2026 NDIA MICHIGAN CHAPTER
GROUND VEHICLE SYSTEMS ENGINEERING
AND TECHNOLOGY SYMPOSIUM
MODELING, SIMULATION, PROTOTYPING, AND VALIDATION (MSPV) TECHNICAL SESSION
AUG. 11-13, 2026 - NOVI, MICHIGAN

INTEGRATING MBSE WITH HIGH-FIDELITY CRASH SIMULATION

Patrick J. Rye[1]

[1]General Motors / Colorado State University

ABSTRACT

Modern electrified ground vehicles introduce complex, multi-domain safety requirements, such as post-crash thermal runaway prevention, that expose the traceability limitations of Document-Based Systems Engineering (DBSE). This paper proposes a four-layer, bidirectional digital thread architecture that integrates Model-Based Systems Engineering (MBSE) with high-fidelity, non-linear Computer-Aided Engineering (CAE) crash simulations. Leveraging SysML, System-Theoretic Process Analysis (STPA), and Python-based orchestration middleware, the framework automates the translation of descriptive safety requirements into explicit finite element boundary conditions. The architecture programmatically extracts key performance indicators from massive binary solver outputs and injects them back into the SysML environment for automated compliance verification. Demonstrated through a simplified electric vehicle side-pole impact case study utilizing LS-DYNA and a 1D thermal model, the framework successfully eliminates manual data handoffs, accelerates multidisciplinary design optimization, and ensures robust, risk-driven requirement traceability across the engineering lifecycle.

Citation: P.J. Rye, "Integrating MBSE with High-Fidelity Crash Simulation," In *Proceedings of the Ground Vehicle Systems Engineering and Technology Symposium* (GVSETS), NDIA, Novi, MI, Aug. 11-13, 2026.

1. INTRODUCTION

Modern ground vehicle systems engineering is increasingly challenged by the intricate interdependencies introduced by electrified powertrains. Crashworthiness engineering has been heavily dominated by primary occupant safety metrics (e.g., dummy kinematics, structural intrusion into the cabin) due to the ease of tracing requirements directly between regulations and direct hazard events. However, in Electric Vehicles (EVs) there are secondary requirements that are equally critical but significantly more difficult to trace and verify. Chief among these is the prevention of thermal runaway in the Rechargeable Energy Storage System (RESS) battery following a collision.

Unlike purely structural occupant safety criteria, assessing thermal runaway risks requires mapping logical safety constraints to multi-domain physical behaviors (structural deformation, fluid leakage, and thermal propagation) [1], [2]. Tracing a system-level requirement such as *"The vehicle shall not exhibit thermal runaway post-crash"*, down to specific targets on coolant pipes is difficult without a robust framework to connect the system-level requirement to component-level requirements.

The necessity for such a framework becomes apparent when evaluating how global regulatory standards dictate post-crash safety thresholds, such as strict limits on electrolyte spillage or mandated electrical isolation [3], [4]. Verifying these regulations necessitates defining

secondary, surrogate requirements because a single structural Finite Element Analysis (FEA) crash model cannot directly calculate multi-domain phenomena. For example since mechanical solvers do not natively simulate fluid dynamics or chemical thermodynamics, regulations preventing electrolyte leakage or delayed thermal propagation [5] must be assessed through structural proxies like seal flange separation limits, minor cooling line deformations that could restrict flow, or volumetric cell intrusion thresholds.

Verifying these secondary requirements is highly complex and can become difficult in traditional Document-Based Systems Engineering (DBSE) [6]. In DBSE, engineering data is captured and managed across static, disconnected files, such as text documents, spreadsheets, and isolated departmental databases [7]. As system complexity increases, this methodology inherently fractures information into domain-specific data silos, causing structural, thermal, and high-voltage electrical teams to maintain isolated versions of the truth that falls out of sync as the vehicle design evolves. This fragmented approach severely complicates requirement traceability throughout system development [8]. When a specification is manually transferred from a simulation tool into a text document, it becomes statically disconnected from the underlying engineering physics and the initial design rationale. Tracing a system-level safety requirement down to a specific subsystem relies on manually updating Requirements Traceability Matrices (RTMs) which is a labor-intensive and error-prone process [9]. Consequently, the change tax on impact analysis is exceptionally high; if an engineer alters a localized parameter, assessing the systemic impact requires manually tracking down and updating every downstream document and risk assessment across different departments [9]. Because DBSE relies on static documents as the primary medium of information exchange, the complex relationships between logical safety constraints and physical implementations becomes obscured [10]. This lack of automated, bidirectional traceability drastically increases the risk of orphaned requirements, where a localized design change unintentionally violates a high-level safety constraint simply because the connection was buried in a different department's siloed documentation [10].

The need for robust requirement traceability is equally critical to U.S. Army modernization strategies. As the military transitions toward Tactical Hybrid Electric Vehicles (THEVs), ensuring battery safety under extreme dynamic loading is paramount. These combat survivability assessments rely on the exact same non-linear explicit CAE solvers used in commercial crashworthiness. Concurrently, the Department of Defense (DoD) has mandated a shift toward Digital Engineering to eliminate the exact DBSE data silos described above. The proposed MBSE-CAE framework directly supports the missions of the U.S. Army Ground Vehicle Systems Center (GVSC) and USMC acquisition programs. By automating the digital thread between STPA risk models, survivability requirements, and high-fidelity physical simulations, this architecture provides a scalable foundation to safely integrate high-voltage systems into combat vehicles while fulfilling DoD digital engineering directives.

To overcome the limitations of document-centric approaches, Systems Engineering has developed the approach of Model-Based Systems Engineering (MBSE), which is increasingly being adopted in complex engineering systems [11]. MBSE transitions the primary artifact of engineering from static text files to an integrated, dynamic system model where logical requirements, functional architectures, and physical components are explicitly linked [12]. This centralized approach fundamentally improves requirement traceability. Requirements traceability is defined as "the ability to describe and follow the life of a requirement in both a forwards and backwards direction" [13]. If an engineer modifies a physical parameter, the MBSE environment can automatically highlight the systemic impact

on all upstream requirements and downstream verification tests across every engineering domain.

A critical "fidelity gap" persists between the descriptive world of MBSE models and the analytic world of high-fidelity Computer Aided Engineering (CAE) [14], [15]. MBSE models are typically topological and logical; they define what a component is, but they do not inherently understand the continuum mechanics required to predict how that component deforms under impact. Conversely, FEA models provide exquisite physical fidelity but are often "black boxes" to the systems engineer, making them disconnected from the requirements they are meant to verify [14], [16].

Bridging this fidelity gap through the integration of MBSE with crash safety FEA models offers two key benefits to ground vehicles: easier requirement traceability, and connecting the latest system risk science with crash safety development. By establishing a bidirectional computational link, system-level requirements and logical architectures can directly drive the boundary conditions, material properties, and geometric parameters of the FEA environment. This integration allows crashworthiness engineers to automatically trace the structural outputs of a simulated vehicle impact directly back to the governing safety constraints within the system model. Orchestrating high-fidelity simulations through an MBSE framework enables automated, multidisciplinary design optimization. Structural modifications can be rapidly evaluated not just for mechanical integrity, but for their systemic impact on coupled thermal and electrical behaviors within an EV. Integrating these domains demystifies the "black box" of CAE for systems engineers, establishing a closed-loop verification process that accelerates development cycles, enhances design confidence, and drastically reduces the risk of late-stage physical prototyping failures. Integrating MBSE also allows for the integration of cutting edge risk science approaches to be used in crash safety, such as System-Theoretic Process Analysis (STPA).

This paper is driven by the following research question: ***What frameworks or architectures can effectively improve risk-based requirement traceability between MBSE and CAE ground vehicle crash analysis?***

To answer this, the paper: (1) reviews existing MBSE-CAE integration literature and identifies remaining gaps specific to non-linear 3D FEA; (2) derives integration requirements from those gaps; (3) proposes a four-layer reference architecture; and (4) demonstrates the architecture's orchestration capabilities using a simplified subsystem case study. The case study models are intentionally lightweight stand-ins chosen to exercise each layer of the pipeline across two fidelity levels; the contribution being demonstrated is the architecture and orchestration framework, not the physical accuracy of the demonstration simulations.

2. LITERATURE REVIEW

2.1. Model-Based Systems Engineering

The International Council on Systems Engineering (INCOSE) defines MBSE as the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities from the conceptual design phase throughout the system's life cycle [17]. MBSE represents a paradigm shift away from DBSE toward a model-centric approach, where an integrated digital model serves as the primary engineering artifact [18].

This transition offers several documented advantages. By utilizing a centralized system model, MBSE establishes a common technical baseline that improves communication across multidisciplinary teams and reduces the ambiguities inherent in natural-language documents [17], [18]. Explicitly defined dependencies within the model ensure that a design change in one area automatically propagates to all impacted subsystems, maintaining consistency and traceability [19]. One of the most significant financial justifications for MBSE is its ability to facilitate early defect prevention: finding and fixing defects during

conceptual and preliminary design phases prevents costly rework during later integration and testing phases [6], [19]. Additionally, the model-based environment supports the storage and reuse of system components across product lines, reducing engineering effort for derivative programs [19].

Despite these advantages, MBSE adoption in domains dominated by high-fidelity physics simulation remains uneven [15]. A persistent barrier is the lack of mature, bidirectional integration between descriptive MBSE models and analytical CAE tools [14].

2.2. System-Theoretic Process Analysis (STPA)

STPA is a top-down, proactive hazard analysis methodology rooted in the System-Theoretic Accident Model and Processes (STAMP) framework [20]. Unlike traditional reliability techniques such as Failure Mode and Effects Analysis (FMEA) or Fault Tree Analysis (FTA), which models safety as a component failure problem, STPA frames system safety as a dynamic control problem. Accidents in modern complex systems frequently arise not from individual component failures but from inadequate control actions, flawed software logic, and dysfunctional component interactions [20], [21].

The STPA methodology proceeds through four iterative steps: (1) defining the purpose of the analysis by identifying losses and system-level hazards; (2) modeling the system's hierarchical control structure; (3) identifying Unsafe Control Actions (UCAs) for each controller in the structure; and (4) determining the causal scenarios through which those UCAs could occur [20].

A primary advantage of STPA for cyber-physical systems such as the Battery Thermal Management System (BTMS) in an EV is its ability to surface hazards that emerge from software logic errors and system interactions, not just hardware wear [20], [21]. Because STPA does not require a finalized physical architecture to begin, it integrates well into MBSE workflows through the standard language of "Risk Analysis and Assessment Modeling Language" (RAAML) [22], enabling safety constraints to actively guide early design decisions in a "safety-by-construction" approach [23]. This alignment with ISO 26262 functional safety processes has been demonstrated in automotive contexts [24], [25].

