

Automating Systems Simulation at Scale

An LLM Pipeline for Multi-Framework Modeling

Raz Saremi, Douglas J. Buettner, Ethan Silverstein · Stevens Institute of Technology
Zoe Szajnfarder · The George Washington University

AI4SE & SE4AI Research and Application Workshop · 22–23 September 2026 · Washington, DC

Research areas: Agentic AI and workflow integration · AI-enabled modeling and simulation

1

A deployed system, not a prototype

37 agents, four paradigms plus hybrid coupling, running against real papers.

2

Provenance built into the artifact

Every generated simulation carries its own extracted mathematics and a verification record.

3

An architectural claim you can test

Structural priors and validation contracts determine trustworthiness — not the backbone.

4

A catalogue of what still fails

Named, reproducible, and open to the room.

1: Choose a paradigm



2: Hand-encode the model

1) Governing equations, 2) state transitions, 3) resource constraints, and 4) rules of engagement.

3: Instrument for the decision



Weeks to months

per scenario, per paradigm.

The binding constraint is not compute. It is the availability of specialists who can write the model.

Trade studies, contingency analyses, and acquisition decisions all wait on it.



Architecture

RQ1

What design decisions are required to turn a generic multi-agent LLM framework into a paradigm-aware, multi-paradigm code-synthesis workflow that an engineering team will actually use?



Speed and trust

RQ2

How does the pipeline compare to expert-authored references on emergent-behaviour metrics, and what is its end-to-end latency relative to manual authoring?

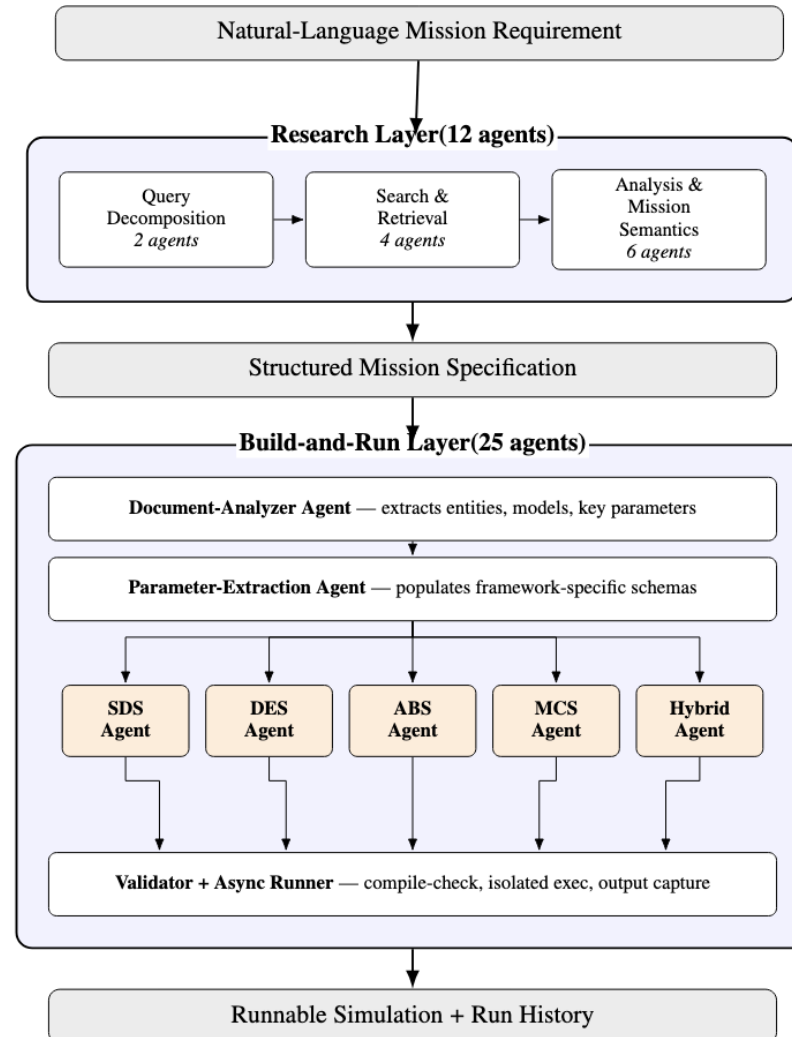


Practicality

RQ3

What recurring failure modes survive even after the full guardrail stack is in place, and what engineering disciplines mitigate them?

