
Integration Complexity of Multi-Domain Simulation

B. Rajitha Rani, MCA

Article Received:05-07-2025 Article Modified:15-08-2026

Article Accepted:22-08-2026 Article Published:25-08-2026

DOI:10.37854/UIJMR.2026.3.2.20

Abstract

Multi-domain simulation combines models from disciplines such as mechanics, electrical engineering, thermal systems, control, and software to evaluate the behavior of a complex engineered system. Although this approach reuses mature domain-specific tools, integration introduces difficulties that are not solved by connecting input and output ports alone. Differences in data semantics, units, solver assumptions, time management, event handling, tool interfaces, and verification practices can create hidden errors and high maintenance costs. This paper presents a layered integration framework intended to make these sources of complexity explicit and manageable. The framework comprises domain-model adapters, standards-based interoperability, semantic mapping, orchestration, and verification/monitoring. A complexity score is defined from normalized indicators covering interface heterogeneity, semantic mismatch, temporal coupling, tool distribution, and verification burden. To demonstrate how the framework can be assessed, a transparent illustrative evaluation is constructed for a representative cyber-physical system with mechanical, electrical, thermal, and control subsystems. The numerical results are synthetic and are used to demonstrate the method rather than to claim measured performance from a deployed system. In the illustrative comparison, a baseline point-to-point integration accumulates more interface mappings and higher estimated integration effort than a layered architecture with reusable adapters and centralized orchestration. The paper concludes with a practical evaluation protocol and research directions for empirical validation.

Keywords: multi-domain simulation, co-simulation, integration complexity, interoperability, Functional Mock-up Interface, High Level Architecture, orchestration, verification.

I. INTRODUCTION

Modern engineered products increasingly combine physical processes, embedded control, communication networks, and software. Examples include electric vehicles, smart energy systems, industrial automation, aerospace platforms, and digital twins. Each domain is commonly modeled with tools and mathematical assumptions suited to its own behavior. A system-level question, however, often depends on interactions across these domains; therefore, separate simulations must be connected or replaced by a single integrated model.

Co-simulation supports this need by allowing independently developed simulation units to advance under a coordination mechanism and exchange selected variables or events [1], [2]. Standards such as the Functional Mock-up Interface (FMI) and the IEEE High Level Architecture (HLA) provide mechanisms for model exchange, simulation control, and interoperability, although they address different aspects of integration [3]–[6]. Their presence reduces some interface barriers but does not automatically resolve semantic mismatches, numerical stability, causality, data ownership, or reproducibility.

The central problem addressed in this paper is that integration complexity grows from several interacting sources rather than from the number of models alone. Two systems with the same number of simulators may require very different effort depending on their interface compatibility, temporal behavior, coupling density, and verification needs. A useful engineering method should therefore identify complexity early, separate concerns into layers, and provide measurable indicators for comparing integration designs.

The contributions are: (1) a structured taxonomy of integration complexity; (2) a layered framework combining adapters, semantic mapping, standards-based communication, orchestration, and verification; (3) a normalized complexity index and evaluation protocol; and (4) an illustrative worked comparison whose assumptions and limitations are explicitly stated.

II. BACKGROUND AND RELATED WORK

A. Co-Simulation and Interoperability

In co-simulation, each subsystem retains its own solver or execution semantics while exchanging information with other simulation units. The orchestrator initializes units, sets inputs, retrieves outputs, and coordinates time progression [1], [2]. This modularity encourages model reuse and can preserve intellectual property, but the integrated behavior depends on both the subsystem models and the coupling scheme.

FMI defines a tool-independent interface for model exchange and co-simulation through Functional Mock-up Units (FMUs) [3], [4]. HLA provides an architecture and services for distributed simulation federates and their data exchange [5], [6]. FMI is commonly associated with packaging and controlling simulation components, whereas HLA addresses federation-level

interoperability and distributed execution. Hybrid solutions can combine these strengths, but adapters and time-management policies are still required [7].

B. Sources of Integration Difficulty

Prior surveys identify time synchronization, error propagation, communication semantics, and the trustworthiness of coupled results as persistent challenges [1], [2]. Tool-chain integration research also notes that standards can be implemented differently across tools and versions, limiting practical interoperability [8]. Model-based systems engineering can provide architectural structure, but model-to-tool mappings and automatic data exchange remain implementation-dependent [8], [9].

