Human Factors in Design, Engineering, and Computing, Vol. 238, 2026, 2592–2602
https://doi.org/10.54941/ahfe1008484

End-to-End MBSE Framework for Lifecycle Traceability Digital Engineering and System-of-Interest Realization

Joshua Adelabu[1], Bhushan Lohar[1], John T. Wade[1], Sean Walker[1], and Carlos Montalvo[2]

[1]Department of Systems Engineering, University of South Alabama, Mobile, AL 36688, USA
[2]Department of Mechanical, Aerospace, and Biomedical Engineering, University of South Alabama, Mobile, AL 36688, USA

ABSTRACT

The increasing complexity of modern engineered systems, systems-of-systems, and digitally enabled enterprises has intensified the need for Model-Based Systems Engineering (MBSE) approaches that provide comprehensive lifecycle integration, architectural continuity, and end-to-end traceability. While existing MBSE frameworks offer valuable modeling methodologies and architectural viewpoints, many remain concentrated on system definition and development activities, providing limited guidance for maintaining traceability across implementation, integration, verification, validation, deployment, sustainment, maintenance, modification, and retirement. This paper presents novel ongoing research findings on E2E MBSE Framework that establishes structured lifecycle architectures spanning the complete realization of a System of Interest (SoI), from evolution to extinction. This research's previous publications introduced conceptual fundamental constructs and formalized the operational architecture. Building upon these foundations, the present work completes the original research objective by integrating all twenty-four lifecycle packages into a coherent architectural structure that bridges both the left and right sides of the Systems Engineering (SE) Vee Model and establishes end-to-end lifecycle continuity. The framework is supported by a traceable architecture of 166 modeling constructs and collectively address evolution, mission analysis, stakeholder analysis, requirements analysis, system analysis, operational analysis, architecture development, implementation, integration, verification, validation, qualification, digital twin, deployment, sustainment, production scalability, in-service maintenance, and retirement or extinction. The principal contribution of this research is the establishment of a lifecycle-integrated framework that bridges longstanding gaps on both the left and right sides of SE Vee model by providing digital continuity through traceability, verification evidence, validation outcomes, and digital engineering artifacts. The paper discusses each package and demonstrates how the framework enables E2E digital engineering. A forthcoming dissertation presents a detailed SoI case study that applies all twenty-four packages to validate the framework's practical utility, completeness, and scalability for complex systems and systems-of-systems environments. The domain-agnostic E2E MBSE Framework can be utilized by any domain for any SoI.

Keywords: Digital engineering, Lifecycle traceability, Digital thread, Virtual twin, Digital twin, End-to-end model-based systems engineering

Received September 16, 2026; Revised October 21, 2026; Accepted October 26, 2026; Available online December 1, 2026

© 2026 The Authors. This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

INTRODUCTION

The increasing complexity of modern engineered systems, systems-of-systems (SoS), and digitally enabled enterprises has intensified the need for Model-Based Systems Engineering (MBSE) approaches that provide comprehensive lifecycle integration, architectural continuity, and end-to-end traceability. As modern engineered systems, SoS, and digitally enabled enterprises become more complex, multidisciplinary, and longevity-prone, engineering approaches must establish and preserve consistency, traceability, and decision continuity throughout the System of Interest (SoI) lifecycle. MBSE addresses these challenges by shifting from document-centric practices to model-centric methods, approaches, and frameworks that improve communication, integration, and lifecycle management. Recent studies highlight MBSE as a foundational component of Digital Engineering by supporting analysis, verification, validation, and collaboration across engineering (Cederbladh et al., 2024).

While existing MBSE frameworks offer valuable modeling methodologies and architectural viewpoints, many remain concentrated on system definition and development activities, providing limited guidance for maintaining traceability across implementation, integration, verification, validation, deployment, sustainment, maintenance, modification, and retirement. Essentially, the right side of the SE Vee Model remains a gap from a framework perspective (Adelabu & Lohar, 2025a, 2025b, 2025c). As a result, organizations often experience fragmented information flows, disconnected lifecycle artifacts, and incomplete digital continuity across engineering environments. Recent digital thread research similarly identifies lifecycle traceability and interoperability as persistent challenges in complex system development (Wu et al., 2024).

