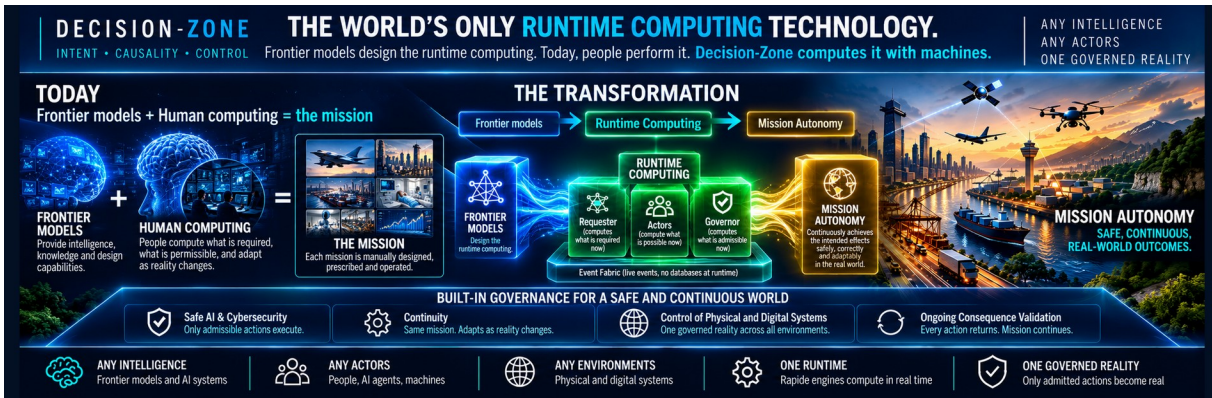




THREE REALMS

Keeping the requester, the thinker and the governor apart. In a life, and in a machine.

Nobody who proposes an action should also approve it.

That is a way of living before it is an architecture.

It is also the one thing missing from every intelligent system built so far.

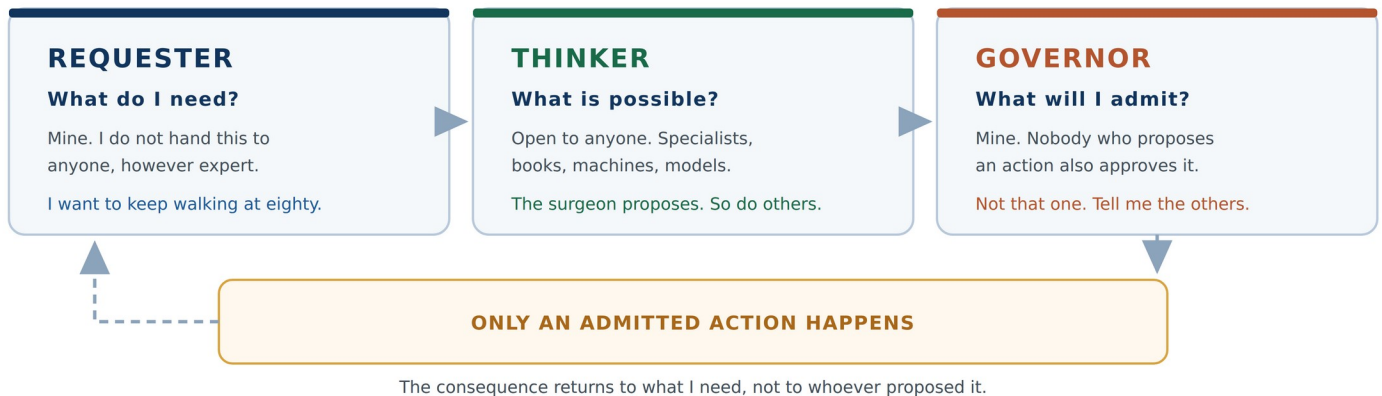
A note written alongside the Runtime Computing architecture.

The separation is lived before it is engineered

Three questions run in a life, all the time. What do I need. What is possible. What will I admit. They are different questions, they belong in different hands, and keeping them apart is a discipline rather than a theory.

A specialist is a source of possibilities. That is the whole of their role and it is a considerable one. What they are not is the person who decides what is required, or the person who approves the action.

THREE REALMS, KEPT SEPARATE, IN AN ORDINARY LIFE



WHAT MOST PEOPLE DO INSTEAD

They hand all three to the specialist at once. Tell me what is wrong, tell me what to do, I will do it.
The specialist becomes requester, governor and actor in one person. The patient becomes the thing acted on.
Nothing in that arrangement is computing what the person actually needs.

