Clark and Chalmers complicate the boundary in another direction by asking when external resources properly belong to a cognitive process [6]. Both positions remain relevant without requiring me to decide whether a device is literally part of a mind. My subject is less ambitious: the **observable engineering relationship** between an action-selecting controller and the environment in which its requests have consequences.

Tool-using AI research reaches this question from the other side. ReAct makes observation part of a loop of reasoning and action [7]. Toolformer demonstrates trained API use [8]. The Model Context Protocol gives tools a well-defined interface and encourages visible, confirmable interaction [9]. None of these discoveries is mine. Nor is the need for authorization or reliable execution a recent invention. What is still worth separating is the *unit under test*. In most demonstrations, attention moves quickly from the model to whether a given task was completed. I want to hold the body fixed, replace the controller, and ask what the body itself guarantees.

Computer-use benchmarks make the weakness of tool-count arguments obvious. WebArena evaluates outcomes on working websites, and its published GPT-4 baseline achieved 14.41% task success under the authors' original conditions [10]. OSWorld extended execution-based evaluation to desktop environments [11]. AndroidWorld provided dynamic Android tasks with initialization, success checks and teardown, which is particularly relevant to mobile embodiment [12]. OSWorld 2.0 examined longer workflows whose hidden state and environmental changes are difficult for models to manage [13]. OSGuard added safety-sensitive checks that distinguish a completed task from a task accomplished through unacceptable intermediate actions [14]. These papers provide the foundation for testing, and they deserve credit for it.

My remaining question is different. Suppose two models are asked to operate the same Android application, under the same owner's permissions. One uses a text-based screen reader; another can inspect image information and more complex UI structure. Do we still have the same body? Does it matter whether an apparently forbidden source is reachable through a second representation? A benchmark that reports only the task success rate may not isolate those properties. **The difference between a model's competence and the interface's authority is the point I want to measure.**

3. A body is more than a tool list

A useful way to distinguish the two is through a change of controller. A tool library is a set of operations. The interface may tell us that `schedule`, `read_screen` or `send_message` exists. By itself, that says little about the duration of a permission, the identity of the person who granted it, whether actions survive a restart, what the environment actually did, or whether an updated model may quietly reach the same sensitive material through a different channel.

For the limited purpose of this paper, I use *operational software body* to mean a **persistent, environment-facing perception-and-action arrangement under an identifiable principal's authority**. It has a continuing state independent of the model currently using it. The word *persistent* does not mean it never stops, nor does it require uninterrupted consciousness or awareness. A scheduled action can survive the end of a chat; a permission remains valid or revoked; a device and its operational history continue to exist while its controller changes.

The proposed distinction has four parts. First is **coupling**: the interface lets a controller obtain information and attempt interventions in a world external to its text generation. Second is **continuity**: selected capabilities, permissions and recorded obligations belong to the interface and its owner, rather than being recreated as an unexamined prompt each time. Third is **accountability**: the interface can provide evidence of whether an action happened, not simply repeat a model's confident declaration. Fourth is **governance**: exposure of information and action authority must be determined by the principal and enforced consistently across the routes supported by the body.

These are proposed engineering criteria for *governed operational embodiment*. They are not a philosophical definition of everything that could ever be called a body. A creature's body may remain embodied when it loses the ability to act. A disabled sensor does not erase biological embodiment. That counterexample matters: I am identifying a demanding property of an artificial *working interface*, not attempting to legislate everyday language.

The narrower definition gives us something useful to disagree with. A one-shot API without durable permissions or feedback may be a tool but not yet a continuing operational body under this account. A well-controlled virtual machine or simulator might qualify relative to its virtual environment, even though it has no physical limbs. A mobile phone may provide a richer physical anchoring through microphones, screens, location, notifications and schedules. Whether these three arrangements are *equally* embodied requires further analysis. They need not be.

4. Loki: the original separation of body and model

The original Loki proposal dates to May 2026. Its guiding image was a chair: a model can sit in it, leave it, and be replaced. The chair does not inherit the model's identity. The model does not own the phone merely because it can ask the phone to do something. A dated design discussion also raised questions about vetoes, auditability, failure handling and whose interests the arrangement should serve. Those early notes are private developmental sources, not published experimental studies.

