

Contract-Based Agentic Systems Engineering: Delegation as Quotient for Trusted Cyber-Physical System Design

Ryan Matheu · Sai Sandeep Damera · John S. Baras

Electrical and Computer Engineering and Institute for Systems Research · University of Maryland, College Park



UNIVERSITY OF
MARYLAND

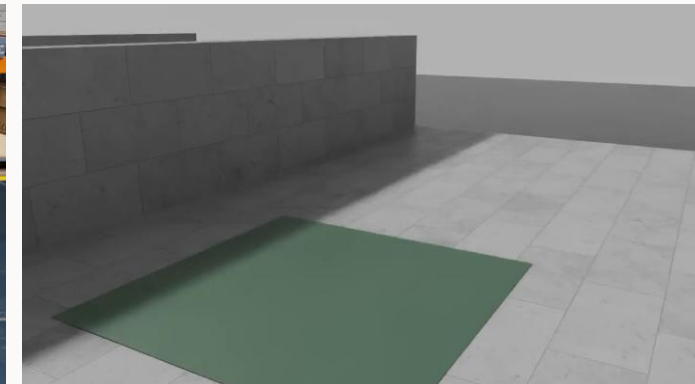
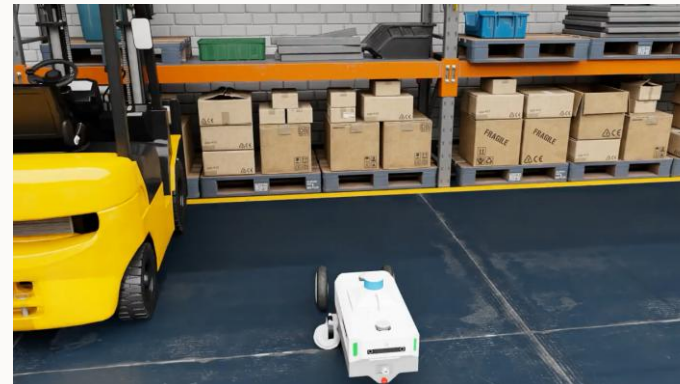
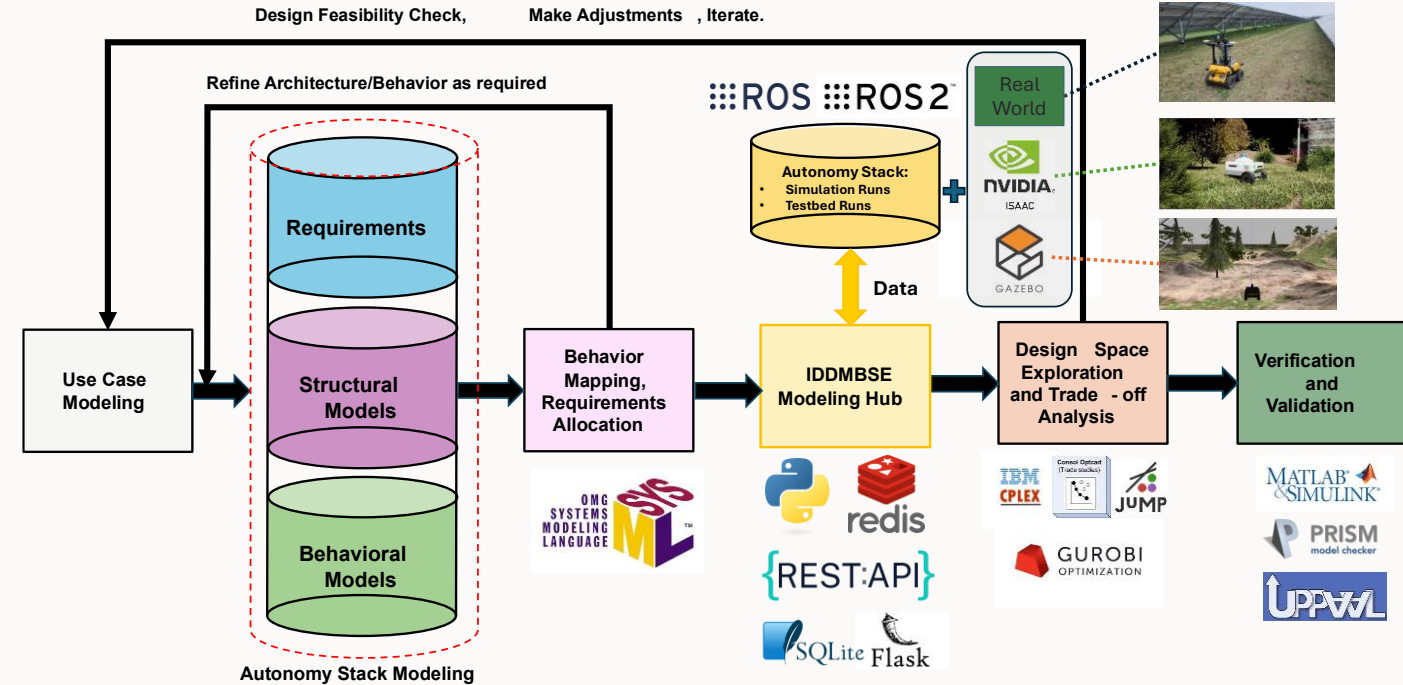
AI4SE Workshop Extended Abstract · Full manuscript in preparation

PART 1

Agentic Systems Engineering

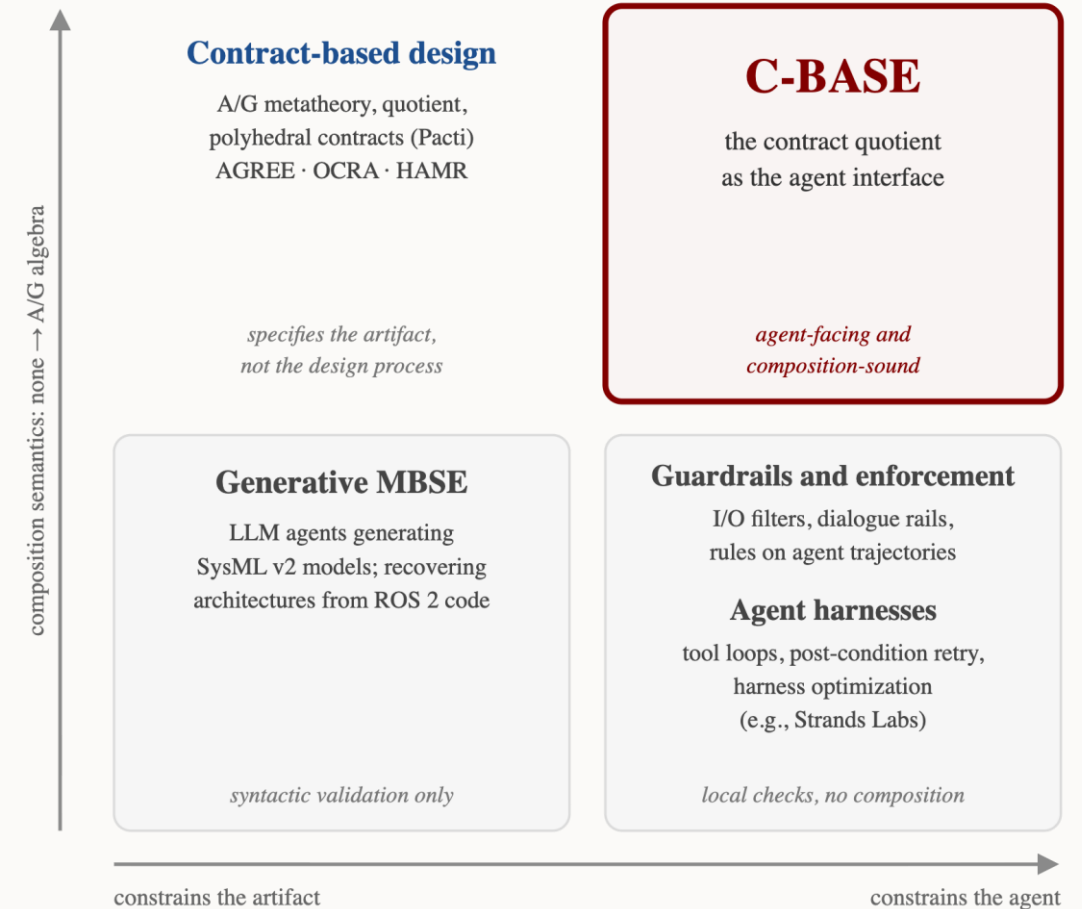
MBSE for Trusted Autonomy in the Agentic Era

- **MBSE provides the discipline.** SysML models and the V-process: requirements decomposition, components realization, systems integration and verification.
- **IDDMBSE adds the data.** Baras et al. place a data-driven loop at every step of the V-model where the physics is known, learning behavioral models where it isn't.
- **Incorporating Agentic AI into the MBSE workflow.** AI agents configured to edit the models and the stack, configured to design and test novel components.
- **Two systems provide an Agentic test bed:** quadruped extreme locomotion and AGV warehouse navigation.
- **The question for this talk:** what do you hand an agent so its work still fits the system it was built for?



