

PALLC AI-GEMS

AFSIM, JSE, NGTS and Beyond

A DEVS Blueprint for the Defense Modeling and Simulation Toolchain

Brad M. Philipbar Philipbar Analytics LLC

29 September 2026

Abstract

Philipbar Analytics LLC's Artificial Intelligence for Gaming, Exercise, Modeling & Simulation (PALLC AI-GEMS) applies PALLC's expertise in the Advanced Framework for Simulation, Integration and Modeling (AFSIM), Joint Simulation Environment (JSE), Next Generation Threat System (NGTS), and heterogeneous digital engineering to a common integration problem: preserving system meaning and behavior across independently developed models and simulators. The Discrete Event System Specification (DEVS) supplies the overarching behavioral formalism. A requirements ontology captures meaning and provenance; a Blueprint Architecture expresses descriptive structure through textual model-based systems engineering (MBSE); explicit contracts govern component behavior, time, state ownership and exchange. Closure under coupling supplies the mathematical basis for recursive composition. Artificial intelligence (AI) assists requirements interpretation, model construction and code generation within a verification and validation (V&V) lifecycle. A conditional closure argument is accompanied by 248,112 finite transition checks and a synthetic timing experiment matching 2,304 analytical service decisions. Native interfaces and intended-use model adequacy have distinct V&V obligations. The resulting requirements-to-verification architecture is designed to accelerate exercises, wargames, acquisition and mission engineering while preserving explicit acceptance criteria.

Keywords: DEVS; Blueprint Architecture; requirements ontology; textual MBSE; AFSIM; heterogeneous simulation; closure under coupling; digital engineering.

1. PALLC's approach to heterogeneous simulation

Philipbar Analytics LLC (PALLC) brings specialist expertise in the Advanced Framework for Simulation, Integration and Modeling (AFSIM), Joint Simulation Environment (JSE), Next Generation Threat System (NGTS), and the wider defense modeling and simulation toolchain. Our approach combines mission engineering, computational science, Discrete Event System Specification (DEVS), and textual model-based systems engineering (MBSE). PALLC AI-GEMS packages these disciplines into Artificial Intelligence for Gaming, Exercise, Modeling & Simulation: a reusable architecture for composing existing capabilities and generating new components.

The integration challenge is behavioral. A mission-analysis model, an interactive training environment, a threat generator and an engineering solver may represent the same system at different resolutions, with different state owners and time conventions. Connecting their data exchanges leaves these differences to be resolved. PALLC begins with requirements, intended use and the governing component contracts. The ontology establishes meaning; the Blueprint Architecture allocates responsibility; DEVS defines when components react, what they emit and how their behavior composes.

Artificial intelligence (AI) assists translation from reviewed intent to textual models, scenarios, adapters and tests. Verification and validation (V&V) govern the resulting lifecycle: verification checks specified behavior and implementation; validation addresses adequacy for the intended use. The same architecture supports Department of War (DoW) acquisition, mission engineering, exercises and wargames. Reusable component contracts carry their assumptions and verification obligations into each selected configuration.

2. A common architecture across the defense toolchain

PALLC treats digital engineering, modeling and simulation (DEM&S) as a connected system of engineering responsibilities. Each tool contributes an analytical capability within a defined experiment. AI-GEMS connects those capabilities through requirements, behavioral contracts and V&V. The breadth of the defense simulation portfolio and its established integration needs are also documented in the defense-industrial-base literature [1].

Table 1: Capability responsibilities and integration obligations in AI-GEMS.

Capability	Engineering responsibility	Integration obligation
AFSIM and NGTS	Mission behavior, battlespace entities and scenario interaction	Preserve event semantics, entity ownership and experiment observables
JSE and training environments	Integrated test, ownship behavior and human participation	Define simulation, system-under-test and operator boundaries
Engineering and physics solvers	Geometry, trajectories and specialized performance	Declare units, frames, validity domains and temporal exchange rules
Industrial and sustainment models	Production, repair, logistics and readiness	Trace resource and completion events into mission consequences
Campaign and acquisition analysis	Aggregate outcomes, alternatives, cost and schedule	Preserve abstractions, dependencies and uncertainty in decisions
MBSE, data and exercise services	Requirements, configuration, observations and assessment	Link model intent, controlled execution and V&V results

The architecture distinguishes scientific responsibility from software connectivity. Two models can describe the same operational situation while using different abstractions and state representations. PALLC assigns each exchanged quantity a meaning, authoritative owner and temporal contract. The selected experiment determines whether a component runs in accelerated analysis, paced training, sampled co-simulation or offline analysis. DEVS composition operates on the declared behavior at that boundary.

Standards and lifecycle alignment

Digital-engineering policy, Gaming, Exercising, Modeling and Simulation (GEMS) guidance, and the Digital System Model (DSM) taxonomy support controlled information across lifecycle decisions [2, 3, 5]. The Defense Science Board's automated-testing work connects digital engineering to test and evaluation [4]. AI-GEMS expresses these responsibilities as controlled model intent, typed interfaces, configuration records and repeatable verification.

Shared ontologies establish meaning across model families; the digital thread connects analytical models to the descriptive system model [6, 7]. Systems Modeling Language (SysML) v2 supplies standardized textual and machine-readable representations for the Blueprint Architecture [8]. Model authority, behavior and intended-use assumptions therefore remain inspectable as the configuration evolves.

Verification checks generated artifacts and coupled execution against specifications. Validation assesses adequacy for the intended system and decision context. Performance studies measure a defined workload, including event density, coupling traffic and pacing. These distinct obligations support reuse across tools while preserving scientific meaning and engineering accountability.

