INCOSE

ASIA-OCEANIA SYSTEMS ENGINEERING CONFERENCE

Utilizing SysML Knowledge in LLMs for Pattern-Based System Architecture Generation in Mission-Critical Systems

INCOSE AOSEC · 2025

AUTHORS

Habibi Husain Arifin, Jirapun Daengdej, Muhammad Khalik Humar, Saulius Pavalkis, and Firas Mahmoud

PUBLICATION DATE

July, 2026

SUBMISSION #

76

INCOSE PUBLICATIONS

Utilizing SysML Knowledge in LLMs for Pattern-Based System Architecture Generation in Mission-Critical Systems

Habibi Husain Arifin
Assumption University of Thailand
Graduate School of Business
habibihusainarifin@gmail.com

Jirapun Daengdej
Assumption University of Thailand
Graduate School of Business
jirapun.dd@gmail.com

Muhammad Khalik Humar
University of Technology Sydney
School of Mechanical and Mechatronic Engineering
muhammadkhalikhumar@gmail.com

Saulius Pavalkis
Dassault Systèmes
CATIA Cyber Systems
saulius.pavalkis@3ds.com

Firas Mahmoud
Dassault Systèmes
Data Science & AI Senior Leader
firas.mahmoud@3ds.com

Abstract

This paper presents a novel method to utilize SysML as a context and process constraint for LLM-based model generation, together with a spreadsheet mapping template to bridge SysML and non-SysML system architects. The approach models a multi-agent system architecture to capture the context, constraints, and process sequences required for generating SysML system architecture models from unstructured/semi-structured specifications. Using an open platform LLM engine, the workflow automates the extraction and structuring of SysML elements, while ensuring human-in-the-loop reviews for accuracy and compliance. The method supports traceability, structured outputs, and alignment with MBSE methodology, enabling consistent integration between modelers and non-modelers. By applying SysML to define interdependencies and sequencing, the approach con-strains LLM outputs to valid SysML constructs. This creates a replicable framework for mission-critical systems, particularly in aerospace and defense, where precision, safety, and regulatory alignment are mandatory.

Keywords

AI4MBSE, MBSE4AI, SysML, Multi-Agent Systems, GenAI, LLM, MBSE.

Introduction

Problem Statement

As Large Language Models (LLMs) continue to make penetration across industries, their application within mission-critical systems such as aerospace and defense must be carefully approached with high precision (Arifin, 2025). As an example, the idea of fully automating safety case generation using GPT-4 without a human in the loop can only achieve approximately 80–90% structural and semantic accuracy, which is insufficient according to safety standards for safety-critical systems (Graydon & Lehman, 2025). A survey highlighted the essential role of interdisciplinary collaboration between researchers, practitioners, and regulators in advancing the safe, ethical, and effective use of LLMs in this domain (Esposito, Palagiano, Lenarduzzi, & Taibi, 2025).

At the heart of this interdisciplinary collaboration among mission-critical systems lies Model-Based Systems Engineering (MBSE), a key enabler of digital engineering (Arifin, 2025). One of the best examples of harmonization between digital engineering and MBSE adoption in the aerospace domain is the NASA MBSE Transformation. A primary digital transformation goal of the NASA engineering center is adopting the approach by tailoring standardized engineering knowledge and business processes and workflows into the integrated toolchains and associated digital thread ecosystem (Pierce, Nicoli, Hill, & Cornford, 2024). To reduce ambiguities in capturing the knowledge and processes, practitioners in this domain often use a formalized/semi-formalized language such as Systems Modeling Language (SysML) (Friedenthal, Moore, & Steiner, 2015). SysML is a graphical modeling language which supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities within a system model. A system architecture model is a structured representation that focuses on the overall system requirements, behavior, structure, properties, and their interconnections. The model provides a holistic view and critical knowledge for decision makers in the mission-critical domain (Hart, 2006).

The intersection of MBSE and AI has led to two complementary approaches: "AI for MBSE / AI Assisted MBSE" (AI4MBSE) (Longshore, Bell, & Madachy, 2024) and "MBSE for AI" (MBSE4AI) (Sprockhoff, et al., 2023). AI4MBSE involves applying AI techniques such as LLMs to enhance the efficiency and automation of systems engineering tasks. When carefully integrated, AI4MBSE can transform how complex systems are designed, developed, verified, validated, and deployed (FAA, 2024) (Longshore, Bell, & Madachy, 2024) (Stefani, et al., 2024). However, an AI technique like an LLM also introduces new challenges which mostly originate from the "black box nature" of complex neural-network-based algorithms (Sprockhoff, et al., 2023). Unlocking the benefits of this integration, however, requires a deep understanding of LLMs' strengths and limitations, and a strong commitment to maintaining human oversight (human-in-the-loop) (Graydon & Lehman, 2025) (Hobbs & Li, 2024) (FAA, 2024). Two examples of LLM potential strengths in mission-critical system lifecycle are: 1. Improving the fidelity and completeness of system/software architecture model, design, development, testing, and requirements traceability; 2. Automating the review of various compliance reports and data, including system safety assessments. LLMs have the potential to learn from the patterns and identify gaps and discrepancies that might be missed by manual review (FAA, 2024). A study by NPS (Longshore, Bell, & Madachy, 2024) leveraged LLMs to generate, modify, and query SysML-based system architecture models. However, LLMs have a limitation of generating spurious or hallucinatory material when they fails to identify the patterns of new criteria (FAA, 2024). MBSE4AI, in contrast, utilizes MBSE to define the context, classify problems to solve, and improve the modularity and traceability of multi-agentic AI in mission-critical systems. A balanced focus on both AI4MBSE and MBSE4AI is necessary to gather both-sided benefits in performance and traceability in digital engineering. Adapting existing methods to the entire development and usage lifecycle of black box AI systems is necessary. An example of this adaptation is using a systematic development framework to design and model the AI-based system, including structure and process definition (Sprockhoff, et al., 2023).