2.3. The Digital Thread in MBSE

The "digital thread" is a data-driven architectural concept that integrates information generated across a product's entire lifecycle, including design, manufacturing, and supply chain management, into a single authoritative communication platform [26]. Originating from aerospace and defense initiatives aimed at eliminating data silos, it ensures the consistency, traceability, and real-time accessibility of enterprise data, models, and design decisions [27], [28], [29].

Within MBSE, the digital thread serves as the connective tissue that bridges the gap between abstract logical and functional system architectures, often formalized in SysML, and their downstream physical implementations [30]. By replacing static, disconnected document-centric processes with an integrated digital ecosystem, the digital thread enables bidirectional data flow, continuous requirement validation, and comprehensive lifecycle management [29], [31].

In practical application, digital threads orchestrate MBSE integration with multidisciplinary physical and behavioral simulations to achieve robust early-stage performance validation [31], [32]. Standardized interoperability protocols, such as the Functional Mock-up Interface (FMI) and Open Services for Lifecycle Collaboration (OSLC), facilitate the export and orchestration of behavioral models as Functional Mock-up Units (FMUs) for co-simulation [31], [33]. This closed-loop framework enables "consistency management," where structural definitions and physical parameters are dynamically synchronized with physics-based simulations and PLM backbones

[27], [32].

2.4. MBSE-CAE Integration Frameworks

The integration of descriptive systems models and analytical behavioral models has been a persistent challenge across aerospace and automotive domains.

D'Ambrosio and Soremekun [34] demonstrated that integrating SysML models directly with domain-specific analysis tools allow requirement changes to automatically propagate into engineering analyses, which supports the generation of traceable, model-based safety cases. Sohier et al. [35] developed frameworks to translate MBSE system architectures into explicit "simulation requests," capturing: (1) the subset of the system to simulate, (2) the simulation objective, (3) requirements on quality, cost, and delivery, (4) test scenarios, (5) calibration and validation data, and (6) the V&V strategy. Their approach, realized as a Papyrus plugin, formalizes *what* must be verified in a structured, traceable manner. Graignic et al. [36] developed a framework that uses system-level logical definitions to aggregate appropriate FE models and automatically export an integrated simulation model to a pre-processor for computation.

While these methodologies successfully establish a digital thread defining the scope and assembly of a simulation, a significant bottleneck emerges when attempting to execute automated requirements verification. Early loop-closing attempts rely heavily on linear parametric solvers. Kaloor and Barosan [37] utilize SysML and IBM Rational Rhapsody's Parametric Constraint Evaluator (PCE) to automatically evaluate hardware configurations against system attributes. Ryan et al. [38] integrate SysML parametric diagrams with MATLAB and Excel to automate architecture trade studies. However, both frameworks are validated using computationally lightweight, 1D analytical models. They are confined to linear mathematical relationships, making them infeasible for the non-linear parameter spaces of explicit crash simulation [37], [38].

Recent work has improved interoperability through API-driven approaches. Castellano [39] proposes a tool-agnostic framework leveraging the SysMLv2 API and Python wrapper functions to map descriptive attributes directly to simulation inputs and outputs, enabling automated compliance checking. Sarathi and Martinez [40] utilize XML Metadata Interchange (XMI) to translate SysML instances into Simulink variants, injecting discrete pass/fail statuses back into the requirements model. While these frameworks demonstrate closed-loop verification, they remain constrained to computationally lightweight behavioral models and have not been extended to the massively parallel, non-linear 3D FE solvers required for crashworthiness analysis [39], [40]. Castellano [39] explicitly notes severe data processing bottlenecks when scaling; Sarathi and Martinez [40] acknowledge that their boolean pass/fail feedback is a forced workaround due to the inability to bidirectionally synchronize complex data structures.

A comprehensive review by Nigischer et al. [41] encapsulates the fundamental limitation of the current MBSE interoperability ecosystem while parameter transfer *to* simulations is maturing, the automated feedback of complex simulation results *into* SysML remains a significant unsolved problem. The primary coupling standards FMI and SysPhS, are explicitly tailored for 1D lumped-parameter co-simulation and discrete-event models. While file-based FMU model exchange approaches exist, they are not designed to interface with the dense binary outputs (`binout`, `d3plot`) characteristic of non-linear 3D FEA and wrapping legacy crash models as FMUs introduces significant re-engineering overhead [41].

2.5. Commercial Toolchain Context

Several commercial ecosystems address subsets of the MBSE-CAE integration problem. Dassault Systèmes' 3DEXPERIENCE platform natively couples CATIA Magic (MBSE) with the SIMULIA/Abaqus suite, enabling direct system model-to-geometry simulation without fragile

file exports. Siemens Teamcenter orchestrates MBSE capabilities through a PLM backbone, linking to the Simcenter portfolio via Simcenter HEEDS for MDAO. Ansys ModelCenter functions as vendor-agnostic orchestration middleware connecting SysML models to Ansys Mechanical, Fluent, and HFSS. While these platforms represent the industrial state of the art, they impose constraints that motivate the proposed open-architecture approach. Enterprise licensing costs are prohibitive for academic and small-scale settings, and vertical integration penalizes the use of external solvers, substituting LS-DYNA for Abaqus, for instance, requires custom middleware that offsets the benefits of native integration. Proprietary orchestration logic also introduces auditability concerns for safety-critical simulation governance, as the data transformation from system model to solver boundary condition is obscured from the engineering team.

2.6. Limitations and Remaining Gaps

Synthesizing the literature, four specific gaps motivate the proposed architecture:

1. **Manual Parameter Instantiation and Scalability:** Automating parameter synchronization between SysML and 3D FE assemblies remains largely manual. Existing prototypes rely on static parameter mappings that require expert intervention whenever geometric topology changes.

2. **Static Data Linkages:** SysML's inherently static data model lacks the dynamic linking required for real-time requirement verification against evolving simulation outputs. This results in disjointed toolchains where crash analysts operate entirely outside the MBSE ecosystem.

3. **Semantic Disconnect in Risk Feedback:** While RAAML can trace hazards to structural components, passing LS-DYNA results back into the risk model to automatically update a compliance state or trigger a redesign iteration remains highly manual.

4. **Data Management Overhead:** High-fidelity crash simulations produce datasets measuring hundreds of gigabytes per run. Traditional MBSE tools are not designed to manage these result files, creating a gap between the descriptive system model and the Simulation Data Management (SDM) environments that host raw data.

3. MBSE-CAE INTEGRATION FRAMEWORK REQUIREMENTS

The synthesis of the literature and the current state of commercial toolsets reveals a critical need for a more robust, open integration strategy capable of addressing the non-linear, binary-output domain of crashworthiness simulation. The proposed reference architecture must satisfy the following five requirements.

Requirement 1 — Standardized Safety and Risk Ontologies: The framework must utilize standardized libraries (RAAML, ISO 26262 stereotypes) to establish explicit, auditable trace links, `mitigates`, `causedBy`, `verifies`, between crash hazards, safety requirements, and the physical vehicle architecture.

Requirement 2 — Formalized Simulation Context via Simulation Blocks: Before executing a CAE model, the Systems Engineer must generate a formalized ⟨⟨SimulationRequest⟩⟩ specifying the exact boundary conditions, target variables, execution metadata (solver version, HPC allocation, contact algorithms, mass scaling limits), and material card references [35]. Without capturing this metadata in the descriptive layer, a simulation result cannot be reliably reproduced and its safety verification is compromised.

Requirement 3 — API-Driven Data Interoperability: The framework must replace static document exports with dynamic parameter mapping via OSLC standards or an SysML API, allowing 0D/1D system properties to be directly pushed into FE input deck variables.

Requirement 4 — SDM Integration for Data Management: Because MBSE tools cannot manage terabytes of crash result data, the framework must integrate an SDM system as the simulation data host. The MBSE environment maintains a lightweight semantic link to the SDM record rather than hosting the binary files directly, preserving the single-source-of-truth principle without overwhelming the system model database.

Requirement 5 — Automated V&V Feedback Loop: The framework must include a programmatic feedback loop that extracts KPIs from FE output files and pushes those values back to the MBSE environment, which automatically flags the associated safety requirement as Pass or Fail and propagates the failure state to upstream risk analyses.

4. PROPOSED REFERENCE ARCHITECTURE

Based on the requirements defined above, a four-layer hub-and-spoke architecture is proposed. Figure 1 shows a high-level overview. This framework is best understood as an open, expandable foundation: PLM governance layers such as access control, IP protection, and multi-supplier data management can be integrated on top of this foundation as organizational maturity and budget allow.

4.1. The Descriptive Layer

This layer acts as the authoritative repository for the vehicle's functional and physical architecture, regulatory compliance, and risk profiles. This layer addresses Requirements 1 and 2.

- **Core Standard:** SysML, extended with RAAML stereotypes for risk and safety ontologies.
- **Ontology & Risk Management:** RAAML structures compliance by establishing explicit, traceable links (`mitigates`, `causedBy`, `verifies`) between defined crash hazards, specific safety requirements, and the physical vehicle architecture.

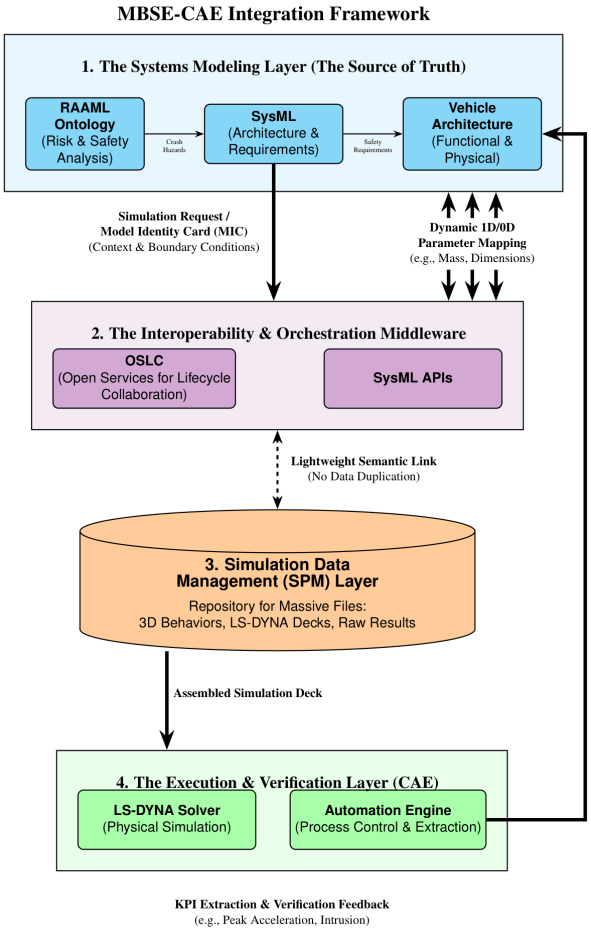

Figure 1: Architecture of the bidirectional hub-and-spoke integration methodology

- **Simulation Context Capture:** The standard ⟨⟨Classifier⟩⟩ is extended into a structured ⟨⟨SimulationRequest⟩⟩ block, following the methodology of Sohier et al. [35]. Core attributes include: *(1) System Scope and Level of Detail*, defining the precise subset of the system architecture to simulate and the target fidelity level (L0–L3); *(2) Simulation Objective and Quality Requirements*, capturing target metrics and the maximum acceptable Type II error probability (β) for invalid results; *(3) Test Scenarios and V&V Directives*, linking to specific environmental instantiations and high-level verification methods; *(4) Execution Metadata*, specifying exact solver version (e.g., LS-DYNA R13.1 Double Precision), HPC node and core allocation, termination time, mass scaling threshold,

and contact algorithm specification; *(5) Material Card References*, stored as version-controlled PLM records rather than hardcoded values; and *(6) Output Extraction Specification*, defining the specific element sets, node IDs, or cross-sectional as a ⟨⟨VirtualSensor⟩⟩ within the descriptive model. This block identifies the specific elements that that the Python binder must interrogate within binary output files to generate the required KPIs.