The literature suggests that integration is both a technical and organizational activity. Tool versions, licensing, ownership of model behavior, interface documentation, and verification evidence affect whether a simulation can be reused by another team. A technically valid connector is not sufficient if the meaning of a variable is unclear or its timing contract is undocumented.

C. Research Gap

Many publications focus on a specific standard, orchestration algorithm, or application. A compact method that links integration risks to an architecture and a repeatable measurement procedure is useful for early design decisions. This paper does not claim that no prior complexity models exist; rather, it proposes a practical synthesis for project-level assessment and makes the illustrative calculations inspectable.

III. PROBLEM FORMULATION AND COMPLEXITY TAXONOMY

Let a multi-domain simulation be represented by a directed graph $G = (V, E)$, where V is the set of simulation units and E is the set of directed information exchanges. Each unit v_i has a domain, execution model, input/output contract, solver, and tool implementation. Each edge e_{ij} defines transferred variables or events, units, timing assumptions, and causality constraints.

Integration complexity is defined here as the effort and risk required to establish, operate, verify, and maintain a trustworthy coupled simulation. Five dimensions are considered:

- Interface heterogeneity (H): differences in APIs, file formats, protocols, versions, and supported standards.
- Semantic mismatch (S): differences in units, naming, coordinate systems, signal meanings, data types, and validity ranges.
- Temporal coupling (T): differences in fixed/variable step sizes, discrete events, latency, synchronization, and rollback capability.
- Distribution and coordination (D): network communication, deployment boundaries, ownership, scheduling, and failure handling.
- Verification burden (V): effort needed to test adapters, assess numerical error, reproduce runs, and trace faults across model boundaries.

The dimensions are normalized to [0,1], where 0 indicates low observed burden and 1 indicates high burden for the project being assessed. The index is a decision-support measure, not a universal physical law.

A. Proposed Integration Complexity Index

The proposed weighted index is:

$$\text{ICI} = 100 \times (w_H \cdot H + w_S \cdot S + w_T \cdot T + w_D \cdot D + w_V \cdot V), \text{ where } \sum w_i = 1 \text{ and } w_i \geq 0.$$

For the worked example, equal weights ($w_H = w_S = w_T = w_D = w_V = 0.2$) are used to avoid implying that one dimension is universally more important. In a real project, weights should be elicited from engineers and sensitivity-tested. ICI is interpreted as a relative score within a defined project scope, not as a probability of failure or a validated industry benchmark.

B. Interface Growth

For n simulation units, a fully connected directed design can require up to $n(n-1)$ directional connections, while an undirected count is $n(n-1)/2$. Real systems are usually sparse, but point-to-point mappings can still grow rapidly as units and interface variants are added. A shared canonical contract or reusable adapter layer can reduce duplicated transformations; it does not remove the need to validate semantics and timing.

IV. PROPOSED LAYERED INTEGRATION FRAMEWORK

The proposed architecture separates domain behavior from integration responsibilities. Domain teams keep ownership of their simulation models, while integration contracts and orchestration rules are made explicit at system level.