Through its three pillars (i.e., methodologies, languages, and tools) (Delligatti, 2014), MBSE serves as a digital backbone for the digital engineering framework, especially in domains where safety is paramount such as aerospace and defense. A modeling language like SysML is one of the MBSE pillars that supports the creation of system architecture models that structure and visualize information, capturing the interdependencies between requirements, systems/subsystems/components, functions, parameters, and constraints (Delligatti, 2014) (Friedenthal, Moore, & Steiner, 2015). When AI is introduced into this framework, it should be treated not as an infrastructural pillar but as an application-layer enhancement (FAA, 2024). AI must be driven by clearly defined use cases – mission-focused, data-

informed, and aligned with regulatory expectations (FAA, 2024). The success or failure of AI in this domain hinges entirely on its grounding in specific and high-value business objectives. Top causes of AI adoption failure are misdefining the problem, optimizing models for the wrong metrics, and not fitting the business/mission workflow (Ryseff, Bruhl, & Newberry, 2024).

One emerging and trending use case that requires both AI4MBSE and MBSE4AI in mission-critical domains (e.g., aerospace & defense) is the need for automation that turns unstructured specifications into SysML-structured data patterns as a starting point for systems engineers to define a system architecture model (Longshore, Bell, & Madachy, 2024) (Stefani, et al., 2024). This application utilizes AI to generate and predict a system architecture by understanding the patterns of specification documents, but at the same time, the AI also needs to understand the context of the system of interest (SOI), SysML constraints, and interdependencies between requirements, structures (i.e., systems/subsystems/components), functions, and parameters. For example, in order to suggest a list of structures and functions, the AI system needs to suggest the list of requirements which these structures and functions need to satisfy (fulfill). These interdependencies are commonly defined in the methodology pillar of MBSE (Delligatti, 2014).

Overall, this paper suggests a novel method to utilize the power of a SysML-based semantic model as a context and constraint when using a black box AI, such as an LLM to suggest a system architecture model extracted from specification documents and/or pre-trained models. To build a systematic context with SysML, this method borrows the concept of multi-agent system architecture by decomposing the system into a set of different agents to handle different problems, i.e., requirement agent, system/subsystem/component agent, function agent, and parameter agent (Sadik & Goerick, 2024) (Maheshwari, Raz, DeLaurentis, Murphy, & Kolawole, 2018) (Bouamra, Poisson, Yun, & Armetta, 2025) (Maheshwari, Kenley, & DeLaurentis, Creating Executable Agent-Based Models Using SysML, 2015) (Mour, Kenley, Davendralingam, & DeLaurentis, 2013) (Jin, Jin, Chen, & Wang, 2024). To ease readers' understanding of the flow of this paper, the authors break it down into the following sections:

- Finding Gaps: Identify where the current approaches fall short and why those gaps matter.
- Contributions of the Paper: Explain what this study adds to close those gaps.
- Proposed Solutions: Present this paper's approach with a SysML contextual model and workflow.
- Scope of the Paper: Define the boundaries (covered and not covered) and assumptions of this paper.
- Results: Interpret the results of the proposed solution.
- Conclusion and Future Work: Summarize the key takeaways and suggest future work.

Finding Gaps

To tailor the problem statement to the proposed solutions, the authors conducted a literature review to identify research gaps, which form the focus of this paper. These gaps highlight existing issues where practitioners attempt to embed AI technology into their systems engineering (SE) processes (AI4SE/MBSE) without adequately considering the need for contextual information in the application. Overall, the authors classify these gaps as follows:

1. **Contextual and data gaps**: Multiple studies related to AI4MBSE, MBSE4AI, and multi-agent systems (MAS) methods demonstrated various use cases to use SysML for modeling and simulation of multi-agent system architecture. However, they often overlook the explicit integration of contextual-information capture and the limitation of today's data for SysML v2 model generation. There remains a gap in using both SysML and MAS to capture, represent, and validate context-rich, sequential processes, and domain involvement for SysML system

architecture generation and verification. These gaps are found in a number of literature studies below:

- A study leverages LLMs to automate generation, modification, and querying SysML-based system architecture model. To improve LLM responses, the study suggested combining several methods, such as re-training, finetuning, retrieval augmented generation (RAG), and prompt engineering (few- and zero-shot) (Longshore, Bell, & Madachy, 2024). The authors show an increasing LLM ability to increase modeling efficiency and enhance the practice of MBSE. However, they didn't specifically touch the importance of context and imbalanced data in pre-training and post-training LLM processes.
- A study leverages AI4MBSE and MBSE4AI (Stefani, et al., 2024):
  - AI4MBSE: Simulation-enabled engineering and digital twins generate data and scenarios that update models, enabling continuous assurance, traceability, and refinement across the Continuous Integration/Development (CI/CD) pipeline of system/software development.
  - MBSE4AI: Using SysML to capture requirements, System-Under-Test (SUT) architecture, and Operational Design Domain (ODD), aligning with EASA's W-model with DevOps for V&V.
- A study found that top AI failure causes are misdefining the problem, models get optimized for the wrong metrics, or do not fit the business/mission workflow. It is suggested to focus on the mission-values problems, not the technology only (Ryseff, Bruhl, & Newberry, 2024). A multi-agent system (MAS) approach offers modularity to break down a large problem into smaller pieces (Sycara, 1998).
- Using SysML to build a context for Multi-Agent Systems (MAS) architecture (Sadik & Goerick, 2024) (Mour, Kenley, Davendralingam, & DeLaurentis, 2013) (Maheshwari, Raz, DeLaurentis, Murphy, & Kolawole, 2018) (Bouamra, Poisson, Yun, & Armetta, 2025) (Jin, Jin, Chen, & Wang, 2024).
  - Two studies using SysML and/or BPMN (Business Process Management Notation) modeling for Multi-Agent Systems Architecture (Mour, Kenley, Davendralingam, & DeLaurentis, 2013) (Sadik & Goerick, 2024). However, both of them focused on the application of multi-agent architecture for different mission-focused modeling and simulation (such as multi-robot system and multi-agent for defense systems). Neither of the studies focused on how to capture contextual information and sequential processes to handle different types of problems.
- Two studies by (Maheshwari, Raz, DeLaurentis, Murphy, & Kolawole, 2018) and (Maheshwari, Kenley, & DeLaurentis, Creating Executable Agent-Based Models Using SysML, 2015) demonstrated how to use SysML to model and execute agent-based architecture. Both studies focused on the application layers but not specifically on using SysML as the context for multi-agent models to generate a SysML system architecture model.
- A study by (Jin, Jin, Chen, & Wang, 2024) used LLM-based agents for requirements engineering (RE). However, the context of AI4MBSE and MBSE4AI is not only about requirements engineering, but also includes the entire system architecture domain of technical process (V-model) in SE lifecycle.

- A study by (Bouamra, Poisson, Yun, & Armetta, 2025) utilized the strength of next generation of SysML, SysML v2 which has richer semantics. The drawbacks of these multi-agent models for SysML v2 are limited domain data as this language is considered a new approach and a lack of contextual modeling to validate LLM outputs against systems engineering logical processes.

2. **Competency and process gaps:** In real-world practice, there are competency and process gaps in many organizations that are trying to adopt MBSE and SysML to capture their system architecture models. One of the main gaps is imbalanced competencies across the team and reluctance to use SysML tools. Therefore, a bridge between SysML modelers and non-modelers is crucial to ensure consistent information between them (Arifin, 2025). Moreover, to mitigate the risk of hallucination, any use of LLM for system model generation requires a human in the loop (HIL) to review, validate, and authorize the generated model artifacts. This additional post-generation process ensures transparency for users and compliance with the ISO 42001 standard regarding AI system operation and monitoring (FINOS AI Governance Framework, 2025).

Contributions of the Paper

No.	Finding Gaps	Expected Contributions
1	**Contextual and data gap**	• **Agent Context Capture and Traceability:** Use SysML as the contextual backbone (requirements, structures (systems/subsystems/components), functions, and parameters) so LLM outputs can be traced back to each agent's prompts/queries and configurations. This contextual information captured in SysML model will be the inputs of LLM when invoking the model generators. • **SysML-Guided LLM Model Generator:** A method where SysML metamodels and dependency relations constrain LLM prompts/queries and output formats, ensuring generated SysML requirements, structures, functions, parameters, etc. align with the modeling methodology and language.
2	**Process and governance gap**	• **Multi-Agent Systems with Agentic Orchestration:** Decompose the modeling methodology process into atomic processes which can be handled/assigned by each specialized agent. For example, agents which handle requirements, structures, functions, and parameters, communicating via the SysML multi-agent model controlled by an orchestrator agent. • **Document-to-Model Pipeline with Human-In-the-Loop (HIL) Review:** An automation pipeline that ingests unstructured/semi-structured documents (e.g., specification documents in PDF format), uses LLMs to extract suggested elements to spreadsheets, and synchronizes them into SysML for human review—bridging non-modelers and modelers (e.g., spreadsheet synchronization patterns highlighted for SysML modeling tools).

Proposed Solutions

This paper proposes a sample of Multi-Agent System Architecture by using SysML semantics. Figure 1 shows a SysML Block Definition Diagram (BDD) which captures contextual information and knowledge structures for a Multi-Agent System (MAS) designed to generate SysML models using an LLM. Here's a breakdown of its orchestrator and specialized agents:

1. **Supervisor Agent Concept** – "AI-Powered Model Generation Sample"
   - This is the core MAS orchestration block which represents the main AI agent ecosystem that coordinates the sub-specialized agents for SysML model generation. The purpose of this block is to act as the hub that connects input data, coordinates agents, and combines the outputs.
2. **Specialized Agents**

- The authors developed a generic LLM agent for model generation called "GenericElementPattern". This agent is reusable and defines how the LLM acquires context for SysML model element generation. It captures not only a SysML block's structural features, but also behavioral features, such as two reusable operations (i.e., WriteToExcel and ReadFromExcel).
- Each blue "ElementType" block represents a specialized agent responsible for a specific atomic process in the model generation workflow. Each of these blocks inherits the features from "GenericElementPattern" block and can be redefined depending on the specific problem to handle:
    - SOI Agent – Generates System of Interest (SOI) blocks based on the provided context.
    - Requirement Agent – Generates SysML requirement elements aligned to the SOI (System of Interest).
    - System Agent – Generates SysML blocks for systems/subsystems/components and their relationships to other elements, such as refinement back to requirements.
    - Function Agent – Generates functions allocated to systems/subsystems/components.
    - Parameter Agent – Generates parameters assigned to systems/subsystems/components.
- These agents operate sequentially, with red dashed "SysML dependency" indicating the process dependency flow between outputs and inputs of the specialized agents.
- The idea of using specialized agents is to ensure high scalability and maintainability for different use cases. For example, the authors defined a requirement metamodel that captures only the name and text. In other applications, an organization may want to map the LLM results to other requirement attributes, e.g., priority (mandatory/optional) or requirement types (functional, non-functional, safety, etc.). For this purpose, it is not necessary to modify the entire architecture, but only to adjust the requirement agent to align with the organizational need. This architecture is easier to adapt to different problems, ensuring it can be scaled and maintained easily.