Q2 is the one where running the system changed our answer. We come back to it.



1 Research layer

12 agent classes

- Query decomposition and optimization
- Parallel search and retrieval
- Dedup, ranking, and summarization
- KPI selection and actor matching

2 Generation layer

25 agents, 6 stages

- Math extraction from paper text
- Paradigm detection and specification
- Typed parameter packs per paradigm
- Deterministic code emission

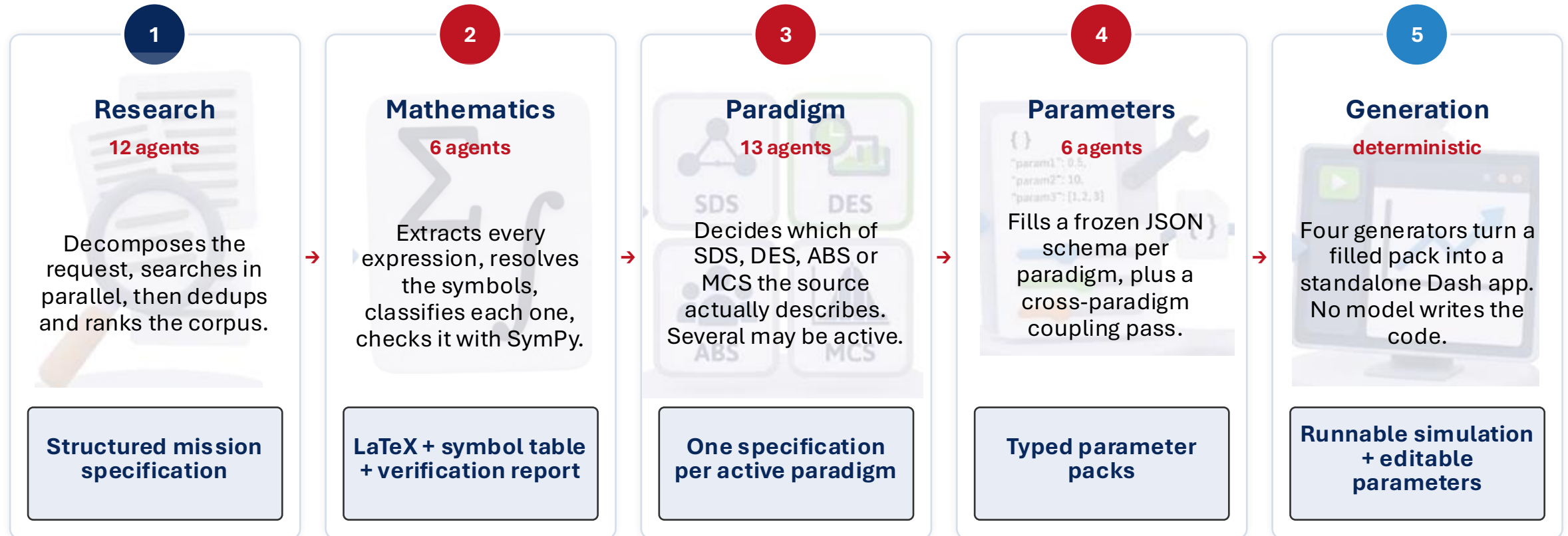
3 Verification layer

runtime + structural

- Compile, execute, inspect gates
- Dynamics-verification agent per paradigm
- SymPy parseability report
- Coupling topology audit

