Oddkin_HAMs

The Evolving Landscape of AI Model Types
and the Next Frontier in Task-Executing AI

Where we are today

The Core Gap (all model modalities)

Across LLMs, agents, multimodal systems, and robotics models:

No existing model is optimized for:

- Verifiable task completion
- Persistent responsibility
- Failure detection and repair
- Outcome-based learning
- Escalation under uncertainty

In short: Today’s models generate outputs. None are trained to reliably *own outcomes*.

Why This Matters To Humans

Real-world delegation will require models that:

- Track commitments over time
- Act in environments with uncertainty
- Detect when plans fail
- Repair or escalate appropriately
- Optimize for *completion*, not plausibility

This is not a scaling problem. It is a **missing abstraction problem**.

Current models weren't designed, trained, or incentivized to take responsibility for real-world outcomes—so they don't

Modern models are trained to optimize **proxy losses** (token likelihood, reward models, preference scores), not **verifiable task outcomes**.

1. They are optimized for the wrong objective

Modern models are trained to optimize **proxy losses** (token likelihood, reward models, preference scores), not **verifiable task outcomes**.

You get fluency, not responsibility, because that's what the loss function rewards.

2. They are trained on static data, not execution trajectories

Most training data consists of:

- text
- images
- code
- demonstrations *without outcomes*

They rarely include:

- real-world state transitions
- confirmation of success or failure
- recovery attempts
- escalation decisions

3. They lack persistent world-state grounding

Current models operate inside:

- short-lived contexts
- ephemeral memory
- self-reported success

They do not maintain:

- durable task identity
- long-horizon commitments
- verified state representations

4. Failure is invisible during training

Training corpora contain:

- successful completions
- clean examples
- polished outputs

Failures, retries, and repairs are underrepresented or filtered out.

5. Safety and liability incentives discourage action

Large labs deliberately avoid training models to:

- take irreversible actions
- own outcomes
- act autonomously under uncertainty

This minimizes legal and safety risk but caps execution capability.

Introducing Healing Action Models (HAMs)

A new class of AI model designed for real-world task execution

What Makes HAMs Different

1. Outcome-Conditioned Intelligence
HAMs are trained to optimize for *verifiable task completion*, not token likelihood or response quality.

2. Persistent Responsibility
HAMs maintain memory across time: commitments, partial progress, failures, constraints, and permissions.

3. Action, Not Orchestration
Actions are learned behaviors, not brittle prompt chains or scripts.

4. Healing-by-Design
Failure detection, recovery, and escalation are first-class model objectives—not edge cases.

5. World-State Grounding
HAMs reason over structured world state and verify that actions actually changed reality.

What's already real technical work (that HAMs would stand on)

Outcome-conditioned policies (decision-making-sequence-modeling)

There's a strong body of work reframing control as "condition on desired outcomes / future states" using transformers and offline RL. Decision-Transformer-style approaches and follow-ons are exactly "policy learning" rather than chat (see: Markov Decision Processes and https://www.youtube.com/watch?v=2iF9PRriA7w)

What's missing vs HAMs: these methods usually live in **closed simulators** with clean state; they don't natively handle messy real-world tools, permissions, partial completion, or human escalation.

Relevant Reading:

https://proceedings.neurips.cc/paper_files/paper/2023/file/51053d7b8473df7d5a2165b2a8ee9629-Paper-Conference.pdf

https://proceedings.mlr.press/v202/xie23b/xie23b.pdf

https://arxiv.org/html/2410.03408v1

https://openreview.net/forum?id=biRNo676X7

Action reasoning models in robotics (vision → plan → actions)

Work like **MolmoAct** explicitly extends VLM/LLM backbones to produce grounded, temporally structured action sequences (and reasoning traces) for manipulation. That's very aligned with "models that act," not "models that talk."

What's missing vs HAMs: robotics is still **embodiment-locked**, data-expensive, and usually not optimized for general "delegated task responsibility" across digital + physical tool interfaces.

Relevant Reading:

https://arxiv.org/html/2508.07917v1

https://allenai.org/blog/molmoact

https://huggingface.co/allenai/MolmoAct-7B-D-0812

"Self-healing" / recovery-oriented agent research (mostly software systems)

There's explicit research framing "self-healing" as diagnosing degradation and selecting corrective actions; e.g., NeurIPS work on self-healing ML systems, and newer work on self-correcting language-model agents and evaluation for recovery.

What's missing vs HAMs: this tends to be **system-level** (controllers, runtime error handling) rather than a unified **model objective** tied to verified outcomes across tasks.