Agentic Systems Engineering Today

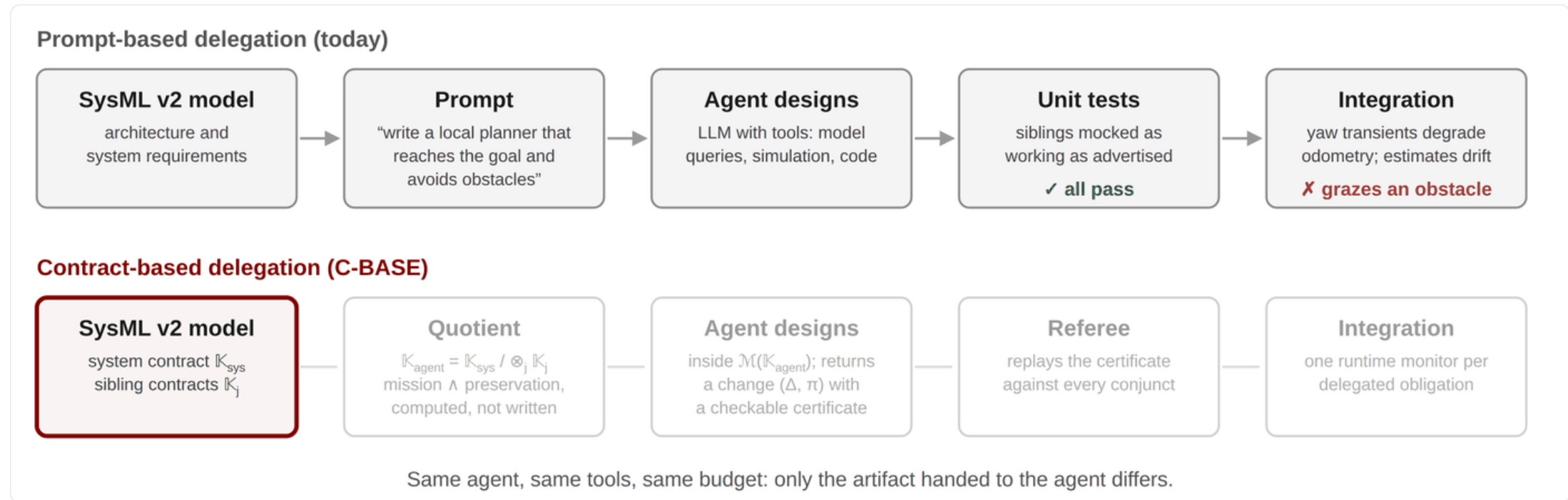
- **NeMo Guardrails** filter an agent's inputs, outputs, and dialogue [Dong et al., 2024; Rebedea et al., 2023], or its action sequences [Wang et al., 2025].
- **Agent harnesses**, such as the AWS Strands Labs open-source project, give agents tools, robot control, post-condition retry, and harness optimization. A delivery is retried until its *local* postconditions pass.
- **Generative MBSE assistants** emit SysML v2 models with syntactic validation and no behavioral obligations [Bouamra et al., 2025; Benchat et al., 2026].
- **Contract-based design** grew from the supplier splits that systems engineers manage with ICDs and requirement flow-down [Benveniste et al., 2018; Nuzzo et al., 2015]. It has the composition semantics, but it has specified the *artifact*, never the *design process*.



The gap, and the proposal. None of these can say whether a locally compliant delivery composes into a globally compliant system. We compute the interface instead of writing it: the agent receives the *quotient* of contract algebra, the weakest contract under which the assembled system still meets its specification.

Delegate Design to Agents with Contracts

- **Contract-Based Agentic Systems Engineering (C-BASE)** hands the agent a contract representing the formal properties a component needs to satisfy.
- Contracts are multi-aspect, with layers focusing on structural conformance, parametric feasibility, and behavioral certification.



PART 2

Assume-Guarantee Contract Theory

Assume/Guarantee Contracts

Assume/Guarantee Contracts provide an algebraic framework for reasoning about formal system requirements.

- Let $\mathcal{B}$ be the universal set of behaviors over all system variables. A property P is a subset of behaviors, $P \subseteq \mathcal{B}$. Similarly, a component M is defined as the collection of behaviors it can exhibit, $M \subseteq \mathcal{B}$. A component is said to satisfy a property, written $M \models P$, when $M \subseteq P$.

Definition (A/G Contract). An Assume/Guarantee (A/G) contract is a pair of properties $\mathcal{C} = (A, G)$ where $A \subseteq \mathcal{B}$ are the assumptions and $G \subseteq \mathcal{B}$ are the guarantees. A component is said to *implement* an AG contract if it satisfies the guarantees of the contract while operating under the assumptions of the contract.

- Contracts over a set of behaviors form a partial order with respect to an order called *refinement*. Given two contracts $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_1$ is said to refine $\mathcal{C}_2$ (denoted $\mathcal{C}_1 \preceq \mathcal{C}_2$) if any implementation of $\mathcal{C}_1$ is an implementation of $\mathcal{C}_2$. Colloquially, when $\mathcal{C}_1$ assumes more and guarantees less than $\mathcal{C}_2$.
- Contract algebra introduces two useful binary operations:
 - Contract Composition** refers to the operation of combining two contracts such that the product is a contract which specifies the behavior of a system constructed via composition of implementations of the two subcontracts.
 - Contract Quotient** allows solving for a missing or residual contract such that its composition refines a top-level contract.

A/G Contracts: Semantics and Operations

- **Contract** $\mathcal{C} = (A, G)$: encapsulate the properties a component assumes of its environment and guarantees in return. Saturated form $(A, G \vee \neg A)$; environments $E \subseteq A$, implementations $M \wedge A \subseteq G$ [Benveniste et al., 2018].
- **Refinement, composition, conjunction, quotient:**

$$\mathcal{C}_1 \preceq \mathcal{C}_2 \Leftrightarrow A_1 \supseteq A_2 \text{ and } G_1 \subseteq G_2$$

$$\mathcal{C}_1 \otimes \mathcal{C}_2 = ((A_1 \wedge A_2) \vee \neg(G_1 \wedge G_2), G_1 \wedge G_2)$$

$$\mathcal{C}_1 \wedge \mathcal{C}_2 = (A_1 \vee A_2, G_1 \wedge G_2)$$

$$\mathcal{C}_1 / \mathcal{C}_2 = (A_1 \wedge G_2, (G_1 \wedge A_2) \vee \neg(A_1 \wedge G_2))$$
- **Contract operations are closed:** When the term algebra is a Boolean algebra, contracts are closed, and iterated delegations can always be computed.
- **SysML v2 [§7.21.2]** reads a requirement as “*if all the assumption constraints are true, all the required constraints must be true*”: the saturated contract. A delegated contract is a native requirement.
- **Contracts become model elements.** The quotient:
$$(A_1 \wedge G_2, (G_1 \wedge A_2) \vee \neg(A_1 \wedge G_2))$$
 - is a requirement with **assume** $A_1 \wedge G_2$ and **require** $G_1 \wedge A_2$

```

requirement def PlannerQuotient {
  doc /* Behavioral residual of the local-planner slot:
  * K_sys divided by the composed perception and gait
  contracts. */
  subject planner : LocalPlanner;
  actor perception : Perception;
  assume constraint perceptionPromise {
    perception.positionError <= perception.errorBound
  }
  require constraint clearance { planner.minClearance >=
    0.40 }
  require constraint goal { planner.goalError <= 0.50 }
  require constraint preserveOdometry {
    planner.maxYawAccel <= 3.0 }
  require constraint preserveDynamics {
    planner.maxDeltaV <= 0.30 and planner.maxDeltaOmega
    <= 0.50
  }
}