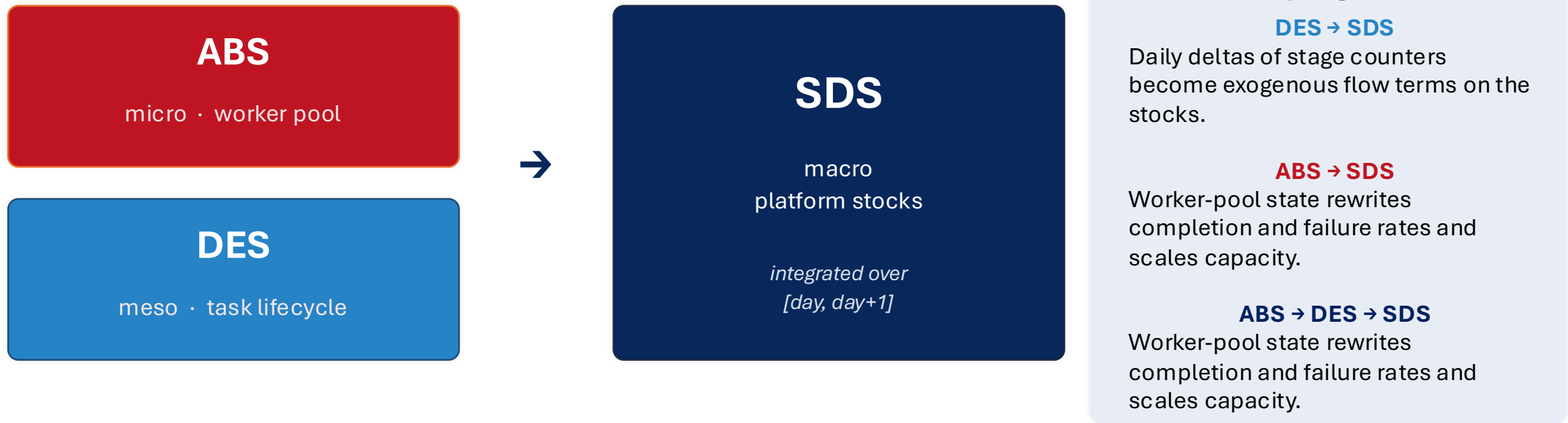
Every inter-layer hand-off is a typed contract the orchestrator enforces before propagation — 11 payload kinds in `plt/contracts.py`.

The agents produce the specification; deterministic generators produce the runnable simulation.



Typed hand-offs make the workflow auditable at every stage.

Real systems are not one paradigm. A sixth stage emits an orchestrator that runs several solo simulations together and passes state between them at day boundaries.



Exogenous terms enter derivatives(), so they are integrated rather than added at display time — mass balance holds by construction.
Solo simulations are imported unmodified and keep running standalone.

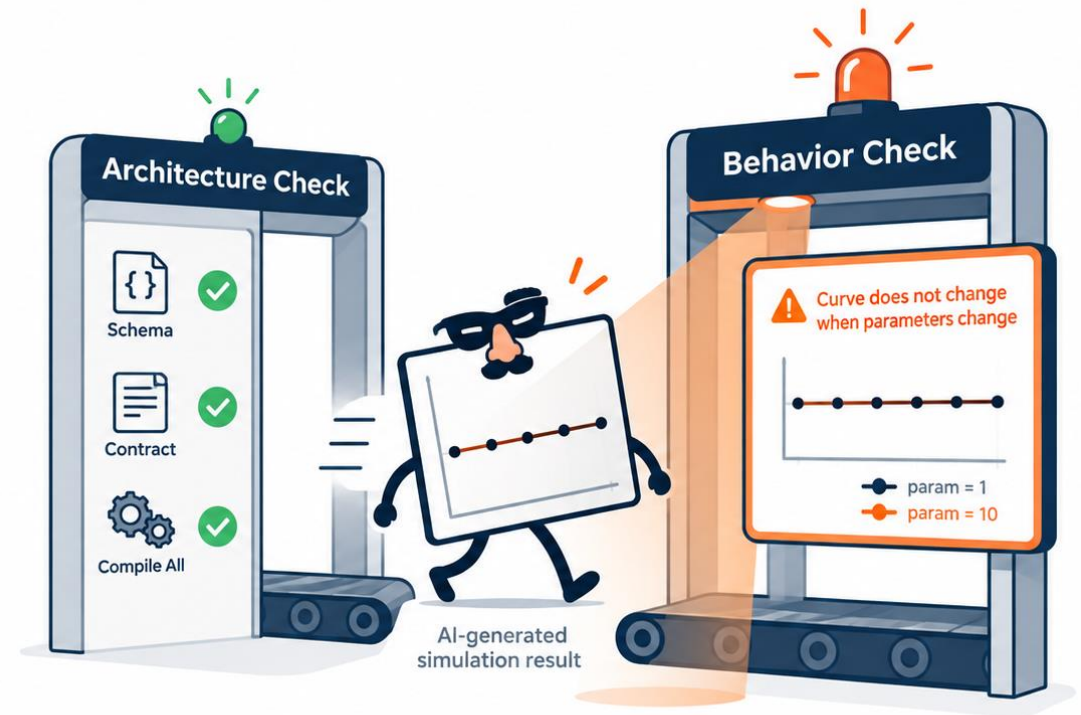
Trustworthiness is architectural, not a property of the model you picked.