3. Zeigler's foundation: composition at every scale

DEVS separates behavioral specification from simulation execution. The real system, model, simulator and experimental frame have distinct responsibilities; adequacy is judged against the intended observations [10, 11]. The governing construction below makes recursive composition explicit.

For an atomic Parallel DEVS component,

$$M_i = \langle X_i, S_i, Y_i, \delta_{int,i}, \delta_{ext,i}, \delta_{con,i}, \lambda_i, ta_i \rangle. \quad (1)$$

Here X_i, Y_i, S_i denote inputs, outputs and state; ta_i is time advance; λ_i emits pre-transition outputs. Parallel DEVS uses finite input bags and internal, external and confluent transitions [12]. Classical DEVS instead resolves imminent components with a selection function.

A coupled specification is

$$N = \langle X_N, Y_N, D, \{M_i\}_{i \in D}, EIC, EOC, IC \rangle, \quad (2)$$

where the coupling relations connect external inputs, external outputs and internal component ports, with declared translations where needed. For a finite admissible composition, closure under coupling constructs a behaviorally equivalent atomic representation. Its state retains every component's state and elapsed clock:

$$S_N \subseteq \prod_{i \in D} \{(s_i, e_i) : 0 \leq e_i \leq ta_i(s_i)\}, \quad ta_N = \min_{i \in D} \{ta_i(s_i) - e_i\}. \quad (3)$$

Let $\Delta = ta_N$, $I = \{i : ta_i(s_i) - e_i = \Delta\}$ and b_j be the complete bag routed to component j , including any simultaneous external inputs. Pre-transition outputs $\lambda_i(s_i)$, $i \in I$, are translated and combined before any state changes. The governing update is

$$s'_j = \begin{cases} \delta_{con,j}(s_j, b_j), & j \in I, b_j \neq \emptyset, \\ \delta_{int,j}(s_j), & j \in I, b_j = \emptyset, \\ \delta_{ext,j}(s_j, e_j + \Delta, b_j), & j \notin I, b_j \neq \emptyset, \\ s_j, & j \notin I, b_j = \emptyset. \end{cases} \quad (4)$$

Elapsed clocks follow the corresponding update:

$$e'_j = \begin{cases} 0, & j \in I \text{ or } b_j \neq \emptyset, \\ e_j + \Delta, & \text{otherwise.} \end{cases} \quad (5)$$

An external input arriving after $0 \leq a < \Delta$ uses elapsed time $e_j + a$ at receiving components and ages unaffected clocks by a . Passive components have infinite time advance and finite elapsed clocks; a wholly passive network awaits input. Output-before-transition is essential: confluence cannot retract an output already emitted.

Conditional closure and logical verification

Assume finite typed composition, total coupling maps, initialized finite clocks, isolated state, terminating transitions, defined confluence and non-Zeno execution. The aggregate and network initially agree. Equal remaining lifetimes select the same next event; equal pre-transition outputs and routing give the same bags; the four cases above yield equal successor states and clocks. Induction over event tags establishes external trace equality, and recursive replacement extends it through the hierarchy [12, 10]. Section 10 verifies this construction over a declared finite domain and tests targeted failure conditions.

Closure supplies mathematical scalability of description and composition. Computational cost also depends on event density, fan-out, state size, communication and solver work. Continuous or approximate components need an explicit hybrid interpretation and an applicable error relation. Their native implementations must satisfy the declared contracts.

4. Requirements ontology and Blueprint Architecture

A Blueprint Architecture is the controlled descriptive model linking purpose, decomposition, interfaces, constraints and verification obligations. AI-GEMS applies this working representation to a heterogeneous defense-engineering portfolio, following the system-theoretic lineage developed by Zeigler, Kewley and Wainer [10]. It records system organization and component purpose; executable DEVS behavior gives selected descriptive elements precise temporal meaning.

Let a requirements ontology be a typed graph $O = (V, E, \mathcal{T}, C)$. The vocabulary $\mathcal{T}$ includes requirement, source, stakeholder, component, port, quantity, frame, behavior, assumption, experiment and V&V result. Constraints C establish identity, allowed relations and required fields. Relations include *derived from*, *allocated to*, *connected to*, *realized by*, *verified by*, *validated for* and *supersedes*. Domain terms retain definitions and source spans, allowing an analyst to distinguish identical names from identical meanings.