```

■ sibling guarantee, assumed
 ■ mission conjuncts
 ■ preservation conjuncts

Contract theory	SysML v2
assumption A , guarantee G	assume, require constraint
$\neg A \vee G$	effective requirement constraint
environment	actor parameter
$M \in \mathcal{M}(\mathcal{C})$	satisfy requirement by part

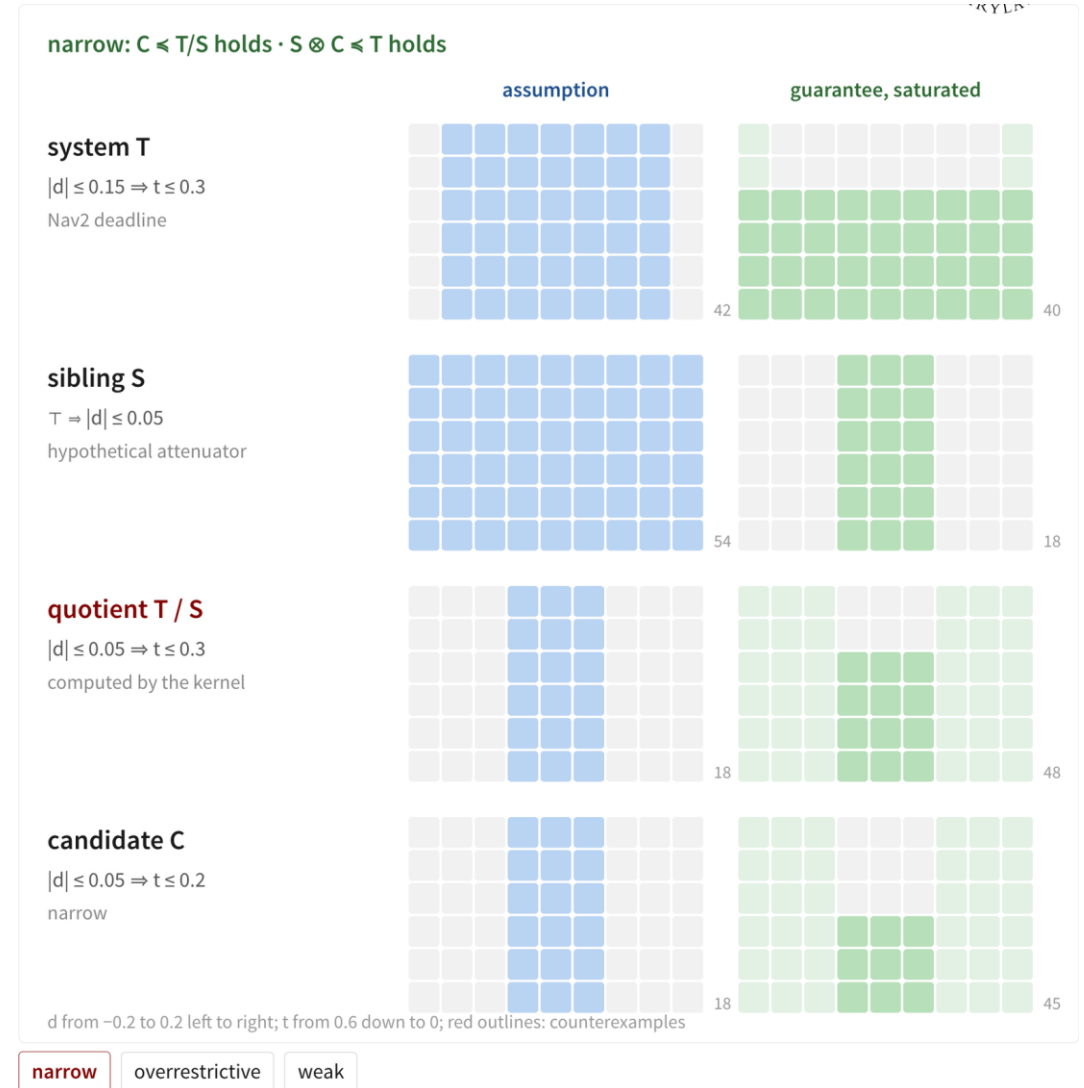
Delegation is the Quotient

- **Problem:** Given a top-level system contract and the fixed siblings C_j , find the least constraining contract that still composes into the system.
- **Answer** [Incer et al., 2018]: the quotient $C_{agent} = C_{sys} / \otimes_j C_j$, the adjoint of composition:

$$C_2 \otimes C \leq C_1 \iff C \leq C_1 / C_2.$$

Theorem (delegation is canonical). Computed aspect by aspect, the quotient composes soundly with the siblings, and every sound specification for the slot refines it.

- **Unfolding Obligations.** With siblings (α, γ) composed, the quotient guarantee is $(\varphi_G \wedge \alpha) \vee \neg(\varphi_A \wedge \gamma)$, specifying the mission, plus *keep the siblings' assumptions true*. E.g., for the quadruped, α bounds the planner's own yaw rate and command smoothness.
- **Hand-written specs keep only the mission, $(\varphi_A \wedge \gamma_{eff}, \varphi_G)$:** wherever a sibling promise is void, a compliant delivery and a compliant sibling compose to violate the system guarantee. **Naive delegation is unsound.**



Multi-Aspect Contracts:

- A delegated multi-aspect contract constrains three key properties of a component:
 - Structure:** *Is the component structurally compliant?*
 - Parameters:** *Are the performance/risk parameters within spec?*
 - Behavior:** *Does the component actually do what was specified?*

Definition (multi-aspect contract). A multi-aspect contract is a tuple:

$$\mathbb{K} := (\partial, \mathcal{C}^{\text{par}}, \mathcal{C}^{\text{beh}}, (m, \varepsilon)),$$

containing a polyhedral contract over the boundary attributes, an STL contract over its signals, and a margin and risk budget. A component implements $\mathbb{K}$ when it realizes ∂ , satisfies $\mathcal{C}^{\text{par}}$, and meets $\mathcal{C}^{\text{beh}}$ at (m, ε) .

- Lineage.** Generalizes multi-viewpoint contracts [Benveniste et al., 2008]: viewpoints share one behavior universe, aspects live in different universes tied at the boundary. Satisfaction is a product, and the quotient is computed **per aspect**: sound, and conservative only where aspects overlap.

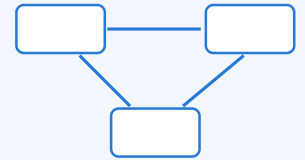
boundary ∂ : typed ports from the SysML v2 model



Structural aspect

typed port graph over the ROS 2 library
conformance: a name-preserving monomorphism
into the observed ROS 2 graph

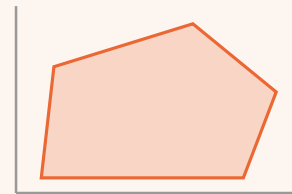
check: generated monitor, linear time



Parametric aspect

polyhedron over the attribute vector a_∂
 $f \geq 10 \text{ Hz} \cdot \ell \leq 100 \text{ ms} \cdot P \leq 60 \text{ W} \cdot m \leq 2.5 \text{ kg}$
composition and quotient in closed form

check: linear-program feasibility

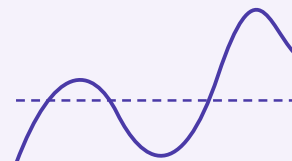


Behavioral aspect

STL pair (Φ_A, Φ_G) over the port signals,
met at margin m and risk ε :