The collapse is the normal case

Handing all three to whoever has the expertise is what most people do, in medicine, in money, in work. It feels like trust. What it actually does is leave nothing computing what the person needs, because the only party still running is the one proposing.

Three ways a life or a system fails

Each of the three can run on its own, and each produces a recognizable failure. None of them look like each other, and only one of them is usually called a problem.

Only the actor running is compulsion: doing, with nothing watching. Only the governor running is paralysis: refusing, with nothing required. Only the thinker running is endless possibility: capability with no judgment attached to it.

ONE PART RUNNING ALONE. THREE DIFFERENT FAILURES.



THE THIRD ONE IS THE MOST COMFORTABLE PLACE TO SPEND A CAREER

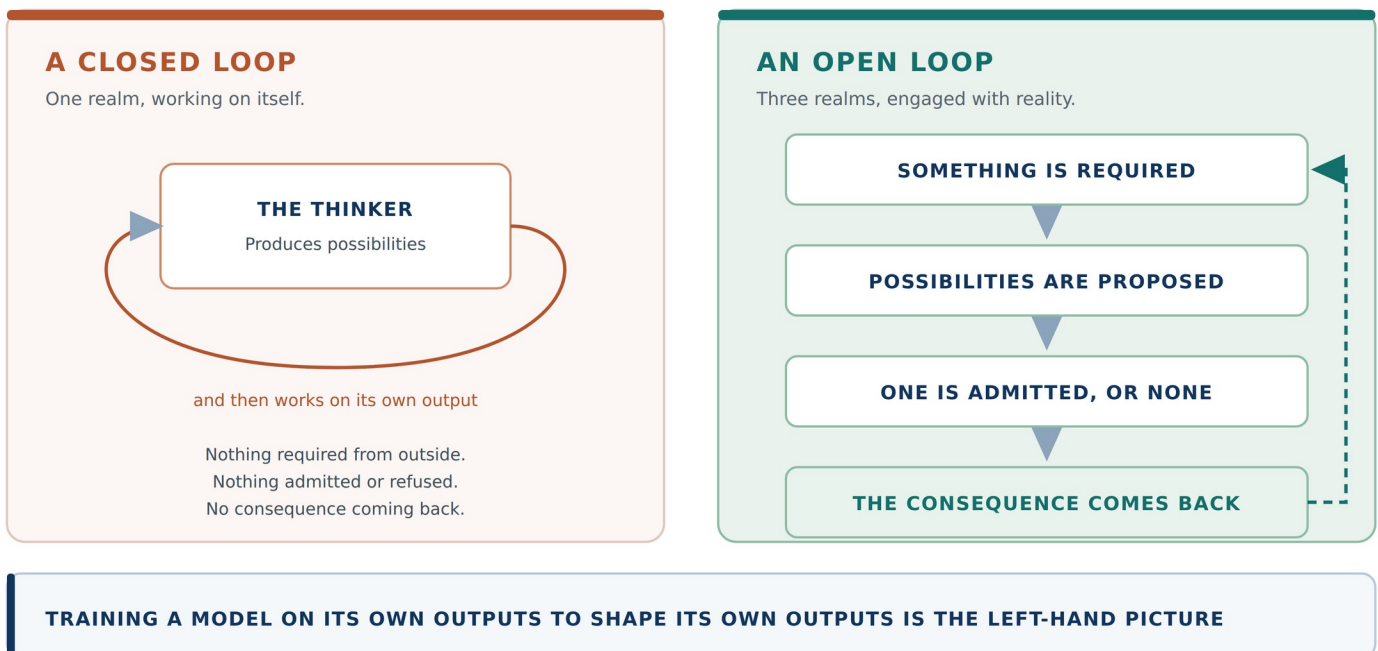
The third one does not look like a failure

It looks like productivity. It looks like research. A person or an institution can spend decades generating possibilities, be rewarded the whole time, and never once compute what was required or what was admissible. That is the state the technology industry is in.

A realm working on itself is still one realm

There is an old idea that the way to improve conduct is to work on your own mind. Quiet it, discipline it, refine what it produces.

Structurally that is the thinker operating on the thinker. Generating and suppressing are the same realm doing both. Nothing is required of it from outside, nothing is being admitted or refused, and no consequence is returning. It is a closed loop with one part running.