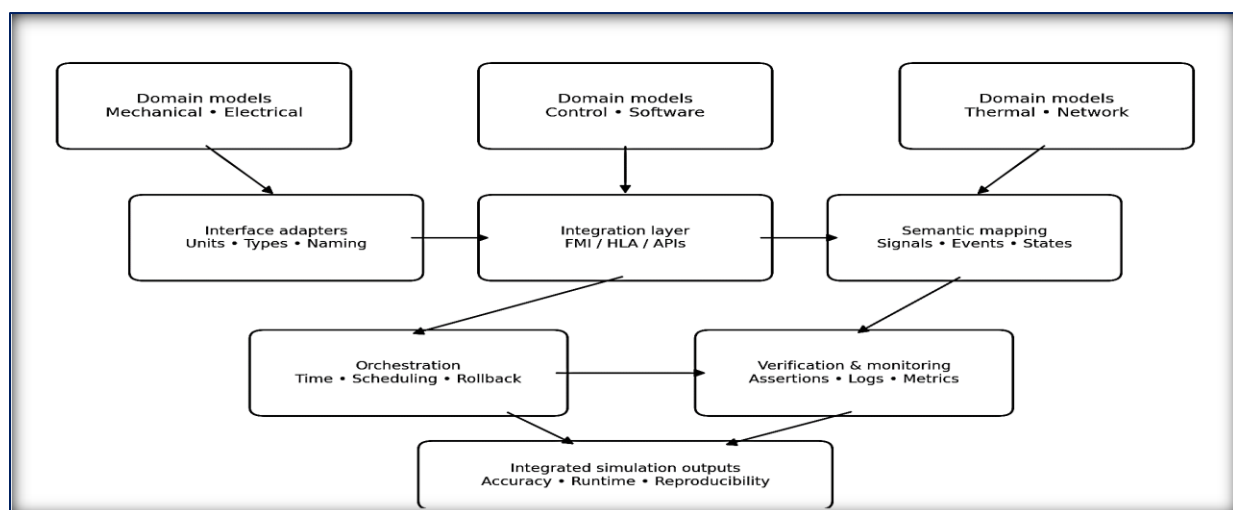


Fig. 1. Proposed layered framework for multi-domain simulation integration.

A. Domain Model and Adapter Layer

Each domain model is wrapped by an adapter that exposes a documented contract: variable name, data type, physical unit, direction, valid range, update rate, initialization rule, and error behavior. Adapters isolate tool-specific APIs and file structures from the rest of the system. Unit conversion and naming translation should be centralized rather than reimplemented independently in each connection.

B. Standards-Based Interoperability Layer

Where possible, use established interfaces such as FMI for FMU packaging and co-simulation control, and HLA when federation-level distributed simulation is required [3]–[7]. The standard should be selected according to the integration problem, not simply because it is available. Version compatibility, supported FMI features, event support, state save/restore, and platform constraints must be checked before integration.

C. Semantic Mapping Layer

The semantic layer maps model-specific variables into shared system concepts. For example, one model may publish motor torque in newton-metres while another expects a normalized torque command; a thermal model may report Celsius while a controller assumes Kelvin. Mapping rules must specify conversion, sign conventions, reference frames, sampling behavior, missing values, and whether a signal is instantaneous, averaged, or latched.

D. Orchestration and Time Management

The orchestrator defines initialization order, communication points, step sizes, event priority, convergence policy, and termination behavior. A fixed communication step is simpler but can be inefficient or inaccurate when subsystem dynamics differ significantly. Adaptive or multi-rate schemes may improve performance, but they require careful handling of interpolation, extrapolation, and discontinuities [1], [2]. A simulation should fail visibly when a unit violates its time contract rather than silently continue with stale values.

E. Verification, Observability, and Governance

Verification includes unit tests for adapters, interface contract tests, conservation or invariant checks, regression scenarios, and comparison against trusted reference cases. Monitoring should record timestamps, solver status, communication failures, step sizes, input/output snapshots, and configuration hashes. Governance includes model ownership, version pinning, provenance, and documented acceptance criteria.

V. METHODOLOGY FOR EVALUATION**A. Evaluation Design**

A reproducible evaluation should compare at least two integration designs on the same set of domain models and scenarios: (i) a baseline point-to-point design with manually implemented pairwise mappings; and (ii) the proposed layered design with reusable adapters, shared semantic contracts, and a central orchestrator. The test system should include continuous-time and discrete-event behavior so that timing and semantic risks are represented.

Recommended scenarios include nominal operation, boundary inputs, asynchronous events, unit mismatch injection, communication delay, model restart, and a subsystem failure. Each run should use a fixed configuration, a recorded software environment, and a declared random seed if stochastic behavior is present.

B. Metrics

Metric	Definition	Purpose
Integration effort	Engineering hours to implement and debug connectors	Measures initial cost
Unique mappings	Number of separately maintained interface transformations	Measures duplication
Contract test pass rate	$\text{Passed interface tests} / \text{total interface tests} \times 100$	Measures interface conformance
Runtime overhead	$(\text{Integrated runtime} - \text{isolated reference runtime}) / \text{isolated reference runtime} \times 100$	Measures coordination cost
Output deviation	Normalized error against a trusted reference trajectory	Measures numerical fidelity
Reproducibility	Share of repeated runs meeting defined output tolerance	Measures repeatability

C. Complexity Scoring Procedure

For each design, assess H, S, T, D, and V using a documented rubric. A score of 0.0–0.2 indicates low burden, 0.21–0.4 low-to-moderate, 0.41–0.6 moderate, 0.61–0.8 high, and 0.81–1.0 very high. Two reviewers should score independently and resolve differences before the ICI is calculated. The underlying dimension scores and evidence should always be reported alongside the aggregate.