Previous paper introduced the conceptual foundations of the End-to-End Model-Based Systems Engineering (E2E MBSE) Framework (Adelabu et al., 2025), while subsequent paper formalized the Operational Architecture package and demonstrated architectural synthesis (Adelabu et al., 2026). Building upon these foundations, this paper completes the original research objective by integrating all twenty-four lifecycle packages into a comprehensive framework spanning evolution through extinction. The resulting framework establishes lifecycle continuity, end-to-end traceability, and digital engineering integration across both sides of the SE Vee Model.

LITERATURE REVIEW

Recent systematic reviews demonstrate that MBSE has matured into a practical engineering architype supporting early validation, verification, architectural analysis, and stakeholder communication (Cederbladh et al., 2024). However, most implementations remain focused on system definition and development phases rather than complete lifecycle realization from end to end (Adelabu et al., 2025).

Concurrently, Digital Engineering (DE) initiatives have expanded the role of models beyond design activities toward enterprise-wide integration and interoperability. Semantic integration approaches that establish authoritative

sources of truth across engineering environments, emphasizing interoperability among heterogeneous tools and disciplines are being proposed (Dunbar et al., 2023). These capabilities are fundamental requirements for DE ecosystems.

Digital thread research further extends this concept by enabling continuous information exchange and traceability throughout the system lifecycle. For example, E2E MBSE Framework's Ops-Arch "provides a robust foundation for tool-agnostic analysis, semantic interoperability, and lifecycle traceability, functioning as a critical semantic juncture in the Digital Thread." (Adelabu et al., 2025). Recent work on traceability management within MBSE environments demonstrates that integration challenges among stakeholders, processes, and models remain significant barriers to achieving lifecycle continuity. Ontology-based approaches have been proposed to improve traceability management; however, most remain focused on information integration rather than complete lifecycle realization (Wu et al., 2024).

Digital twin research similarly emphasizes lifecycle integration. Recent studies argue that digital twins can extend MBSE beyond design activities by maintaining synchronization between virtual and physical systems throughout operation and sustainment (Madni et al., 2023). These studies highlight the importance of integrating digital twins within broader digital thread architectures to support lifecycle decision-making, verification, validation, and performance optimization.

Collectively, the literature demonstrates substantial progress in MBSE, digital engineering, digital threads, and digital twins. Nevertheless, a significant gap remains: few frameworks provide an integrated architectural structure that spans the complete realization of a SoI from evolution to extinction while maintaining traceability and digital continuity across both sides of the SE Vee Model. This gap motivates the development of the E2E MBSE Framework.

METHODOLOGY

This research adopts a Design Science Research (DSR) methodology to develop and refine an architectural framework capable of addressing identified lifecycle integration and traceability gaps. DSR is appropriate because the primary objective is the creation of a novel engineering artifact intended to solve a recognized problem within SE and DE practice. DSR is defined as the creation and rigorous evaluation of artifacts ranging from constructs, models (including patterns, (Bhushan, 2022) and meta-models), methods, instantiations, providing solutions to concrete problems with emphasis on relevancy or context definition, design or experiment execution, and rigorous analysis or iteration (Pastor et al., 2024).

Framework development occurred across three progressive phases. The first phase established the conceptual foundations of the E2E MBSE Framework (Adelabu & Lohar, 2025a) and identified twenty-four lifecycle packages (Adelabu & Lohar, 2025b) spanning the complete realization of a SoI (Adelabu & Lohar, 2025c), (Adelabu et al., 2025). The second phase formalized the Operational Architecture package and demonstrated

architectural synthesis through structured modeling constructs (Adelabu et al., 2026). The third phase, presented in this paper, integrates all twenty-four comprehensive lifecycle packages as the E2E MBSE Framework supported by 166 modeling constructs organized across both sides of the SE Vee Model. The framework establishes traceable relationships among stakeholder needs, requirements, architectures, implementation artifacts, verification evidence, validation outcomes, operational data, sustainment information, and retirement decisions.

Traceability is maintained through interconnected modeling constructs that collectively support digital thread continuity from evolution to extinction. The framework also provides a foundation for Virtual Twin and Digital Twin integration by connecting lifecycle information across development, operation, and sustainment environments (Körtvélyi et al., 2026).

RESULTS AND DISCUSSION

24 Packages End-to-End MBSE Framework and Architectural Spines

The 24-package E2E MBSE Framework with its associated architectural spines numericized in percentages depicted in Figure 1 is discussed below.