The “fake mustache” problem: structurally valid,
behaviorally suspicious.

Architecture checks the ID.
Behavior checks the alibi.

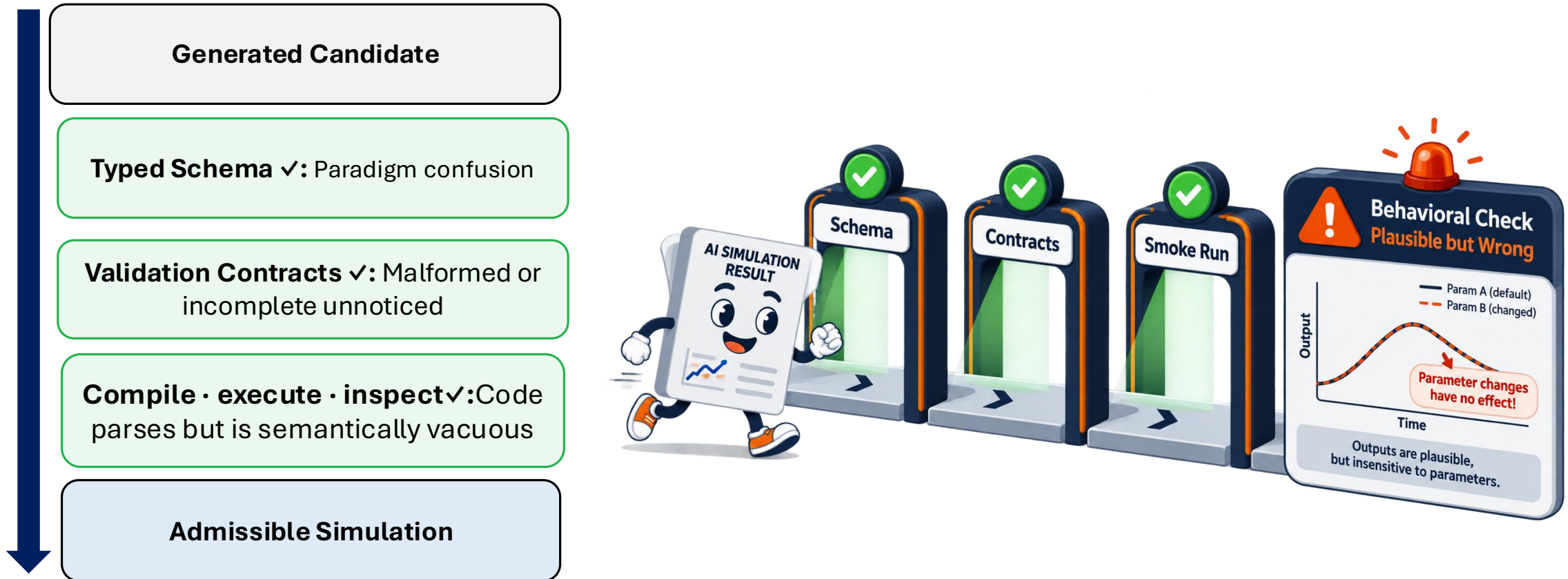
Presence ≠ effect

A curve can still be wrong.

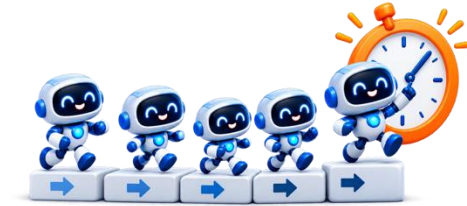


Three checks to make code admissible, not trustworthy.

The harder failure is a model that still looks plausible while ignoring meaningful parameter changes.