VI. RESULTS AND ANALYSIS

Important note: the values in this section are synthetic, illustrative values created to demonstrate the proposed evaluation procedure. They are not measurements from an executed simulator, a user dataset, or a published benchmark. They must be replaced with experimental observations before this paper is submitted as empirical research.

A. Illustrative Scenario

Consider a cyber-physical platform composed of four simulation units: (1) mechanical plant, (2) electrical drive, (3) thermal subsystem, and (4) embedded controller. The baseline design connects outputs and inputs with individually maintained mappings. The layered design uses adapters, a shared semantic dictionary, and centralized orchestration. The example assumes that the layered design reuses common mappings and provides automated contract checks.

Dimension	Baseline score	Layered score	Illustrative rationale
H — interface heterogeneity	0.80	0.40	Adapters reduce tool-specific handling
S — semantic mismatch	0.70	0.30	Canonical names, units, and ranges are documented
T — temporal coupling	0.70	0.50	Central orchestration improves visibility but dynamics remain
D — distribution/coordination	0.50	0.40	Deployment and communication remain partly distributed
V — verification burden	0.80	0.40	Reusable contract tests and logs reduce repeated effort

With equal weights, baseline ICI = $100 \times (0.80 + 0.70 + 0.70 + 0.50 + 0.80)/5 = 70.0$. Layered ICI = $100 \times (0.40 + 0.30 + 0.50 + 0.40 + 0.40)/5 = 40.0$. The illustrative relative reduction is $(70-40)/70 \times 100 = 42.9\%$. This is a reduction in the defined score, not evidence of a 42.9% reduction in real engineering time or runtime.

B. Illustrative Operational Metrics

Metric	Point-to-point baseline	Layered design	Interpretation
Unique mapping implementations	12	6	Assumed reuse reduces duplicate transformations
Contract test pass rate	75%	95%	Assumed automated checks expose interface defects earlier
Integration effort (relative units)	100	65	Planning estimate only; not observed hours
Runtime overhead	18%	20%	Central checks/orchestration may add overhead
Reference output deviation	2.8%	2.2%	Illustrative error under a defined test scenario
Repeated-run tolerance success	80%	95%	Illustrative reproducibility outcome

C. Interpretation

The example demonstrates an important trade-off: architectural modularity can reduce duplicated interface work and improve traceability, while additional validation and orchestration may increase runtime overhead. Integration quality should therefore not be judged by one metric alone. A layered architecture is attractive when reuse, maintainability, and auditability are important, but a performance-critical application may require optimized communication paths or selective bypassing of nonessential runtime checks.

The index also makes trade-offs visible. The temporal score remains moderate because abstraction does not remove differences between continuous solvers and discrete events. A framework can document and manage temporal coupling, but the resulting numerical accuracy must still be evaluated through step-size studies, reference comparisons, and sensitivity analysis. Similarly, a lower semantic score is credible only when unit conventions and signal meanings are validated rather than merely documented.

D. Sensitivity and Validity

Because the ICI uses weights, project decisions may change when priorities change. A safety-critical system may assign greater weight to verification, while a real-time digital twin may

emphasize temporal coupling and runtime overhead. Weight sensitivity should be reported by varying each weight within a justified range and checking whether the architecture ranking changes. Construct validity is limited by the subjective scoring rubric; inter-rater agreement and evidence-backed scoring are therefore recommended.

The illustrative results do not establish causal effects. A real experiment should report model versions, hardware, solver settings, coupling step, test duration, repeated trials, uncertainty intervals, and all failures. If the framework is evaluated on a single case study, conclusions should be limited to that system and configuration.