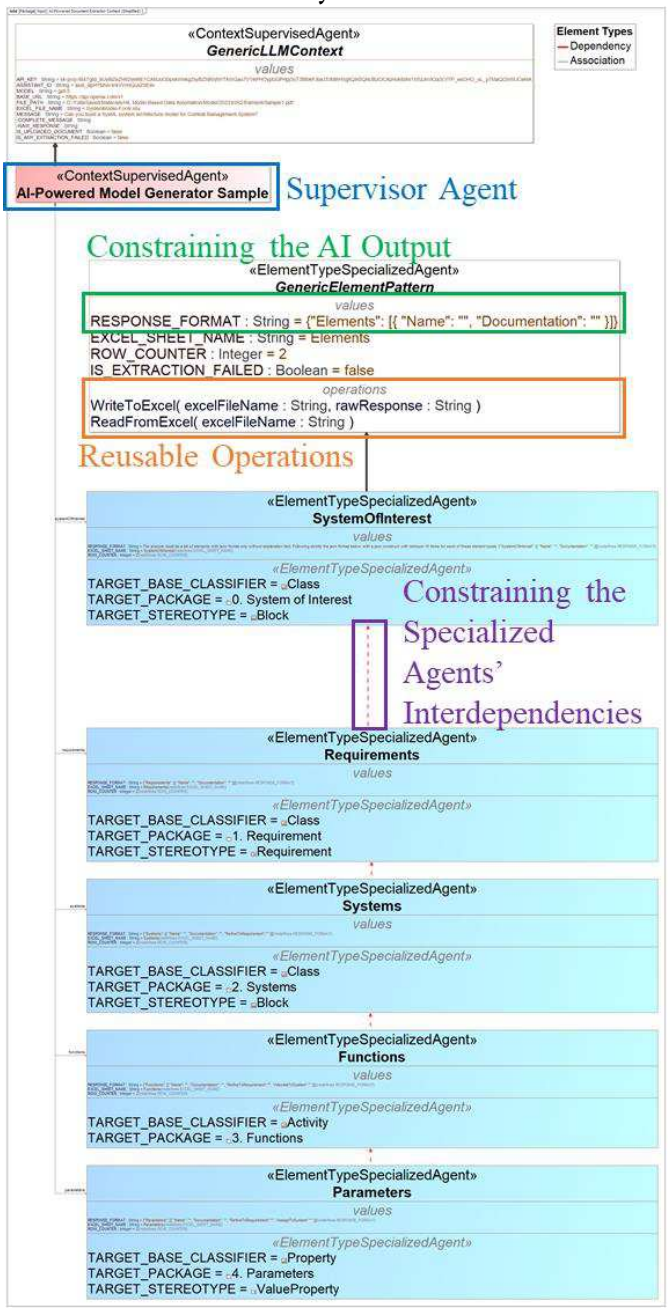

Figure 1. An Example of Multi-Agent System for SysML System Architecture Model Generation

Table 1 lists the essential input parameters and values needed to configure and execute the LLM model generator by using the context and processes captured in the SysML multi-agent model.

Input Parameters	Example of Value	Description
MODEL	gpt-5	The name of the LLM model to be used for processing the request.
BASE_URL	https://api.openai.com/v1	The endpoint URL for making API requests.
API_KEY	[YOUR_API_KEY]	The authentication key required to access the API.
ASSISTANT_ID	[YOUR_ASSISTANT_ID]	The unique identifier for a configured AI assistant. It's used to associate a specific set of instructions, tools, and model with a conversation thread, allowing for personalized and task-specific interactions.
FILE_PATH	.../.../ Sample1.pdf	The local or relative path to the input document file (i.e., PDF Specification Document).
EXCEL_FILE_NAME	SystemModel-ForAI.xlsx	The name of the spreadsheet file where results from LLM will be stored and retrieved.
MESSAGE	Can you build a SysML system architecture model for Combat Management System?	The main user query/prompt for the LLM model.
COMPLETE MESSAGE	Can you build a SysML system architecture model for Combat Management System? The answer must be a list of elements with json format only without explanation text. Following strictly the json format below with a json construct with a minimum of 10 items for each of these element types: {"SystemOfInterest": [{ "Name": "", "Documentation": "" }]} {"Requirements": [{ "Name": "", "Documentation": "" }]} {"Systems": [{ "Name": "", "Documentation": "", "RefineToRequirement":"" }]} {"Functions": [{ "Name": "", "Documentation": "", "RefineToRequirement":"", "AllocateToSystem":"" }]} {"Parameters": [{ "Name": "", "Documentation": "", "RefineToRequirement":"", "AssignToSystem":"" }]}	The combination of user's basic query and additional query constructed by each agent.
RAW_RESPONSE	{"SystemOfInterest": [{"Name": "Combat Management System - Surface Combatant", "Documentation": "Primary system of interest integrating sensors, weapons, and command functions for a surface combatant."}]} {"Requirements": [{"Name": "RQ-01 Situational Awareness", "Documentation": "The system shall provide a coherent, timely tactical picture to support decision-making."} ...]} {"Systems": [{"Name": "Sensor Integration Subsystem", "Documentation": "Interfaces with shipboard and offboard sensors; normalizes inputs.", "RefineToRequirement": "RQ-02 Sensor Fusion"} ...]} {"Functions": [{"Name": "Acquire Sensor Inputs", "Documentation": "Collect raw data from onboard and offboard sensors.", "RefineToRequirement": "RQ-02 Sensor Fusion", "AllocateToSystem": "Sensor Integration Subsystem"} ...]} {"Parameters": [{"Name": "Sensor Input Latency Budget (ms)", "Documentation": "Maximum allowable end-to-end latency for sensor ingestion.", "RefineToRequirement": "RQ-02 Sensor Fusion", "AssignToSystem": "Sensor Integration Subsystem"}]}	The unprocessed output returned by the LLM model. In this case, the format is JSON.
IS_UPLOADED_DOCUMENT	True	A flag indicating whether a document has been uploaded.
IS_ANY_EXTRACTION_FAILED	False	A flag indicating if any data extraction step failed. This flag will be true if the