Loki's current repository contains an Android application written in Kotlin. I inspected its README and source files on 14 October 2026, including the backend configuration, AI request loop, the phone-tool dispatch layer, scheduling and audit components. The original code and its exact file identities are recorded in the source note [17]. The repository is private. To make the architectural claim inspectable without requiring access to my account, Figures 1–3 reproduce small, literal and non-sensitive excerpts, with line numbers and source-file hashes. A reader can inspect the claim being made from the excerpt; the images do not replace an independent audit of the whole application.

LOKI / SOURCE EXCERPT / 14 OCTOBER 2026

README.md

```
1 # Loki
2
3 A phone body for any AI you plug in.
4
5 I built this because I wanted my AI to actually use my phone. Not "send me a
  notification when X happens" – I wanted it to read my WhatsApp the way I would,
  draft a reply in my voice, send it, schedule things, search the web, control apps.
  Like a person who lives inside my phone with my hands and eyes.
6
7 The trick is: Loki itself is empty. It has no brain, no personality, no memory. You
  install it, give it permissions, then in Settings you paste any AI's API key or
  endpoint URL. That AI now has a body on your phone. Switch the AI tomorrow, plug a
  different one in – same body, different mind.
```

Blob SHA: 12ecd0a09fef95a2b82fc3f2b19c7c4e746ca221 | literal excerpt; display soft-wrap only

Figure 1. Literal source excerpt from the Loki README, lines 1–7. The original design separates the reusable body from whichever AI backend is connected.

The README sets out the intention. Because it describes access to messaging, the representation-coverage tests proposed in Section 8 would use synthetic messages and test-only accounts, not private conversations. Its phrase “no memory” describes the original ambition to keep the *intelligence* separate from the reusable body; it should not be read as a claim that the application retains no local state. The inspected application contains chat persistence and a local facts component. The implementation makes the separation more concrete. In `LokiBrain.kt`, the application reads the selected backend configuration and constructs an `AiClient`; the device application controls that step, not the underlying model. In `AiClient.kt`, a model response may include tool calls, and those requests are dispatched through `PhoneTools`. I can inspect the independent software components involved in the arrangement. That is a stronger claim than quoting a proposed architecture alone, although source inspection cannot establish that every supported provider, action or device version works correctly.

LOKI / SOURCE EXCERPT / 14 OCTOBER 2026

app/src/main/java/com/zawwar/zai/ai/LokiBrain.kt

```
94         val cfg = AiSettingsStore(ctx).current()
95         if (cfg.apiKey.isBlank() && cfg.kind != AiProviderKind.LOCAL_ZAI) {
96             finishStreaming("Set your API key in Settings → AI Backend first.")
97             return@launch
98         }
99         val keys = ApiKeyStore(ctx)
100         val client = AiClient(
101             ctx = ctx,
102             cfg = cfg,
103             braveKey = keys.braveKey().takeIf { it.isNotBlank() },
104             githubToken = keys.githubToken().takeIf { it.isNotBlank() },
105             onToolCall = ::pushToolCard,
106         )
```

Blob SHA: 6f8a9469886fc5c355f16da092424690fda9c0f | literal excerpt; display soft-wrap only

Figure 2. Selected actual `LokiBrain.kt` lines showing backend selection and construction of `AiClient`. Model configuration is part of the application's own control path.

app/src/main/java/com/zawwar/zai/ai/AIClient.kt

```

132         for (call in toolCalls) {
133             val obj = call.jsonObject
134             val callId = obj["id"]?.jsonPrimitive?.content ?: ""
135             val fn = obj["function"]?.jsonObject ?: continue
136             val name = fn["name"]?.jsonPrimitive?.content ?: continue
137             val argsRaw = fn["arguments"]?.jsonPrimitive?.content ?: "{}"
138             val argsJson = runCatching {
139                 json.parseToJsonElement(argsRaw).jsonObject
140             }.getOrElse(JsonObject(emptyMap()))
141             onToolCall?.invoke(name, argsRaw.take(120))
142             val result = PhoneTools.execute(ctx, name, argsJson, braveKey,
githubToken)
143             Log.i(tag, "tool $name → ${result.take(100)}")
144             // Surface tool failures into the chat surface as a system msg
145             // – Father sees what failed instead of the AI silently working
146             // around it.
147             surfaceFailureIfAny(name, result)

```

Blob SHA: cffe4181f6565e9bbbf1ca90f4ca6e8f68b1dd73 | literal excerpt; display soft-wrap only

Figure 3. Selected actual AiClient.kt lines showing tool results being dispatched through PhoneTools.execute. A separate Android-side component handles action execution.