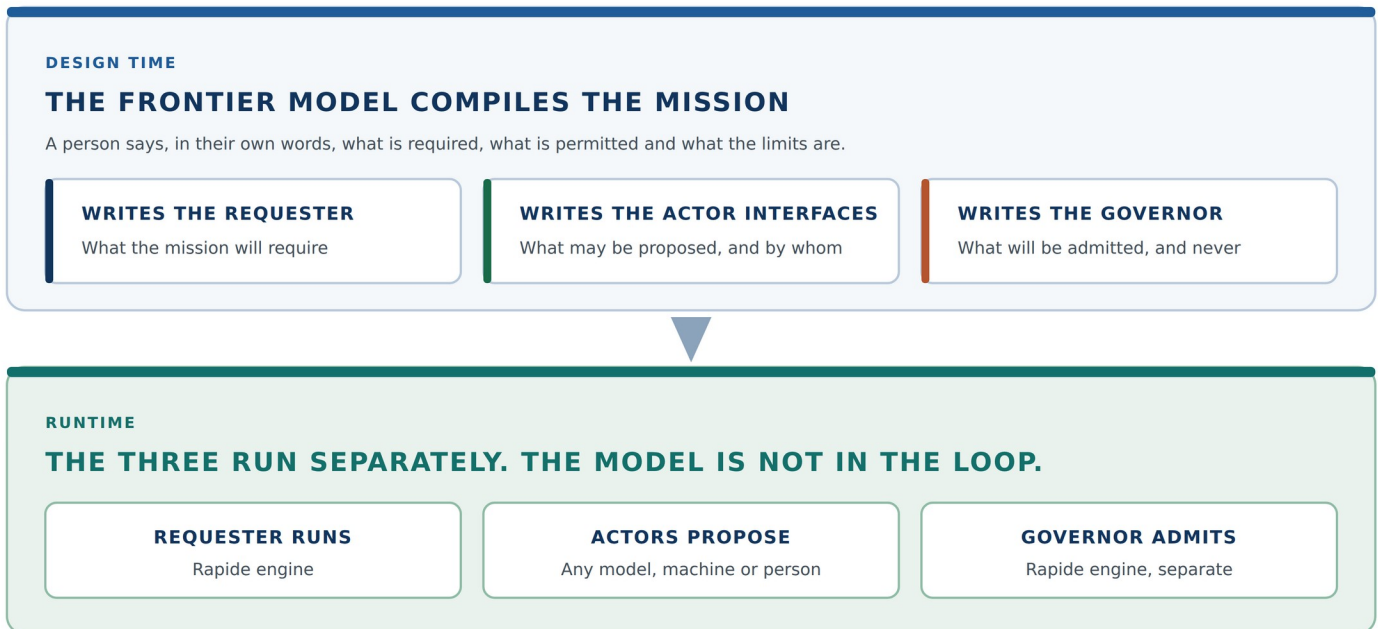
This is also what model alignment is

A model trained on its own outputs, to shape its own outputs, is the left-hand picture exactly. There is no requester outside it stating what is required now, and no governor outside it deciding what is admissible now. The result is a better behaved thinker. It is not a governed system, and no amount of further training makes it one.

What the frontier model is for, and it is a great deal

Frontier models are the most capable artifacts anyone has ever built. They deserve the standing they have. In this architecture they hold the role that makes everything else possible, which is the compilation of the mission itself.

A person states, in their own words, what is required, what is permitted and what the limits are. The model turns that into a running program: it writes the requester, it writes the actor interfaces, and it writes the governor. Without a model capable of doing that, declaring a mission would need engineers, and the whole idea would stay out of reach of the people who actually hold the missions.



A compiler can emit something stricter than itself.

What makes a governor a governor is that it runs apart and nothing can bypass it.