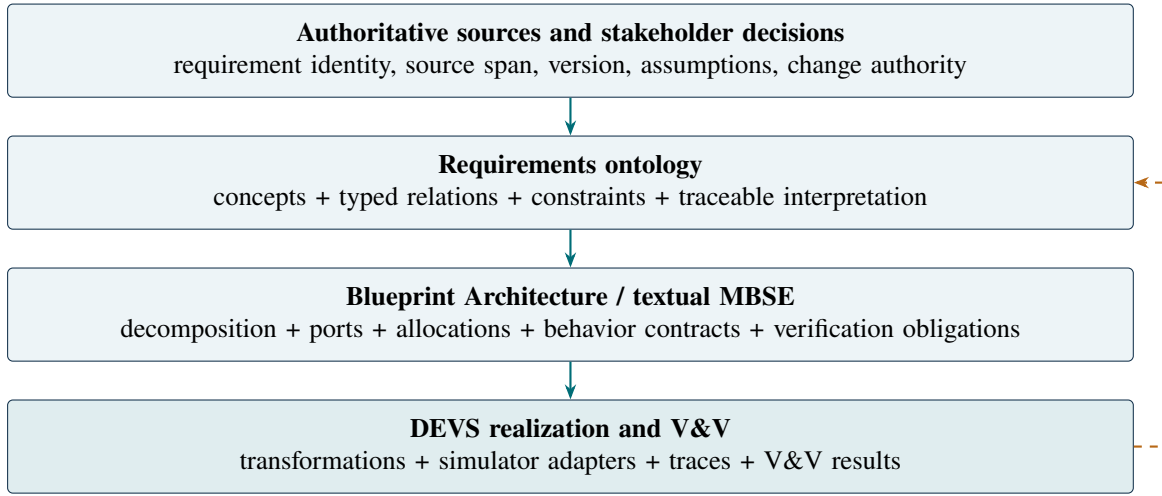


Figure 1: The descriptive blueprint and the executable realization are connected by explicit mappings. V&V results and change findings return to the controlled requirement interpretation.

For each accepted requirement r , the blueprint must retain its source and interpretation, allocate responsibility, and identify an acceptance predicate P_r over declared observations. Coverage can be checked as

$$\forall r \in R_{accepted}, \quad source(r) \neq \emptyset, \quad allocation(r) \neq \emptyset, \quad verification(r) \neq \emptyset. \quad (6)$$

Every accepted exchange additionally requires a defined schema, direction, owner, time convention and applicable unit/frame semantics. A controlled policy specifies cardinalities: one authoritative writer per state attribute per ownership epoch; one selected implementation per required component role in a realized configuration; and one or more relevant checks for every accepted requirement. Legitimate multiple sources or implementations remain representable until a configuration decision selects them.

Ontology reasoning and graph validation serve different purposes. The Web Ontology Language (OWL) supports logical entailment under its semantic assumptions; Shapes Constraint Language (SHACL) performs graph validation, verifying conformance to declared shapes [13, 14]. An absent record in an open-world ontology is not automatically false. AI-GEMS therefore applies explicit completeness checks to the accepted engineering baseline and retains unknown, disputed and superseded assertions. Semantic similarity can suggest a link; source provenance and engineering review establish it.

Verification and validation of the blueprint

Verification and validation (V&V) address distinct questions [7]. Verification checks conformance to requirements and specifications: structural consistency, traceability, transition rules, adapter contracts and composed traces satisfying P_r . The governing-equation and analytical-oracle checks are verification. Validation assesses model adequacy for its intended use against relevant system observations and stakeholder needs, with declared uncertainty. The blueprint assigns each verification obligation and intended-use validation criterion to the responsible component and experiment.

5. Textual components and a cross-lifecycle example

An illustrative equipment-availability study connects production, sustainment, mission activity and communications access. Manufacturing owns completion events; sustainment owns readiness; AFSIM owns selected mission behavior; a geometry service supplies access observations. Costs and lead times retain explicit analytical components and source records. Each allocation identifies the responsible state owner.

Table 2: Illustrative requirements traced into a descriptive and executable architecture.

ID	Accepted requirement interpretation	Allocation and acceptance observation
R1	Activity requires ready equipment and a valid contact at the decision epoch	Mission component consumes complete readiness/access bag; trace shows both predicates true
R2	A completed repair changes readiness once, with source and event identity	Sustainment component owns readiness; duplicate replay leaves the same state
R3	A design/configuration change invalidates dependent analytical predictions	Dependency graph selects invalidated analytical predictions; stale revision is rejected
R4	The same accepted inputs and seeds reproduce the declared observations	Realization records versions, initial states and seeds; differential trace check passes
R5	Reported cost and delay preserve their scenario and uncertainty assumptions	Assessment component retains source quantities and sampling design beside each result

A small textual SysML view illustrates the structural boundary [8]. Port conjugation reverses the declared flow direction at the consumer. The fragment is an explanatory interface design, not an execution or native-import receipt.

```
package AIGEMSBlueprint {
  item def Availability {
    attribute ready : ScalarValues::Boolean;
  }
  port def ReadinessPort {
    out item status : Availability;
  }
  part def Sustainment { port readiness : ReadinessPort; }
  part def MissionEngine { port readiness : ~ReadinessPort; }
  part integration {
    part support : Sustainment;
    part mission : MissionEngine;
    connect support.readiness to mission.readiness;
  }
}
```

The ontology links R1 and R2 to these identities. Executable mappings add states, elapsed-time rules and observations. At tag τ , the mission transition evaluates

$$permit(\tau) = ready(\tau) \wedge access(\tau) \wedge authorized(\tau), \quad (7)$$

using the accepted same-tag snapshot. Simultaneous repair completion and access closure enter the declared bag/phase policy before evaluation. Future completions remain unavailable to an earlier decision.

System Entity Structure (SES) describes decomposition and alternatives; pruning selects a configuration for synthesis from a model base [15, 16]. A detailed manufacturing solver or an approved aggregate lead-time component can expose the same readiness contract. The experimental question and intended-use validation results govern that selection. Interface equivalence preserves only the properties established by the refinement argument.

The example links production and sustainment decisions to mission consequences through explicit ownership and behavior. Section 9 verifies the temporal service mechanism; each manufacturing, repair or cost component requires its own V&V.

6. AFSIM, JSE and NGTS within one architecture

PALLC assigns each capability the role it serves best. AFSIM supplies mission behavior and effectiveness analysis; JSE supplies an integrated test and training environment; NGTS supplies battlespace entities and scenario behavior. JSE is itself a composed environment, so an NGTS service may already sit within the selected JSE boundary. The blueprint identifies the actual exposed interfaces, assigns state ownership and prevents duplicate authority across nested components.