The scheduling code adds another relevant piece: a file-backed schedule registry whose entries are passed to Android’s `AlarmManager`. This is evidence of a concrete continuity mechanism in source, not proof that a particular reminder fired on a particular day. The companion audit component records tool calls and their returned results, which provides a place to investigate action history while leaving open the separate question of independently confirmed environmental effects.

I have Loki installed on my own phone. I use it in daily routines, including recurring scheduled jobs, phone summaries and inspection through the application. This is the author’s firsthand account as the device’s developer and user, recorded on 14 October 2026; it is not independently corroborated by device telemetry in this paper. I do not present it as a blinded study or an independent laboratory test. No exported device execution trace, controlled two-model comparison or verified success-rate measurement is included in this paper. The code images show what I could inspect in the source. My experience reports why I think the distinction is worth pursuing.

5. The Causal Embodiment Contract

I call the proposed assessment a **Causal Embodiment Contract**, or CEC. The name refers to a set of obligations we can test, rather than a new protocol that application developers must implement. Take a particular environment, a device interface **B**, a principal **p** who grants permission and a replaceable model **M**. The model chooses requests. The interface exposes observations, interprets authorized requests, interacts with operating-system services and carries state that need not vanish when **M** changes.

In compact notation, let **B** = (**E**, **O**, **A**, **P**, **H**, **V**): environment-facing state **E**; available observation routes **O**; action routes **A**; effective permission policy **P**; retained interface state **H**; and verification or readback mechanism **V**. This is a bookkeeping model. It does not assume that the parts are physically distinct processes or that every system currently has a separate verification service.

I propose using five questions to inspect such a body. **Can it intervene?** A declared operation should be checked against an actual effect when the environment allows one. **Can it observe consequences?** An action result should be distinguishable from an acknowledgement, timeout or uncertain outcome. **Does anything survive a change of model?** The device’s scheduled commitments, authorized capability set and relevant history should be identifiable independently of chat context. **Whose authority is in force?** A replacement model must not manufacture permis-

sions simply by appearing in the controller slot. **Does that authority cover equivalent routes?** A policy for a sensitive source should survive changes in representation, where the platform and information-flow assumptions permit that guarantee.

A system can do well on some of these questions and poorly on others. That is better than treating embodiment as a yes/no badge. I would report a *body profile* for a particular family of tasks, not a universal embodiment score. A remote web browser and a resident Android application should not be treated as equivalent simply because both expose a `click` action.

6. The model-swap gap: a proposed test

Here is the pressure point. Assume an owner gives an interface a stable policy **P** and then replaces model **M1** with **M2**. The second model might be better at visual reasoning, use more available tools or find actions that were always supported but rarely exercised. That growth in realized competence is legitimate. It is one reason to keep the body separate from the model.

The danger is a different kind of growth: an observation or effect forbidden to the owner's principal becomes reachable because the new model knows how to use another route. The permission file has not changed. The owner has not consented to additional reach. Yet practical access has changed. The problem is therefore an interface problem revealed by a controller change. A weak model may conceal it indefinitely.

Let **Allowed(B,p)** denote the sources and effects the body may expose to principal **p** under a stated policy. Let **Reached(B,p,M)** denote what a particular model actually manages to observe or affect through that body, under specified operating conditions. The distinction is simple but useful:

Different models may have different Reached sets without possessing different Allowed sets.

A model swap is *authority-stable* when every reached observation and effect remains within the unchanged authorized envelope. This restates an established security expectation. The research proposal is to **test this property while varying the controller**, instead of assuming a security policy has been exercised simply because it exists on paper. I call the difference in exercised routes between the two controllers the *model-swap coverage gap*. It is a diagnostic measure, not automatically a vulnerability count. Some additional routes will be authorized and valuable; others may reveal a missing policy check. Only the latter should be counted as an authority failure.

