Science Innovation
2025, Vol. 13, No. 3, pp. 26-32
https://doi.org/10.11648/j.si.20251303.12

Research Article

A Full-Stack Systems Engineering Framework for Complex Systems-of-Systems Simulation

Bo Qin, Wei Ren*

China Academy Electronics and Information Technology, China Electronics Technology Group Corporation, Beijing, China

Abstract

This paper presents a comprehensive full-stack systems engineering framework designed to enhance the adaptability, scalability, and interoperability of complex systems across various domains, including defense, transportation, and industrial automation. The proposed framework is organized into five interconnected layers: the Task Layer, which defines mission objectives and stakeholder needs; the System Architecture Layer, which captures high-level system behavior and structural decomposition; the Subsystem Layer, responsible for modeling domain-specific subsystems; the Component Layer, which encapsulates functional elements and their interactions; and the Hardware and Software Interface Layer, which manages the integration of physical components with software control logic. A key feature of this framework is its ability to enable seamless transitions from high-level requirements to detailed component specifications, ensuring traceability and coherence throughout the development lifecycle. The integration of standardized interfaces, such as the Functional Mock-up Interface (FMI), enables plug-and-play subsystem integration, promoting modularity and reusability. The framework leverages SysML for architecture modeling, discrete event simulation for subsystem behavior analysis, and a co-simulation environment for synchronized software-hardware interaction. This holistic approach supports robust system verification, validation, and iterative optimization in both design and operational phases. By enabling multi-level abstraction, cross-domain integration, and simulation-based evaluation, this structured framework provides a scalable and flexible platform for addressing the growing complexity of modern systems. It serves as a valuable asset for engineers, architects, and decision-makers seeking to accelerate development cycles, reduce integration risk, and enhance overall system performance.

Keywords

System Engineering, SoS, Simulation System, Framework

1. Introduction

Modern engineering challenges often involve the integration of multiple autonomous systems, from robotics and transport networks to communications infrastructure and control systems [1]. This complexity defines the nature of systems of systems (SoS), where individual systems must operate closely to achieve overall goals.

The dynamic nature of contemporary technological advances requires the rapid development, testing, and deployment of these integrated systems [2]. Traditional engineering approaches are often inadequate in dealing with complex SoS,

*Corresponding author: Renwei10@cetc.com.cn (Wei Ren)

Received: 17 April 2025; **Accepted:** 3 June 2025; **Published:** 11 June 2025

leading to challenges in interoperability, scalability, and adaptability [3]. Therefore, there is an urgent need for a comprehensive framework to facilitate the design, simulation and validation of SoS throughout its lifecycle.

This paper presents a full-stack system engineering framework for complex system simulation. By integrating simulation tools at all layers, from conceptual design to hardware and software interfaces, the framework aims to enhance decision making, reduce development time, and improve system reliability.

2. Background and Related Work

2.1. Systems-of-Systems in Engineering Context

In various engineering domains, SoS refers to the coordinated integration of multiple autonomous systems to achieve complex objectives [4]. While these systems can operate independently, they must work together in a larger operating environment. Frameworks like the Digital Engineering Framework for Integration and Interoperability (DEFII) provide guidelines for modeling such complex architectures, The importance of interoperability and modularity is emphasized [5].

2.2. Simulation in Systems Engineering

Simulation plays a key role in systems engineering, making it possible to test and verify system behavior in a variety of situations. Tools like the Multi-Physics Object-oriented Simulation Environment (MOOSE) provide system modeling and analysis capabilities across different domains, helping to make informed decisions during the design and development phases [6].

2.3. Full-Stack Systems Engineering

Full-stack systems engineering includes the overall integration of all system layers - from high-level objectives to low-level hardware and software components [7]. This approach ensures that changes or issues in one layer can be effectively tracked and handled throughout the system [8]. Incorporating simulation at each layer enhances the ability to predict system performance and identify potential failures early in the development process [9].

2.4. Existing Frameworks and Limitations

Current frameworks often focus on specific aspects of systems engineering or simulation, lacking a unified approach that spans the entire system stack [10]. For instance, while Model-Based Systems Engineering (MBSE) provides tools for system modeling, it may not fully integrate simulation capabilities across all layers [11]. Similarly, traditional simulation tools might not offer the flexibility required for modeling complex SoS architectures [12]. This fragmentation underscores the need for a comprehensive framework that bridges these gaps.

