DECISION-ZONE · TECHNICAL BACKGROUND

Mission Computing

Runtime causal discovery, and why it is not prescription

Rajeev Bhargava · Decision-Zone Inc. · November 2026

ABSTRACT

All computing to date executes stored prescription. Traditional systems have humans author that prescription; agentic systems have frontier models author it. In both, the running system executes frozen code and discovers nothing. Mission computing does not prescript. Intent directs the attention of a frontier model to discover the causal relationships that bear on a mission and to propose actions at runtime, inside a governed interaction between a persistent Requester that holds the mission and an independent Governor that computes admissibility on the present. Nothing is frozen into rules. This paper sets out the technical background: the prescription paradigm and its limit, the mission computing architecture, the precise sense in which it is non-prescripted, and why it cannot be reconstructed by arranging actors.

1 Output calculation execution: the common root

Every computing system in use today performs one operation. It calculates an output and executes it. Store, retrieve, calculate, emit. Databases do it, applications do it, services do it. We call this output calculation execution, and it has been the only paradigm.

In traditional engineering, the intelligence sits in people. A human observes the world, forms a causal model of it, and freezes that model into prescripted code. The code runs, logs accumulate, and later a human analyzes those logs, discovers what the model missed, and re-authors the code. The discovery is human, it is offline, and it is historical. The running system itself discovers nothing. It executes what was decided before it ran.

2 The agentic extension

Frontier models are the apex of output calculation execution. They provide natural-language access to calculation, a natural-language query over everything that has been written. A request is stated in words, and the model computes an output.

Agentic AI applies this to authorship. The model reads data and writes the prescripted code, or it selects and sequences a set of tool calls fixed in advance. The author has moved, from a human to a model. The structure has not moved. At runtime, the system still executes frozen prescription.

So traditional and agentic computing are the same where it counts. Each discovers first, freezes the discovery into code, and then executes the freeze. The only difference between them is who

writes the freeze. **Agentic AI did not remove prescription. It automated its authorship.**

3 The limit of prescription

A prescripted system can meet only what its author foresaw. A situation the author did not anticipate either breaks the system or waits for new code to be written.

The deeper limit is structural. Prescripted code executes and terminates. It produces an output and stops. It holds nothing. No part of the system keeps a purpose alive across time and reconciles it with a changing present. The purpose lives outside the system entirely, carried by the people who hold it by hand. This is not a failing of those people. It is a property of a technology that was built to compute actors and to leave the mission to humans.

That gap, between a frozen rule and the live present, is where prescripted systems fail and must be patched, again and again, for as long as they run.

Figure 1. Three paradigms on one axis: where discovery happens, and what runs.

	Traditional engineering	Agentic AI	Mission computing
WHO DISCOVERS	Humans, from logs	The model, from data	Model attention, set by intent
WHEN	Before runtime, offline	Before runtime, offline	At runtime, continuously
WHAT RUNS	Frozen prescripted code	Frozen prescripted code	A governed loop, nothing frozen
THE PRESENT	Read after the fact	Read after the fact	Computed on, as it happens

4 Mission computing

Mission computing begins from intent, not instruction. The author states what is to be held true, the mission, rather than the steps to carry out.

Intent compiles the mission into a running structure, and at runtime it directs the attention of a frontier model to discover the causal relationships that bear on the mission now, and to propose the action that serves it. The attention of the model is the discovery mechanism. Intent is what points that attention at the mission rather than at a stored answer.

The structure that holds and bounds this is a triad.

- **The Requester is persistent.** It holds the mission and continuously computes what is required now. It does not terminate. This is what makes it a Requester and not a function call.
- **Actors propose actions.** A frontier model, directed by intent, is such an actor. Its attention discovers the causal relationships and proposes the action that serves the mission on the present.

- **The Governor is independent.** It computes the admissibility of a proposed action against the present. It admits, or it does not.
- **A commit boundary separates proposal from effect.** Only admitted actions take effect.

The figuring out happens while the system runs, against what is actually happening, and it never becomes a stored rule. This is runtime computing. Discovery and governance, both computed on the present, continuously, for as long as the mission is held.

5 The precise sense of non-prescription

Mission computing is non-prescripted in a strict sense, and the distinction carries the whole architecture.

- **The Governor does not look up stored rules.** It computes admissibility on the present. Its invariants are properties to be held, not entries in a table to be matched.
- **Governance is not in the model.** No frontier model contains a Governor, and none can be given one by training. The Governor is external structure. This is why admissibility is computed rather than generated, and why it cannot be talked around.
- **An actor cannot defeat a mission.** Because the Governor sits outside the action space and admits on the present, an actor can only propose. What takes effect is what the Governor admits, and nothing else.

The model operates at runtime, but only as a proposing actor. Its power is discovery, not decision. The decision, the commit, belongs to structure that is not a model.

6 Why it cannot be assembled from actors

The common attempt to reproduce this is to build a Requester and a Governor as actors, agents given those names. This does not produce a weak triad. It produces no triad.

An actor computes an output and terminates. The Requester never terminates. It holds. A holder cannot be built from a thing whose nature is to finish. An actor sits inside the action space. The Governor sits outside it, at the commit. A boundary cannot be built from a participant. The moment the Governor is made an actor, other actors can out-propose and override it, and the invariant collapses, because actors defeat actors. A Governor built as an actor also runs logic to emit its decision, which is prescription returning under a new name.

So what exists outside this architecture is actors, which is output calculation execution, which is the paradigm that was always there. The triad and runtime computing are not present elsewhere to be built badly. **This is absence, not imitation.**

7 What this establishes

Mission computing is a distinct category of computing, not an improvement inside the prescripted one. For the first time, a system holds a purpose and reconciles it with the present continuously, without freezing that reconciliation into rules.

It also changes what a frontier model is for. The model is no longer asked to write the program, and it is no longer asked to be the decision. It is asked to discover, live, inside a governed mission. Two intelligences are at work, the one that compiles intent and the one that runs on the present, and neither is a stored rule.

The demonstrations follow directly from this. A mission system meets situations no one scripted because nothing was scripted. The causal relationship is discovered at runtime, not retrieved from a frozen rule. What looks, from the outside, like adaptability is in fact the absence of prescription.

> Prescription froze the solution and ran it. Mission computing holds the mission and discovers the solution on the present, every moment, and never freezes it.