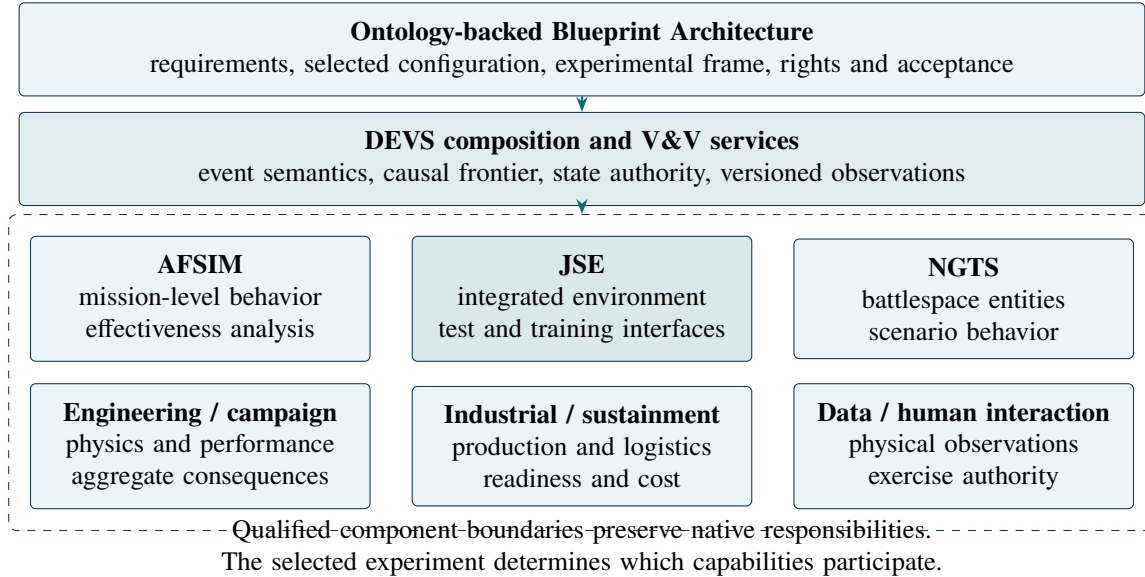


Figure 2: PALLC's federation blueprint. Boxes identify capability roles, not interchangeable simulators or mandatory pairwise connections. NGTS can participate within a JSE configuration; its ownership and time contract are specified at the actual exposed boundary.

Collaborative-aircraft experimentation in JSE and automated NGTS scenario generation illustrate the demand for reusable scenario and integration services [20, 21]. AI-GEMS connects scenario intent to formal component behavior, controlled execution and V&V across these environments. Each integration maps the installed system's interfaces to a declared behavioral contract.

Three integration modes with explicit guarantees

Native event-coordinated mode requires boundary disclosure or safe output bounds, controllable advancement, complete endpoint outputs and defined input injection. **Sampled co-simulation mode** specifies exchange periods, interpolation and event-detection limitations, with error and sensitivity studies for the intended quantities. **Offline analytical mode** exchanges versioned scenarios, response surfaces or complete trajectories before dependent execution. An interface manifest identifies the mode supported by the actual installed release; licensing and access authority remain supplier/customer responsibilities.

High Level Architecture (HLA), Distributed Interactive Simulation (DIS), Test and Training Enabling Architecture (TENA), and the Functional Mock-up Interface (FMI) occupy different interoperability roles. HLA federation agreements and time services can implement part of the execution contract; DIS and TENA exchanges require application-specific semantics and qualification; FMI capability flags govern supported stepping, clocks and optional state handling [17, 18, 19]. A single authoritative time-management scheme governs each composition boundary. Nested coordinators must translate grants and observations consistently.

Engineering tools such as Systems Tool Kit (STK) participate through the analytical contract appropriate to their role. Data, terrain, exercise-control and planning services carry provenance and configuration responsibilities. Human owners govern requirements, resources and acceptance. The same formalism can represent relevant workflow behavior while preserving those institutional responsibilities.

7. One behavioral contract across many simulators

A reusable integration contract must preserve causal order, simultaneous-event meaning, state ownership and complete observations. Shared displayed time is only one observable. PALLC applies the same system-theoretic contract to a mission-analysis component, an interactive JSE boundary, NGTS-generated behavior or an engineering service. The installed interface determines which guarantees can be implemented and qualified.

Each event carries run identity, scenario revision, logical tag, unique identifier, source, port, payload and schema. Logical tags $\tau = (n, \mu)$ contain an integer time tick n and a microstep μ . A run fixes the epoch, time scale, tick size, conversions and rounding policy. Each state attribute has one declared writer; truth state and participant-perceived state are separate quantities. Units, frames and freshness are explicit boundary semantics. Real-time pacing has its own deadline and overrun policy, separate from logical event order.

Let g be the committed frontier, b_i a disclosed next boundary, h_i a maximum admitted step, ℓ_i a certified lower bound on undisclosed output, and Q the pending queue. A candidate endpoint is

$$t^* = \min\{T_{stop}, \min_i b_i, \min_i (g + h_i), \min_i \ell_i, \min_{e \in Q} t(e)\}. \quad (8)$$

A missing b_i or ℓ_i is omitted only when an alternative certified contract constrains that component; missing all such information rejects event-coordinated admission. Empty queues contribute $+\infty$. In-flight events must be accounted for, completeness established over $[g, t^*)$, and all endpoint events supplied before dependent transitions commit. A candidate equal to g requires same-time resolution or a diagnosed deadlock.