If a design iteration passes a requirement but exceeds the defined mass scaling threshold, the MBSE framework must flag the result as a numerical artifact rather than a valid physical outcome.

Figure 2 shows the stereotype extensions defined by this framework.

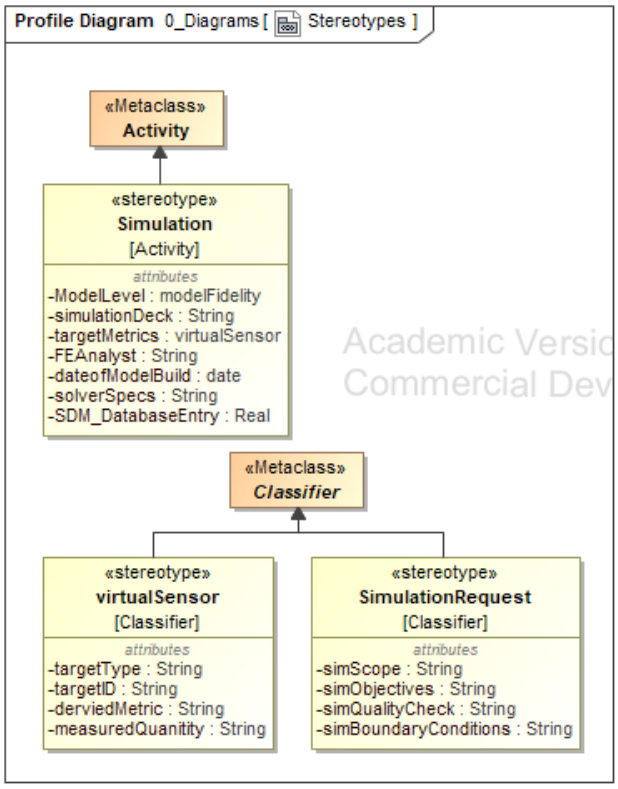

Figure 2: Expanded Stereotypes

4.2. The Interoperability & Orchestration Middleware

This layer bridges the semantic gap, enabling dynamic communication between the system model and the execution environment without duplicating heavy data. This layer fulfills Requirement 3 by replacing manual mapping with API-driven parameter transfer.

- **Core Standards:** OSLC and SysML APIs, REST.
- **Function:** Replaces manual 3D FE parameter mapping and static document exports. Dynamically maps 0D/1D system properties (e.g., pipe wall thickness, impact velocity, material selection) from the MBSE environment directly to variables within the simulation input deck.
- **Advantages over Commercial Middleware:** (1) *Solver-native* — directly interrogates FE binary output files using Python libraries without transcoding results into an intermediate format; (2) *Architecturally transparent* — the data transformation logic is fully auditable and modifiable by the engineering team; (3) *Standards-independent* — imposes no requirements on the CAE solver's external interface beyond standard file output, preserving compatibility with legacy model libraries that cannot be re-wrapped as FMUs without significant effort.

4.3. Simulation and Data Management (SDM) Layer

MBSE databases are not designed to host terabytes of raw crash data. This dedicated infrastructure handles the heavy lifting of the simulation files. This layer satisfies Requirement 4 by integrating a dedicated SDM system to host the raw data.

- **Function:** Acts as the high-volume data bridge, storing LS-DYNA input decks and raw result files natively.
- **Integration Mechanism:** The MBSE environment maintains a lightweight semantic link to the specific SDM record rather than hosting the files directly,

preserving the single source of truth without compromising system model performance. After results are processed, the SDM record retains the input deck and extracted KPIs; raw result files are archived or purged according to a defined data lifecycle policy.

4.4. The Execution & Verification Layer (CAE)

This is where physical simulation occurs and where results are translated back into systems-level intelligence. This implementation solves Requirement 5 by extracting KPIs and automatically pushing pass/fail outcomes back into the MBSE model.

- **Core Solver:** Non-linear explicit FE solvers such as LS-DYNA.
- **Automation Engine:** Programmatic control closes the verification loop, integrating well with multidisciplinary optimization tasks.
- **Function:** Programmatically extracts vital KPIs (e.g., peak plastic strain, intrusion depth, cross-sectional flow area) from FE output files into SysML constraint blocks, where the MBSE tool checks the pass/fail metric against the governing requirement.

4.5. The Digital Thread Workflow

The automated lifecycle of a safety requirement within this architecture proceeds as follows, see Figure 3 on the following page.

1. **Define:** A crash hazard and its mitigating safety requirement are defined in the MBSE tool using RAAML.
2. **Request:** The Systems Engineer uses the ⟨⟨SimulationRequest⟩⟩ block to generate a request to the CAE analyst, defining the exact configuration, boundary conditions, and output extraction specification.
3. **Map:** Via OSLC/SysML APIs, 0D/1D parameters are dynamically pushed to construct the FE input deck.
4. **Execute & Store:** The simulation runs on the HPC cluster. Result files are stored natively in the SDM system.
5. **Extract & Verify:** API wrappers parse through the SDM results, extract target KPIs, and push values back to the MBSE environment.
6. **Flag:** The MBSE tool automatically flags the original safety requirement as Pass" or Fail." If a requirement fails, an automated impact analysis traces the failure back to the specific system components and associated UCAs requiring redesign.

4.6. Hierarchy of Model Fidelity

Modern engineering design is characterized by the ongoing tension between increasing system complexity and finite computational resources [42]. To optimize computational budgets without compromising analytical rigor, model fidelity is formalized into a four-level hierarchy (Table 1 on the next page), based on the L* nomenclature of Zhang et al. [43].

5. CASE STUDY: EV SIDE POLE IMPACT

The following case study is a process demonstration of the proposed integration architecture, not a physically certified crash analysis. The LS-DYNA subsystem model and 1D lumped-parameter thermal model are intentionally simplified stand-ins, selected to exercise each layer of the digital thread pipeline across two fidelity levels while remaining computationally accessible. Specifically, this case study is mean to represent a subsystem verification loop on a segment of the cooling system. The verification loop would verifies the allowable deformation the cooling pipes can endure before risk of fracturing or the constrained pipe would reduce the coolant flow such that the cells began to overheat. A full production deployment would substitute these demonstration models with validated, full-vehicle models; the contribution demonstrated here is the orchestration framework that connects them.

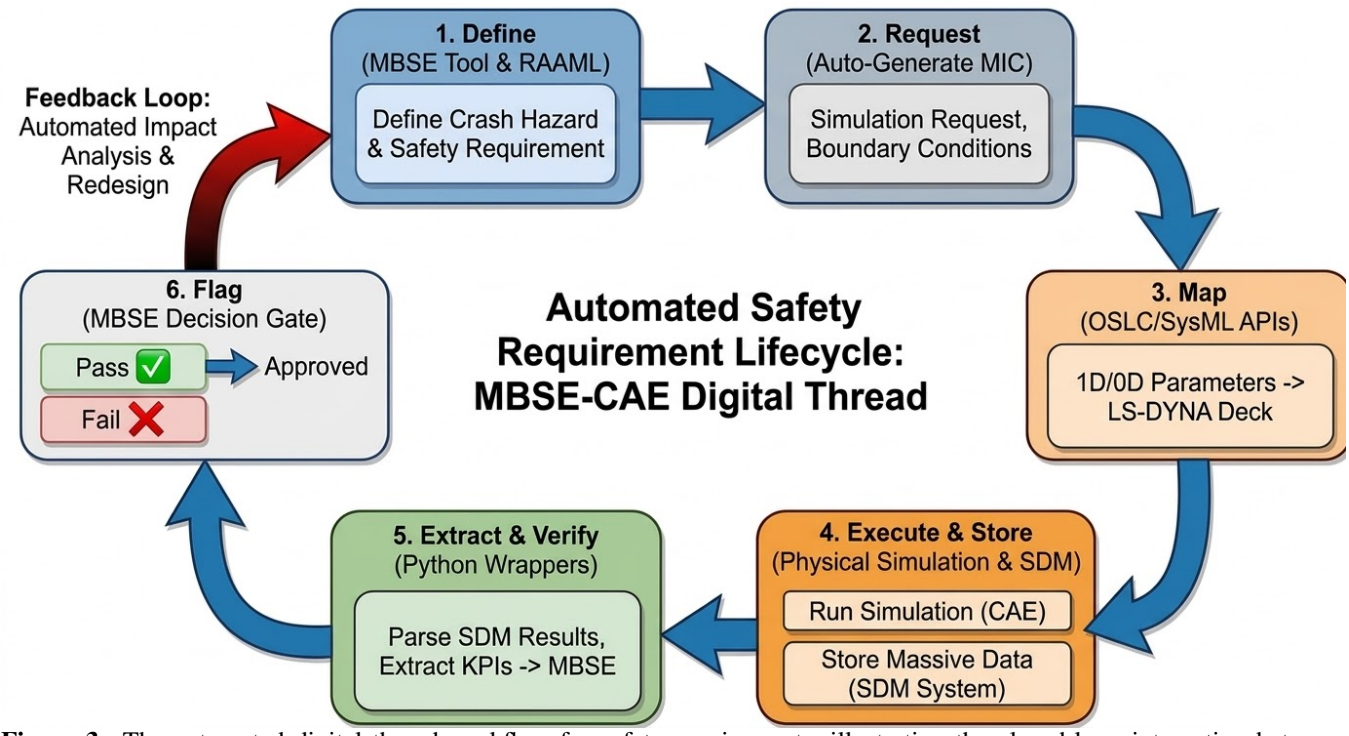

Figure 3: The automated digital thread workflow for safety requirements, illustrating the closed-loop integration between MBSE definition, CAE execution, and automated verification.