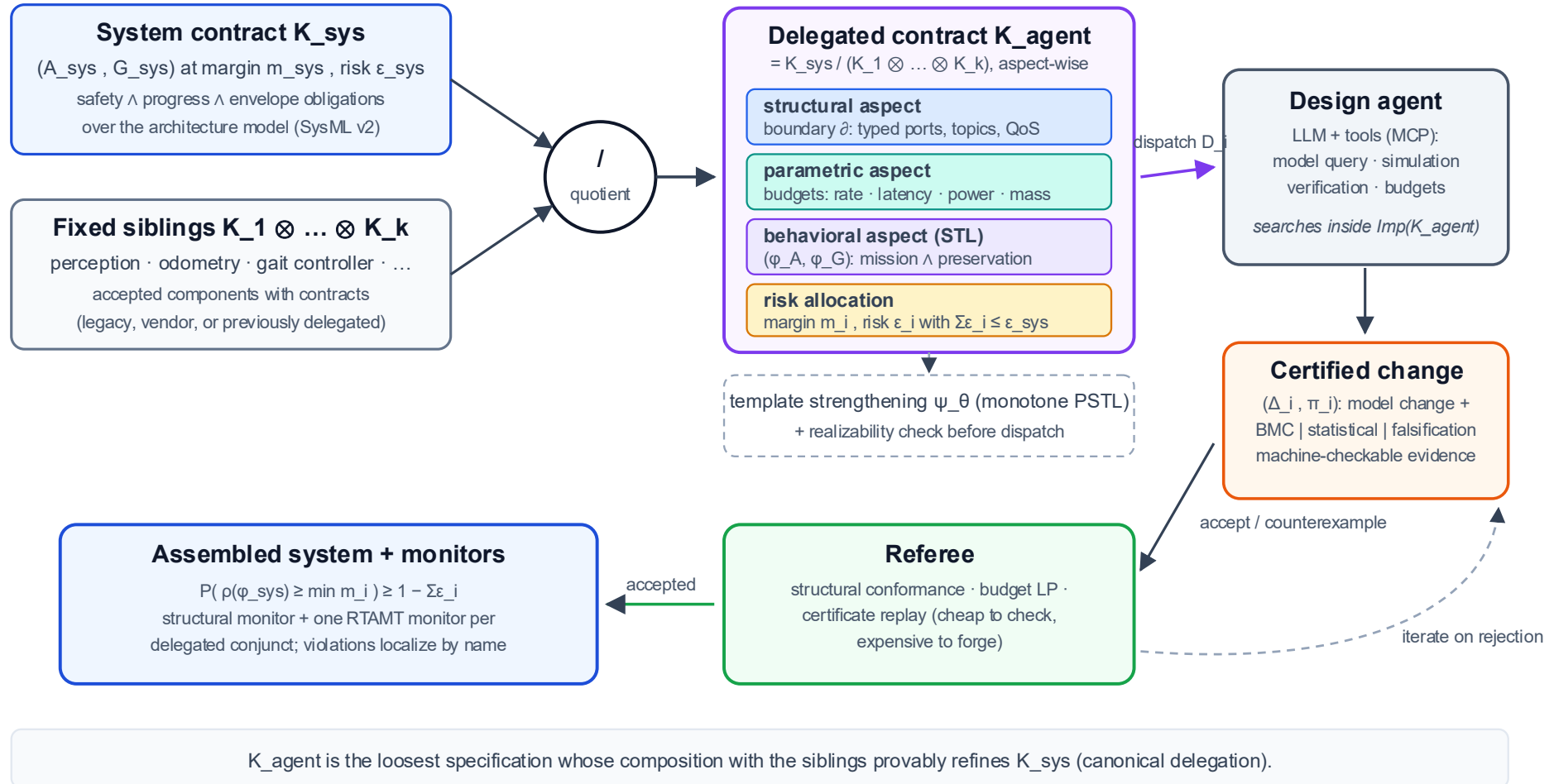
$$\mathbb{P}(\rho(\Phi_G, W) \geq m) \geq 1 - \varepsilon$$

check: certificates, bounded or statistical



Multi-Aspect Contract-Based Agentic Delegation Framework

Delegation is division: the residual obligation of a design slot is computed, not written



Multi-Aspect Fragment: SysMLv2 Structural Conformance Contract

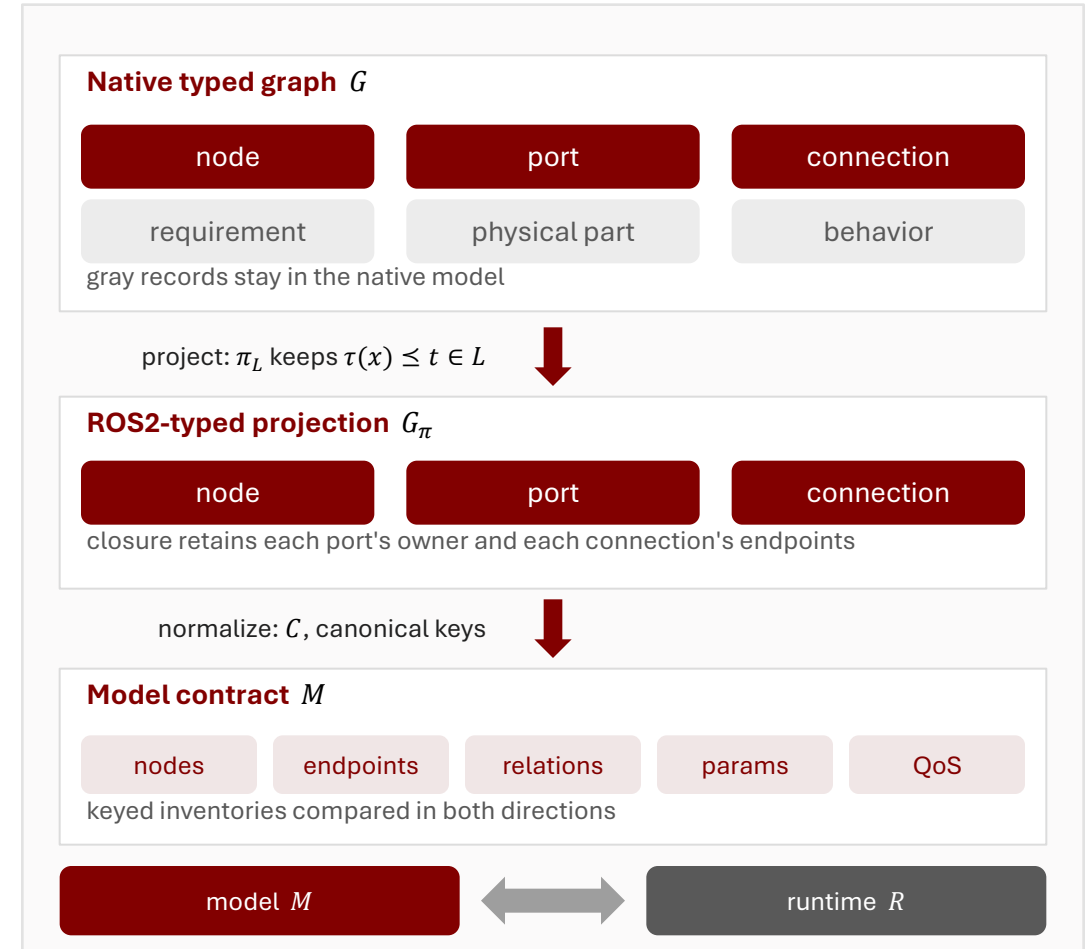
- **Typed graph.** $G = (N, P, E, T, \tau, \rho, \alpha)$ — node, port and connection occurrences, types τ ordered by specialization $\leq$, ownership $\rho: P \rightarrow N$, and evaluated values α (channels, interface types, QoS, parameters).
- **Domain projection.** The library vocabulary $L \subseteq T$ of ros2-sysmlv2 selects the software view:

$$G_\pi = \pi_L(\text{flat}(G)) = \text{flat}(G) \upharpoonright \{x: \exists t \in L, \tau(x) \leq t\}$$
- **Closure.** With L closed under owners and endpoint kinds, $\pi_L(Y)$ is the unique largest L -typed subgraph, so every extractor selects the same graph. Requirements, physical structure and behavior stay native.
- **Keyed contract.** $M = C(G_\pi)$ normalizes names and endpoint roles into keyed inventories:

$$k(p) = (\text{endpoint}, \text{fqOwner}(p), \text{kind}(p), \text{resolvedChannel}(p))$$

Key fields. $\upharpoonright$ = restriction, the subgraph induced on the selected records · fqOwner(p) = fully qualified ROS2 name of the node owning p (/perception/scan_filter)
kind(p) = publisher or subscriber · resolvedChannel(p) = topic name after namespacing and remapping (/scan)

Definition (structural conformance). Writing $R \models M$: coverage is complete, records are well keyed, $X_M^c = X_R^c$ for every inventory, and every required same-key value agrees.