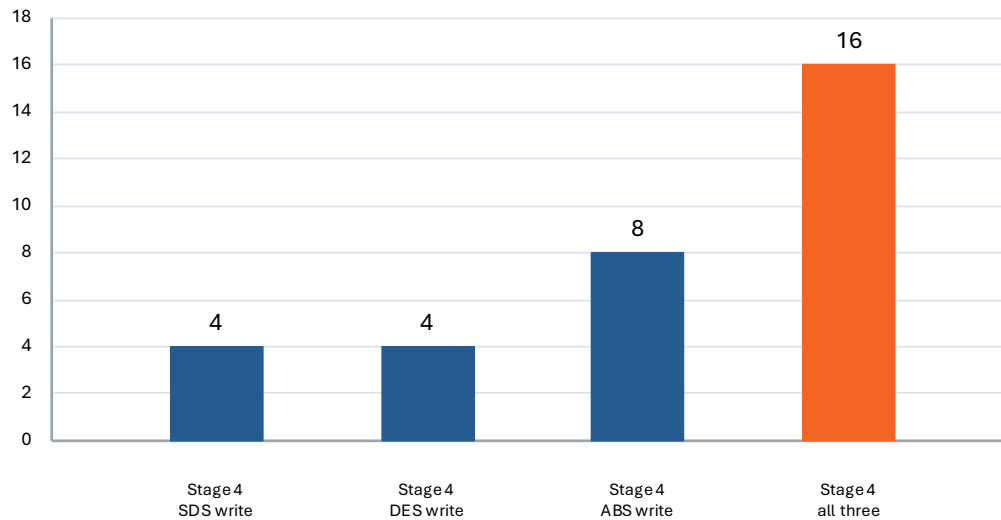


Code Generation is not the expensive part. The sequential agent stages are.



Sequential agent stages = wall clock time

Deterministic generation, measured (seconds)



End-to-end : [10, 26] minutes



Dominated by agent latency

Stage 1 alone runs six sequential agent calls. Stage 3 runs one per active paradigm plus a coupling pass. These, not code emission, set the wall clock.



Phase 6 is the long pole

Full-page diagram scanning costs two to five minutes per paper. It runs in a subprocess with a timeout and is optional — disable it and the pipeline is unaffected.

They are the failures that survived the current guardrails — and became the next validation agenda.

01

Topology bug

Diagrams give structure, not coefficients.

02

Crossing-link bug

Curved links that cross are still lost.

03

Backbone-drift bug

Model upgrades can re-open old failures.

04

Semantic-validation bug

Runs can pass presence checks yet drift outside tolerance.