Table 1: Model Fidelity Hierarchy

Level	Model Type	Application Phase	Use Case
L0	Analytical	Concept	Energy management strategy
L1	Simplified and Linearized Models	Preliminary Design	Topology optimization, load path definition.
L2	High-Fidelity FEA	Verification	Compliance check (LS-DYNA, Radioss).
L3	Surrogate / ROM	Risk Assessment	Monte Carlo simulation (10k+ runs)

5.1. STPA Safety Analysis of the Battery Thermal Management System

STPA was conducted on the Battery Thermal Management System (BTMS) to identify the hazard scenarios governing the simulation design. The analysis is scoped to the post-crash thermal management control loop; a complete STPA of the full BTMS would additionally address high-voltage isolation, electrolyte containment, and crash-triggered shutdown sequences, which are outside the scope of this case study.

5.1.1. Losses and Hazards

Losses: L-1: Occupant or first-responder injury from thermal runaway propagation. L-2: RESS destruction from uncontrolled thermal propagation. L-3: Vehicle fire from post-crash electrolyte ignition.

System-Level Hazards: H-1: Cell temperature exceeds thermal runaway onset threshold post-crash. H-2: Coolant circulation continues through a mechanically compromised line, masking flow restriction from the ECU. H-3: BTMS fails to detect degraded cooling capacity within the post-crash thermal propagation window. **Hazard link:** H-2 $\rightarrow$ H-1 $\rightarrow$ L-1, L-2, L-3.

5.1.2. Unsafe Control Actions

The BTMS ECU receives coolant temperature and flow rate feedback from the cooling circuit and issues pump speed and valve position commands in response. The critical post-crash vulnerability lies in the feedback path: a partially crushed cooling line reduces internal cross-sectional area and mass flow rate, but if

the line is not fully severed, pump pressure readings may remain within nominal bounds. Therefore the ECU has no observable fault signal distinguishing nominal operation from thermally degraded operation through a restricted line. Table 2 on the following page presents the UCAs for the post-crash pump control subsystem. The analysis is explicitly scoped to pump-level control actions.

The highest-consequence loss pathway is the causal scenario linking UCA-1 and UCA-2: the BTMS commands continued circulation through a restricted line, receives a no fault signal because partial restriction does not exceed sensor detection thresholds, and degraded heat extraction allows cell temperatures to rise toward thermal runaway. This scenario is particularly hazardous because it is silent, and a no fault code is generated during a window when rescue operations may be underway.

5.1.3. Operationalization in the Simulation Framework

This STPA analysis governs two concrete simulation design decisions. First, UCA-1 and UCA-2 motivate the selection of post-crash internal cross-sectional flow area as the primary surrogate metric extracted from the LS-DYNA output: it is the reduction in flow area, undetectable by the ECU, that makes continued pumping dangerous. Second, the causal scenario conditions the 1D thermal model on the post-crash restricted flow area rather than nominal flow, ensuring the verification chain interrogates the specific coupled failure scenario STPA identifies as highest consequence.

Within MSOSA, the ⟨⟨UnsafeControlAction⟩⟩ blocks for UCA-1 through UCA-4 are linked directly to the ⟨⟨FailureMode⟩⟩ block for `CoolingLineRupture`, which connects to the ⟨⟨Simulation⟩⟩ blocks verifying the strain and temperature requirements. Any design change affecting cooling line routing, pipe material, or BTMS control logic will automatically flag the associated UCAs and their downstream verification simulations for review.

5.2. The Descriptive Layer

The descriptive layer is modeled in Magic Systems of Systems Architect (MSOSA) in SysML. This tool was selected based on available software licensing; the Python middleware was developed using its implementation of OSLC REST API, SimulationWebClient library. But this is not a definitive implementation constraint, the architecture is applicable to any SysML environment with an accessible API.

The requirements diagram (Figure 4 on the next page) connects the ⟨⟨FailureMode⟩⟩ block for `CoolingLineRupture` to the higher-level system requirement of maintaining post-crash thermal stability, and to the specific failure plastic strain limit of the chosen material, 6063-T5 aluminum at 17.1%. The failure plastic strain limit of 17.1% is sourced from Barsoum et al. [44] work on AL 6063-T5 extruded tubes, and physical testing. Two ⟨⟨Simulation⟩⟩ blocks are mapped to verify these requirements, each with a *ModelLevel* attribute mapped to the fidelity hierarchy in Table 1 on the preceding page.

A ⟨⟨VirtualSensor⟩⟩ block is included as part of each ⟨⟨Simulation⟩⟩ block to define precisely what information is drawn from each simulation binouts for the KPIs. For the LS-DYNA model, this block specifies the node set comprising the coolant pipe, enabling nodal displacements to be extracted and the resulting cross-sectional restriction to be calculated using the Shapely Python library[1]. This restriction value is then passed as a boundary condition to the 1D thermal simulation.

Figures 5 on page 13, and 6 shows the block diagram of the model, including the mapping of the RAAML ⟨⟨UnsafeControlAction⟩⟩ of inadequate cooling to the ⟨⟨FailureMode⟩⟩ of thermal runaway, and the ⟨⟨UnsafeControlAction⟩⟩ of continued pump operation to the ⟨⟨FailureMode⟩⟩ of cooling line rupture.

[1]The open source library is available at: https://shapely.readthedocs.io/en/stable/

Table 2: STPA Unsafe Control Actions BTMS Post-Crash Cooling

UCA ID	Control Action	Type	Unsafe Control Action	Hazard
UCA-1	Pump speed command	Provided	BTMS commands continued circulation when line cross-sectional area is reduced below effective flow threshold post-crash	H-2, H-1
UCA-2	Pump shutoff	Not provided	BTMS does not shut off pump when flow is restricted by line deformation, because sensor feedback remains within nominal range	H-3, H-1
UCA-3	Pump speed command	Stopped too soon	BTMS reduces pump speed in response to elevated cell temperature, further reducing already-restricted flow	H-1
UCA-4	Valve position command	Provided	BTMS opens bypass valve on a module already receiving restricted flow, compounding thermal isolation of affected cells	H-1