#1 Evolution: It establishes the SoI strategic origin and rationale. Its spine (Origin → Objective → Order → Outcome → Opportunity) captures the progression from problem recognition to future value creation. This package defines why the system should exist, aligns stakeholder intent with enterprise objectives, and provides the foundational context for all subsequent modeling activities. Results demonstrate that Evolution creates traceable justification for system development, reducing ambiguity while ensuring architectural continuity across the End-to-End MBSE Framework.

#2 Mission Analysis: It transforms strategic intent into an executable mission context. Its spine (Mission → Mission Intent → Mission Outcome → Mission Effect → Mission Context → Mission Constraint → Mission Success Criteria → Mission Definition) defines what success means, the effects, the environment, and governing limitations. This package establishes mission-driven intent that could shape requirements and inform operational boundaries before solution development begins. Results indicate that Mission Analysis improves stakeholder alignment, strengthens operational relevance, and provides a rigorous basis for subsequent stakeholder, requirement, and architectural analyses within the E2E MBSE Framework.

#3 Stakeholder Analysis: It identifies and characterizes all entities that influence, use, support, regulate, or are affected by the SoI. Its spine (Stakeholder → Stakeholder Kind → Stakeholder Classification → Stakeholder Lifecycle Participation → Stakeholder Directional Activity → Stakeholder Behavioral or Engagement State → Stakeholder Needs Definition) captures stakeholder needs, desirements, and expectations. Results demonstrate improved stakeholder traceability, conflict resolution, and alignment between mission objectives, system requirements, and establishes a robust foundation for requirements engineering.

#4 Requirement Analysis: It formalizes stakeholder needs into structured, verifiable system requirements. Its spine (Requirement Intent → Requirement Definition → Requirement Class → Requirement Type Requirement

Structure Requirement Organization → Requirement Subset, Requirement Subtype → Requirement Meta-View Semantic Ontology → Requirement Formalization and Computability → Requirement Integrity, Validity, Maintenance, and Governance → Requirement Transition, Interoperability, and Reusability) progressively transforms stakeholder expectations into measurable engineering requirements. This package establishes the authoritative requirements baseline and enables lifecycle traceability. Results indicate improved requirement quality, consistency, verification readiness, and allocation accuracy, reducing ambiguity and supporting architecture development throughout the framework.

#5 System Analysis: It evaluates potential solution before architectural commitment. Its spine (System Exploration Context → Potential Solution Architecture Design Space → Candidate Architectures → System Analysis Architecture Modeling → System Analysis Evaluation Criteria → System Analysis Live Analytical Model/Real-Time Analytical Model → System Analysis Trade Study → System of Interest (SoI) Decision-Making → System of Interest (SoI) Definition) supports systematic assessment of feasible concepts. This package balances performance, cost, risk, operational effectiveness, etc. Results include enhanced decision quality, improved architectural justification, and transparent rationale for selecting solution(s) that best satisfy stakeholder, mission objectives, and requirement.

#6 Operational Analysis: It defines how the SoI performs within its intended operational environment. Its spine (Operational Intent and Capability → Operational Concept, Context, and System Usage → Operational Behavior and Execution → Operational Actors, Agency, and Roles → Operational Interactions, Interfaces, Ports, and Flows → Operational Virtual Physical Entity and Structural Representation → Operational Intelligence and Interoperability) captures mission execution characteristics and external influences. This package validates operational feasibility prior to architecture development. Results show stronger alignment between needs and system capabilities, reducing operational gaps and improving readiness for use case and functional analysis.

#7 Use Case Analysis: It translates operational needs into structured interactions between actors and the SoI. Its spine (Use Case Context and System Boundary → Use Case Identification (System Use Case Model) → Actor Identification (External Entities Interacting with the System) → Use Case Diagram (Actor-System Interaction Modeling) → Use Case Specification → Use Case Scenarios (System Usage Scenario Modeling) → Scenario Refinement and Activity Behavioral Flows → Use Case Relationships and Use Case Decomposition → Use Case Traceability and Intelligence) captures system usage across normal, alternate, and exception conditions. This package bridges operational concepts and functional requirements. Results demonstrate improved understanding of user interactions, enhanced behavioral traceability, and clearer definition of system responsibilities supporting downstream functional and architectural development.