3. Proposed Framework: Full-Stack Systems Engineering for SoS Simulation

3.1. Overview

This section presents a conceptual framework for integrating simulation at all levels of Systems-of-Systems (SoS) engineering across systems. The framework leverages a full-stack systems engineering approach - in which system elements, from task-level requirements to software and hardware interfaces, are represented, simulated, and validated. The framework is designed to ensure that simulation is not an isolated effort, but is embedded throughout the engineering lifecycle, supporting decision making from the strategic to tactical level.

3.2. Architectural Layers of the Framework

The framework is structured as a five-layer stack, each representing a level of abstraction and engineering concern: mission layer, system architecture layer, subsystem layer, component layer and software/hardware interface layer.

3.2.1. Mission Layer

As the base layer of the full stack system engineering framework, the task layer establishes the overall goal, theory and operation scenario to guide the development and integration of complex SoS. This layer translates high-level strategic objectives into structured mission scenarios that provide a clear context for system design and analysis. Key inputs in this phase include Operational Concepts (CONOPS), which outline the envisioned use of the system in a combat environment, and detailed mission threads that represent the sequence of operational activities and interactions between system components [13]. To model and evaluate these scenarios effectively, the mission layer employs tools such as campaign-level simulations and wargame simulation systems that enable stakeholders to assess potential outcomes, identify risks, and make informed decisions early in the system development process. This structured approach ensures that subsequent engineering work is aligned with the intended mission objectives and operating environment.

3.2.2. System Architecture Layer

The system architecture layer acts as a key intermediary in full-stack systems engineering, translating high-level mission objectives into detailed system designs. This layer captures the structural and behavioral aspects of the system, detailing how the various subsystems interact, their interdependencies,

and the overall functional architecture. Key components include command and control (C2) structures, communication networks, and integrated models of system behavior to ensure consistent operation throughout the system. To model and manage these complex interactions effectively, engineers use a model-based systems engineering (MBSE) approach, using tools such as SysML to create standardized visual representations of system architecture. These tools help with system analysis, design, validation, and validation to ensure that all components meet planned business objectives and can adapt to changing requirements [14]. The system architecture layer achieves seamless integration and coordination among subsystems by providing clear and detailed system structure and behavior diagram, laying the foundation for the successful implementation and operation of the system.

3.2.3. Subsystem Layer

The subsystem layer is a key component of the entire framework, focusing on the detailed design and analysis of individual subsystems such as physical platforms, sensors, and other important components. This layer pays particular attention to the internal structure and key performance parameters of each subsystem to ensure they operate at optimal performance within the broader system environment. To achieve this goal, engineers typically employ Discrete Event Simulation (DES) technology, which models the system's operation process as a series of discrete independent events in time to evaluate the behavior of subsystems under various scenarios [15].

In addition, hardware-in-the-loop (HIL) simulators are also used in the process of integrating actual hardware components into the simulation environment, providing an effective platform for subsystem performance testing and verification under real-time conditions [16]. This approach can identify and solve potential problems at an early stage of the development process, thereby enhancing the reliability and efficiency of the entire system. By focusing on the fine modeling of subsystems, this layer ensures that each component meets its specified standards and can effectively contribute to the system's task objectives.

3.2.4. Component Layer

The component layer is the key layer in the full-stack system engineering framework, which focuses on the modeling, design and simulation of individual hardware and software modules. It includes the control logic part, the embedded system part and the specific function unit, and focuses on the interface design at the module level and the performance and characteristics of the components.

To design these components in greater detail and precision, engineers use a range of proprietary tools. Such as Simulink and MATLAB, these software are widely used to design control systems and embedded software, providing a graphical environment foundation for dynamic system modeling. Another example is Modelica, an object-oriented, equation based design language for modeling complex physical environments and systems. For specific Functional Mock-up Units (FMUs), the Functional Mock-up Interface (FMI) standard is followed, which facilitates information exchange and co-simulation of dynamic models between different platforms. For example, FMUs can be integrated into Simulink models, enabling seamless interoperability between tools and enhancing modularity of system components [17].

By leveraging these tools and standards, the component layer ensures that the behavior of each module is thoroughly validated and optimized, thereby improving the overall reliability and efficiency of complex systems.

3.2.5. Software/Hardware Interface Layer

The software/hardware interface layer is a key component of the overall framework, and its main role is to facilitate the integration and interaction between virtual simulation and physical hardware components. This layer simulates the firmware state, the firmware running behavior and the interaction between the simulated firmware to realize the integration test of hardware and software. At the same time, this layer ensures that software and hardware can better work together.

Several technical frameworks are included in this layer, such as the Co-simulation framework. It allows multiple simulation tools to operate simultaneously to achieve precise interactions between different system components. These frameworks support the synchronization of various models, so that virtual and physical elements of the system can be tested synchronously effectively.