		output formats do not follow the JSON standard, which causes failure in parsing them.

Table 1. An Example of Used Parameter Set for Multi-Agent System for LLM Model Generator

Figure 2 shows an example of using SysML behavioral diagrams (i.e., Activity Diagram (ACT)) to model a process automation workflow for generating a SysML system architecture model using an LLM, while also capturing context and process sequences. The diagram illustrates a multi-agentic orchestration to automate the following end-to-end processes:

1. Prepare contextual (input) parameters for the LLM, including the prompts/queries. The top-left action (process) prepares the input parameters, i.e., URLs, model ID, and constructed prompt, and an extraction control flag (to identify if there is any error during the automation).
2. Encapsulate and send the parameters into a message to the LLM via the OpenAI API. The purple box invokes "SendMessageWithFileAndResponseFormat" by passing the input parameters and receives "rawResponse" in a structured format (e.g., JSON). This is where the LLM generates architecture content (e.g., system of interest, requirements, systems, functions, and parameters).
3. Process and parse the LLM's output. Each specialized agent will handle a different scope of system architecture, according to the constraints and interdependencies between elements. This post-processing of the LLM response allows the automation to parse the results and write to Excel (green box). Using Excel as a medium enables human offline review, traceability, and a bridge between SysML and non-SysML system architects.
4. Read from Excel for subsequent model generation after validation by a human. The orange box action allows integration with SysML modeling tools without manual modeling.
5. After being validated, generate the system architecture model based on the extracted data.

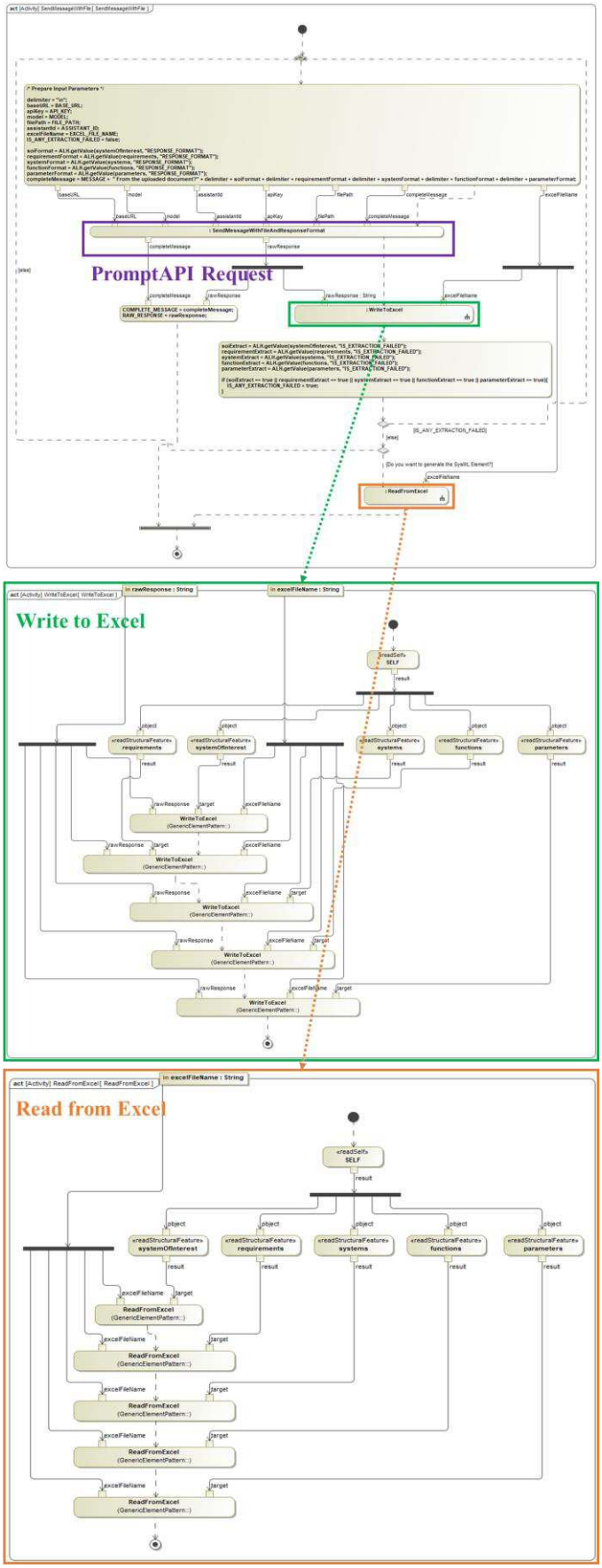

Figure 2. Process Automation for SysML System Architecture Model Generation

Figure 3 shows a SysML model library containing reusable functions for Excel integration in an automation workflow. Two main operations are defined: ReadFromExcel (to load data from a specified Excel file, sheet, and row) and WriteToExcel (to store raw LLM output into Excel with file, sheet, and row parameters). By standardizing these functions, the automation process can be reused across projects, improving speed, reducing manual errors, and ensuring consistent execution.