#8 Functional Analysis: It defines what the SoI must do independent of implementation. Its spine (Functional Structure and Functional Behavior → Functional Invocation Trigger Event Input Output → Functional Interaction, Interfaces, Ports, and Flows → Functional Orchestration → Functional

Traceability, Intelligence, and Allocation Readiness) captures system functionality and logical behavior. This package establishes the functional baseline for architecture synthesis and requirement allocation. Results indicate improved completeness of system behavior representation, enhanced traceability, and stronger support for logical architecture development.

#9 Logical Architecture: It organizes system functions into logical structures that describe behavior independent of physical implementation. Its spine (Logical Context → Logical Structure, Hierarchy, and Organization → Logical Interactions, Interfaces, Ports, and Flows → Logical Sequence, Logical Behavior, and Logical States → Logical Traceability, Intelligence, and Allocation Readiness) establishes the conceptual architecture of the system. This package bridges functional analysis and physical realization. Results demonstrate improved architectural consistency, enhanced traceability, and a structured framework for allocating requirements and evaluating solution alternatives.

#10 Requirement Allocation Architecture: It allocates requirements to logical architectural elements and establishes responsibility for requirement satisfaction. Its spine (Requirement Tree Hierarchy Decomposition → Requirement to Functional Allocation → Function to Logical Allocation → Logical to Requirement Allocation → Requirement Coverage Analysis → Requirement Verification Method Definition → Requirement Allocation Matrix: RAM) ensures requirements are fully assigned and traceable. This package strengthens architectural accountability and supports verification planning. Results show improved requirement coverage, reduced allocation gaps, and enhanced end-to-end traceability across the system lifecycle.

#11 Physical Architecture: It transforms logical constructs into realizable physical solutions. Its spine (Physical Context → Physical Constraints Inheritance and Drivers → Physical Decomposition → Physical Interfaces, Ports, Connections, and Flows → Physical Allocation → Physical Architecture Element Candidate Architectures → Physical Architecture Element Candidate Architectures Evaluation → Physical Architecture Selected Elements Baseline for Implementation) defines system structure, integration boundaries, and implementation characteristics. This package establishes the foundation for realization and manufacturing. Results demonstrate improved design feasibility, integration readiness, and architectural traceability from stakeholder needs through physical system realization.

#12 Detailed Design (Virtual Twin): It develops implementation-ready virtual representations of the SoI through comprehensive digital engineering models. Its spine (Detailed Discipline Component Design → Detailed Interfaces, Ports, and Connectors Design Specifications → Detailed Behavior Models → Detailed Engineering Constraint Models → Detailed Parametric Performance Models → Modeling Application Simulation Analysis (MASA) Models → Virtual Integration Environment → Virtual Twin Model → Design Verification Simulation → Authoritative Design Baseline Release) creates a high-fidelity digital representation supporting analysis and verification. Results indicate improved design maturity, reduced development risk, earlier issue identification, and enhanced readiness for implementation within the E2E MBSE Framework.

#13 Implementation: It realizes approved physical designs through acquisition, fabrication, assembly, and deployment preparation. Its spine (Implementation Intent and Decision Definition: I2D2 → Implementation SCOUT: Sourcing, Costing, Ordering, Utility, Transportation → System Element Realization (Build, Assembly & Manufacturing – BAM) → Implementation Configuration and Traceability Control → Implementation Verification → → Implementation Support and Delivery Artifacts) translates design into tangible system elements. Results demonstrate improved implementation traceability, optimized sourcing decisions, reduced realization risk, and stronger alignment between engineering intent and delivered system components.

#14 Component/Unit Integration and Testing: It verifies individual components satisfy allocated requirement(s) pre, higher-level integration. Its spine (Component or Unit Integration Testing Definition, Strategy, Enablement, and Mechanism → Component Assembly Integration Execution → Component Interface Verification → Component Integration Test Execution → Component Integration Defect Detection, Identification, and Resolution → Component Integration Evaluation, Qualification, Regression, and Stability Validation → Component Integration Baseline and Subsystem Readiness) ensures each element performs as intended. This package establishes confidence in building-block performance. Results show early defect identification, reduced downstream integration issues, and improved verification traceability throughout the realization process.