Another common technique in this layer is digital twins, which create real-time digital copies by providing a dynamic model of a real-world physical system. These digital copies support predictive analytics and operational optimization, enabling engineers to monitor system performance and predict the likelihood of problems before they arise [18]. By integrating digital twin technology into an analog environment, engineers can gain a deeper understanding of system behavior and make more appropriate decisions throughout the development lifecycle.

By utilizing these tools and methods, the software/hardware interface layer ensures tight integration of virtual models and real-world physical components, ensuring system robustness and reducing the risk of integration issues during deployment.

Table 1. Five Layers Framework.

Layers	Defines	Inputs	Tools
Mission	Defines operational goals, doctrine, and mission scenarios.	CONOPS mission threads	Campaign simulation wargaming systems
System Architecture	Captures system interactions, interdependencies, and high-level functions.	C2 communication networks system behavior models	SysML models MBSE platforms
Subsystem	Represents physical platforms, sensors, and subsystems.	/	DES HIL
Component	Models hardware and software modules including control logic and embedded systems.	Module-level Interfaces	Simulink MATLAB Modelica FMUs
Software/Hardware Interface	Enables integration testing, communication between virtual and physical components.	Emulates firmware Real-time behavior Device-level interactions	Co-simulation frameworks Digital twins RTOS

Each layer is integrated with simulation tools aligned with specific SE lifecycle phases:

Table 2. Layers vs. SE Lifecycle.

SE Phase	Simulation Focus	Tool Examples
Requirements Analysis	Mission and scenario validation	Wargaming tools
Design	Architectural modeling Functional verification	SysML MBSE
Implementation	Module-level simulation Interface testing	Simulink
Testing & V&V	System-level & end-to-end simulation	HLA
Maintenance	Runtime monitoring Digital twins	Cloud-based simulation dashboards

3.3. Interoperability and Modularity

In a complex SoS environment, ensuring that the system is modular in design and has good interoperability is an important prerequisite to ensure that the system components can be effectively developed, integrated and evolved. The following framework can help SoS achieve these prerequisites:

Standard Interfaces: The framework leverage established standards such as Advanced Architecture (HLA) and Functional Simulation Interface (FMI) to improve interoperability between different simulation tools and platforms. HLA is a general architecture of countermeasures for distributed computer simulation systems, enabling seamless interaction between different simulation components. On the other hand, FMI improves the interoperability of simulation tools and the ability to reuse models by allowing the exchange and joint simulation of dynamic models [19].

Plug-and-play modules: In order to adapt to the dynamic nature of SoS, the framework supports plug-and-play functionality, allowing the subsystem to be replaced and upgraded at any time without the need to reconfigure various parameters. This modular approach allows components to be added, removed, or replaced with minimal impact on the overall system, enhancing the flexibility and maintainability of the system.

Multi-Resolution Modeling: Recognizing the varying levels of detail required across different system layers, the

framework incorporates multi-resolution modeling. This approach allows simulations to operate at different levels of fidelity, enabling high-level abstractions for system-wide analysis and detailed models for specific subsystems. Such flexibility ensures that resources are allocated efficiently, and simulations remain scalable and relevant to the analysis objectives [20].

By integrating these strategies, the framework ensures that complex systems can be developed and evolved efficiently, with components interacting seamlessly and simulations tailored to the specific needs of each analysis.

4. Enabling Technologies

The successful implementation of a full-stack systems engineering framework for Systems-of-Systems (SoS) simulation relies on a suite of advanced technologies. These enabling technologies facilitate the integration, interoperability, and scalability required to model complex military operations effectively. Below are key technologies that underpin the proposed framework.

4.1. Digital Twins

A digital twin is a virtual copy of a physical system that can be monitored, simulated, and analyzed in real time. In military applications, digital twins can model platforms such as aircraft, vehicles, and weapon systems, and validate the digital models to provide improvements in performance, maintenance needs, and potential failures for deployment. The USAF's Model One program enhances decision correctness and AI training of decision strategies in Digital Warfare [21] by integrating multiple live simulations into a highly aggregated digital twin environment.

4.2. Agent-Based Modeling (ABM)

Agent-Based Modeling involves simulating the actions and interactions of autonomous agents to assess their effects on the system as a whole. In defense scenarios, ABM is used to model complex behaviors of individual units, such as soldiers or vehicles, within a larger operational context. This approach aids in understanding emergent behaviors and testing strategies under various conditions [22].