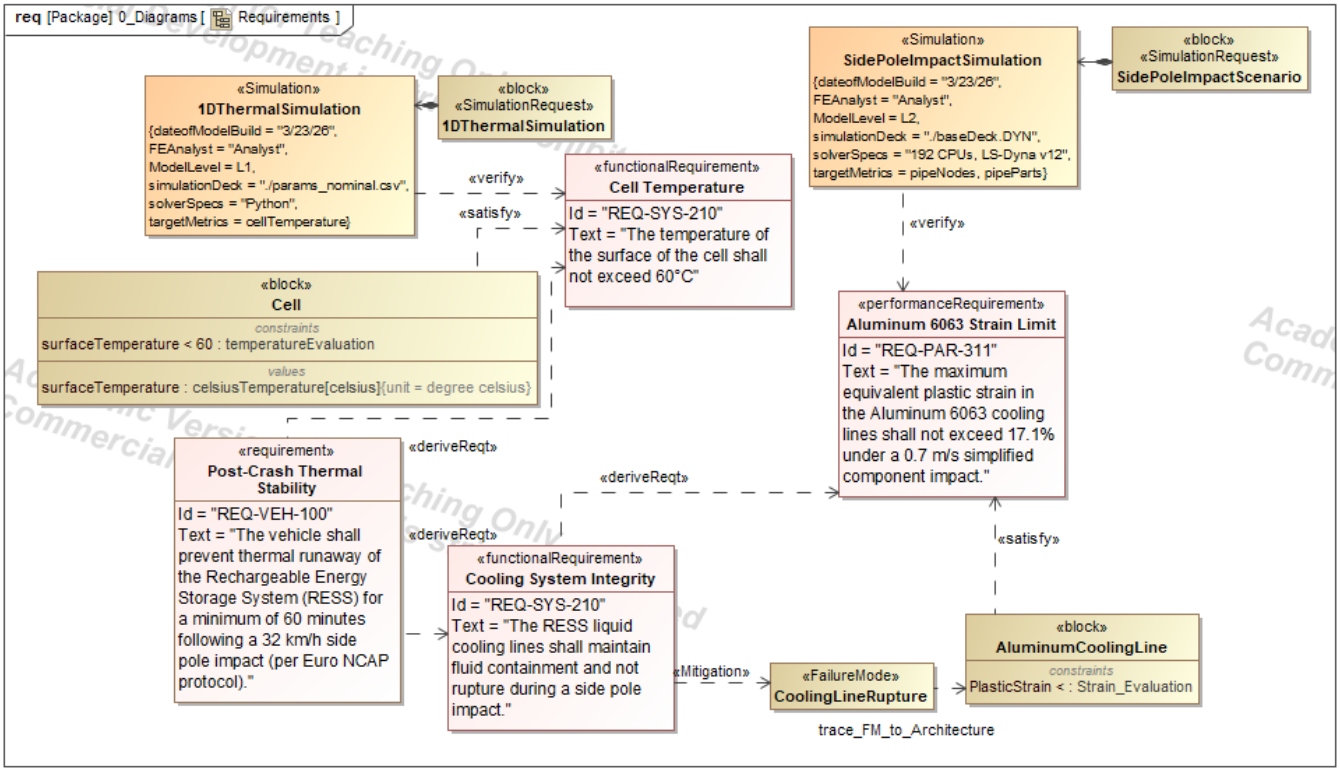

Figure 4: SysML Requirements diagram of case study

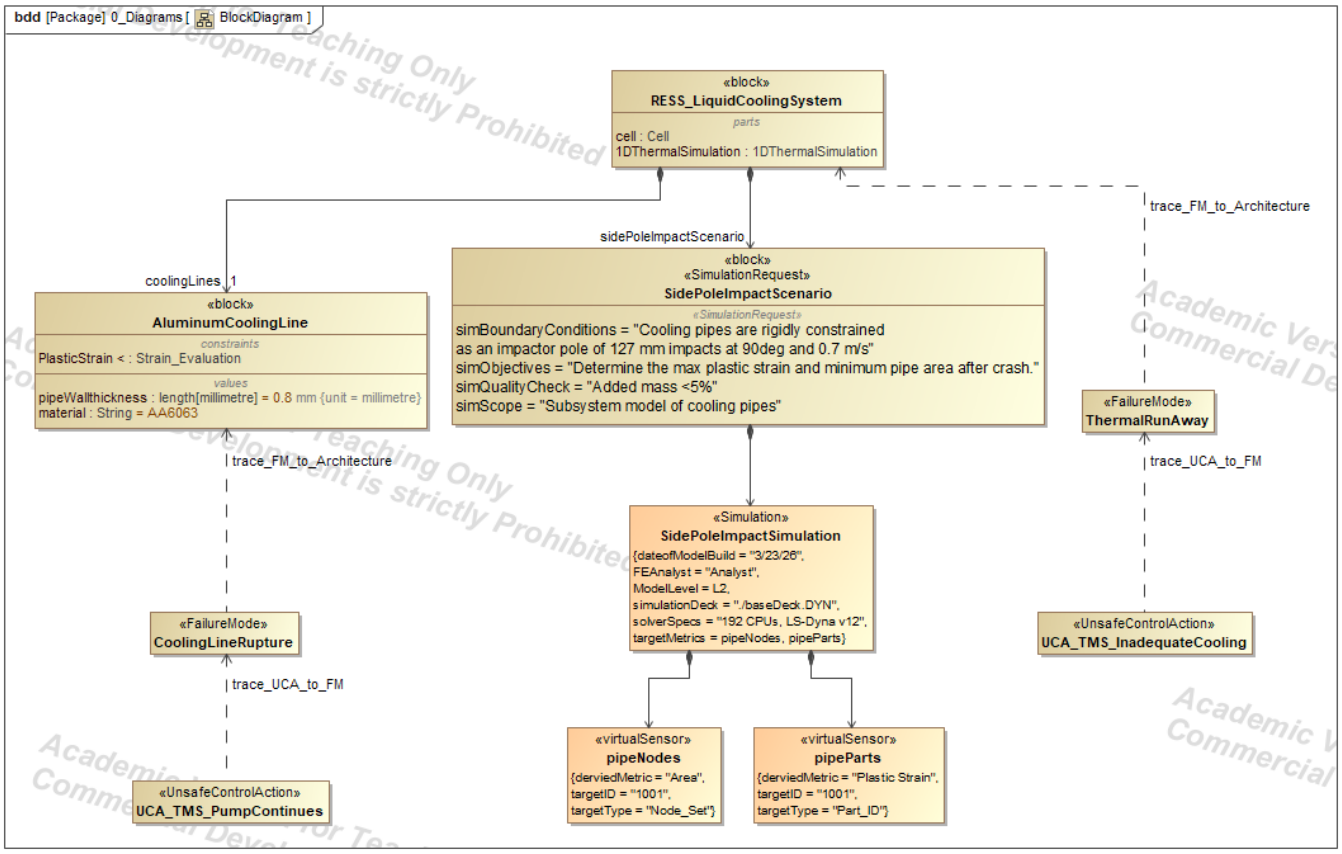

Figure 5: SysML Block Diagram of case study - LS-DYNA

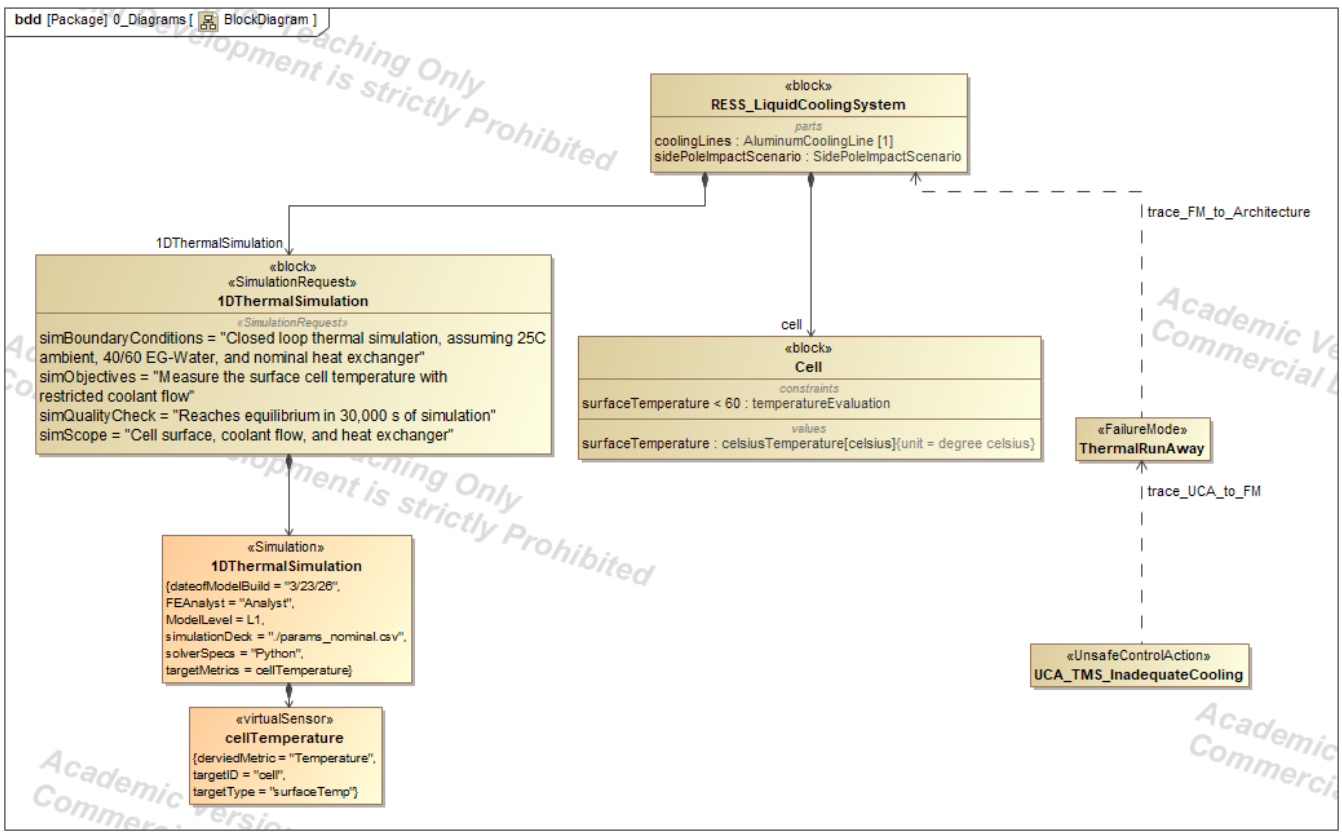

Figure 6: SysML Block Diagram of case study - 1D Thermal

5.3. Orchestration Middleware & SDM

Python serves as the orchestration middleware, executed through the MSOSA SysML Model Execution Plugin via a Jupyter Notebook and its SimulationWebClient API functionality[2].

The script polls the SysML model for simulation input parameters, impact angle, impact velocity, model file path, pipe wall thickness, and the cooling line material, then modifies the base LS-DYNA keyword file with the updated parameters, creates a versioned SDM workspace record, and submits the simulation to the HPC cluster. The script polls for job completion; in a production implementation, the architecture would integrate with the HPC scheduler API to receive a completion signal rather than polling.

The python script creates a record in a SQL database. Its schema is based on these important entries: Simulation ID, Simulation MetaData, Deck File Artifact Location, Input Parameters, Virtual Sensor Information, Extracted KPIs, Binout File Location. These

- **Simulation ID**
  SQL column: `simulation_id` *Type:* `UUID`
  Description: Unique identifier for a single simulation run; used to connect Simulation to SDM.

- **Simulation MetaData**
  SQL column: `simulation_metadata`
  Type: `JSONB`
  Description: Structured metadata about the run: solver version, HPC queue ID, node/core count, termination criteria, timestamps, etc.

- **Deck File Artifact Location**
  SQL column: `deck_file`
  Type: `VARCHAR`
  Description: Path or URI pointing to the LS-DYNA keyword deck on the file system or object store (e.g., `/hpc/runs/123/master.k`.

- **Input Parameters**
  SQL column: `input_parameters`
  Type: `JSONB`
  Description: Value map of inputs for the run (wall thickness, impact velocity, material ID, boundary condition flags) as extracted from the SysML model.

- **Virtual Sensor Information**
  SQL column: `virtual_sensor_def`
  Type: `JSONB`
  Description: Encodes how to select the data to be post-processed (e.g., LS-DYNA part IDs, element sets, node sets, geometric subset definition) plus intended KPI type and units.

- **Extracted KPIs**
  SQL column: `kpi_output`
  Type: `JSONB`
  Description: Scalar metrics computed from the simulation (max plastic strain, min cross-sectional area, final cell temperature) linked back to a simulation, a virtual sensor, and a SysML requirement.

- **Binout File Location**
  SQL column: `binout_file`
  Type: `VARCHAR`
  Description: Path to the LS-DYNA `binout` result file used by the Python post-processor to derive KPIs.

Once results are available, the Lasso Python library[3] extracts the maximum plastic strain from the part ID specified in the ⟨⟨VirtualSensor⟩⟩. If this value exceeds the target defined in the requirements, the script returns the value to the model and flags a high-risk event. If the value

[3]The open source library is available at: https://github.com/open-lasso-python/lasso-python

[2] Pseudo-code and MSOSA model for this case study is available on GitHub at: https://github.com/CoderMonad/Non-Linear-FEA-MBSE-Digital-Thread-Simulation-Pipeline

is below the threshold, the script proceeds to execute the 1D thermal simulation using the post-crash pipe area as a boundary condition, and exports the resulting cell temperature back to the SysML model for comparison against the thermal requirement.

5.4. The Analytical Layer

The analytical layer consists of two simulations: an L2 LS-DYNA model simulating the non-linear deformation of the cooling pipes, and an L1 1D thermal simulation implemented in Python.

5.4.1. L2 - LS-DYNA model

Figure 7 shows the model setup in LS-DYNA from a top-down view. The LS-DYNA model was correlated to a physical subsystem test, and contains two coolant pipes with opposed directional flow, impacted by a rigid pole. The model is parameterized such that values from the descriptive layer are injected directly into the input deck via the Python binder.

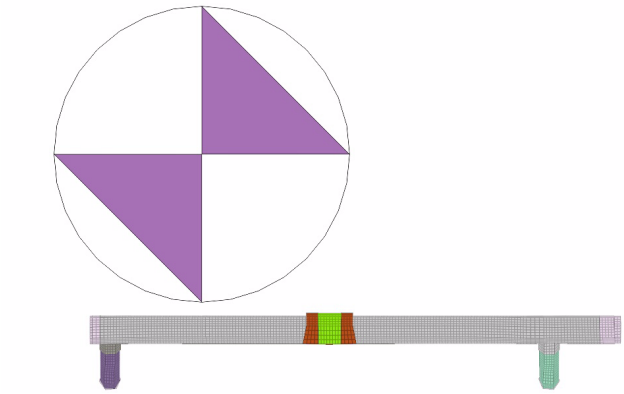

Figure 7: Top-down view of the LsDyna Subsystem model

Mesh sizing follows a graded approach: 1mm element size in the impact contact zone for deformation resolution, transitioning to 5mm elsewhere. The coolant pipes are rigidly constrained at their ends. The impactor is a rigid cylinder of 127mm (5in) diameter and 22kg mass, traveling perpendicular to the pipe axes at 0.7m/s. The pipes are modeled as 6063-T5 aluminum using `*MAT_024` Piecewise Linear Plasticity. The specific material card developed from coupon testing on sample 6063-T5 aluminum.