Writing the three is not being the three

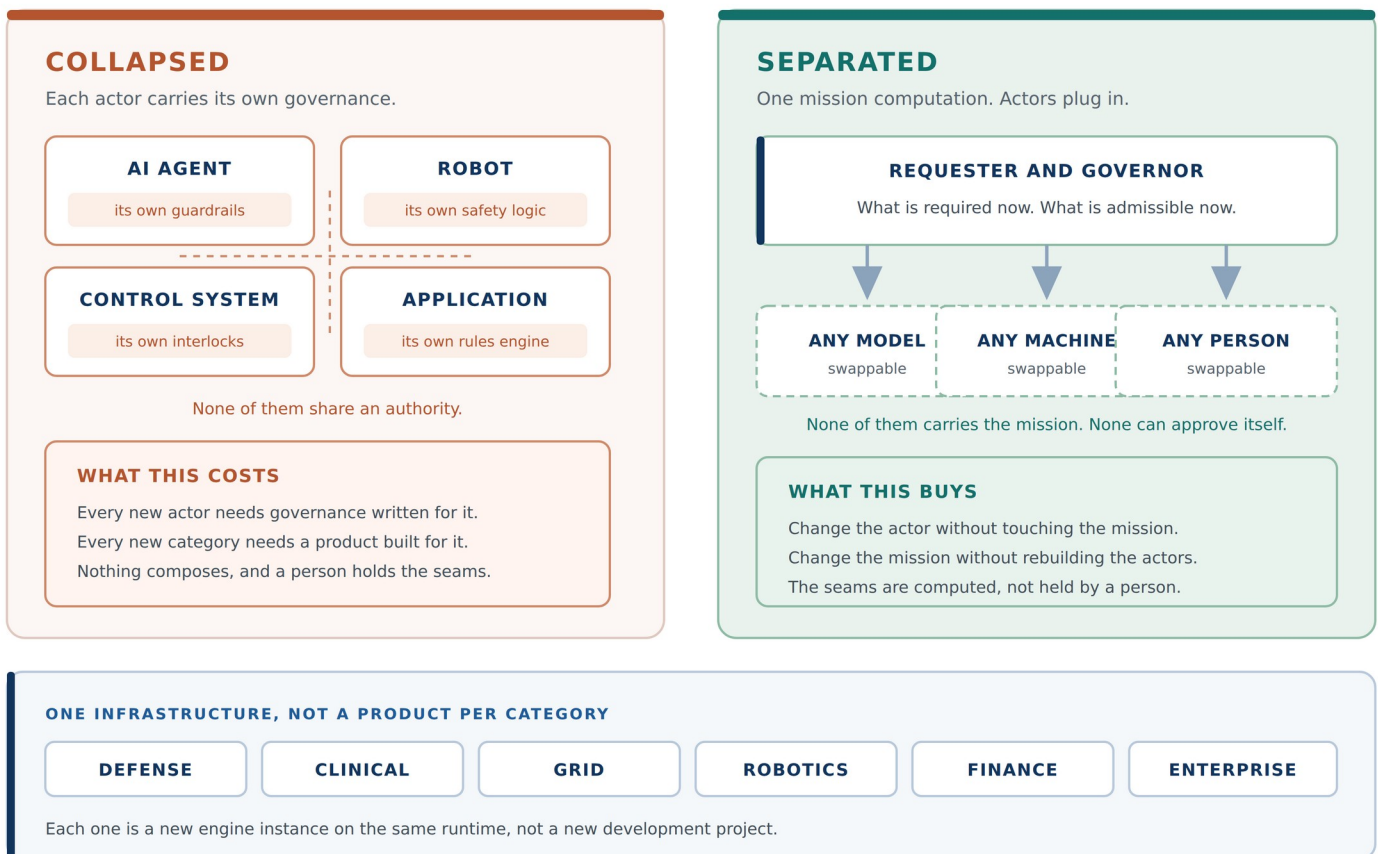
A compiler routinely emits something stricter than itself. What makes a governor a governor is not the words it is made of, it is that it runs apart and nothing can bypass it. So the model expresses all three at design time, and at runtime it is not in the loop. That is not a demotion. It is the arrangement that lets the most capable actors ever built be used on consequential work at all.

05 / COMPOSABILITY

Separate them and almost anything can be built

This is the engineering consequence, and it is the reason one company can address defense and clinical and grid without building three products.

Once the mission computation sits outside the actors, the actors become swappable. Any model, any machine, any person, in any combination, because none of them carries the mission and none of them can approve itself.



Collapse them and you get the opposite

Every actor needs governance written for it, every category needs a product built for it, and a person holds the seams between them. That is why nothing composes today. It is not a tooling problem. It is what happens when the computation that should sit above the actors is distributed into each of them.

THE POINT

The separation is the whole thing

Not a constraint added to intelligence. Not a rule layer bolted onto a capable actor. The separation is what makes any of it safe to run, and it is as true of a person as it is of a system.

Nobody who proposes an action also approves it.

The consequence returns to what was required, not to whoever acted.

Runtime Computing is that, built so it cannot be collapsed.

Three independent computations against live reality. A frontier model to express them in a person's own words. An enforced boundary that only an admitted action crosses, and a consequence that returns to the same mission.

**A person who lives in one realm becomes what that realm does.
A machine is no different, and it has no habit of its own to fall back on.**

ANY INTELLIGENCE. ANY ACTORS. ONE GOVERNED REALITY.