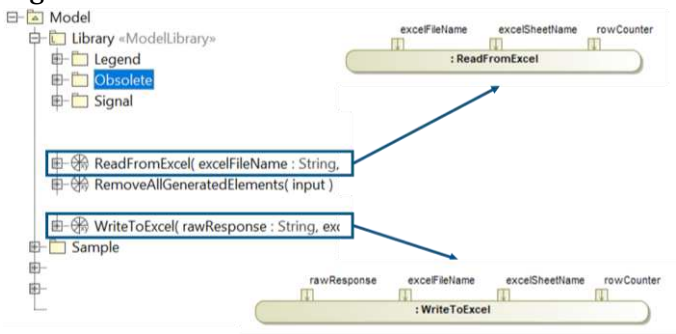

Figure 3. Reusable Behaviors/Functions for Specialized Agents

In summary, the holistic workflow shown in Figure 4 focuses on 3 pillars, i.e., Input, Process, and Output:

- **Input:** Unstructured specification documents (PDF/Word) are analyzed by LLM engines to extract all architectural elements defined in the agents' context, such as system of interest, requirements, systems/subsystems/components, functions, and parameters.
- **Process:** The extracted data are organized in an Excel spreadsheet according to the constrained mapping defined in the agents' context, enabling humans (non-modelers) to review and refine the information without using modeling tools directly.
- **Output:** The validated spreadsheet content is synchronized into a SysML modeling environment, where the automation generates the formal system architecture model with traceable requirements, structures, functions, and parameters.

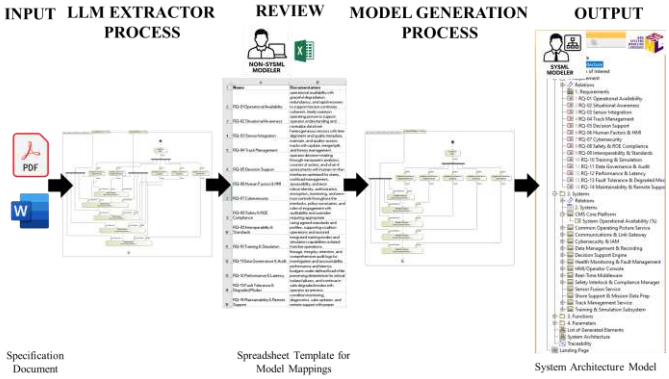

Figure 4. End-to-End Steps for SysML Model Generation with LLM Multi-Agent Systems

Scope of the Paper

The scope of this paper is to apply SysML as a context and process constraint for LLM-based model generation and develop a spreadsheet mapping template to bridge SysML and non-SysML system architects. The details, including assumptions and limitations, are as follows:

1. **SysML as an LLM context and process constraint for model generation:** Using SysML to model multi-agent system architecture, which will construct the context and capture the knowledge to construct the inputs of LLM prompts. At the same time, the SysML architecture also captures the processes and their interdependencies to constrain the sequences of LLM prompts.
2. **SysML and spreadsheet mapping template for model generation:** Building a template to bridge SysML modelers and non-modelers via a spreadsheet template.
3. **LLMs' "accuracy" is not measured:** This study focuses only on the workflow of how to utilize context and knowledge captured in SysML Multi-Agent Architecture for generating a system architecture model using LLM models. Therefore, this study will not cover a comparison study with various LLM models, engineered prompts, configurations, or any other optimization to measure the accuracy of the models. This can be covered in future work.
4. **Semantic "accuracy" is not measured**: The semantic accuracy, e.g., INCOSE requirement quality guideline (INCOSE, 2023), is not

evaluated in this study. However, in the future work section, the authors discuss some possible ways to include semantic checking as a guardrail on LLM implementation Rules (Marchiori, et al., 2024).

5. **Prompt Engineering for Agents**: The idea of having different agents for different SysML elements can be more powerful if the implementation goes deeper into the prompt engineering topic. For example, using different prompts for requirements, structures, behaviors, and parametrics. However, this study does not capture this specifically, which can be investigated in future work.
6. **Tools used for demonstration:** The modeling tools used in this demo are CATIA Magic Cyber Systems Engineer (MCSE) and Magic Model Analyst (MMA). MCSE is mainly used to capture system/software architecture models with SysML (Dassault Systèmes, 2025). MMA supports execution, animation, and debugging of executable SysML models, with process customization through action scripting and structured expressions (CATIA Magic, 2025). For the LLM engine, the authors use open platform models (ChatGPT 3.5, 4, and 5) to prove the concept. Other tools with SysML modeling and simulation features can also be used to replicate this use case.

Results

Figure 5 shows the final SysML system architecture model automatically generated by the LLM multi-agent system.