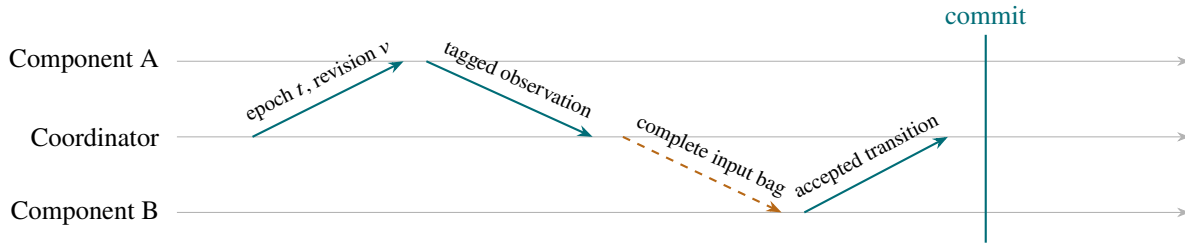


Figure 3: Proposed causal transaction. Horizontal spacing represents protocol order, not measured latency. Native adapters must demonstrate every required boundary operation.

The coordinator requests provisional evaluations, checks epoch/revision/completeness, collects outputs from pre-transition states, routes complete bags, applies internal/external/confluent transitions, drains finite zero-time consequences and commits the accepted result. Inputs that alter a trajectory invalidate dependent predictions and cached analyses. A stale result, overshoot, conflicting duplicate or late event fails the contract before an invalid commitment. Global recovery requires coherent participant state or verified deterministic replay.

Non-anticipation obligation. Pre-transition state depends on earlier admitted inputs; post-transition state may additionally depend on the complete current bag. Neither depends on a later input. Complete event disclosure, monotone advancement, finite same-time closure and correct local transitions preserve this invariant by induction over committed tags.

Adapter qualification will demonstrate the stated assumptions against the selected licensed releases. The finite reference verifies scheduling; native grants, transactional state commitment and stepping require the specified interface tests. These tests form the verification baseline for each delivered integration.

8. Why AI changes the engineering economics

DEVS provides explicit transition, time and coupling semantics for AI-assisted engineering. System Entity Structure and DEVS markup established methods for selecting, describing and assembling models well before contemporary large language models (LLMs) [15, 16]. AI-GEMS applies that formal foundation to requirements interpretation, textual MBSE, component construction and code generation.

PALLC directs AI-generated artifacts toward declared DEVS contracts and deterministic verification. Each transformation retains requirement identities, semantic types and behavioral obligations. The engineering objective is greater accepted-artifact throughput and lower translation effort under consistent V&V criteria.

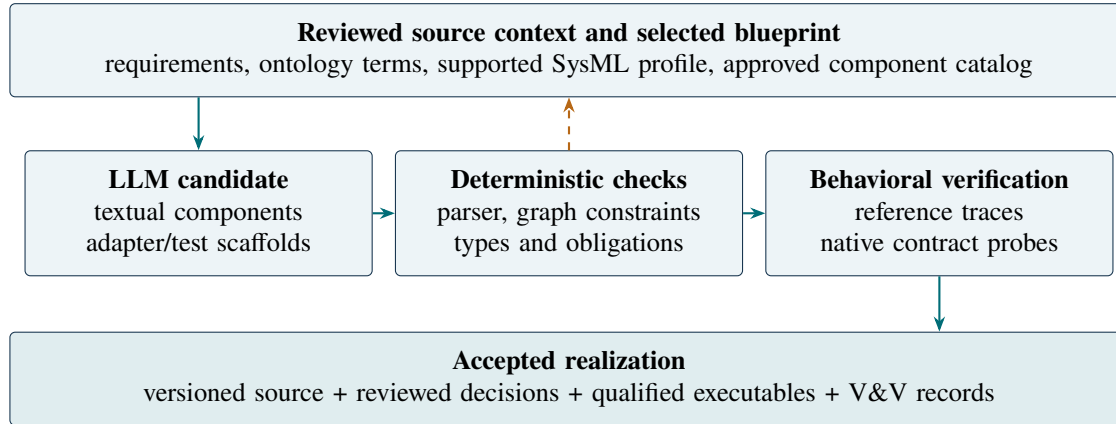


Figure 4: AI accelerates candidate construction inside a controlled generation loop. Parsing verifies syntax; behavioral verification and intended-use validation address separate obligations.

LaVoie and colleagues report 151 SysMBench prompts across four LLM backends: 604 cases in which production-checker acceptance increased from 51.16% for single-shot generation to 100% after checker-driven repair [9]. This measures conformance to the implemented rules on that dataset. Specification-driven DEVS generation and Parallel DEVS statechart verification extend this research toward behavioral models [22, 23]. AI-GEMS assigns requirements fidelity, behavioral verification and intended-use validation distinct acceptance criteria.

The AI-GEMS architecture specifies a typed intermediate representation containing ports, states, initial conditions, transitions, time advance, confluence, units/frames, random sources and adapter capabilities. Each generated behavior maps to a requirement or accepted design decision. Unsupported constructs generate a diagnostic. A semantics-preserving transformation must establish that every accepted generated trace satisfies the corresponding blueprint obligations within the declared abstraction and experiment.

A measurable generation study

A future controlled evaluation will compare manual and assisted engineering on the same requirement sets and acceptance suite. It will report engineer time, accepted components, requirement omissions, semantic defects, repair rounds and regression failures. Held-out scenarios will test transfer beyond the generation examples. LLM/backend configuration, retrieved sources, candidate artifacts, human edits and accepted hashes will be preserved where customer information policy permits. Runtime learned policies, if used, become bounded components with explicit inputs, outputs, timing, admissible actions and failure dispositions.