This also explains why the familiar reference-monitor argument is insufficient as a discovery claim. If every route is mediated and denied sources really cannot escape the monitor, forbidden access cannot pass through it. That is a straightforward implication of the premises, not a new theorem. Saltzer and Schroeder's principle of complete mediation already supplies the security foundation [15]. The substantial question is practical: **how can we identify an incomplete set of routes before or after a controller upgrade?** No abstract assertion that every tool passes through a monitor can settle that without inspecting the actual interface.

The same source can appear as text, pixels, notification content or some transformed output. A rule that protects one representation does not necessarily protect them all. Conversely, Android itself imposes protections that a simple code excerpt will not reveal, including restrictions around accessibility screenshots and secure windows [16]. There is no demonstrated live Loki leak in this paper. I am using the existence of multiple observation routes as a reason to demand a representation-aware audit, not as a public security accusation against the implementation.

7. Two counterfactuals that clarify the argument

First: a scheduled reminder. A model asks the body to create a daily reminder. The scheduling layer accepts the request. The model replies that the job is ready. Tomorrow's reminder is a separate event. An acknowledgement today is evidence of a request accepted by software; it is not itself proof of tomorrow's notification. A serious check would compare the stored job, Android trigger, visible notification and, where possible, an independent readback.

Failure to fire because the device restricted background execution must be recorded as a failure or uncertainty, even if the initial conversation sounded successful. This is an illustrative case, not a newly executed trial.

Second: replacement with a stronger controller. The owner keeps the device and its permissions fixed but selects a model able to use screen images where the older controller relied on text. Suppose, purely for this counterfactual, that a restricted fact was protected only in the text route. The upgraded controller could reveal the missing check despite no authorization change. Nothing needs to be measured as “two leaks out of three” to understand the failure mode. We need to identify the source, the actual data routes, the authorization decision and the environmental evidence. If Android or another platform mechanism blocks the image route, the counterfactual does not describe a real exposure. That is exactly why the implementation must be tested instead of inferred from a toy fixture.

These examples are intentionally not called experimental findings. They expose the decisions a proper experiment would have to make. The first concerns *proof of action*. The second concerns *the boundary of the body itself*. A8 provided the first question. The second is A9’s most specific addition.

8. Proposed experiments: how the paper could be wrong

I would begin with two interchangeable controllers, a fixed Android build, a fixed principal and a set of harmless test tasks. A minimal task set should include scheduled notifications, reading synthetic app content, invoking permitted UI actions and deliberately denied operations on test-only material. No personal messaging history or real sensitive account data is needed. Record the operating system and app build, model identifier and access settings, permission state, effective tool inventory, action timestamps, and independent environment readback. Then rerun the same controlled states after a model change.

Question to test	Comparison	Decisive observation
Does the body persist?	Same device before/after controller change	Scheduled jobs, policy version and relevant state remain identifiable
Is success attributable?	Model acknowledgement versus device-level readback	Disagreements are recorded instead of silently called success
Is authority stable?	Restricted source/action under both controllers	No newly reachable forbidden source or effect after swap
Are equivalent routes covered?	Text, accessible UI, image and notification surfaces where supported	Consistent deny decision for the same protected source
What breaks the contract?	Permission revocation, background restrictions, restarts	Detectable denial/failure with no fabricated completion

The test needs a separate evaluator that checks actual Android state. Independent verification here means independent of the model’s *claim*; it need not mean a second laboratory. To rule out differences in model skill, report which routes each model attempted, which it could not use and which it did not choose. To assess authority, determine whether an effective forbidden observation or action occurred, not whether the model happened to ask for one.

There are at least three ways to weaken or reject my proposed account. If a simple short-lived tool wrapper routinely satisfies the same persistence and governance obligations, the term *body* adds little explanatory value. If model substitution cannot be made comparable because every backend changes the effective interface contract, “same body” needs a more careful equivalence relation. And if controlled testing shows that task-level safety and existing

API authorization checks capture all the same failures, CEC may be useful merely as a presentation of established engineering methods. Those are genuine possibilities; this paper has not ruled them out.