5.4.2. L1 - 1D thermal Simulation model

The 1D thermal simulation evaluates whether the post-crash cooling system retains sufficient capacity to prevent battery cell temperatures from exceeding the thermal runaway design threshold. Implemented as a lumped-parameter transient model in Python, the simulation represents the RESS cooling circuit as three coupled physical processes.

Heat exchange between the battery cells and circulating coolant is governed by the NTU effectiveness method:

$$\epsilon = 1 - \exp\left(\frac{-UA}{\dot{C}}\right) \tag{1}$$

where $\dot{C} = \dot{m}c_p$ is the coolant capacity rate. Heat rejection at the radiator is modeled using an assumed thermal effectiveness. Cell temperature is updated each timestep using a lumped thermal mass formulation, where the net heat retained by the cell stack drives the transient temperature rise.

All key thermal model parameters, along with their sources and justifications, are provided in Table 3 on the next page.

5.5. Results and discussion

The initial simulation, representing the baseline design configuration, was parameterized directly from the SysML model and submitted to the High-Performance Computing (HPC) cluster, utilizing 192 CPU cores. The Python middleware successfully orchestrated the data handoff, automatically extracting the required structural metrics from the LS-DYNA binary output files without manual intervention.

The internal cross-sectional area of the deformed pipe was computed dynamically using the Shapely Python library. The minimum cross-sectional area was extracted along the deformed pipe length to characterize the localized flow restriction over time, as shown in Figure 8 on the following page. This geometric

Table 3: Thermal parameters for 1D and justification

Parameter	Value	Basis / Justification	Citation
Coolant Type	40/60 EG-Water	Common RESS Coolant	
Coolant Specific Heat (c_p)	1,200 J/(kg·K)	40/60 EG, water at 60°C	[45]
Nominal mass flow rate ($\dot{m}$)	0.054 kg/s	Calculated from nominal pipe area and pump flow	
Overall heat transfer coefficient (UA)	50 WK^{-1}	Estimated for a compact plate-type heat exchanger consistent with RESS cooling subsystem scale	
Radiator thermal effectiveness	0.7	Representative value for automotive cross-flow heat exchangers	
Initial coolant temperature	63.5°C	Calculated from a steady-state analysis from RESS parameters	
Thermal runaway onset threshold	70°C cell surface	Conservative design limit adopted as a margin relative to the accelerated aging onset temperature for NMC Li-ion chemistry (∼80–120°C)	[46]
Cell thermal mass	200 kg	Mass based on a 5 cell representative model	
Cell Heat generation	1250 W	Heat gen based on a 5 cell representative model	
Cell specific heat	945 J/(kg·K)	Cell specific heat for an Automotive NMC Cell	[47]
Ambient temperature	25 °C	Assumed ambient temperature	
Simulation termination	30,000s or $\lvert\Delta T\rvert < 1 \times 10^{-4}$°C/step	Equilibrium criterion	

data serves as the critical surrogate metric connecting the structural impact to potential cooling system degradation.

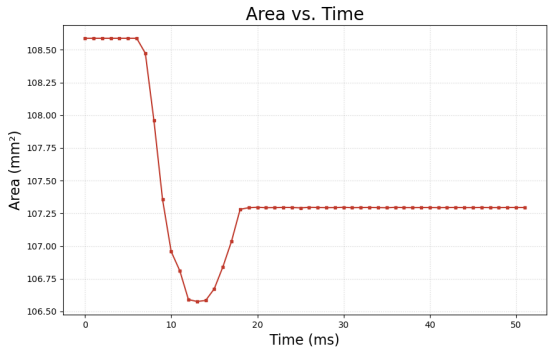

Figure 8: Coolant pipe area over time - default values

For this baseline configuration, the Verification Interface calculated a maximum plastic strain of 21.1%. Because this value exceeds the predefined safety threshold, indicating an unacceptable risk of material rupture and subsequent coolant leakage, the Python API automatically extracted this failure metric and injected it back into the corresponding SysML constraint block. Within the system model, the requirement status was instantly flagged as a failure, the result shown in the simulation console of MSOSA, seen in figure 9.

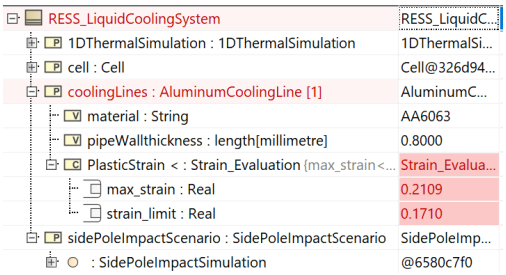

Figure 9: Simulations results showing plastic strain exceeding requirement

For this instance, the Python layer did not perform the 1D thermal simulation, as the plastic strain exceeded the requirement.

This demonstrates a core advantage of the proposed architecture: it provides immediate, automated bidirectional traceability. A localized failure in a computationally heavy 3D structural model violated a logical safety constraint, alerting the systems engineer to the specific physical hazard (Coolant Line Rupture) identified in the RAAML risk analysis.

To resolve this failure, a parameteric sweep can set up with the framework to quickly verify what thickness the coolant wall needs to be to pass the requirements.

Several instances were made, varying the coolant wall thickness to 0.2, 0.4, 0.6, and 1.0 mm thick. The automated verification loop was re-initiated. Each modified instance was subjected to the identical simulated side pole impact, with the 1D thermal simulation running on instances that do not exceed the 17.1% plastic strain.

Table 4 on the following page shows the results of the parameter sweep.

Based on the results, it can be seen that a wall thickness of 0.4 mm is the first feasible solution in this sweep, with a max plastic strain of 15%, and a final cell temperature below 70° C, even with the constricted final area.

A check that the results make intuitive sense can be performed by recalling that per Wierzbicki and Suh [48], decreasing a tube's wall thickness reduces peak plastic strain by shifting the deformation behavior from localized plastic hinging at the point of impact to distributed global ovalization. Because bending stiffness drops proportionally to t^3 with thinner walls, impact energy spreads across a much larger material volume instead of concentrating in a single localized zone. While this structural transition effectively lowers the maximum local strain, the increased post-crush ovalization permanently restricts the internal cross-sectional area. This creates an operational trade-off, as the resulting flow reduction introduces a potential risk for localized temperature spikes within the system. This behavior is not one that could be captured outside of non-linear CAE analysis.

The parameter sweep of the system architecture was able to identify a design choice that passed two competing safety requirements while allowing easy traceability between the failures.

5.6. Orchestration layer performance

To evaluate the practical viability of the proposed framework, the computational overhead of the Python digital binder was characterized independently from the LS-DYNA solver execution time. The full orchestration cycle, spanning SysML parameter extraction, master deck generation, HPC job submission, result polling, surrogate metric computation, and SysML requirement status update, was instrumented and timed across both the baseline and modified design configurations. The performance was measured for each step across the five instances ran. The mean completion time and standard deviations were calculated from the results. The 1D thermal simulation only shows the mean time to run since there were only two instances ran.

The LS-DYNA structural simulation itself, executed on 192 CPU cores required a wall-clock time of approximately 90.4 ± 4.5 seconds per run, after a several minute wait in queue. The Python binder overhead was measured separately and found to be: SysML parameter extraction via the MSOSA API query and master deck generation took 0.54 ± 0.12 seconds, and post-processing, including Shapely-based cross-sectional area computation and `binout` parsing via the Lasso library,completed in 21.63 ± 3.5 seconds, the longest part of which was loading in the `d3plot`, taking 21.55 ± 3.5 seconds. This time reflects only loading the specific elements identified in the VirtualSensor. Since the Lasso library allows for only the specified parts into memory this reduces the overall loading time compared to loading in the entire model.

The downstream 1D thermal simulation, executing on a local workstation, reached equilibrium in under 0.02 seconds for both instances run. The total wall-clock time for a complete single-configuration orchestration cycle, from parameter extraction to SysML

Table 4: Results from the parameter sweep

Wall thickness [mm]	Max Plastic Strain [%]	Plastic Strain Req	Final Pipe Area [mm^2]	Resulting Temperature [°C]	Temperature Req
0.2	13.9%	Passed	68.0	73.8	Failed
0.4	15.0%	Passed	94.9	63.3	Passed
0.6	18.2%	Failed	105.4	NA	NA
0.8 [baseline]	21.1%	Failed	107.3	NA	NA
1.0	23.0%	Failed	107.6	NA	NA

requirement update was dominated by solver queue wait time and HPC execution, which remains true for larger full-vehicle models that can take hours to run.

6. COMPARATIVE TRACEABILITY BASELINE

To empirically validate the framework's capacity to eliminate "orphaned requirements" and reduce the engineering "change tax" associated with DBSE, a simulated late-stage change management scenario was executed. The objective of this baseline is to quantify workflow efficiency and error susceptibility by measuring "Traceability Hops" (the number of discrete data transitions between tools or documents) and the "Time to Impact Assessment" (the overall time of the change).

6.1. Scenario Definition

Following the successful parameter sweep (Table 4) which identified the 0.4 mm wall thickness as a feasible design, a hypothetical design update to the cell chemistry is introduced. This update reduces the safe thermal runaway onset threshold from 70°C to 60°C. The engineering objective is to determine if the current baseline design still complies and to identify the associated systemic risk if it fails.

6.2. Traditional DBSE Workflow Evaluation

In a traditional DBSE paradigm, assessing the systemic impact of this requirement change relies on manual navigation across disparate data silos. The process necessitates four distinct traceability hops:

1. **RTM Update:** A systems engineer updates the text within the central RTM document.
2. **Domain Handoff:** The engineer must manually identify the impacted domains and notify the thermal and structural domain owners via email or ticketing systems.
3. **CAE Report Extraction:** The thermal analyst must search archived, static CAE PDF reports or PowerPoint decks to retrieve the prior parameter sweep data. They manually verify that the selected 0.4 mm wall thickness yielded a final cell temperature of 63.3°C, discovering it now violates the new 60°C threshold.
4. **Risk Mapping:** A safety engineer must then manually cross-reference an isolated FMEA or STPA spreadsheet to determine that this thermal failure maps directly to the undetected flow restriction, thereby triggering UCA-1 and UCA-2.

The Time to Impact Assessment in this manual workflow is highly variable, often spanning hours to days depending on personnel availability, and carries a severe risk of the new threshold becoming an "orphaned requirement" if any human handoff is missed.

6.3. Proposed MBSE-CAE Digital Thread Evaluation

Conversely, within the proposed bidirectional digital thread, the change propagates systematically and instantaneously.

1. **Single-Point Update:** The systems engineer updates the `ThermalRunawayThreshold` parameter within the SysML descriptive model from 70°C to 60°C.
2. **Automated Re-evaluation:** Because the previous simulation Key Performance

Indicators (KPIs) are stored and parametrically linked, the SysML constraint evaluator instantaneously evaluates the stored 63.3°C result against the new 60°C threshold. The ⟨⟨Simulation⟩⟩ block for the 0.4 mm configuration is automatically flagged as "Failed" in real-time, without requiring a CAE solver re-run.