9. Reproducible verification of temporal behavior

The retained software experiment isolates one coupling mechanism: whether an exchange policy preserves service opportunities in known synthetic availability windows. It uses integer microseconds, exact exogenous intervals and instantaneous unlimited service. No orbit, manufacturing process or native simulator is executed. For request j with release r_j and exclusive deadline d_j , and half-open contact intervals $[a_k, b_k)$, an independent interval oracle computes

$$t_j^{oracle} = \min_k \{ \max(r_j, a_k) : \max(r_j, a_k) < \min(d_j, b_k) \}, \quad (9)$$

with no service if the set is empty. Open, close, release and deadline events form microstep-zero bags; service follows at microstep one. The polling baseline evaluates exact membership only at grid points $m\Delta$. An event-reporting fixed-step method is outside this polling-only comparison.

Seed 20260929 generates 12 schedules, 576 windows and 2,304 requests; 1,152 requests are serviceable. Durations cycle through 0.05, 0.20, 0.75, 2, 7 and 15 seconds. Starts follow $30i + 10 + U$ seconds, with seeded microsecond-grid $U \in [0, 1)$. For 5 s and 10 s polling, starts therefore occupy a restricted phase range. The cases test sensitivity, rather than estimating average operational error rates [25].

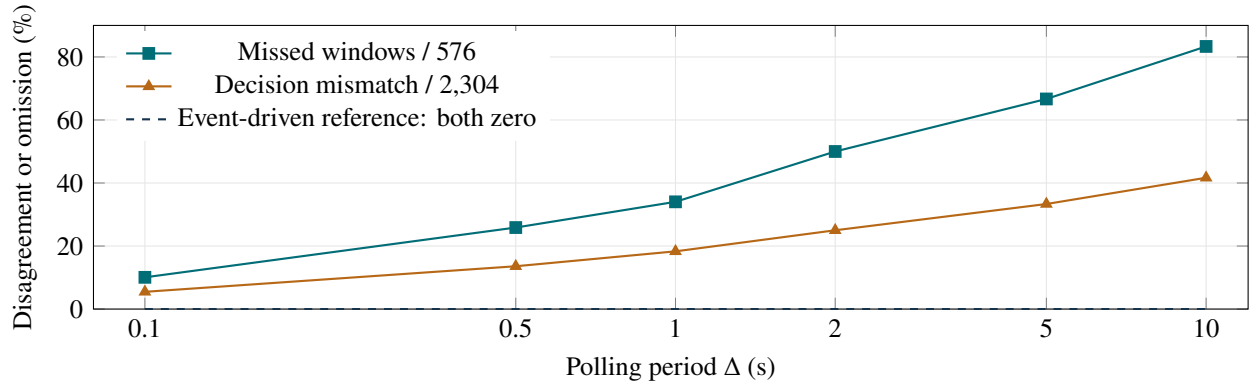


Figure 5: Measured synthetic sensitivity to polling period. Points are exact aggregate fractions from the saved experiment. The event-driven reference is independent of the displayed polling-period axis. No confidence bands are implied for this deliberately constructed suite.

Table 3: Service-time error, conditioned on both methods delivering service.

Method / grid	Both served	Missed windows	Mean error (s)	Maximum (s)
Event reference	1,152	0	0	0
Polling / 0.1 s	1,026	58	0.048621	0.099843
Polling / 0.5 s	839	149	0.237716	0.499843
Polling / 1 s	730	196	0.455672	0.999843
Polling / 2 s	576	288	1.182418	1.998852
Polling / 5 s	384	384	3.923454	4.990616
Polling / 10 s	192	480	6.994561	9.984350

The event-driven reference matches all 2,304 analytical decisions and all delivered service timestamps. Fourteen tests exercise boundary conditions, simultaneous events, seeded oracle scenarios and corrupted inputs. Independent recomputation checks every request-method outcome [25]. Complete event boundaries preserve the declared service opportunities for these inputs. This verifies the synthetic scheduling behavior; wall-clock performance and native simulator fidelity require separate evaluation.

10. Governing-equation verification and acceptance

A reproducible checker independently exercises the governing construction using exact integer clocks, three-state components, nine admissible state/elapsd pairs, four coupling topologies and four input-bag choices. Flat and grouped execution share the declared atomic transition function; a separately coded absolute-deadline/arithmetic oracle checks both across three explicit two-level regroupings. This is exhaustive verification of the declared finite domains; the general closure argument remains conditional on Section 3’s assumptions [24].

Table 4: Executed mathematical checks and their exact scope.

Governing property	Cases/checks	Verified observation
Remaining lifetime	5,832	Minimum and regrouped minima agree, including scalar zero and infinite residuals
Four transition cases	248,112	Successor state, elapsed clock, output and complete bag match the independent oracle
Hierarchical regrouping	744,336	Three partitions agree with the reference on the same one-step cases
Bag and edge ordering	248,112	Reversed bag/edge order preserves multiplicity, states and outputs
Deliberate defects	6	Targeted missing-confluence, post-transition-output, corrupted-clock/bag and invalid-advance witnesses are rejected
Supporting predicates	Multiple	Half-open service, frontier minima, conservation, probability normalization and progress checks pass

The domains exercise 51,744 internal, 232,416 confluent, 376,576 external and 83,600 unaffected component updates. Fourteen new tests include hand-derived expectations and negative controls; together with the temporal suite, all 28 tests pass. A silent-trace counterexample demonstrates why trace inclusion alone cannot establish required progress. The reproducibility record specifies the finite domains and maps each governing equation to its verification procedure [24].