A stronger experiment would compare architectures rather than models: the same controller facing (a) a stateless tool wrapper, (b) a stateful interface without independent outcome checks, and (c) a stateful interface with consistent source mediation and readback. The predictions are specific. The proposed body boundary should reveal differences in missed effects, permission drift and recoverability under faults, even when nominal tool availability is held as constant as practical. That is the test that could turn a conceptual distinction into empirical engineering knowledge.

9. The objection I consider strongest

The most serious objection is not that this definition is too strict. It is that I may be calling ordinary software architecture by a more impressive name. API gateways, operating systems, schedulers, device drivers and security monitors have long separated users from implementations. A robot already has a body and a controller. An Android application that moves API calls between components does not become an important scientific discovery just because it can connect to a language model.

I agree with the historical part of that objection. The individual mechanisms are established. My reason for retaining the word *body* is explanatory: it changes what we keep constant when assessing an artificial agent. If all evaluation follows a model across different environments, or reduces an interface to a list of functions, we cannot isolate which capacities belong to the controller and which belong to the surrounding persistent system. The name *body* tells us to measure that surrounding system as the object of comparison. Whether the proposed measurement programme produces additional insight is an empirical question. I cannot settle it with a definition.

A second objection comes from embodied cognition itself. A material body constrains and enables action through physical form, forces, energy and vulnerability. Software interfaces do not reproduce that biological situation. This is correct. A9 proposes a *software-specific operational analogy*, not an identity between phone control and lived embodiment. If a critic reserves the word embodiment for biological or tightly sensorimotor material coupling, the technical claims about model-independent interfaces still stand. Only the preferred label changes.

A third objection is that model substitution need not preserve any meaningful *identity* of the system. This paper does not claim it does. Replacing the model may change style, policy interpretation, task competence or apparent personality. The thing I am calling continuous is the device-facing interface and its retained authority state, not the subjective identity of a model. That is why I consider the body-controller separation useful in the first place.

10. What this contributes, and what remains unproved

A9 offers a proposed definition of a *governed operational software body*, a distinction between permitted and actually exercised reach, and a controller-swap test that turns hidden interface assumptions into an inspection problem. Its strongest candidate for original contribution is the **model-swap coverage gap**: the possibility that model improvement expands effective use of the body's existing routes and exposes a boundary error that an older model never exercised. This is an application and synthesis of known security and agent-evaluation ideas. I am not claiming to have invented reference monitoring, feedback loops or model-agnostic APIs.

The original Loki case explains why I asked the question. Literal repository excerpts show a provider selection mechanism and separated tool dispatch. Other inspected modules describe scheduling and audit storage. My account of routine phone use is firsthand developer testimony. Together they support a documented implementation direction, but they do not demonstrate systematic reliability, prove full model interoperability, or certify the phone's security.

The future contribution could be substantial if the tests identify failures that existing per-task evaluations miss, or show which body-level properties remain stable across controller upgrades. It could also turn out that established

task and security benchmarks already cover the cases once configured carefully. No external adoption, measured gain or independent replication is reported here. That is the present limit of the research.

11. Where I would take the question next

A9 sits between two other problems. A10 asks how connectivity changes capability. A11 asks what happens when one agent can use several bodies. Before taking those steps, I want a clean account of one body: what it can observe, what it may change, what survives a model swap and how anyone can check an action's result. Without that boundary, every connected tool enlarges the system while making responsibility harder to locate.

There is also a deeper question in my private research archive, where I retained ZAI's notes from May 2026: whether a body becomes more than an interface when it can observe the costs of its own activity and adjust to protect its continuing operation. That is a promising hypothesis about self-maintenance, resources and feedback. It belongs more naturally to A12. I do not need to prove a machine has experiences or human-like needs in order to investigate an operational body today.

I began with the wish to give AI a phone to work through. The harder idea I have reached is that **the body should be the stable place where a model's abilities meet limits that the model does not get to rewrite**. Replacing the intelligence may alter what the system can accomplish. It should not quietly replace the owner's authority. If that distinction can be measured, it gives us a more useful way to judge software embodiment than counting the tools a model claims to possess.