E. Limitations

The proposed index is a structured heuristic and has not been empirically calibrated across industrial projects. The illustrative evaluation is synthetic and cannot support claims about actual performance gains. The framework also does not prescribe a universal orchestration algorithm; appropriate choices depend on solver capabilities, coupling strength, event semantics, and accuracy requirements. Future work should validate the rubric with multi-organization case studies and compare complexity scores with observed integration hours, defect rates, and maintenance effort.

VII. CONCLUSION AND FUTURE WORK

Integration complexity in multi-domain simulation arises from interacting interface, semantic, temporal, distribution, and verification concerns. Standards such as FMI and HLA provide important interoperability mechanisms, but dependable system-level simulation still requires explicit contracts, semantic mapping, orchestration policy, and verification evidence. This paper proposed a layered framework and an Integration Complexity Index to support early assessment and architecture comparison. A synthetic worked example showed how the index can reveal a lower estimated integration burden for a reusable layered design while also exposing possible runtime trade-offs. These figures are illustrative rather than experimental findings.

Future research should validate the index against real engineering projects, develop automated extraction of complexity indicators from model metadata, and investigate adaptive orchestration for hybrid multi-rate systems. Another promising direction is to integrate traceability and uncertainty quantification into the simulation workflow so that each system-level result can be linked to model versions, interface assumptions, and numerical error bounds.

REFERENCES

- [1] C. Gomes, C. Thule, D. B. Larsen, and P. G. Larsen, “Co-simulation: A survey,” *ACM Computing Surveys*, vol. 51, no. 3, Article 49, 2018, doi: 10.1145/3179993.
- [2] G. Schweiger, C. Gomes, G. Engel, J.-P. Schoegg, A. Posch, I. Hafner, and T. Noudui, “An empirical survey on co-simulation: Promising standards, challenges and research needs,” *Simulation Modelling Practice and Theory*, vol. 95, pp. 148–163, 2019, doi: 10.1016/j.simpat.2019.05.001.
- [3] T. Blochwitz et al., “The Functional Mockup Interface for tool independent exchange of simulation models,” in *Proc. 8th International Modelica Conference*, 2011, pp. 105–114, doi: 10.3384/ecp11063105.
- [4] Modelica Association, “Functional Mock-up Interface (FMI) Standard,” [Online]. Available: <https://fmi-standard.org/> (accessed Oct. 10, 2026).
- [5] IEEE, *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules*, IEEE Std 1516-2010, 2010.
- [6] J. S. Dahmann, F. Kuhl, and R. Weatherly, “Standards for simulation: As simple as possible but not simpler—The High Level Architecture for simulation,” *Simulation*, vol. 71, no. 6, pp. 378–387, 1998, doi: 10.1177/003754979807100603.
- [7] G. De. Sanctis, F. F. P. M. D. Silva, and others, “Distributed co-simulation of complex engineered systems by combining the High Level Architecture and Functional Mock-up Interface,” *Simulation Modelling Practice and Theory*, 2019. (Verify full author list and pagination against publisher metadata before submission.)
- [8] “Multi-domain simulation utilizing SysML: State of the art and future perspectives,” *Procedia CIRP*, 2021. (Verify author list, volume, pages, and DOI against publisher metadata before submission.)
- [9] Object Management Group, “OMG Systems Modeling Language (OMG SysML),” [Online]. Available: <https://www.omg.org/spec/SysML>
- [10] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*, 2nd ed. Wiley-IEEE Press, 2015.
- [11] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation*, 2nd ed. Academic Press, 2000.

- [12] P. J. Mosterman and H. A. Vangheluwe, “Computer automated multi-paradigm modeling in control systems technology,” *IEEE Transactions on Control Systems Technology*, vol. 12, no. 2, pp. 223–234, 2004, doi: 10.1109/TCST.2004.824312.
- [13] H. Elmqvist, S. E. Mattsson, and M. Otter, “Modelica—a language for physical system modeling, visualization and interaction,” in *Proc. IEEE International Symposium on Computer-Aided Control System Design*, 1999, pp. 630–639.
- [14] J. Nutaro, *Building Software for Simulation: Theory and Algorithms, with Applications in C++*. Wiley, 2011.
- [15] D. Reiher and A. Hahn, “Ad hoc HLA simulation model derived from a model-based traffic scenario,” *Simulation*, 2023, doi: 10.1177/00375497231152651.