Compositional refinement and its limits

Let B be a blueprint, $T(B)$ its generated realization, F the experimental frame, and α an observation abstraction. A useful acceptance obligation is

$$\alpha(Tr_F(T(B))) \subseteq Tr_F(B), \quad (10)$$

Here Tr_F contains complete or maximal behaviors under admissible inputs. Receptiveness, progress, outcomes and deadlines are also checked, preventing vacuous acceptance of a silent or deadlocked implementation. The blueprint must supply behavioral semantics and observables. Local refinement composes under compatible interfaces, scheduling, isolated state and admissible environments; cross-component properties also require composed-system checks.

Predicates P_r express ordering, bounded response, ownership, conservation or permitted transitions. A resource example is $inventory(t) = inventory(0) + receipts(t) - issues(t)$ under a declared accounting convention. Continuous physics requires equation-specific conservation, convergence and integration-error checks. Event coordination preserves its own contract alongside parameter and observation uncertainty.

Scaling and fidelity become measured quantities

Runtime studies will vary component count, event density, fan-out, horizon and worker count under a comparable scientific workload. Throughput, latency, memory, communication and reproducibility will include adverse synchronization cases. Specification complexity, execution complexity and fidelity remain separate axes [10]; the present experiment measures temporal correctness.

Aggregate surrogates require a declared state abstraction and validation frame. If N_{ab} counts eligible transitions from state a to b , the conditional estimate is $\hat{P}_{ab} = N_{ab}/\sum_c N_{ac}$ when the denominator is positive. Holding-time dependence can require a semi-Markov representation. DEVS executes the accepted transition/time model; experimental design, parameter inference and held-out validation establish its usefulness.

For a digital twin, additional obligations identify the physical referent, observation provenance, update interval, calibration, staleness policy and uncertainty. For a human exercise, participant perception, exercise authority and learning measures are distinct from simulated truth. Both can use the same composition framework while requiring distinct intended-use validation criteria.

11. PALLC AI-GEMS: an integrated offering

PALLC’s AFSIM subject-matter expertise encompasses multidisciplinary scenario engineering, custom scripts and plugins, and mission-effectiveness analysis. We combine that expertise with computational science, DEVS systems engineering and source-traceable textual MBSE [27]. AI-GEMS organizes these capabilities into an integrated offer for rapid scenario construction, heterogeneous execution and V&V.

Our portfolio connects SysML AI Code Gen, ModelShift’s explicit model-to-adapter bindings and engineering after-action review, experimental-frame methods, and DEVS contract verification. ForceWeaver mission-rehearsal and Decision Revision Engine work extends the design into scenario and acquisition workflows. The full repository ontology identifies the reusable methods and their dependencies; the supporting crosswalk retains repository coverage, source identities and verification scope [26]. Each delivered integration carries its own acceptance baseline.

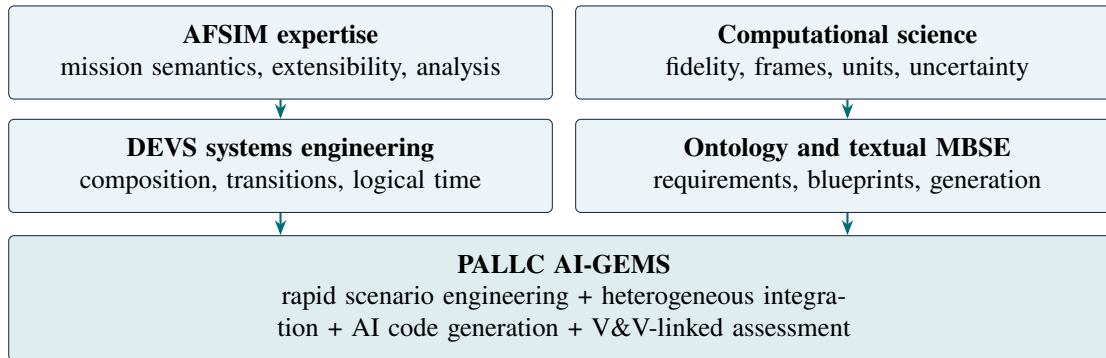


Figure 6: PALLC expertise organized into AI-GEMS delivery capabilities. Each customer configuration binds the selected tool versions, experimental frame and V&V criteria.

Customer deliverables and acceptance

We will deliver a source-linked requirement ontology and Blueprint Architecture; selected components; reviewed adapters/scenarios; reproducible execution; and assessment linked to decision or training objectives. Each configuration records rights, ownership, fidelity and V&V results. Native interface verification will demonstrate input admission, observable behavior, time control, same-time semantics and repeatable traces for selected AFSIM, JSE, NGTS and other components.

The 24-hour engineering demonstration

PALLC’s rapid-delivery objective is a 24-hour architecture increment. Starting from an agreed requirement set, accessible data and qualified reusable components, the demonstration will produce a reviewed ontology, textual blueprint, selected scenario, reference run and acceptance record. Elapsed time, engineer effort, cost, rework and deferred obligations will be recorded from accepted inputs to delivery. The target covers a bounded increment; comparative speedup requires a baseline with matched scope and acceptance.

Reusable contracts, automated transformations and immediate defect feedback accelerate delivery while preserving model rights, engineering review and intended-use validation.

Conclusion

Zeigler’s DEVS formalism supplies recursive behavioral composition; the requirements ontology preserves meaning; the Blueprint Architecture organizes descriptive structure and proof obligations. Textual MBSE and governed AI accelerate their construction. Governing equations, bounded verification and qualified interfaces give PALLC’s exercise, wargaming, acquisition and mission-engineering offering a reproducible engineering foundation.