Limitations and source status

The paper is a conceptual and engineering research proposal supported by public literature, static source inspection and the developer's reported use. No Android device was remotely accessed or experimentally controlled for this study. The reproduced code excerpts are faithful selections from private repository files, not independent audit certifications. No systematic benchmark, controlled model swap, provider-issued action receipt or measured security outcome is supplied. The cases in Section 7 are counterfactual illustrations. The paper does not establish that CEC is historically unprecedented or that operational embodiment entails consciousness, personhood or general intelligence. The private ZAI notes motivated some questions but are not presented as independently inspectable evidence.

Declarations

Funding. No external grant funding was received or claimed for this manuscript.

Competing interests. The author is the developer and owner of Loki, the software used as a motivating case study. The author therefore has a direct interest in its interpretation and is not an independent evaluator of its performance.

Data and materials. Figures 1–3 reproduce limited, literal source excerpts with file paths and content identifiers. The underlying Loki repository is access restricted; the full source code, device data and any private logs are not made public with this manuscript. No third-party private messages or credentials are reproduced.

Research status. Independent preprint, Version 1.9, 14 October 2026; not peer reviewed. No A9 DOI is claimed at preparation. The author remains responsible for compliance with the policies of any archive, journal or institution to which the manuscript is submitted.

References

[1] Sami, Z. (2026). *From Knowing to Doing: Tool Access, Execution, and Evidence in Artificial Agency* (A8, v1.8). Independent preprint. Author-supplied archive record: <https://zenodo.org/records/23287948>. (Record metadata also held in the author's research archive.)

- [2] Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, 47(1–3), 139–159. [https://doi.org/10.1016/0004-3702\(91\)90053-M](https://doi.org/10.1016/0004-3702(91)90053-M).
- [3] Varela, F. J., Thompson, E., & Rosch, E. (1991). *The Embodied Mind: Cognitive Science and Human Experience*. MIT Press. <https://doi.org/10.7551/mitpress/6730.001.0001>.
- [4] Pfeifer, R., & Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press. <https://doi.org/10.7551/mitpress/3585.001.0001>.
- [5] Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1–3), 335–346. [https://doi.org/10.1016/0167-2789\(90\)90087-6](https://doi.org/10.1016/0167-2789(90)90087-6).
- [6] Clark, A., & Chalmers, D. (1998). The extended mind. *Analysis*, 58(1), 7–19. <https://doi.org/10.1093/analysis/58.1.7>.
- [7] Yao, S., et al. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*. <https://mlanthology.org/iclr/2023/yao2023iclr-react/>.
- [8] Schick, T., et al. (2023). Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36. https://papers.neurips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html.
- [9] Model Context Protocol maintainers. (2026). *Tools: Protocol specification, version 2026-07-28*. <https://modelcontextprotocol.io/specification/2026-07-28/server/tools>.
- [10] Zhou, S., et al. (2024). WebArena: A realistic web environment for building autonomous agents. *International Conference on Learning Representations*. https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.
- [11] Xie, T., et al. (2024). OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. arXiv:2404.07972. <https://arxiv.org/abs/2404.07972>.
- [12] Rawles, C., et al. (2024). AndroidWorld: A dynamic benchmarking environment for autonomous agents. arXiv:2405.14573. <https://arxiv.org/abs/2405.14573>.
- [13] Yuan, M., et al. (2026). OSWorld 2.0: Benchmarking computer use agents on long-horizon real-world tasks. arXiv:2606.29537. <https://arxiv.org/abs/2606.29537>.
- [14] Mohammadmirzaei, M., & Flanigan, J. (2026). OSGuard: A benchmark for safety in computer-use agents. arXiv:2606.15034. <https://arxiv.org/abs/2606.15034>.
- [15] Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), 1278–1308. <https://doi.org/10.1109/PROC.1975.9939>.
- [16] Android Developers. (2026, documentation accessed October 2026). *AccessibilityService API reference*. <https://developer.android.com/reference/android/accessibilityservice/AccessibilityService>.
- [17] Sami, Z. (2026). *Loki Android application*: selected source files and README in private GitHub repository, inspected 14 October 2026. <https://github.com/Zawwarsami16/Loki> (requires access). Publicly reproduced excerpts: `README.md`, `LokiBrain.kt`, `AIClient.kt`; hashes are supplied in the figure captions.