05

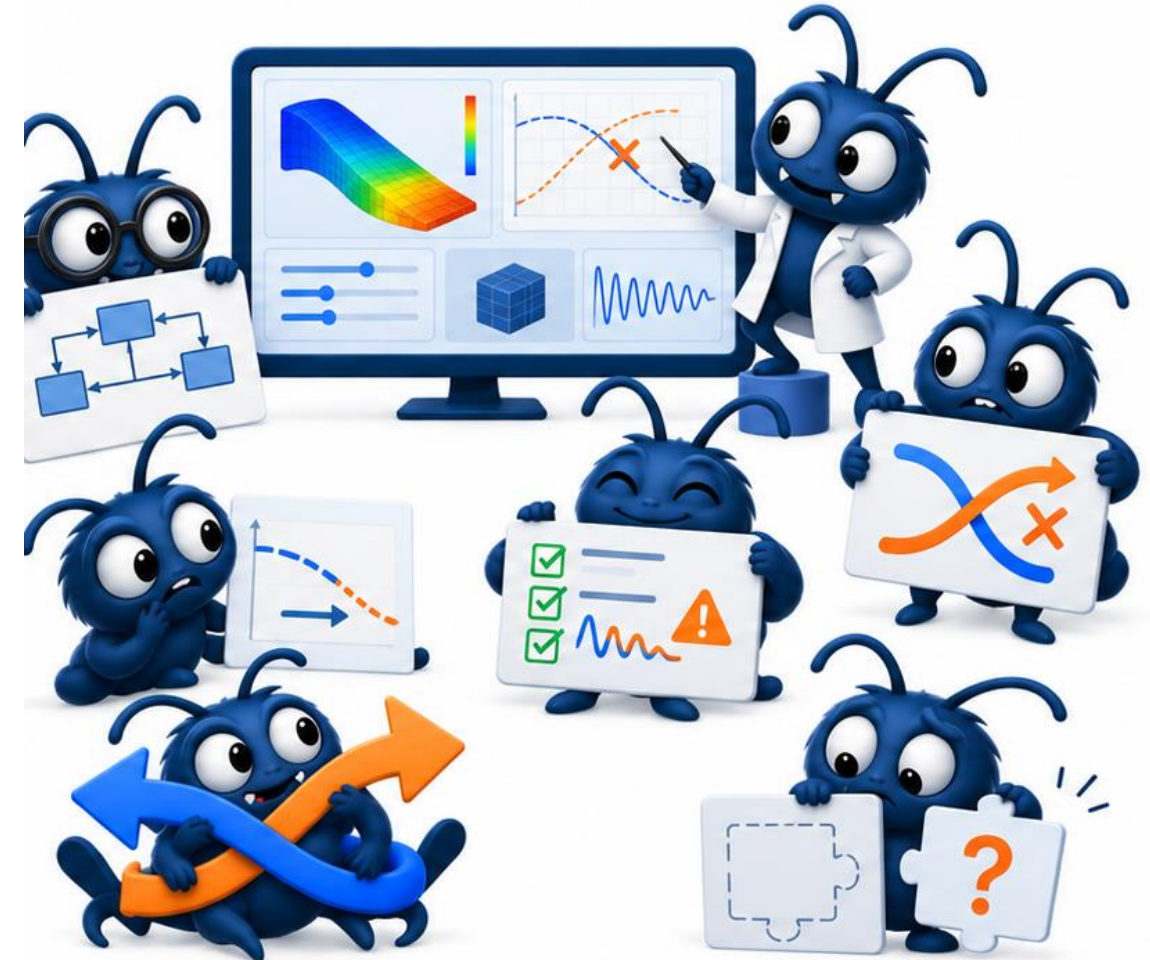
Hybrid-gap bug

Only two coupling channels are wired end to end.

06

Dangling-dependency bug

A missing Stage 5 repair module silently no-ops.



Every “bug” is a test case.

Q1

Architecture

Three disciplines carry the result: 1) typed schemas as structural priors, 2) inter-stage validation contracts, and 3) compile-execute-inspect validation. The pipeline is deployed and runs against real papers end to end.

Q2

Speed and trust

Speed: Simulation systems builds with in minutes
Trust: behavioral tests help to check the trustworthiness, and is still failing.

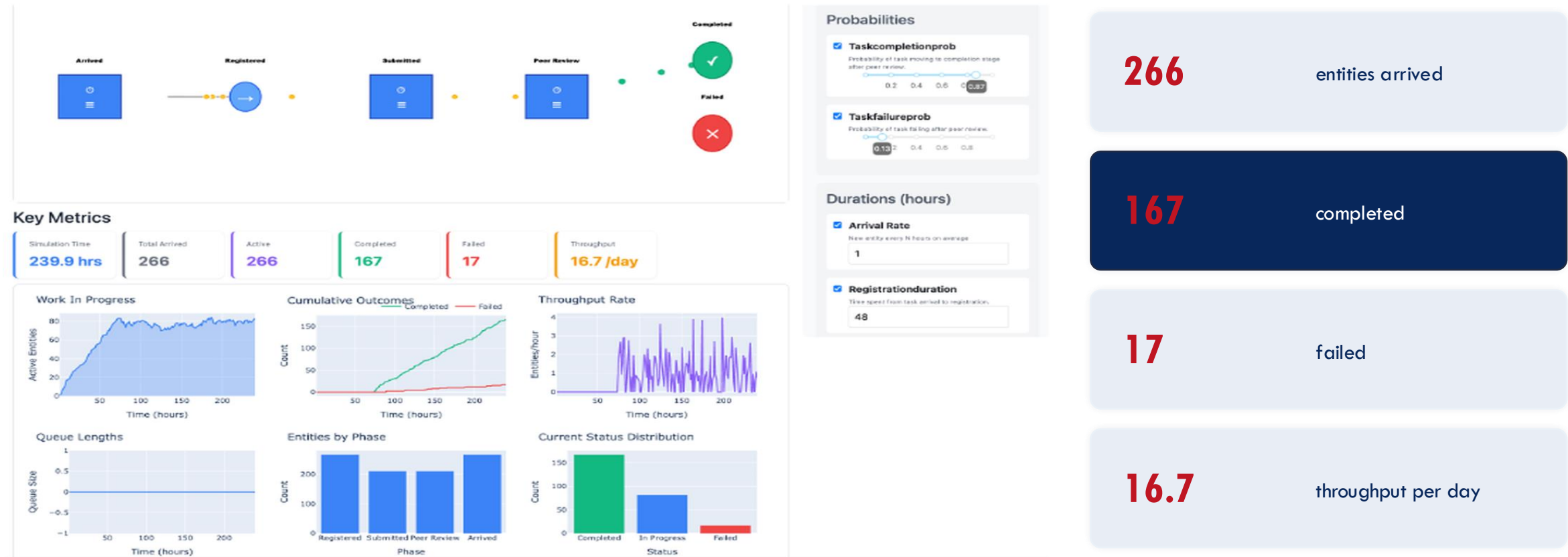
Q3

Practicality

Six recurring failure modes survive the full guardrail stack and are named in the validation bug zoo: topology, crossing-link, backbone-drift, semantic-validation, hybrid-gap and dangling-dependency.