4.3. Cloud-Based Simulation Platforms

Cloud computing provides scalable resources for running complex simulation systems without the need for extensive infrastructure build-out on premises. Platforms such as Rescale provide high performance computing platforms for military simulation applications, enabling them to quickly implement simulation modeling and modeling analysis. This approach of cloud computing can reduce the development time and cost of simulation systems, while allowing the construction of a wider range of simulation test scenarios [23].

4.4. Integrated Modeling Environments (IME)

Integrated Modeling Environments facilitate the collaboration of various modeling and simulation tools within a unified framework. The U.S. Navy's adoption of IME supports Model-Based Systems Engineering (MBSE) and allows for the seamless integration of different simulation models, enhancing the efficiency and effectiveness of system development processes [11].

5. Discussion

The full-stack system engineering framework based on SoS simulation system proposed in this paper can effectively improve the adaptability and scalability of complex systems. This section discusses the benefits of the framework, its potential limitations, and its positioning relative to existing approaches.

With respect to advantages, by covering all layers - from mission objectives to hardware and software interfaces - the framework ensures that each component and subsystem is aligned with the overall system's building objectives. This consistency ensures the consistency of the system behavior and improves the efficiency of the task. The framework also emphasizes the use of unified standard interfaces, such as HLA and RPR FOM, to improve interoperability between different systems. The framework integrates digital twins and continuous simulation to support system evaluation and maintenance throughout the life cycle. This capability ensures continuous performance and adaptability in dynamic operating environments.

In terms of limitations, implementing a full-stack simulation framework requires significant resources, including professionals, tools, and infrastructure. Therefore, the rational allocation of resources and the necessary investment are a major challenge to realize the mine construction. In addition, the framework relies on extensive data exchange and cross-layer integration, and therefore requires a strong data management strategy to ensure data consistency, security, and accessibility. In addition, while the framework promotes the use of standard interfaces, integrating interfaces with various simulation tools and models remains a challenge. Differences in tool functionality and formats can hinder interoperability, which requires additional coordination. The proposed framework builds on established methodologies such as MBSE. By extending these methods, comprehensive modeling and requirements analysis can be carried out across layers.

6. Conclusion

The complexity of modern systems is increasing, so engi-

neers need a comprehensive systems engineering approach to address complex system design, especially for SoS simulation systems. The full-stack systems engineering framework proposed in this paper aims to meet this need by integrating modeling and simulation across all system layers, from mission planning to hardware and software interfaces. This full-stack system engineering framework ensures that each component and subsystem is aligned with the overall objectives to enhance the consistency of system objectives and the effectiveness of system functions.

By leveraging technologies such as digital twins, advanced architecture (HLA), cloud-based simulation platforms, and integrated modeling environments, the framework enables enhanced simulation system interoperability, system modularity, and lifecycle management.

Although this framework can bring significant advantages to simulation system design, challenges coexist. These challenges include the complexity of implementing the framework, data management challenges, and harmonizing tool integration and interface standardization. Addressing these challenges is critical to the successful implementation of the framework.

In short, the full stack system engineering framework provides a solid foundation for the development of complex simulation systems. By building a comprehensive approach, it can effectively enhance the reliability, adaptability and effectiveness of simulation systems in dynamic operating environments. Future work will focus on refining the framework, addressing the many challenges faced during implementation, and exploring its application in various scenarios to further validate the effectiveness of the framework.

Abbreviations

MBSE	Model-Based Systems Engineering
SysML	Systems Modeling Language
DEFII	Digital Engineering Framework for Integration and Interoperability
MOOSE	Multi-Physics Object-oriented Simulation Environment
CONOPS	Operational Concepts
SoS	Systems-of-Systems
C2	Command and Control
DES	Discrete Event Simulation
HIL	Hardware-in-the-loop
FMUs	Mock-up Units
FMI	Functional Mock-up Interface
HLA	Advanced Architecture
ABM	Agent-Based Modeling
IME	Integrated Modeling Environments

Author Contributions

Qin Bo: Conceptualization, Methodology, Investigation
Wei Ren: Validation, Project administration

Funding

This work is supported by CETC.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Gorod, A., Sauser, B., & Boardman, J. (2008). System-of-systems engineering management: A review of modern history and a path forward. *IEEE Systems Journal*, *2*(4), 484-499.

[2] Hirshorn, S. R., Voss, L. D., & Bromley, L. K. (2017). NASA systems engineering handbook (No. HQ-E-DAA-TN38707).

[3] Kossiakoff, A., Sweet, W. N., Seymour, S. J., & Biemer, S. M. (2011). Systems engineering principles and practice (Vol. 83). John Wiley & Sons.