- **Top section:** A hierarchical SysML diagram linking four main element types—Requirements, Systems/Subsystems/Components, Functions, and Parameters—with dashed lines indicating dependencies and traceability (e.g., SysML refinement and allocation).
- **Bottom section:**
    - Requirement's list (e.g., RQ-01 Operational Availability, RQ-02 Situational Awareness).
    - Systems/Subsystems/Components' list (e.g., CMS Core Platform, Cybersecurity & IAM).
    - Functions' list (e.g., Authenticate & Authorize Users, Evaluate Threats).
    - Parameter's list (e.g., System Operational Availability %, Sensor Input Latency Budget).

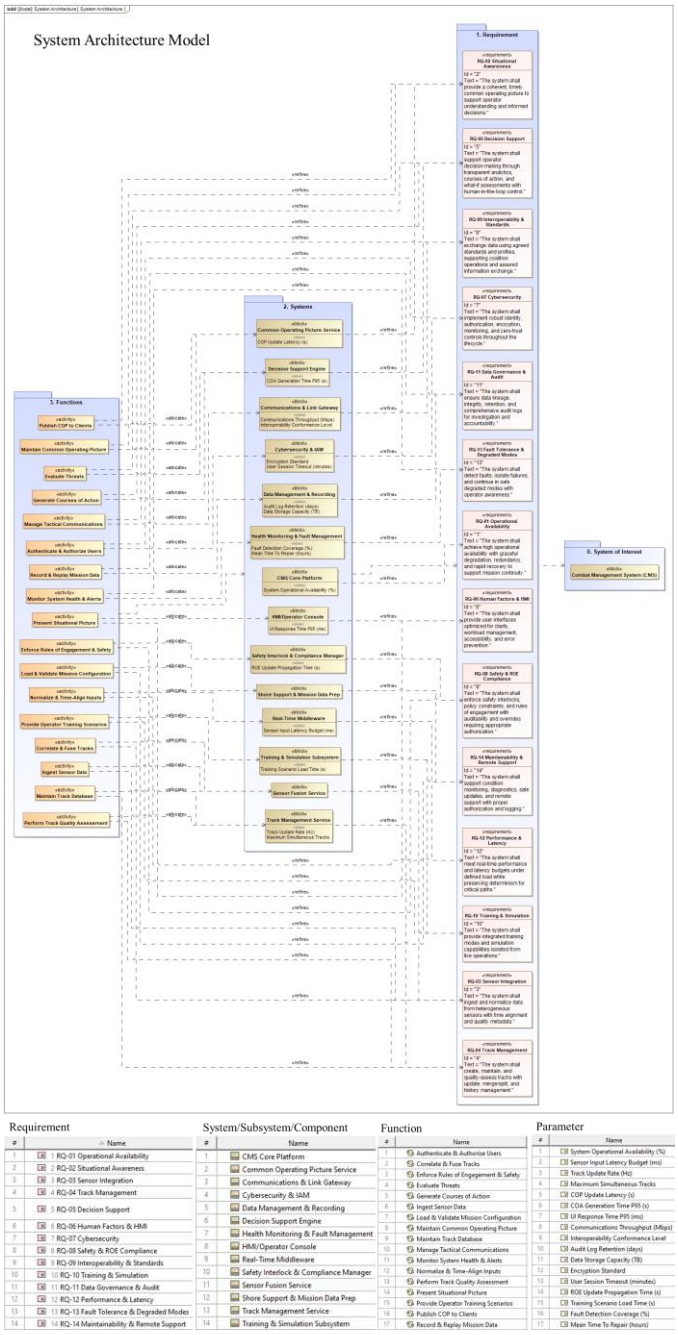

Figure 5. An Example of Generated SysML System Architecture Model for CMS

The matrix shown in **Figure 6** visualizes the relationships and traceability between all SysML elements generated by the LLM. Rows list the generated elements grouped as Requirements, Systems, and Functions. Columns represent the same

element groups, allowing cross-mapping of dependencies. Red arrows indicate Refine relationships (e.g., requirements refined by systems or functions). Blue arrows indicate Allocate relationships (e.g., functions allocated to systems).