3. **Semantic Traceability:** Utilizing the embedded RAAML ontology, the user executes a single dependency query (one click) to traverse the *verifies* and *mitigates* links. The environment instantly highlights the downstream impact, visually linking the thermal failure back to the `CoolingLineRupture` ⟨⟨FailureMode⟩⟩ and the governing UCA-1.

6.4. Comparative Metrics

Table 5 on the following page summarizes the performance of the proposed architecture against the DBSE baseline. By transforming static documents into a dynamic, API-driven graph, the framework reduces traceability hops by 75% and compresses the impact assessment time from hours to seconds. Most critically, it entirely eliminates the risk of orphaned requirements, as the logical safety constraints and the physical simulation results are perpetually synchronized.

7. LIMITATIONS AND FUTURE WORK

While the proposed reference architecture successfully demonstrates a bidirectional digital thread between descriptive MBSE models and high-fidelity CAE solvers, several limitations remain that warrant further investigation.

The current framework operates deterministically, relying on idealized nominal values to produce single scalar results. This limits its ability to realistically assess vehicle crash dynamics and thermal propagation, as it fails to account for stochastic uncertainties like manufacturing tolerances, material variability, and mesh discretization errors. To overcome this, the model must transition to a probabilistic approach. This requires updating the MBSE environment, specifically the SysML/RAAML constraints and the Python digital binder, to utilize statistical distributions such as Gaussian instead of static scalar values.

Another limitation of the proposed framework is that its automated orchestration capabilities are bounded by the parametric abilities of the pre-built FEA model. While the Python binder can modify scalar inputs such as wall thickness or impact velocity, the underlying mesh topology and geometric architecture must still be manually constructed by a specialist CAE engineer outside of the MBSE environment. The framework therefore cannot respond to design changes that are topological rather than parametric, a repositioned coolant line, an added bracket, or a modified cross-sectional profile cannot be propagated into the analytic spoke automatically, and the mesh must be rebuilt by hand. Resolving this would require integration with geometry-driven meshing pipelines, such as ANSA or HyperMesh scripting, capable of translating MBSE-defined geometric changes directly into updated FEA discretizations without manual intervention.

The proposed approach also has limitations relative to commercial platforms in its absence of a managed PLM backbone. Toolchains such as Teamcenter provide robust data governance, access control, and IP protection features that are essential in multi-supplier development environments. The Python binder, in its current form, does not address these concerns. The proposed framework is therefore best understood not as a commercial platform replacement, but as a reference architecture demonstrating that bidirectional MBSE-CAE integration for non-linear crash simulation is achievable without enterprise infrastructure, and as a foundation upon which PLM governance layers could be added as organizational maturity increases.

8. CONCLUSION

The electrification of ground vehicle powertrains has introduced complex, multi-domain safety requirements that traditional DBSE can no longer effectively manage. This

Table 5: Traceability Metric Comparison

Metric	Traditional DBSE / Manual Workflow	Proposed MBSE-CAE Architecture	Improvement Factor
Active Engineering Time (per iteration)	≈ 15 minutes (manual deck edit, post-processing)	0.54s (parameter extraction) + 21.6s (post-processing)	≈ 40x reduction
Manual Data Handoffs	4 (SysML → CAE → Thermal script → SysML)	0 (Fully API-driven)	100% elimination
Traceability Hops	Manual search across 3 disparate documents	1 (Automated semantic link)	N/A
Risk of Data Transcription Error	High	Zero	N/A

paper addressed the critical "fidelity gap" between descriptive MBSE architectures and high-fidelity CAE by proposing an open, four-layer reference architecture. By embedding STPA within a formalized digital thread, the proposed framework establishes robust, bidirectional requirement traceability across the system lifecycle, and with risk at the center of model development.

Through the simplified EV side pole impact case study, the architecture successfully demonstrated the automated orchestration of non-linear structural and lumped-parameter thermal simulations. The Python-based interoperability middleware effectively mapped descriptive parameters to solver input decks, executed the HPC workflow, and programmatically extracted critical Key Performance Indicators (KPIs) from dense binary outputs (`d3plot`, `binout`). By automatically injecting these metrics back into the SysML environment to trigger pass/fail compliance states, the framework enables closed-loop, multidisciplinary design optimization. The parameter sweep validation demonstrated that competing physical constraints, such as structural strain limits and thermal runaway thresholds, can be balanced rapidly without manual data translation. Furthermore, execution profiling confirmed that the orchestration middleware introduces negligible computational overhead relative to the requisite explicit solver runtimes.

References

[1] Y.-M. Cheng, D.-X. Gao, F.-M. Zhao, and Q. Yang, "Early warning method for charging thermal runaway of electric vehicle lithium-ion battery based on charging network," *Scientific Reports*, vol. 15, no. 1, p. 7895, Mar. 6, 2025, ISSN: 2045-2322. DOI: 10.1038/s41598-025-92738-7. Accessed: Feb. 25, 2026. [Online]. Available: https://www.nature.com/articles/s41598-025-92738-7.

[2] V. Hemakumar, V. Chakravarthy, S. Surendranath, V. Gundu, M. Ramkumar Prabhu, and S. Hari Chandra Prasad, "A multi-scale modeling approach for predicting and mitigating thermal runaway in electric vehicle batteries," *Thermal Science and Engineering Progress*, vol. 56, p. 103 029, Dec. 2024, ISSN: 24519049. DOI: 10.1016/j.tsep.2024.103029. Accessed: Feb. 25, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2451904924006474.

[3] NHTSA, *Federal motor vehicle safety standards; FMVSS no. 305a electric-powered vehicles: Electric powertrain integrity*, 2024.

[4] UNECE, *Uniform provisions concerning the approval of vehicles with regard to the protection of the occupants in the event of a lateral collision*, 2011.

[5] SAMR and SAC, *GB 38031-2025: Electric vehicles traction battery safety requirements*, 2025.

[6] E. R. Carroll and R. J. Malins, "Systematic literature review: How is model-based systems engineering justified?" Sandia National Laboratories, Albuquerque, New Mexico, Tech. Rep. SAND2016-2607, Mar. 2016.

[7] P. Dutta Banik and D. S. Sengupta, "Compare and analysis MBSE benefits with document-centric traditional system engineering approach," in *Proceedings of the International Conference on Industrial Engineering and Operations Management*, Detroit, USA: IEOM Society International, Oct. 9, 2024, ISBN: 9798350717297. DOI: 10.46254/wc01.20240082. Accessed: Feb. 25, 2026. [Online]. Available: https://index.ieomsociety.org/index.cfm/article/view/ID/25326.

[8] I. M. Siddique, "MBSE implementation in small satellite systems: Rationale for adoption over traditional document-based systems engineering," *International Journal of Geoinformatics Science and Technology*, vol. 1, no. 1, pp. 1–13, Jan. 7, 2025. DOI: 10.46610/IJGST.2025.v01i01.001. Accessed: Mar. 17, 2026. [Online]. Available: https://matjournals.net/engineering/index.php/IJGST/article/view/1295.

[9] O. Odarushchenko, O. Odarushchenko, O. Striuk, V. Shamanskyi, and P. Hroza, "Automating requirements traceability in project documentation using TraceTrend tool: Model, design and application," *Radioelectronic and Computer Systems*, vol. 2025, no. 4, pp. 236–246, Dec. 8, 2025, ISSN: 2663-2012, 1814-4225. DOI: 10.32620/reks.2025.4.16. Accessed: Mar. 17, 2026. [Online]. Available: https://nti.khai.edu/ojs/index.php/reks/article/view/reks.2025.4.16.

[10] A. M. Madni and M. Sievers, "Model-based systems engineering: Motivation, current status, and research opportunities," *Systems Engineering*, vol. 21, no. 3, pp. 172–190, May 2018, ISSN: 1098-1241, 1520-6858. DOI: 10.1002/sys.21438. Accessed: Feb. 25, 2026. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/10.1002/sys.21438.

[11] A. Akundi, W. Ankobiah, O. Mondragon, and S. Luna, "Perceptions and the extent of model-based systems engineering (MBSE) use – an industry survey," in *2022 IEEE International Systems Conference (SysCon)*, Montreal, QC, Canada: IEEE, Apr. 25, 2022, pp. 1–7, ISBN: 9781665439923. DOI: 10.1109/SysCon53536.2022.9773894. Accessed: Mar. 17, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/9773894/.

[12] J. M. Borky and T. H. Bradley, *Effective Model-Based Systems Engineering* (SpringerLink Bücher). Cham: Springer International Publishing, 2019, 779 pp., ISBN: 9783319956695.

[13] O. Gotel and A. Finkelstein, "Extended requirements traceability: Results of an industrial case study," in *Proceedings of ISRE '97: 3rd IEEE International Symposium on Requirements Engineering*, Annapolis, MD, USA: IEEE Comput. Soc. Press, 1997, pp. 169–178, ISBN: 9780818677403. DOI: 10.1109/ISRE.1997.566866. Accessed: Mar. 17, 2026. [Online]. Available: http://ieeexplore.ieee.org/document/566866/.

[14] A. Busch, R. Dresibach, and F. Popielas, "What do simulation engineers need to know about (model- based) systems engineering," *Engineering Modelling, Analysis and Simulation*, vol. 3, no. 1, 2026, ISSN: 30059763. DOI: 10.59972/fzvc41wt. Accessed: Feb. 25, 2026. [Online]. Available: https://emas.nafems.org/index.php/v1/article/view/fzvc41wt.

[15] H. Kim, D. Fried, P. Menegay, G. Soremekun, and C. Oster, "Application of integrated modeling and analysis to development of complex systems," *Procedia Computer Science*, vol. 16, pp. 98–107, 2013, ISSN: 18770509. DOI: 10.1016/j.procs.2013.01.011. Accessed: Feb. 25, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050913000124.

[16] B. P. Zeigler, S. Mittal, and M. K. Traore, "MBSE with/out simulation: State of the art and way forward," *Systems*, vol. 6, no. 4, p. 40, Nov. 15, 2018, ISSN: 2079-8954. DOI: 10.3390/systems6040040. Accessed: Jan. 7, 2026. [Online]. Available: https://www.mdpi.com/2079-8954/6/4/40.

[17] M. Chami and J.-M. Bruel, "A survey on mbse adoption challenges," in *The Systems Engineering Conference of the Europe, Middle-East and Africa (EMEA) Sector of INCOSE (EMEASEC 2018)*, Berlin, Germany, Nov. 2018.

[18] A. Akundi and V. Lopez, "A review on application of model based systems engineering to manufacturing and production engineering systems," *Procedia Computer Science*, vol. 185, pp. 101–108, 2021. DOI: 10.1016/j.procs.2021.05.011.