[4] Nielsen, C. B., Larsen, P. G., Fitzgerald, J., Woodcock, J., & Peleska, J. (2015). Systems of systems engineering: basic concepts, model-based techniques, and research directions. *ACM Computing Surveys (CSUR)*, *48*(2), 1-41.

[5] Dunbar, D., Hagedorn, T., Blackburn, M., Dzielski, J., Hespelt, S., Kruse, B.,... & Yu, Z. (2023). Driving digital engineering integration and interoperability through semantic integration of models with ontologies. *Systems Engineering*, *26*(4), 365-378.

[6] Icenhour, C., Keniley, S., DeChant, C., Permann, C., Lindsay, A., Martineau, R.,... & Shannon, S. (2018). *Multi-physics object oriented simulation environment (moose)* (No. INL/CON-18-51061-Rev000). Idaho National Lab.(INL), Idaho Falls, ID (United States).

[7] Taivalsaari, A., Mikkonen, T., Pautasso, C., & Systä, K. (2021, May). Full stack is not what it used to be. In *International conference on web engineering* (pp. 363-371). Cham: Springer International Publishing.

[8] Blanchard, B. S. (2004). System engineering management. John Wiley & Sons.

[9] Rainey, L. B., & Tolk, A. (2015). Modeling and Simulation Support for System of Systems Engineering Applications.

[10] Gregory, J., & Salado, A. (2024, July). Towards a Systems Engineering Ontology Stack. In *INCOSE International Symposium* (Vol. 34, No. 1, pp. 1304-1318).

[11] Riva, M., Zanutta, A., Genoni, M., Scalera, M. A., & Balestra, A. (2024, August). MBSE or no MBSE: is MBSE the final answer to system engineering?. In Modeling, Systems Engineering, and Project Management for Astronomy XI (Vol. 13099, pp. 115-120). SPIE.

[12] Sinha, R., Paredis, C. J., Liang, V. C., & Khosla, P. K. (2001). Modeling and simulation methods for design of engineering systems. *J. Comput. Inf. Sci. Eng.*, *1*(1), 84-91.

[13] DoD, D. I. (2008). 5000.02: Operation of the defense acquisition system. *US Department of Defense, Washington, DC*.

[14] Hart, L. E. (2015, July). Introduction to model-based system engineering (MBSE) and SysML. In *Delaware Valley INCOSE Chapter Meeting* (Vol. 30). Mount Laurel, New Jersey: Ramblewood Country Club.

[15] Babulak, E., & Wang, M. (2010). Discrete event simulation. *Aitor Goti (Hg.): Discrete Event Simulations. Rijeka, Kroatien: Sciyo*, 1.

[16] Bullock, D., Johnson, B., Wells, R. B., Kyte, M., & Li, Z. (2004). Hardware-in-the-loop simulation. *Transportation Research Part C: Emerging Technologies*, *12*(1), 73-89.

[17] Broman, D., Brooks, C., Greenberg, L., Lee, E. A., Masin, M., Tripakis, S., & Wetter, M. (2013, September). Determinate composition of FMUs for co-simulation. In *2013 Proceedings of the International Conference on Embedded Software (EMSOFT)* (pp. 1-12). IEEE.

[18] Kannapinn, M., Schäfer, M., & Weeger, O. (2024). TwinLab: a framework for data-efficient training of non-intrusive reduced-order models for digital twins. *Engineering Computations*.

[19] Hällqvist, R., Munjulury, R. C., Braun, R., Eek, M., & Krus, P. (2022). Realizing interoperability between mbse domains in aircraft system development. *Electronics*, *11*(18), 2901.

[20] Shanks, G. (2003). Real-time Platform Reference Federation Object Model (RPR FOM) Version 2.0 D17. *Simulation Interoperability Standards Organization*.

[21] Singh, M., Fuenmayor, E., Hinchy, E. P., Qiao, Y., Murray, N., & Devine, D. (2021). Digital twin: Origin to future. *Applied System Innovation*, *4*(2), 36.

[22] Sabzian, H., Shafia, M. A., Bonyadi Naeini, A., Jandaghi, G., & Sheikh, M. J. (2018). A review of agent-based modeling (ABM) concepts and some of its main applications in management science. *Interdisciplinary Journal of Management Studies (Formerly known as Iranian Journal of Management Studies)*, *11*(4), 659-692.

[23] Taylor, S. J., Kiss, T., Anagnostou, A., Terstyanszky, G., Kacsuk, P., Costes, J., & Fantini, N. (2018). The CloudSME simulation platform and its applications: A generic multi-cloud platform for developing and executing commercial cloud-based simulations. *Future Generation Computer Systems*, *88*, 524-539.