#15 Subsystem Integration, Verification, and Testing: It integrates verified components into functional subsystems and evaluates their collective performance. Its spine (Subsystem Integration, Verification, and Testing Strategy → Subsystem Assembly Integration Execution → Subsystem Interface Verification → Subsystem Functional Verification → Subsystem Integration Testing → Subsystem Verification Evidence and Traceability → Subsystem Validation Readiness Assessment) focuses on validating subsystem behavior and interfaces. Results demonstrate improved subsystem maturity, early identification of interface incompatibilities, and increased confidence prior to full system integration activities.

#16 System Integration, Verification, and Testing: It assembles subsystems into the complete SoI and evaluates overall system performance. Its spine (System Integration, Verification, and Testing Strategy → System Assembly Integration Execution → System Verification → System Validation Readiness Transition) ensures all elements function cohesively. Results indicate improved system interoperability, enhanced verification coverage, and confirmation that integrated system capabilities satisfy allocated requirements and architectural intent.

#17 System Validation and Operational Testing: It confirms that the integrated system fulfills stakeholder needs and mission objectives within realistic operational environments. Its spine (System Validation, Operations, Test, Planning, and Strategy → System Validation ABCDE (Actors, Boundary, Context, Definition, Environment) → Operational Architecture and Use Case Validation → System Validation Execution → System of Systems Validation Execution and Emergent Observations → System Operational Testing and

Evaluation (OT&E) → System Validation Results, Analysis, and Evidence → System Validation Non-Conformance, Risk, Feedback, and Resolution → System Digital Twin and Qualification Acceptance Compliance Readiness) focuses on operational effectiveness and suitability. Results demonstrate stakeholder satisfaction, mission accomplishment capability, and validation of intended operational outcomes.

#18 Digital Twin: It establishes a dynamic digital representation synchronized with operational SoI. Its spine (Digital Twin Strategy, Architecture, Framework, Environment (DT SAFE) → Digital Twin Types, Structure, and Model Development → DT Bi-direction Relational Analytics Intelligence Network (DT BRAIN) → Digital Twin Verification, Validation, and Uncertainty Quantification (VVUQ) → Digital Twin Achieved) enables predictive insight and lifecycle decision-making support. Results demonstrate enhanced situational awareness, performance optimization, and continuous digital-thread-enabled system learning. It also integrates predictive capabilities for users.

#19 Qualification, Acceptance, and Compliance Testing: It demonstrates that the system satisfies regulatory, contractual, safety, and certification requirements. Its spine (Qualification Strategy Structure Planning and Definition → Qualification Testing Execution → Acceptance Testing (Factory and Operational) → Compliance Verification and Certification → Test Evidence Traceability Report Audit (TREAT) → Non-Conformance Waiver and Risk Disposition → Deployment Readiness Evaluation Assessment Decision (DREAD)) establishes formal readiness for operational deployment. Results indicate increased stakeholder confidence, verified compliance, reduced certification risk, and documented evidence supporting system acceptance.

#20 Deployment, Operations, and Sustainment Testing: It evaluates system performance during operational deployment and sustainment activities. Its spine (Deployment Testing → Operational Testing → Sustainment Testing → Continuous Operational Evaluation → Deployment Operation Sustainment (DOS) Assessment) ensures continued effectiveness in real-world environments. Results demonstrate sustained mission capability, operational resilience, and continuous system performance assessment, supportability, and service life security.

#21 Post-Deployment Verification and Validation: It assesses system performance after deployment to ensure continued mission effectiveness and compliance. Its spine (Post-Deployment Verification → Post-Deployment Validation → Degradation and Emergent Behavior → Risk and Safety Revalidation → Independent Verification and Validation (IV&V) → Change Identification Feedback Upgrade and Modification Readiness) supports lifecycle learning and improvement. Results reveal emerging operational insights, stakeholder confidence maintenance, and enable updates to engineering baselines.

#22 Production, Scalability, and Modification: It manages system growth, replication, modernization, and adaptation throughout its operational lifecycle. Its spine (Production Planning → Scalability Assessment → Modification Definition → Change Implementation → Configuration

Re-Baselining) supports evolving mission and enterprise needs. Results demonstrate improved adaptability, controlled configuration evolution, enhanced production efficiency, and preservation of lifecycle traceability despite changing operational requirements.