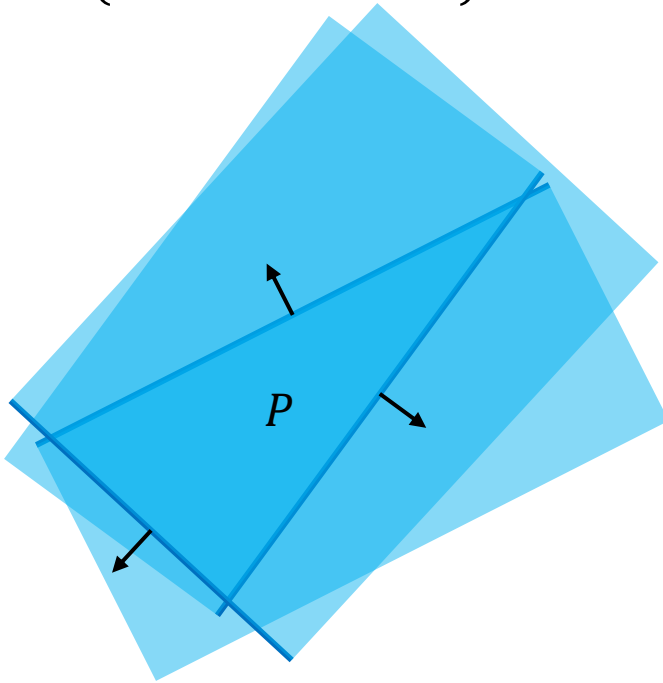
Our published SysML v2 domain library
that makes this projection executable:
sysand.com/projects/seil-isr/ros2-sysmlv2



Multi-Aspect Fragment: Polyhedral Contracts

A polyhedron is the intersection of finitely many halfspaces. A Polyhedral contract is a pair $(\mathcal{A}, \mathcal{G})$ where $\mathcal{A}$ and $\mathcal{G}$ are polyhedrons representing the set of assumptions and guarantees respectively. Let the vector x be the collection of all system variables. Polyhedral contracts are general enough to reason about system parameters.

$$P := \left\{ x : \bigwedge_{k=1}^N a_k^T x \leq b_k \right\} = \{x : A_P x \leq b\}$$



Algorithms for refinement verification are implemented as linear programs. For instance, given three polyhedral contracts:

$$\mathcal{C}_i = \left(\{x : A_i^{\mathcal{A}} x \leq b_i^{\mathcal{A}}\}, \{x : A_i^{\mathcal{G}} x \leq b_i^{\mathcal{G}}\} \right),$$

for $i = 1, 2, 3$, the statement:

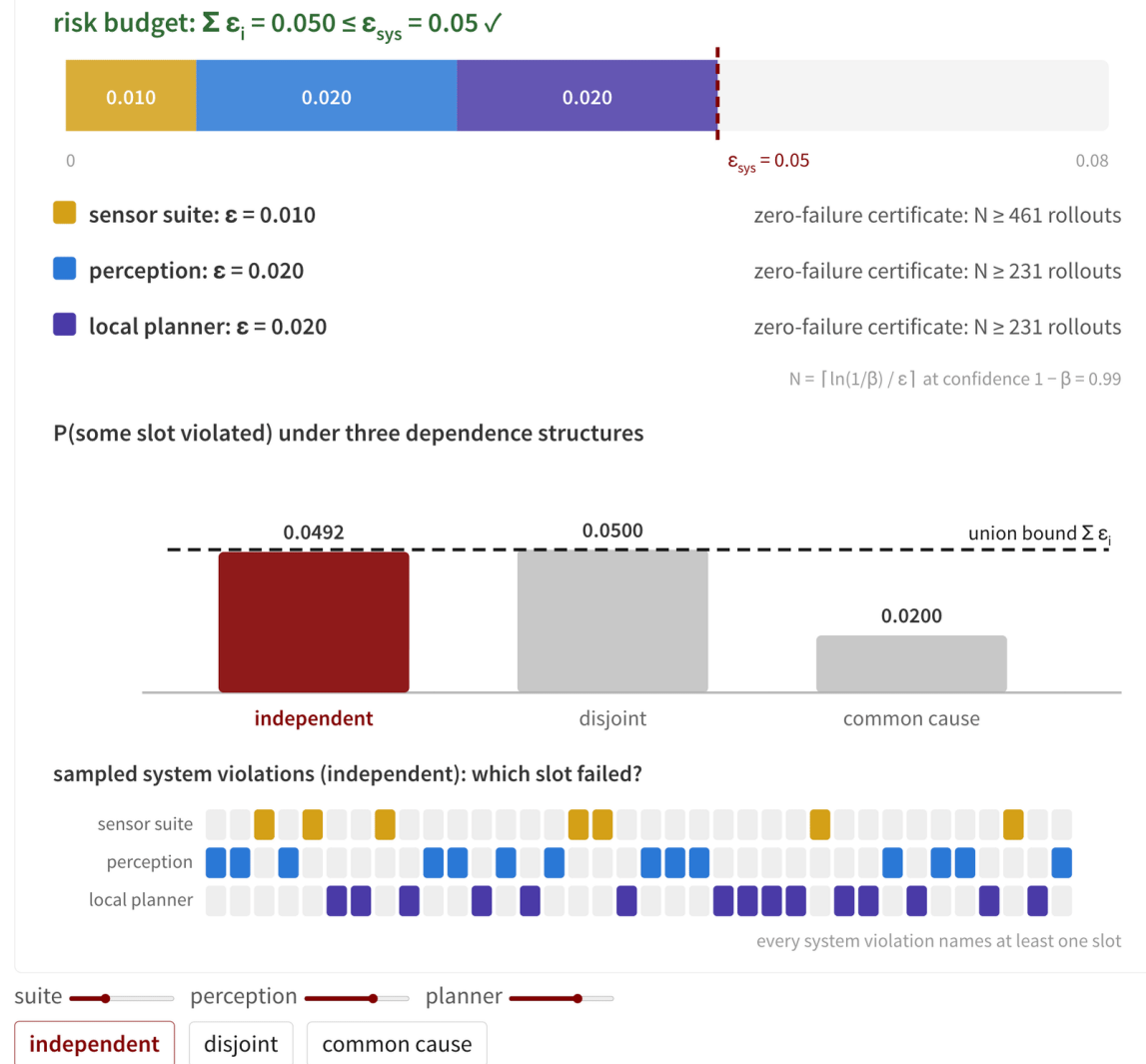
$$\mathcal{C}_1 || \mathcal{C}_2 \preceq \mathcal{C}_3 \leftrightarrow (\mathcal{A}_1 \cap \mathcal{A}_2 \cup (\mathcal{G}_1 \cap \mathcal{G}_2)^c \supseteq \mathcal{A}_3) \wedge (\mathcal{G}_1 \cap \mathcal{G}_2 \subseteq \mathcal{G}_3)$$

can be verified algorithmically by solving an LP through invocation of Farka's lemma. Thus reasoning about Polyhedral contracts is efficient and decidable.