Relevant Reading:

https://proceedings.neurips.cc/paper_files/paper/2024/file/4a86ec12e94ef1fe306362e7bdcd5894-Paper-Conference.pdf

https://arxiv.org/pdf/2509.25238

So are HAMs “new”?

The name: yes, it’s our coined definition.

The substance: no—HAMs are best positioned as a **new unifying abstraction** that combines:

- **Outcome-conditioned policy learning** (from offline RL / decision transformers)
- **Grounded action generation + intermediate planning** (from Vision-Language-Action (VLA) reasoning models)
- **Recovery / self-healing as a first-class target** (from self-healing ML + self-correcting agents)
- **Plus** the piece that’s least standardized today: **verifiable completion under delegation** (explicit “done-ness,” escalation, irreversibility penalties, persistent commitments) as the core eval + training signal.

That last bullet is where we credibly claim novelty: not “we made agents,” but **we formalized and trained for responsibility + completion + repair** as the central objective.

HAMs are “outcome-oriented execution policies.”

“Outcome-Oriented” (this is the key difference)

Most AI models optimize **proxy objectives**:

Model Type	Optimizes For
LLMs	Next-token likelihood
Agents	Tool invocation correctness
Planners	Plan optimality
VLMs	Perception accuracy
Robotics	Control reward functions

HAMs optimize for outcomes. That means:

- Success is defined by *external state change.*
- The model receives reward only if the world is measurably different in the intended way.
- Partial success, failure, and recovery are all explicitly modeled.

If the model:

- Writes an email but it never sends → failure
- Schedules a meeting but wrong time → failure
- Books something but doesn’t notify stakeholders → failure
- Encounters an error and doesn’t repair → failure

Execution” (not planning or advising)

Execution means:

- Calling tools
- Triggering APIs
- Sending messages
- Scheduling, updating, coordinating
- Taking irreversible steps
- Waiting for confirmations
- Retrying or repairing when things break

An execution model is judged on:

- *Did the thing actually happen?*
- *Was it done safely and correctly?*
- *Did it recover when it didn’t?*

Not on fluency or reasoning elegance.

“Policy” (in the RL / control sense)

A **policy** is a function that decides *what action to take next* given the current situation.

Formally:

$$\pi(a_t \mid s_t, g, m_t)$$

- s_t = current world state
- g = goal / commitment
- m_t = memory (what’s already been tried, constraints, failures)
- a_t = next action to execute

This is very different from a language model, which outputs **tokens**, not actions.

LLMs optimize probabilities. HAMs optimize consequences.

A Concrete Example

“Reschedule my dentist appointment to next week and let my partner know.”

LLM (Advice-Oriented)

- Explains how to do it
- Maybe drafts a message
- Claims success

Agent (Orchestration-Oriented)

- Calls scheduling API
- Sends message
- Assumes success
- Breaks if API response changes

HAM (Outcome-Oriented Execution Policy)

- Checks existing appointment state
- Identifies valid time windows
- Attempts reschedule
- Verifies confirmation
- Updates calendar
- Sends notification
- Confirms partner received message
- Detects failure if any step breaks
- Repairs or escalates
- Ends only when predicates are true

Why This Is a New Model Class

Because **outcome orientation changes everything**:

Dimension	LLMs / Agents	HAMs
Objective	Plausible output	Verified completion
Time horizon	Single turn / short loop	Until done
Memory	Ephemeral	Persistent
Failure	Ignored or hallucinated	Detected and repaired
Evaluation	Human judgment	World-state changes
Trust	Low / fragile	Grows / Earned through execution

HAMS are outcome-oriented execution policies: models trained to take responsibility for real-world tasks by acting, verifying, repairing, and completing—not just advising.

Why Agentic Architectures Don’t Solve This

1. Agents orchestrate behavior; they don’t learn execution

Agentic systems are typically:

- an LLM
- wrapped in prompts
- with tool-calling logic
- plus retries and heuristics

All the “intelligence” about what to do when things go wrong lives in:

- prompts
- control code
- hand-written policies

When an agent fails, it fails the same way next time unless an engineer intervenes.Orchestration ≠ learning.

2. Agents optimize steps, not outcomes

Agents are implicitly optimized for:

- calling the right tool
- producing the next reasonable step
- following a plan

They are not optimized for:

- verifying the task actually completed
- staying responsible over time
- repairing failures
- minimizing irreversible mistakes

They stop when the loop ends, not when the world changes.