#23 In-Service Maintenance and Operation: It sustains operational availability, reliability, and mission effectiveness throughout service life. Its spine (In-Service Operational Mission Execution → Preventative and Scheduled Maintenance → Corrective Maintenance and Repair → Condition-Based and Predictive Maintenance and Performance Optimization → Condition-Based and Predictive Maintenance and Performance Optimization → Sustainment Support → Operational Risk Safety Anomaly Management → End-of-Service to End-of-Life (EoS to EoL) Transition Signals Verification) enables proactive asset management. Results demonstrate improved availability, reduced lifecycle cost, enhanced safety, and optimized long-term operational performance.

#24 Extinction: It manages the orderly retirement, disposal, recovery, and knowledge preservation of the SoI. Its spine (Extinction and Disposal (END) Strategy Identification and Definition (STRIDE) → Operational Decommissioning (OpsDecoms) → END Asset Recovery and Reclamation → Knowledge Management and Retirement Data Archival → Extinction and Disposal System Assessment and Final Elimination (END SAFE) → Extinction and Disposal Verification and Validation (END V&V) → Extinction and Disposal Type (END Type)) ensures responsible end-of-life execution. Results demonstrate preservation of enterprise knowledge, regulatory compliance, environmental stewardship, and completion of full lifecycle traceability within the End-to-End MBSE Framework.

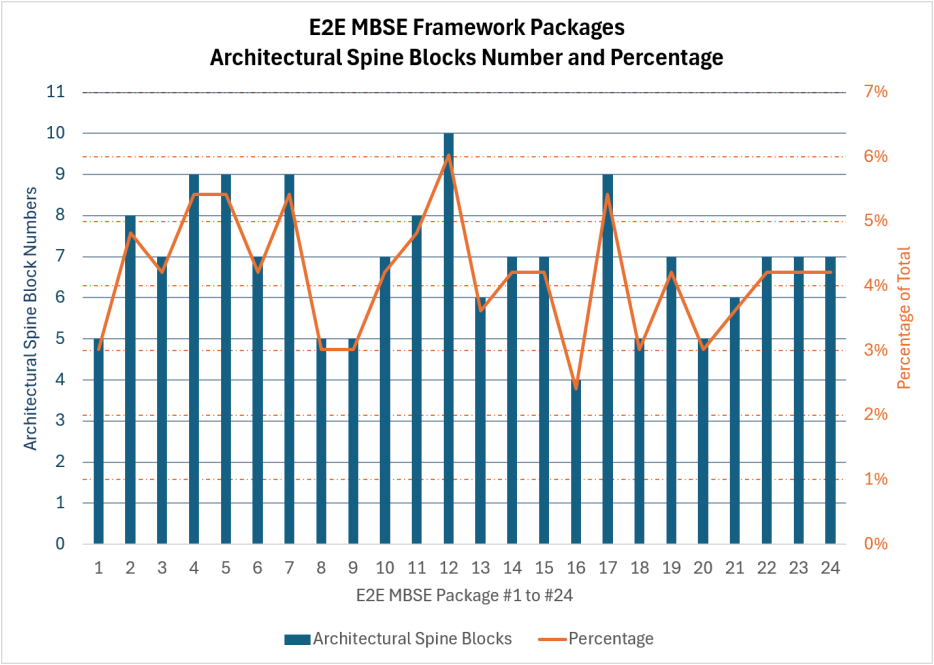


Figure 1: E2E MBSE framework's 24 packages architectural spine blocks.

CONCLUSION AND FUTURE WORK

This research addresses a fundamental challenge in MBSE: the absence of a comprehensive lifecycle framework capable of maintaining architectural continuity, traceability, and DE integration from system evolution to extinction. In response, this work presents the End-to-End MBSE Framework, a lifecycle-integrated architecture that spans the complete realization of a SoI from evolution to extinction.

The research makes three essential contributions to the SE and DE body of knowledge. First, it completes the development of a lifecycle-integrated framework consisting of twenty-four interconnected packages that collectively bridge longstanding coverage gaps across both the left and right sides of the SE Vee Model. Unlike many existing approaches that focus primarily on requirements, architecture, and design activities, the E2E MBSE Framework explicitly incorporates implementation, integration, verification, validation, qualification, deployment, sustainment, modification, production scalability, maintenance, and retirement within a single coherent lifecycle structure.

Second, the framework establishes end-to-end lifecycle traceability through 166 interconnected modeling constructs. This traceability architecture enables continuity across stakeholder needs, requirements, architectural decisions, implementation artifacts, verification evidence, validation outcomes, operational performance, and extinction decisions, thereby preserving digital continuity throughout the system lifecycle.