Multi-Aspect Fragment: STL Contracts and Design Risk Quantification

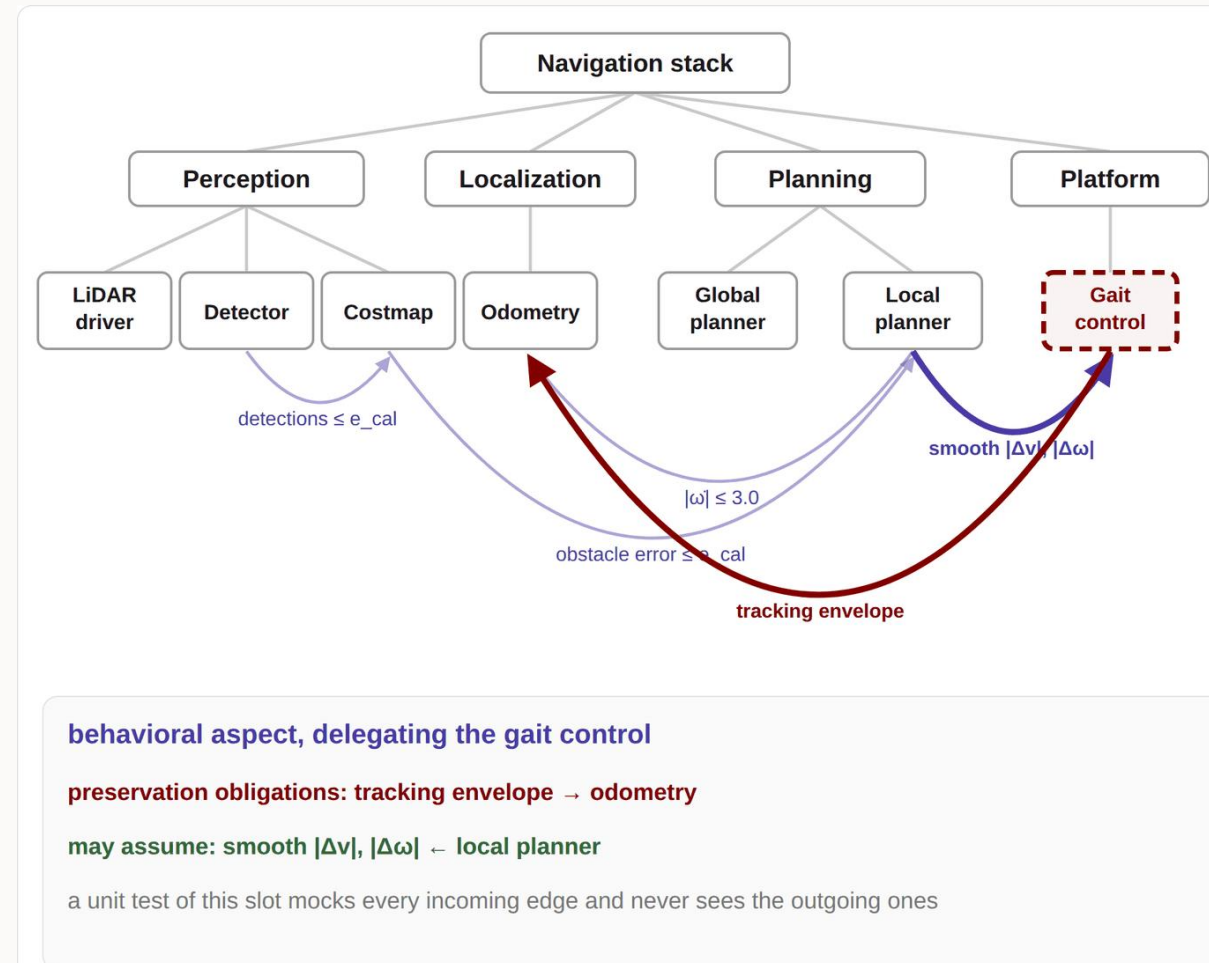
- **Signal Temporal Logic (STL).** Formulas over $\varphi := \top \mid \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_{[a,b]} \varphi_2$ carry a real-valued robustness $\rho(\varphi, x)$, with $\rho(\varphi, x) \geq 0 \Leftrightarrow x \models \varphi$ [Fainekos and Pappas, 2009].
- **Contracts are a Specification.** A pair (φ_A, φ_G) becomes the implication $\varphi_C = \neg\varphi_A \vee \varphi_G$, quantified by $\rho(\varphi_C, x) = \max(-\rho(\varphi_A, x), \rho(\varphi_G, x))$. Positive robustness means the implementation satisfies the contract.
- **Probabilistic measure of satisfaction.** A component with closed-loop signal distribution W satisfies at margin m and risk ε when:

$$\mathbb{P}(\rho(\varphi_C, W) \geq m) \geq 1 - \varepsilon,$$
 taken on the assume/guarantee implication.
- **Proposition (risk composes).** For a robustness-monotone decomposition, $\mathbb{P}(\rho(\varphi_{sys}, W) \geq \min_i m_i) \geq 1 - \sum_i \varepsilon_i$ with no independence assumed, and every runtime violation localizes to one delegation.
- **Risk budget.** Allocated like mass or power: a slot proved by model checking carries $\varepsilon_i = 0$ and releases budget to slots certified by sampling.



Delegation Descends Trees and Obligations Cross Couplings

- **Two structures per aspect.** A decomposition *tree*: SysML v2 containment is strict, and mass, power, and cost allocate down it. A *coupling* structure that is not a tree: connections cross branches, latency composes along paths, and behavioral conjuncts bind shared signals.
- **Delegation descends the tree:** the quotient is computed at a node. Classical budget allocation is iterated quotient.
- **Preservation obligations cross the couplings:** they are exactly the coupling edges that leave the slot toward a sibling that relies on them.
- **Unit tests are tree-local;** the failures live on cross-boundary couplings [cf. [interdependent multigraphs](#), Baras, 2014].



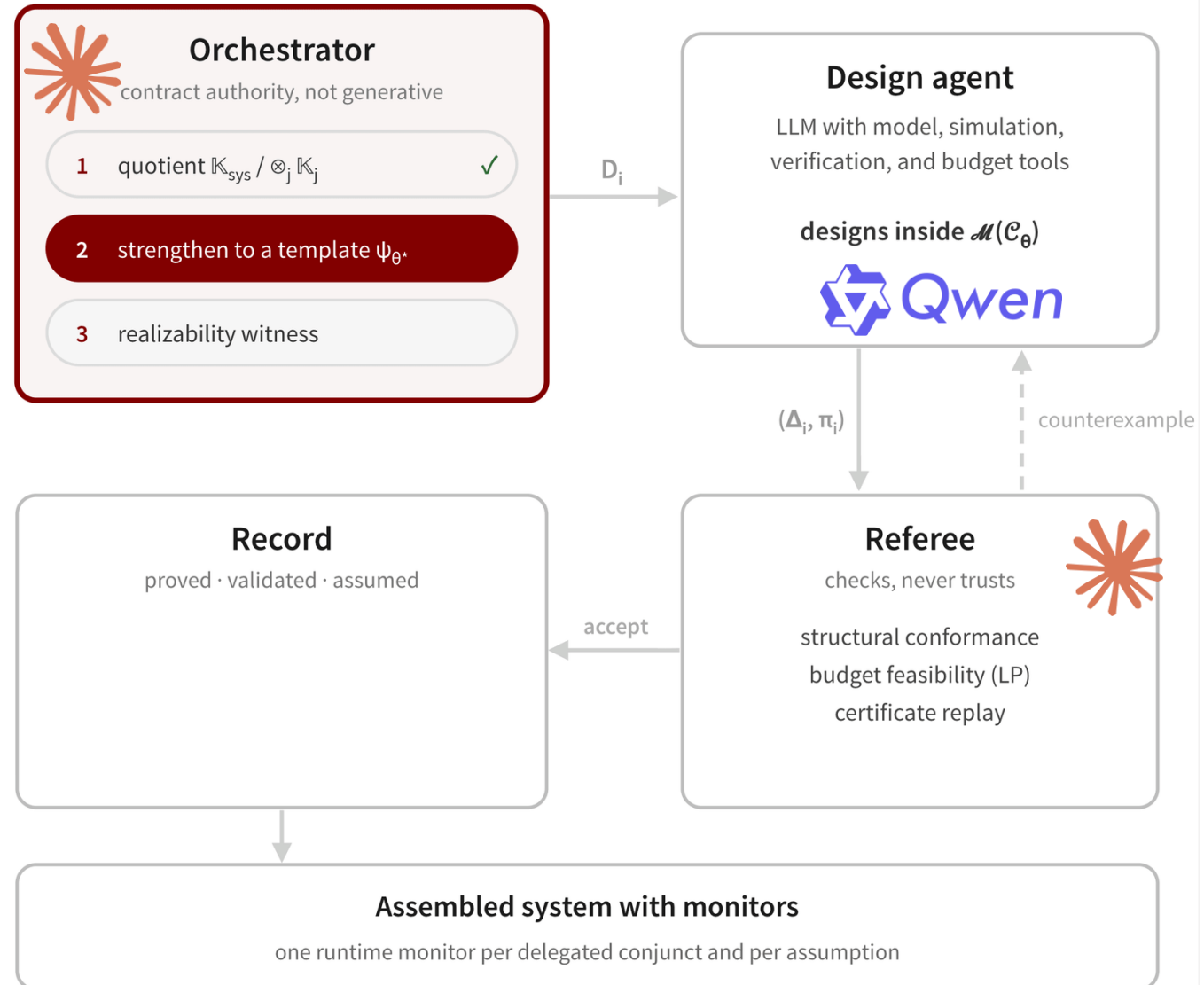
PART 4

From Theory to Practice: Demonstration on AGR Navigation Stack

The C-BASE Design Protocol: Check, Never Trust

- **Orchestrator.** The contract authority, deliberately not generative: it computes quotients aspect by aspect, strengthens each into a monotone STL template [Asarin et al., 2012], checks realizability by witness synthesis [Raman et al., 2014], and allocates risk.
- **Design agent.** Returns a certified design change — a typed model-and-code delta with a certificate that is cheap to check [Necula, 1997]: model checking (*proved*), conformal or zero-failure bounds (*validated*), falsification survival (*adversarial*).
- **Referee and record.** Accept by checking, never by trusting; every claim is recorded as proved, validated, or assumed [Leroy, 2009], and monitors ship with the system [Yamaguchi et al., 2024].

