DECISION-ZONE · RUNTIME COMPUTING

Decided in advance, or decided on the present.

Rajeev Bhargava, Founder, Decision-Zone Inc. · November 2026

There are two ways to run a system toward a goal. One follows decisions made in advance. The other computes each decision at the moment, on the present. They look the same until something no one planned for arrives. This explains the difference, and walks through the simulator that shows it.

PART 1

The difference

Every system that acts toward a goal is built one of two ways.

PRESCRIPTED RUNTIME EXECUTION

It runs a script by itself.

The decisions are written in advance: a rulebook, a plan, a policy, a trained model. The system executes them at runtime. Its autonomy is in the doing, it runs the prescription without a human. This builds **agentic autonomy**.

NON-PRESCRIPTED RUNTIME COMPUTING

It computes each decision on the present.

No script. A Persistent Requester computes what the goal requires now. An Independent Governor admits only a move that holds the goal now, from what is available. Its autonomy is in the deciding. This builds **mission autonomy**.

For every situation that was anticipated, both work, and they look identical. The difference appears only when the unforeseen arrives: the case the script was never written for. The prescripted system has no decision for it, or applies one that no longer fits, and it breaks. The non-prescripted system computes a decision for it, because it computes decisions rather than retrieving them. It reads the present and determines what holds the goal now, so it holds on the case no one planned for.

The piece that makes this possible is a **Governor**. There is no governor inside a script, an agent, or a trained model. Their restraint is whatever was written or trained into them in advance, applied by the same thing that acts. A Governor is a separate component with separate authority that decides, on the present state, whether a move holds the goal. That separation is what holds the unforeseen, and it is what the prescripted world does not have.

> A script holds only what was written. A Governor holds the case that arrives. The difference is whether the decision is retrieved from the past or computed on the present.

PART 2

What the simulator shows

Both systems hold one goal: keep a value inside a safe band. The same events hit both at the same time. The left system runs a fixed rulebook. The right system runs a Requester and a Governor. Nothing is pre-timed, both compute live, and the unforeseen arrives on its own.

1. **Normal operation**
   Disturbances push the value around, and both systems hold it in the band. Every event so far is one the rulebook was written for, so there is no visible difference. That is expected. The foreseen is handled by both.

2. **The unforeseen arrives**
   The big correctors fail under sustained load, a case the rulebook never anticipated. This happens by itself, you do not have to trigger it.

3. **The script breaks**
   The left rule still calls for the corrector that is gone, and it has no fallback. The value drifts out of the band and the breach count climbs. The panel reads: no Requester, no Governor, one rule, executed.

4. **The Governor holds**
   On the right, the Requester flags that the value is outside the goal and a correction is required. The Governor evaluates every available corrector against the goal on the present: it denies the one that is unavailable, denies the ones that move the wrong way, and admits a

smaller corrector that lands back in the band. It commits that one. The value holds, and the breach count stays at zero.

WHAT TO WATCH

- **The two lines.** The scripted line leaves the band and turns red; the governed line stays inside it.
- **The breach counts.** They diverge on their own, the scripted side climbing, the governed side holding at zero.
- **The right panel's mechanism.** The Requester stating what is required now, and the Governor admitting or denying each corrector, every cycle. That is mission autonomy working in the open.
- **The left panel's absence.** No Requester, no Governor, a single rule looked up and executed. That is agentic autonomy, and why it cannot hold the unforeseen.

The divergence is not staged. Both sides compute the same disturbances live, and the outcome follows only from whether each decision is retrieved from a script or computed on the present. That is the whole demonstration: the script and the Governor are identical until the moment the present stops matching the past.

Agentic autonomy holds what was written. Mission autonomy holds what arrives. One is a script executed by itself. The other is a Governor deciding on the present. The value is in the second, and it is a category of its own: Runtime Computing.

Decision-Zone Inc. · Intent, Causality, Control · companion to the Runtime Computing simulator. UMB-001, the Universal Mission Builder, is the product entrance to Runtime Computing.