[19] A. M. Madni and S. Purohit, "Economic analysis of model-based systems engineering," *Systems*, vol. 7, no. 1, p. 12, 2019. DOI: 10.3390/systems7010012.

[20] N. G. Leveson, *Engineering a safer world: systems thinking applied to safety* (Engineering systems), New paperback edition. Cambridge, Massachusetts London, Englang: The MIT Press, 2017, 534 pp., ISBN: 9780262533690.

[21] S. M. Sulaman, A. Beer, M. Felderer, and M. Höst, "Comparison of the fmea and stpa safety analysis methods–a case study," *Software Quality Journal*, 2017.

[22] *Risk analysis and assessment modeling language (RAAML) libraries and profiles*, Dec. 2022. [Online]. Available: https://www.omg.org/spec/RAAML/1.0/PDF.

[23] A. Ahlbrecht, U. Durak, and W. Zaeske, "Model-based stpa: Towards agile safety-guided design with formalization," in *2022 8th IEEE International Symposium on Systems Engineering (ISSE)*, IEEE, 2022.

[24] A. Mallya, V. Pantelic, M. Adedjouma, M. Lawford, and A. Wassyng, "Using stpa in an iso 26262 compliant process," in *Computer Safety, Reliability, and Security (SAFECOMP)*, Springer, 2016, pp. 117–129.

[25] Y. Li, W. Liu, Q. Liu, X. Zheng, K. Sun, and C. Huang, "Complying with iso 26262 and iso/sae 21434: A safety and security co-analysis method for intelligent connected vehicle," *Sensors*, vol. 24, no. 6, p. 1848, 2024.

[26] "Global horizons united states air force global science and technology vision," United States Air Force, AF/ST TR13-01, Jun. 21, 2013. [Online]. Available: https://www.hsdl.org/?view&did=741377.

[27] V. Singh and K. E. Willcox, "Engineering design with digital thread," *AIAA Journal*, vol. 56, no. 11, pp. 4515–4528, Nov. 2018, ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.J057255. Accessed: Feb. 21, 2026. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.J057255.

[28] E. M. Kraft, "The air force digital thread/digital twin-life cycle integration and use of computational and experimental knowledge," in *54th AIAA Aerospace Sciences Meeting*, 2016. DOI: 10.2514/6.2016-0897.

[29] Q. Zhang, S. Zheng, C. Yu, Q. Wang, and Y. Ke, "Digital thread-based modeling of digital twin framework for the aircraft assembly system," *Journal of Manufacturing Systems*, vol. 65, pp. 406–420, 2022. DOI: 10.1016/j.jmsy.2022.10.004.

[30] P. M. Khanolkar, J. Gopsill, and A. Olechowski, "Decoding the digital thread digitalization approach for product design and development: Benefits, challenges, and extensions," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 39, e23, 2025. DOI: 10.1017/S0890060425100140.

[31] Q. Zhang, J. Liu, and X. Chen, "A literature review of the digital thread: Definition, key technologies, and applications," *Systems*, vol. 12, no. 3, p. 70, Feb. 23, 2024, ISSN: 2079-8954. DOI: 10.3390/systems12030070. Accessed: Feb. 25, 2026. [Online]. Available: https://www.mdpi.com/2079-8954/12/3/70.

[32] S. Kang, M. Gottschall, S. Cho, and T. Blochwitz, "Requirements-based, early stage architecture performance validation on a brake system use case," presented at the Asian Modelica Conference 2024, Jeju, Korea, December 12 – 13, 2024, Nov. 13, 2025. DOI: 10.3384/ecp21793. Accessed: Mar. 3, 2026. [Online]. Available: https://ecp.ep.liu.se/index.php/modelica/article/view/1494.

[33] C. Coïc, M. Williams, J. C. Mendo, J. M. Alvarez-Rodríguez, and M. Richardson, "Linking design requirements to fmus to create lotar compliant mbse models," in *FMI User Meeting*, 2023.

[34] J. D'Ambrosio and G. Soremekun, "Systems engineering challenges and MBSE opportunities for automotive system design," in *2017 IEEE international conference on systems, man, and cybernetics (SMC)*, IEEE, 2017, pp. 2075–2080.

[35] H. Sohier, P. Lamothe, S. Guermazi, M. Yagoubi, P. Menegazzi, and A. Maddaloni, "Improving simulation specification with MBSE for better simulation validation and reuse," *Systems Engineering*, vol. 24, no. 6, pp. 425–438, Nov. 2021, ISSN: 1098-1241, 1520-6858. DOI: 10.1002/sys.21594. Accessed: Feb. 24, 2026. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/10.1002/sys.21594.

[36] P. Graignic, T. Vosgien, M. Jankovic, V. Tuloup, J. Berquet, and N. Troussier, "Complex system simulation: Proposition of a MBSE framework for design-analysis integration," *Procedia Computer Science*, vol. 16, pp. 59–68, 2013, ISSN: 18770509. DOI: 10.1016/j.procs.2013.01.007. Accessed: Feb. 24, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050913000082.

[37] T. Kaloor and I. I. Barosan, "A MBSE framework for the design and analysis of complex automotive systems using SysML and PCE," in *2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C)*, Hyderabad, India: IEEE, Jun. 4, 2024, pp. 191–198, ISBN: 9798350366259.

DOI: 10.1109/ICSA-C63560.2024.00042. Accessed: Feb. 24, 2026. [Online]. Available: https://ieeexplore.ieee.org/document/10628178/.

[38] J. Ryan, S. Sarkani, and T. Mazzuchim, "Framework for architecture trade study using MBSE and performance simulation," in *Selected Papers Presented at MODSIM World 2011 Conference and Expo*, 2012.

[39] P. Castellano, "A new space MBSE framework for automating requirements verification with SysMLv2 and python," Ph.D. dissertation, Politecnico di Milano, Milan, Italy, Apr. 3, 2025.

[40] T. Sarathi and R. Martinez, "A framework for seamless interoperability: Linking mission models, system models, and high-fidelity simulations for defense applications," in *INCOSE International Symposium*, vol. 35, Wiley Online Library, 2025, pp. 444–456.

[41] C. Nigischer, S. Bougain, R. Riegler, H. P. Stanek, and M. Grafinger, "Multi-domain simulation utilizing SysML: State of the art and future perspectives," *Procedia CIRP*, vol. 100, pp. 319–324, 2021, ISSN: 22128271. DOI: 10.1016/j.procir.2021.05.073. Accessed: Feb. 24, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2212827121005382.

[42] M. G. Fernández-Godino, "Review of multi-fidelity models," *Advances in Computational Science and Engineering*, vol. 1, no. 4, pp. 351–400, Dec. 1, 2023. DOI: 10.3934/acse.2023015. Accessed: Feb. 24, 2026. [Online]. Available: https://www.aimsciences.org/en/article/doi/10.3934/acse.2023015.

[43] M. Zhang, A. Jungo, and N. Bartoli, "Disciplinary data fusion for multi-fidelity aerodynamic application," in *6th CEAS Aerospace Europe Conference*, 2017.

[44] I. Barsoum, F. Khan, A. Molki, and A. Seibi, "Modeling of ductile crack propagation in expanded thin-walled 6063-t5 aluminum tubes," *International Journal of Mechanical Sciences*, vol. 80, pp. 160–168, Mar. 2014, ISSN: 00207403. DOI: 10.1016/j.ijmecsci.2014.01.012. Accessed: Mar. 18, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020740314000253.

[45] A. M. Azmi Roslan, N. Tukiman, N. N. Ibrahim, and A. R. Juanil, "The effects of ethylene glycol to ultrapure water on its specific heat capacity and freezing point," *J. Appl. Environ. Biol. Sci*, vol. 7, no. 7, pp. 54–60, 2017.

[46] S. Abada, G. Marlair, A. Lecocq, M. Petit, V. Sauvant-Moynot, and F. Huet, "Safety focused modeling of lithium-ion batteries: A review," *Journal of Power Sources*, vol. 306, pp. 178–192, Feb. 2016, ISSN: 03787753. DOI: 10.1016/j.jpowsour.2015.11.100. Accessed: Mar. 18, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S037877531530598X.

[47] A. Loges, S. Herberger, P. Seegert, and T. Wetzel, "A study on specific heat capacities of li-ion cell components and their influence on thermal management," *Journal of Power Sources*, vol. 336, pp. 341–350, Dec. 2016, ISSN: 03787753. DOI: 10.1016/j.jpowsour.2016.10.049. Accessed: Mar. 18, 2026. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0378775316314410.

[48] T. Wierzbicki and M. Suh, "Indentation of tubes under combined loading," *International Journal of Mechanical Sciences*, vol. 30, no. 3-4, pp. 229–248, 1988. DOI: 10.1016/0020-7403(88)90057-4.

9. CONTACT INFORMATION

Patrick J. Rye
Propulsion VPIM at General Motors
Doctoral Candidate at Colorado State University — Systems Engineering Department
E-mail: patrick.rye@gm.com

DISCLAIMER

The views, analyses, and conclusions expressed in this paper are solely those of the author. They do not represent the official positions, policies, or endorsements of any affiliated organization.

ACKNOWLEDGMENTS

The author acknowledges Dr. Jeremy Daily, who kindly provided valuable editorial and technical comments to improve the quality of the paper. The author also acknowledges Emily Rye for her assistance with proofreading the manuscript.

The author acknowledges that Generative AI was utilized to assist in early drafting, grammar refinement, editorial clarity, word count reduction, and early code development. All AI-generated content was verified, revised, and approved by the author. The conceptual development, data interpretation, and final conclusions remain entirely the work of the author.

ACRONYMS

Acronym	Definition
API	Application Programming Interface
BTMS	Battery Thermal Management System
CAE	Computer-Aided Engineering
DBSE	Document-Based Systems Engineering
ECU	Electronic Control Unit
EV	Electric Vehicle
FE	Finite Element
FEA	Finite Element Analysis
FMEA	Failure Mode and Effects Analysis
FMI	Functional Mock-up Interface
FMU	Functional Mock-up Unit
FTA	Fault Tree Analysis
HPC	High-Performance Computing
KPI	Key Performance Indicator
MBSE	Model-Based Systems Engineering
MDAO	Multidisciplinary Design Analysis and Optimization
MSOSA	Magic Systems of Systems Architect (Specific software tool)
NTU	Number of Transfer Units (Thermal effectiveness method)
OSLC	Open Services for Lifecycle Collaboration
PCE	Parametric Constraint Evaluator
PLM	Product Lifecycle Management
RAAML	Risk Analysis and Assessment Modeling Language
RESS	Rechargeable Energy Storage System (Battery)
ROM	Reduced Order Model
RTM	Requirements Traceability Matrix
SDM	Simulation Data Management
STAMP	System-Theoretic Accident Model and Processes
STPA	System-Theoretic Process Analysis
SysML	Systems Modeling Language
UCA	Unsafe Control Action
XMI	XML Metadata Interchange