strengthening the residual into a monotone template



The C-BASE Demonstration

- **One slot, one residual.** The local controller of an authored navigation stack for Nova Carter in Isaac Sim; six siblings are fixed. The work order is the per-aspect quotient, as source-bound JSON.
- **Three aspects, one boundary.** 47 endpoints with declared QoS; ten sampled-STL conjuncts; eight metric bounds; a risk claim, $\varepsilon \leq 0.1$ over ≥ 275 episodes.
- **Evidence and authority ride along.** Retained verdicts with digests; the agent may change controller, clearance, detector — nothing else. A proposal is never an acceptance.

the system target: what the assembled robot must do

$\mathbb{K}_{\text{sys}}$: Carter system target

reach north aisle, south aisle, home
never below 0.40 m of an obstacle
terminal 0.25 m / 0.35 rad, $\varepsilon_{\text{sys}} \leq 0.1$

$\otimes$ fixed siblings $\mathbb{K}_j$

$\mathbb{K}_{\text{sys}} / \otimes_j \mathbb{K}_j$ computed per aspect by the contract kernel

work order for the local-controller slot

structural

behavioral

performance and risk

coupling

every value is from the authored contract and the retained evaluations

target

siblings

quotient

structural

behavioral

risk

coupling

authority

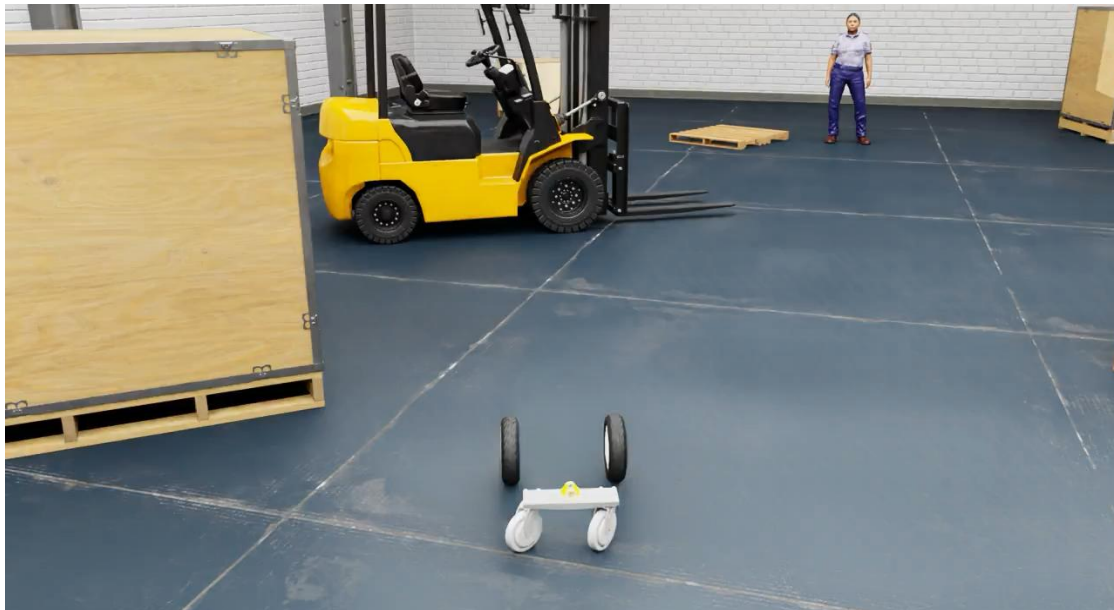
Agentic Design Workflow

- **Setup.** Qwen 3.8, 27B at 8-bit, resident on the workstation: temperature 0, fixed seed, schema-constrained JSON, thinking on. Two rounds, every request and response retained.
- **Round 1, 70 s: decline.** Both controllers violate the immutable 50 ms compute bound in retained live evidence (pursuit 86.2 ms, MPPI 69.8 ms); the agent refuses to weaken the threshold or to read inconclusive as satisfied.
- **Round 2, 83 s: decline again.** After the orchestrator restated, without changing, that admission is measurement rather than acceptance, five obligations are named as outside its authority.
- **What the contract bought.** The timing obligation is routed to its owner: selection cannot discharge it, so the orchestrator must re-derive the residual. No hidden fallback, no fabricated repair.



Results

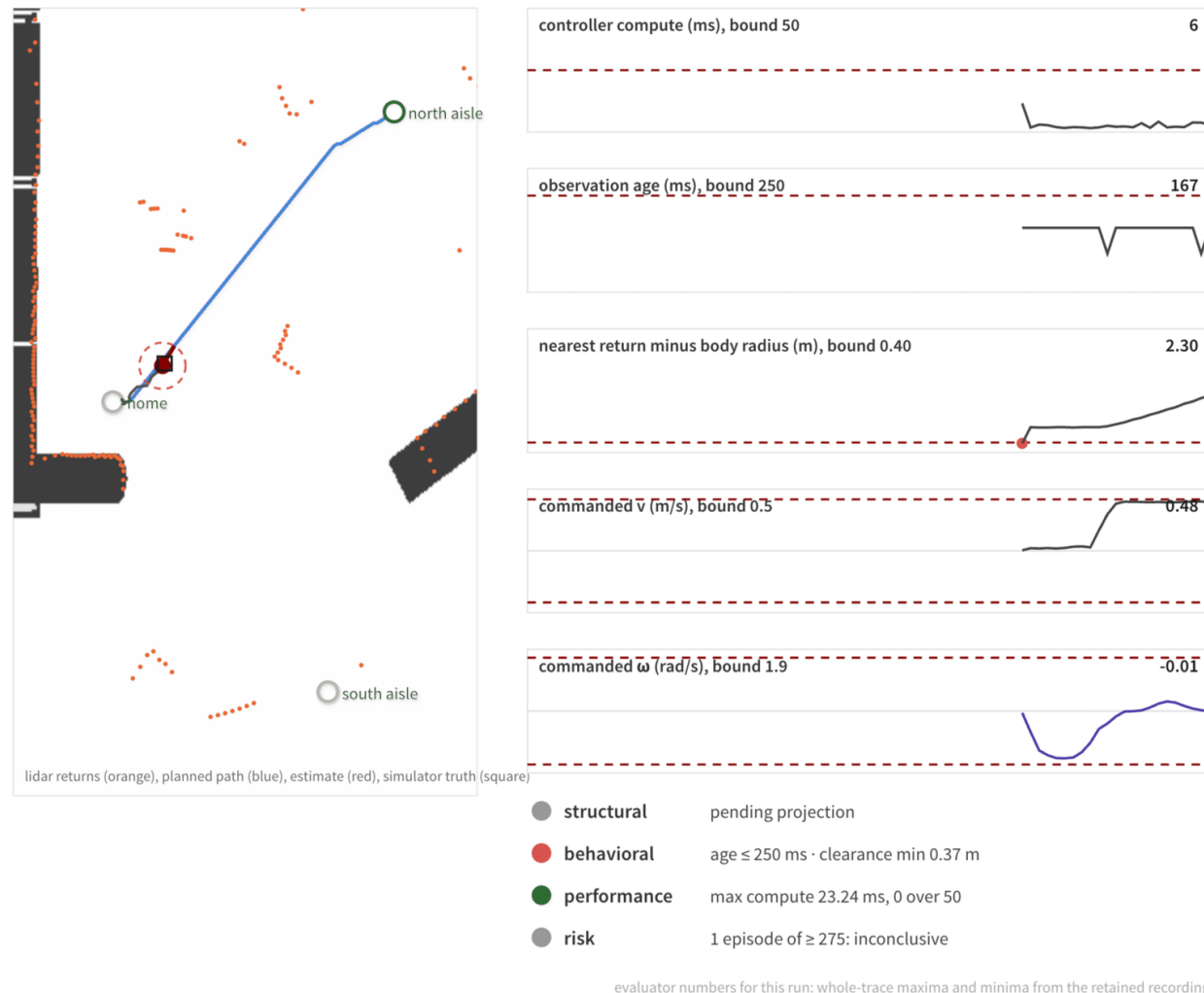
- **Problem.** Given the system contract and the fixed siblings $\mathcal{C}_j$, find the least constraining contract that still composes into the system.