Third, the framework formalizes the integration of digital thread, virtual twin, and digital twin concepts within a unified lifecycle architecture. As a domain-agnostic approach, the framework provides a foundation for supporting complex systems and systems-of-systems.

Collectively, these contributions complete the original research objective and establish a lifecycle-complete MBSE framework that extends systems thinking beyond development and into operations, sustainment, evolution, and extinction. The significance of this work lies in its potential to transform fragmented lifecycle activities into a continuously connected engineering ecosystem capable of supporting informed decision-making, traceability preservation, and digital continuity across the entire system lifecycle.

Future work will focus on dissertation-level validation through the application of all twenty-four packages to a representative SoI case study. Planned activities include SysML implementation, digital thread realization, scalability assessment, and evaluation of framework completeness, practical utility, and applicability across diverse engineering domains.

REFERENCES

Adelabu, J., & Lohar, B. (2025a). End-to-End (E2E) Model-Based Systems Engineering (MBSE) Framework. *Shelby Hall Graduate Research Forum Presentations*, *1*(1).

Adelabu, J., & Lohar, B. (2025b, March 27). End-to-End (E2E) MBSE Framework. *Shelby Hall Graduate Research Forum Posters*.

Adelabu, J., & Lohar, B. (2025c, September 15). End-to-End (E2E) Model-Based Systems Engineering (MBSE) Framework for Complex Systems Life Cycle Design. *Advanced Systems Research Laboratory (ASRL) Day (1st Founding Anniversary)*.

Adelabu, J., Lohar, B., T Wade, J., Walker, S., & Montalvo, C. (2025). Building Blocks for Effective Modeling and Simulation of Systems Lifecycle using an End-to-End Model-Based Systems Engineering Framework. *Human Factors in Design, Engineering, and Computing. AHFE (2025) International Conference. AHFE Open Access, Vol 199. Applied Human Factors and Ergonomics (AHFE) International, USA.*, 2406–2416. https://doi.org/10.54941/ahfe1007053

Adelabu, J., Lohar, B., Wade, J., Walker, S., & Montalvo, C. (2026). Analysis for an End-to-End MBSE Operational Architecture. *Human Factors in Simulation, Software and Systems Engineering. AHFE (2026) International Conference. AHFE Open Access, Vol 220. AHFE International, USA.*, 27–37. https://doi.org/10.54941/ahfe1007691

Bhushan, L. (2022). *Development of New Space Systems Architecture in SYSML Using Model-Based Pattern Language* [University of South Alabama (USA)]. https://jagworks.southalabama.edu/theses_diss/75

Cederbladh, J., Cicchetti, A., & Suryadevara, J. (2024). Early Validation and Verification of System Behaviour in Model-based Systems Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology*, *33*(3), 1–67. https://doi.org/10.1145/3631976

Dunbar, D., Hagedorn, T., Blackburn, M., Dzielski, J., Hespelt, S., Kruse, B., Verma, D., & Yu, Z. (2023). Driving digital engineering integration and interoperability through semantic integration of models with ontologies. *Systems Engineering*, *26*(4), 365–378. https://doi.org/10.1002/sys.21662

Körtvélyi, N., Schichl, T., & Jungwirth, M. (2026). Formalizing the Digital Thread: A Standards-Based MBSE Integration Framework. *Procedia Computer Science*, *277*, 545–553. https://doi.org/10.1016/j.procs.2026.02.096

Madni, A. M., Purohit, S., & Madni, C. C. (2023). Exploiting Digital Twins in MBSE to Enhance System Modeling and Life Cycle Coverage. In *Handbook of Model-Based Systems Engineering* (pp. 527–548). Springer International Publishing. https://doi.org/10.1007/978-3-030-93582-5_33

Pastor, O., Segooa, M. A., & Panach, J. I. (2024). Teaching Design Science as a Method for Effective Research Development. In *Handbook on Teaching Empirical Software Engineering* (pp. 479–499). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-71769-7_17

Wu, S., Wang, G., Lu, J., Hu, Z., Yan, Y., & Kiritsis, D. (2024). Design ontology for cognitive thread supporting traceability management in model-based systems engineering. *Journal of Industrial Information Integration*, *40*, 100619. https://doi.org/10.1016/j.jii.2024.100619