3. Agents have no grounded notion of “done”

In agentic systems:

- “done” is inferred
- success is assumed
- completion is often self-reported

There is no canonical, external definition of:

- completion predicates
- world-state verification
- obligation release

If the system can’t answer “how do you know this is done?” it cannot be trusted.

4. Failure handling is heuristic, not trained

When agents encounter failure:

- retry
- re-prompt
- ask the user
- switch tools

These behaviors are not learned from data. They are brittle and don’t improve with experience.

HAMs, by contrast, train directly on failure and recovery trajectories.

5. Agents don’t accumulate responsibility over time

Agentic loops are:

- session-bound
- context-window-limited
- memory-light

They do not maintain:

- long-lived commitments
- task ownership across days or weeks
- accountability for downstream consequences

Once the loop ends, responsibility ends.

6. Agents avoid irreversible action by design

Agent frameworks intentionally:

- sandbox actions
- simulate effects
- require human confirmation
- avoid autonomy under uncertainty

That’s good for safety — but it means:
They never cross into true delegation.

Notes

What You Explicitly Do *Not* Do in Year 1

This is critical.

You do **not**:

- Build a consumer app
- Brand it as a personal assistant
- Chase demos
- Compete on chat quality
- Market “agents”
- Over-index on embodiment

Those are downstream.

What Investors Should Believe After 12 Months

If you execute this plan, the belief is:

“Oddkin is doing for computational execution what early vision labs did for computational perception — defining the abstraction, the benchmark, and the training loop. Products come later.”

That’s a fundable lab.

We make a consumer product **after** Healing Action Models demonstrate *reliable responsibility*, not before.

The “Now It Makes Sense” Moment… consider a consumer product only when:

1. **Verifiable task completion is boring**
    - The model reliably finishes tasks
    - Success is no longer surprising
    - Failures are rare and well-handled

2. **Recovery works without prompt hacks**
    - Failure detection is automatic
    - Repair improves over time
    - Humans are consulted, not burdened

3. **Escalation is calibrated**
    - The model knows when *not* to act
    - Users feel safe delegating
    - Irreversible mistakes are extremely rare

4. **You can define a narrow promise**
    - One job the system does end-to-end
    - Clear boundaries
    - Clear success criteria

Why OpenAI (and Similar Labs) Are Unlikely to Do This

OpenAI optimizes for scalable intelligence and safe advice; Healing Action Models optimize for responsibility and real-world execution—those are fundamentally different goals.

Why OpenAI (and Others) Are Unlikely to Build Healing Action Models

- **Different optimization target**
  Large labs optimize for general intelligence and safe advice; HAMs optimize for outcome ownership, execution, and recovery. These goals are structurally misaligned.
- **Execution increases liability**
  Models that take real-world actions create legal, regulatory, and safety exposure that large platforms deliberately avoid.
- **Training requires interactive environments**
  HAMs need execution substrates, trajectory-level data, failure-heavy training, and verification loops—high-friction investments that don’t scale with existing data pipelines.
- **Agents are a local maximum**
  Tool-using agents deliver value quickly without changing training paradigms, making them “good enough” for incumbent roadmaps.
- **Business models conflict**
  Execution models end tasks; platform models maximize ongoing interaction, usage, and engagement.
- **Benchmark ownership doesn’t reinforce platform dominance**
  HAMs are moated by benchmarks, data loops, and outcome definitions—orthogonal to platform network effects.
- **History favors focused labs**
  New model classes almost always emerge from small, focused labs willing to ignore incumbent incentives.

Why Now: Healing Action Models

1. Foundation Models Are Finally Capable Enough: LLMs and multimodal models now possess sufficient reasoning, perception, and generalization to serve as the cognitive core of execution systems—something that was not true even 3–5 years ago.

2. The Limits of "Agents" Are Now Obvious: Tool-based agents plateau quickly: brittle orchestration, poor recovery, and no learning from execution have exposed a ceiling that cannot be fixed with prompting alone.

3. Infrastructure for Real-World Execution Now Exists: APIs, automation layers, and verifiable digital state make it possible to measure real-world task completion at scale without robotics-first constraints.

4. Data Scarcity Is No Longer the Bottleneck: Synthetic environments, trajectory mining, and preference learning enable scalable training on execution behavior—even when real-world data is expensive.

5. The Cost of Non-Execution Is Now Visible: As AI systems move closer to delegation, failures to act, recover, or escalate correctly create real operational and trust gaps that advisory models cannot bridge.

6. Large Labs Are Structurally Misaligned: Major foundation-model labs optimize for safe, general intelligence—not outcome ownership—leaving space for focused labs to define execution-first model classes.