Appendix

A crowdsourced software development failure-prediction paper, given to the pipeline as a PDF with no other input. Below is the generated discrete-event application running, unedited.



Stage lifecycle, live metrics, and the probability and duration controls are all emitted by the generator — no hand-editing.

System Dynamics Simulation (SDS) SDS

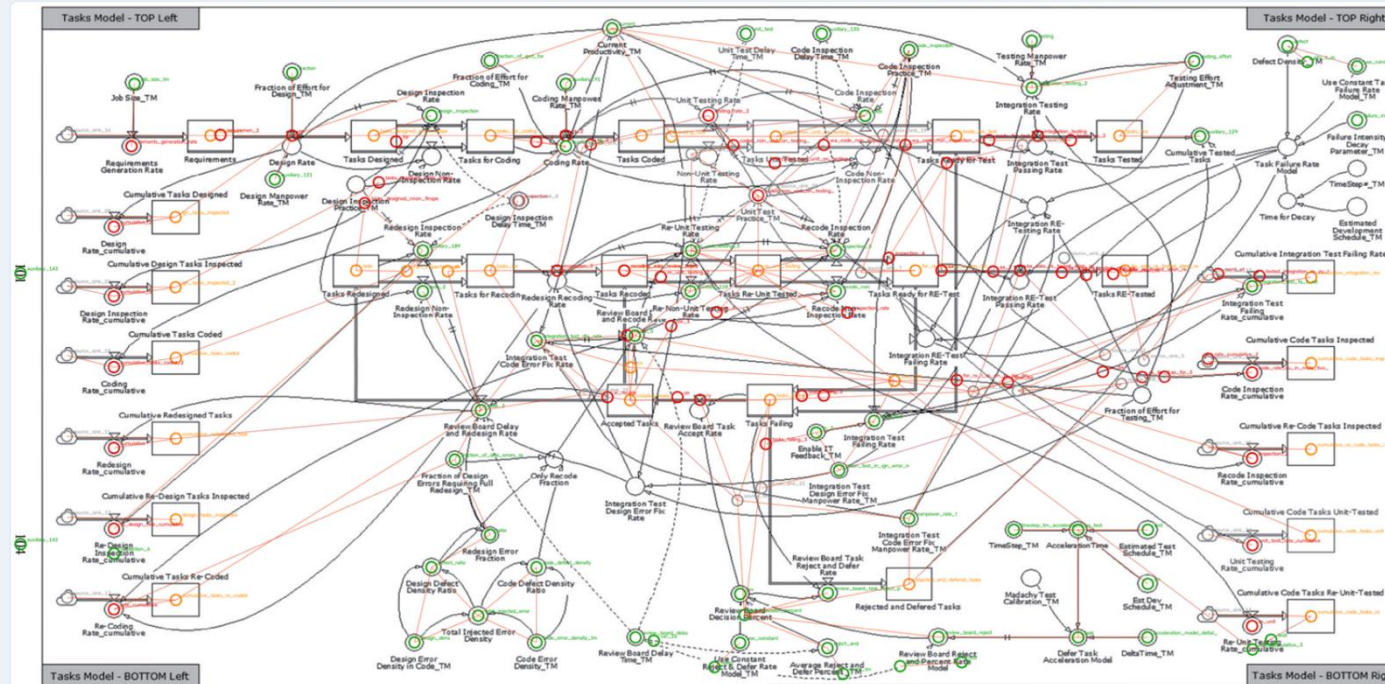
Paper: upload_Half-Acquisition Games and Software Defect

Input Parameters Generated Code Source Diagram Run Simulation

figure: assembled_system_dynamics_pp102_105_modified_tasks_models · 32 stocks · 58 flows · 94% of labels read · QA: review before trusting