References

- [1] T. W. Simms, OUSD(R&E). *Using Modeling and Simulation to Increase Speed to Market, Reduce Risk, and Foster Interoperability in the Industrial Base*. Congressional briefing, 1 Dec. 2022.
- [2] U.S. Department of Defense. *DoDI 5000.97: Digital Engineering*, 21 Dec. 2023.
- [3] OUSD(R&E). *Advancing Gaming, Exercising, Modeling and Simulation (GEMS) Capabilities*, 1 Oct. 2022. Upload month differs from cover date.
- [4] Defense Science Board. *Digital Engineering Capability to Automate Testing and Evaluation*, 2024. Public release 16 Aug. 2024.
- [5] Office of Systems Engineering and Architecture. *Digital Engineering, Modeling and Simulation* portal and linked Digital System Model Data Taxonomy. Digital-engineering reference catalog.
- [6] Systems Engineering Research Center. *Handbook on Digital Engineering with Ontologies*, v2.0, May 2025.
- [7] Office of Systems Engineering and Architecture. *Systems Engineering Guidebook*, Sept. 2026, §§3.1–3.4, 6.2.6–6.2.7; pp. 17–24, 137–138.
- [8] Object Management Group. *Systems Modeling Language*, v2.0. Adopted Sept. 2025; language document formal/2026-03-02, March 2026.
- [9] C. LaVoie, E. Andujar Lugo, T. G. Topcu, L. B. Kara. “Natural-Language to SysMLv2 Translation via Conformance-Driven Iterative Refinement.” arXiv:2607.14162v1, 14 July 2026. Author-reported ASME IDETC/CIE acceptance.
- [10] B. P. Zeigler, R. Kewley, G. Wainer. “DEVS Closure Under Coupling, Universality, and Uniqueness: Enabling Simulation and Software Interoperability from a System-Theoretic Foundation.” *Computers* 14(12), 514, 2025. §§3, 5, 7.4, 7.6–7.7; Apps. B, F.
- [11] D. Fournes, V. Albert, A. Nketsa. “A new specification-based qualitative metric for simulation model validity.” *Simulation Modelling Practice and Theory* 66, 1–15, 2016.
- [12] A. C.-H. Chow. “Parallel DEVS: A parallel, hierarchical, modular modeling formalism and its distributed simulator.” *Trans. Society for Computer Simulation International* 13(2), 55–67, 1996.
- [13] W3C. *OWL 2 Web Ontology Language Document Overview*, second edition, Recommendation, 11 Dec. 2012.
- [14] W3C. *Shapes Constraint Language (SHACL)*, Recommendation, 20 July 2017. Later draft work is separately versioned.
- [15] B. P. Zeigler. “Extending the Hierarchy of System Specifications and Morphisms with SES Abstraction.” *Information* 14, 22, 2023; published 29 Dec. 2022.
- [16] S. Mittal, S. A. Douglass. “DEVSMML 2.0: The Language and the Stack.” SpringSim 2012. Domain-specific textual languages and transformations.
- [17] Modelica Association Project FMI. *Functional Mock-up Interface Specification*, v3.0.2. Capability flags, clocks and co-simulation sections.
- [18] Simulation Interoperability Standards Organization. *Space Reference Federation Object Model*, SISO-STD-018-2020, §§4.2–4.5. Federation-specific conventions.
- [19] Y. Vanommeslaeghe et al. “Adapting parallel DEVS semantics to FMI 3.0 co-simulation using synchronous clocks.” *SIMULATION*, 2025. Author manuscript archived.
- [20] Naval Air Systems Command. “F-35 pilots and Navy Collaborative Combat Aircraft hone tactics in Joint Simulation Environment.” 5 Jan. 2026.
- [21] Department of the Navy. “Live, Virtual, Constructive (LVC) Afloat: Automated Scenario Generation,” topic N251-023. 2025 topic; technical answers, 9 and 17 Jan. 2025.
- [22] Z. Chen, H. Zhuang, Z. Li, C. Li. “Specification-Driven Generation and Evaluation of Discrete-Event World Models via the DEVS Formalism.” arXiv:2603.03784v2, 21 May 2026; initial submission 4 March.
- [23] V. K. Vasa, H. S. Sarjoughian, E. J. Yellig. “LLM-based Framework for Generating and Verifying Parallel DEVS Statecharts.” arXiv:2608.14956v1, 15 Aug. 2026.
- [24] Philipbar Analytics LLC. *Bounded verification of AI-GEMS governing equations*, 29 Sept. 2026. `scripts/verify_governing_equations.py`, `docs/results/governing-equations.json`, and the equation inventory in the verification directory. Exact domains, negative controls and hashes retained.
- [25] Philipbar Analytics LLC. *AI-GEMS synthetic timing experiment*, 29 Sept. 2026. `src/ai_gems_timing.py`; runner, tests and complete inputs/traces in `docs/results/`. Seed 20260929; Python 3.9.6.
- [26] Philipbar Analytics LLC. *Full-portfolio ontology crosswalk for the AI-GEMS blueprint*, 29 Sept. 2026. Portfolio blueprint crosswalk and companion source ledger in `docs/research/`. Full inventory reconciliation, source hashes, revisions and seven proposed reuse mappings.
- [27] Philipbar Analytics LLC. *AI-GEMS capability provenance and verification record*, 29 Sept. 2026. Repository research directory. Exact source revisions, claim locators, conformance records and hashes.