Orchestrator-owned diagnostic run in Isaac Sim: the authored Python stack drives Nova Carter north aisle, south aisle, home; chase camera at 8 frames per simulated second, played at 5x. The widget follows this clip's clock

Same recording, four verdicts. Structural from the graph snapshot and QoS; behavioral from ten STL conjuncts over the trace; performance from controller events; risk only from a campaign; coupling from the sibling obligations.

MPPI, cluttered warehouse: t = 5.7 s of 99.4 s



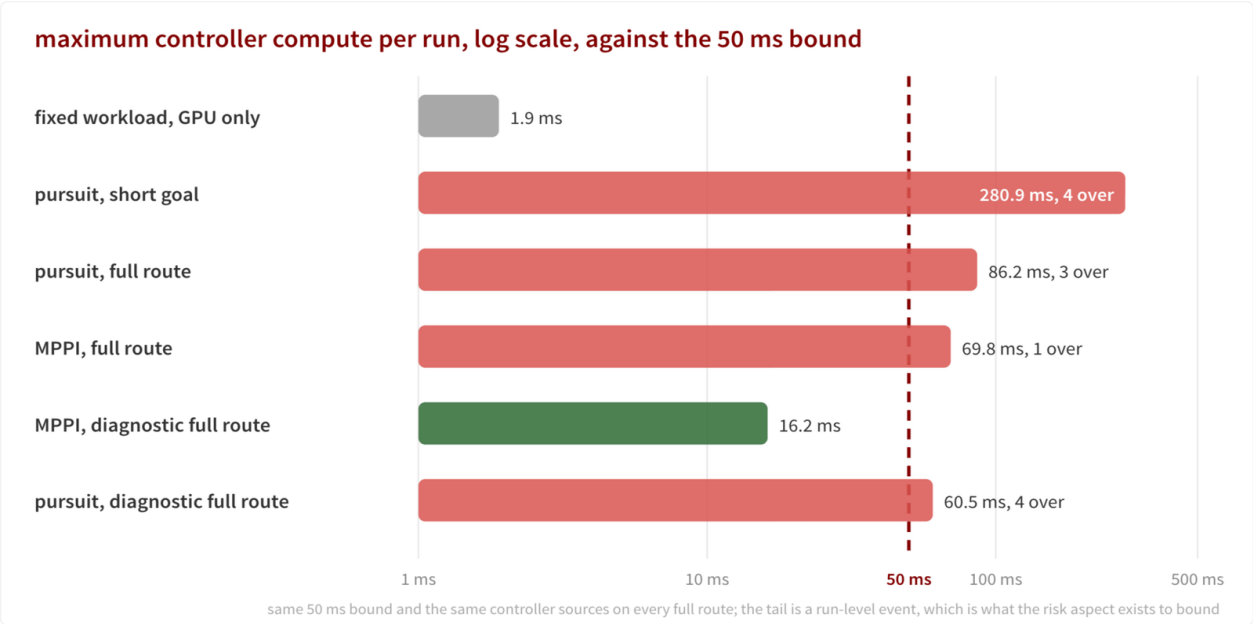
MPPI, cluttered warehouse

MPPI, empty warehouse

pursuit, empty warehouse

What Each Aspect Caught, and What Stays Open

Aspect	Finding on the live Python stack	Owner and status
Structural	14 of 47 endpoints off the declared sensor QoS: depth 5 not 10, TF subscriptions RELIABLE at depth 100	sibling runtime configuration; violated
Behavioral	observation age 0.367 s against 0.25 s, repaired by time-consistent localization to 0.20 s and 0.167 s; clearance held at ≥ 0.42 m, its proof waits on a measured localization-error bound	siblings, orchestrator; age satisfied since
Performance	compute tail 280.9, 86.2, 69.8, then 16.2 ms against 50 ms, while the fixed-workload GPU record reads 1.2 to 1.9 ms	platform; intermittent, beyond selection
Risk and coupling	$\varepsilon \leq 0.1$ needs ≥ 275 independent episodes, five exist; the residual proof waits on the structural fix and the error bound	next: the campaign



The insight. Every full route reached its three goals with terminal errors ≤ 0.08 m and ≤ 0.16 rad inside every envelope: a mission-only harness would accept all of them. The three aspects localize three defects to three owners, and the agent’s decline routed the fourth, the timing tail, to the orchestrator. The risk campaign that turns single trials into a certificate is the next step.

Takeaways

What we showed

Delegation has a canonical interface. The aspect-wise quotient is the weakest sound specification for a slot, computed from the architecture model rather than written; it carries the siblings' assumptions as literal conjuncts, the exact obligations hand-written specifications drop and unit tests cannot see; and per-slot certificates add by a union bound into a system certificate, so every violation localizes to one delegation.

What's next

Open agentic frameworks

Wire the orchestrator, agent, and referee into AWS Strands-Robots and comparable open-source agentic design stacks, so the contract layer is portable across agent runtimes.

Hardware in the loop

Take the diagnostic trial from Isaac Sim to hardware-in-the-loop design of robot platforms, with the same contracts and monitors scoring the physical run.

Native SysML v2 library

A domain library that builds the C-BASE structure on native SysML v2 requirements, so quotients, certificates, and monitors live in the model rather than beside it.

Thank You

Contract-Based Agentic Systems Engineering: Delegation as Quotient for Trusted Cyber-Physical System Design

Ryan Matheu, Sai Sandeep Damera, John S. Baras · {rmatheu, sdamera, baras}@umd.edu · University of Maryland, College Park

References



1. Dong, Y., Mu, R., Jin, G., Qi, Y., Hu, J., Zhao, X., et al. (2024). Position: Building Guardrails for Large Language Models Requires Systematic Design. ICML.
2. Rebedea, T., Dinu, R., Sreedhar, M. N., Parisien, C., Cohen, J. (2023). NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails. EMNLP.
3. Wang, H., Poskitt, C. M., Sun, J. (2025). AgentSpec: Customizable Runtime Enforcement for Safe and Reliable LLM Agents. ICSE 2026.
4. Bouamra, Y., Yun, B., Poisson, A., Armetta, F. (2025). SysTemp: A Multi-Agent System for Template-Based Generation of SysML v2.
5. Benchat, M., Briechle, D., Chanchad, R., Chauhan, M., Chavda, et al. (2026). Modeling and Recovering Hierarchical Structural Architectures of ROS 2 Systems from Code and Launch Configurations.
6. Benveniste, A., Caillaud, B., Nickovic, D., Passerone, R., et al. (2018). Contracts for System Design. Foundations and Trends in Electronic Design Automation.
7. Nuzzo, P., Sangiovanni-Vincentelli, A. L., Bresolin, D., Geretti, L., Villa, T. (2015). A Platform-Based Design Methodology With Contracts and Related Tools for the Design of Cyber-Physical Systems. Proceedings of the IEEE.
8. Object Management Group (2026). OMG Systems Modeling Language (SysML), Version 2.0, formal/2026-03-02, §7.21.2.
9. Incer Romeo, I., Sangiovanni-Vincentelli, A., Lin, C.-W., Kang, E. (2018). Quotient for Assume-Guarantee Contracts. MEMOCODE.
10. Incer, I., Badithela, A., Graebener, J. B., Mallozzi, P., et al. (2025). Pacti: Assume-Guarantee Contracts for Efficient Compositional Analysis and Design. ACM Transactions on Cyber-Physical Systems.
11. Benveniste, A., Caillaud, B., Ferrari, A., Mangeruca, L., Passerone, R., Sofronis, C. (2008). Multiple Viewpoint Contract-Based Specification and Design. FMCO 2007.
12. Fainekos, G. E. and Pappas, G. J. (2009). Robustness of Temporal Logic Specifications for Continuous-Time Signals. Theoretical Computer Science.
13. Asarin, E., Donzé, A., Maler, O., Nickovic, D. (2012). Parametric Identification of Temporal Properties. RV 2011.
14. Raman, V., Donzé, A., Maasoumy, M., Murray, R. M., Sangiovanni-Vincentelli, A., Seshia, S. A. (2014). Model Predictive Control with Signal Temporal Logic Specifications. CDC.
15. Yamaguchi, T., Hoxha, B., Ničković, D. (2024). RTAMT: Runtime Robustness Monitors with Application to CPS and Robotics. STTT.
16. Necula, G. C. (1997). Proof-Carrying Code. POPL.
17. Leroy, X. (2009). Formal Verification of a Realistic Compiler. Communications of the ACM.