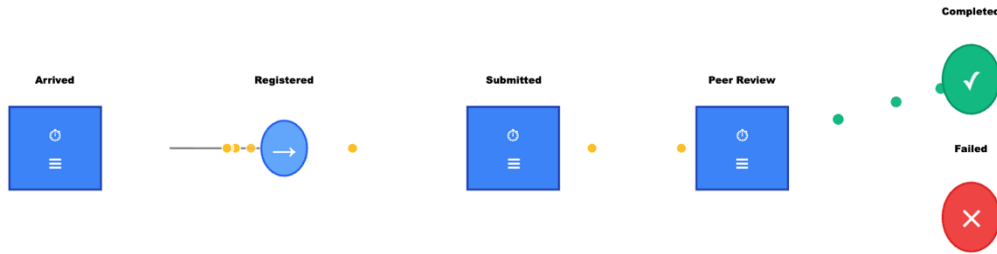
The overlay shows what the extractor detected. Anything on the page but not in the overlay did not make it into the model.

Detected structure (overlay)

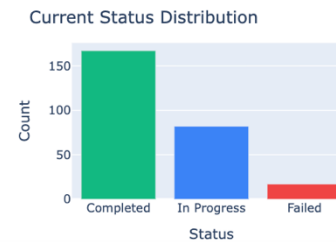
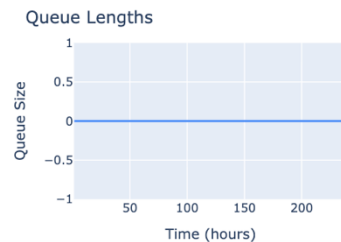
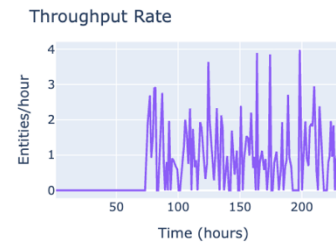
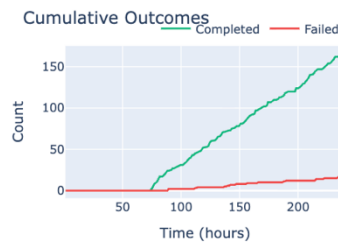


assembled_system_dynamics_pp102_105_modified_tasks_models.overlay.png

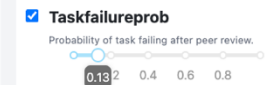
[illegible]



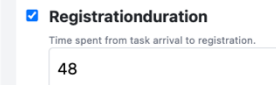
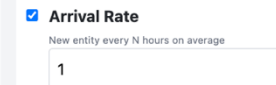
Key Metrics



Probabilities



Durations (hours)



Agent-Based (ABS)

Paper: CrowdGim: A Hybrid Simulation Model for Failure Prediction in Crowdsourced Software Development.pdf

Post #886

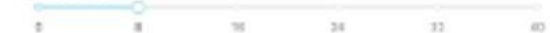
Agent-Based Simulation

Agent-Based Simulation

Random Seed

7

Task Arrival Rate



Agent Arrival Rate



Registration Threshold



Submission Rate



Review Pass Threshold



Initial Tasks at Reset

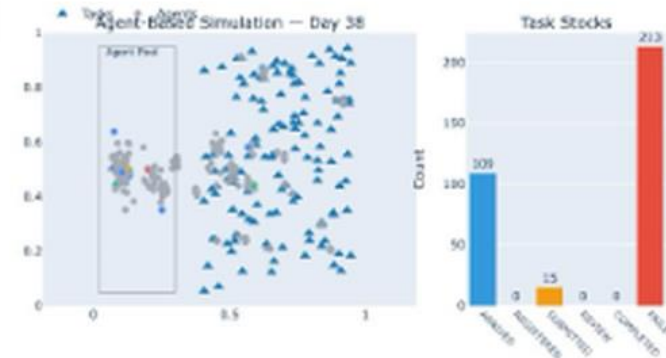


Reset Simulation

Play / Pause

Step 1 Day

Speed (ticks/sec)



Day 38	Tasks - Total 337	Tasks - Completed 0	Tasks - Failed 213	Pass Rate 0	Tasks - Open 124
Avg Reg Delay (s) 1.74	Avg Sub Delay (s) 1.74	Avg Outcome Cycle (s) 13.77	Agents - Count 490	Agent Utilization 0.735	