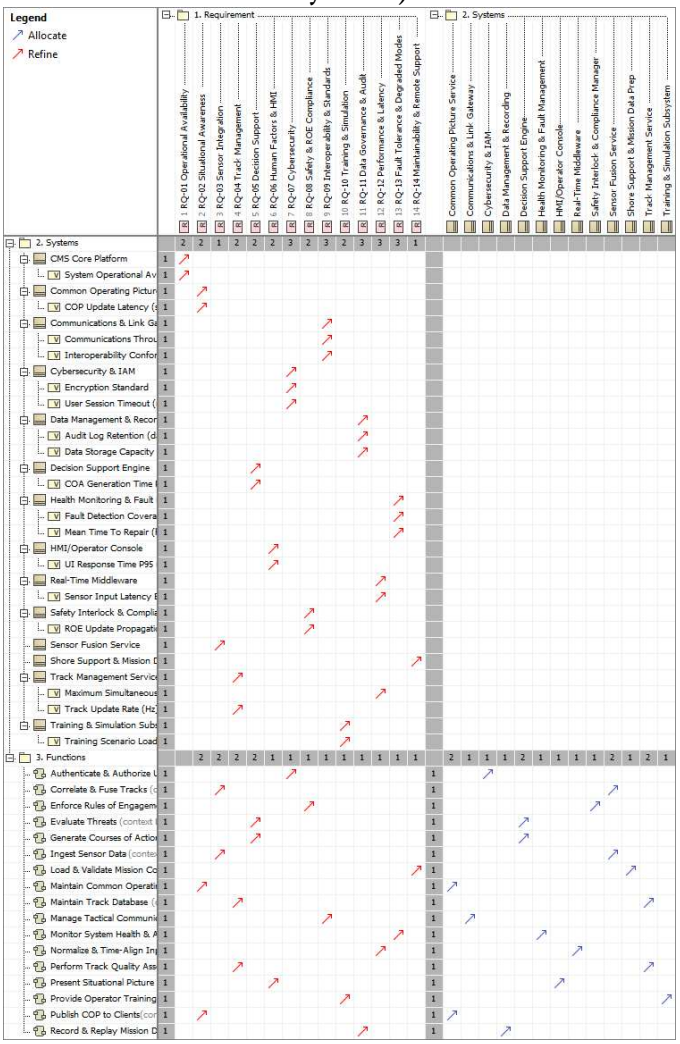

Figure 6. Traceability Among Model Elements Generated by LLM Multi-Agent System

Conclusion and Future Work

In conclusion, this study demonstrates a novel method to utilize SysML as a context and process constraint for LLM-based model generation by adopting a multi-agent system architecture and a spreadsheet mapping template to bridge SysML and non-SysML system architects. The proposed workflow captures the context, knowledge, constraints, and process sequences to generate a SysML system architecture model from unstructured/semi-structured specifications, ensuring traceability, structured outputs, and human-in-the-loop validation. This approach provides a replicable framework for any generative AI (e.g., LLM) adoption in mission-critical systems where precision, safety, and regulatory alignment are mandatory.

However, here are some limitations that can be the foundation for future work:

1. **Addressing LLMs' accuracy comparison:** The main limitation of this study concerns quantitative evaluation. Future work can include comparative analyses across multiple LLM engines, fine-tunings, and configuration settings.
2. **Semantic accuracy checks with guardrail:** The framework currently demonstrates how SysML can capture not only LLM outputs but also the context and process definition of the multi-agent architecture. However, for mission-critical systems, it is essential to assess semantic accuracy—for example, by validating SysML requirements using rule-based and/or ontology-based algorithms aligned with INCOSE requirement-quality guidelines. Similar methods can be applied to other SysML elements, such as values and units, functions, and properties.
3. **Explainability in AI (XAI) gap:** In the mission-critical domains, a lack of sufficient explainability in AI systems can limit human ability to make accountable decisions and conduct thorough assessments.
4. **Hybrid AI integration with multi-agent systems approach:** There's no silver bullet. Different problems need different tools. For a requirements-extraction agent, a hybrid works best—e.g., use NLP (or multiple LLMs) to extract requirements and an expert/rule-based system to check them.
5. **Impact on SysML v2:** The study is conducted on the SysML v1 metamodel. Implementation for SysML v2 is expected to be simpler, as manual string-to-element encoding/decoding is no longer required; LLMs can be trained or prompted to generate SysML v2 syntax directly.

References

Arifin, H. (2025, July 14). *Human-in-the-Loop AI in Model-Based Systems Engineering*

Turbocharges Defense Manufacturing. Retrieved August 6, 2025, from https://blog.3ds.com/brands/catia/ai-in-model-based-systems-engineering/

Bouamra, Y., Poisson, A., Yun, B., & Armetta, F. (2025). SysTemp: A Multi-Agent System for Template-Based Generation of SysML V2. arXiv - Cornell University.

Delligatti, L. (2014). *SysML Distilled: A Brief Guide to the Systems Modeling Language.* Addison-Wesley Professional.

Esposito, M., Palagiano, F., Lenarduzzi, V., & Taibi, D. (2025). On Large Language Models in Mission-Critical IT Governance: Are We Ready Yet? *arXiv - Cornell University*.

FAA. (2024). *Roadmap for Artificial Intelligence Safety*. Federal Aviation Administration (FAA).

FINOS AI Governance Framework. (2025). *Human Feedback Loop for AI Systems.* Retrieved August 13, 2025, from https://air-governance-framework.finos.org/mitigations/mi-11_human-feedback-loop-for-ai-systems.html

Friedenthal, S., Moore, A., & Steiner, R. (2015). *A Practical Guide to SysML: The System Modelling Language.* Elsevier Inc. Retrieved July 19, 2017

Graydon, M. S., & Lehman, S. M. (2025). *Examining Proposed Uses of LLMs to Produce or Assess Assurance Arguments.* NASA.

Hart, L. E. (2006). Introduction To Model-Based Systems Engineering (MBSE) and SysML. Lockheed Martin Corporation.

Hobbs, K. L., & Li, B. (2024). Safety, Trust, and Ethics Considerations for Human-AI Teaming in Aerospace Control. AIAA Scitech Forum.

INCOSE. (2023). *INCOSE Guide to Writing Requirements V4 – Summary Sheet.* INCOSE.

Jin, D., Jin, Z., Chen, X., & Wang, C. (2024). MARE: Multi-Agents Collaboration Framework for Requirements Engineering. arXiv - Cornell University.

Longshore, R., Bell, R., & Madachy, R. (2024). *Leveraging Generative AI to Build, Modify, and Query MBSE Models*. Naval Postgraduate School (NPS).

Maheshwari, A., Kenley, C. R., & DeLaurentis, D. A. (2015). Creating Executable Agent-Based Models Using SysML. Bellevue, WA: INCOSE.

Maheshwari, A., Raz, A., DeLaurentis, D. A., Murphy, A., & Kolawole, O. (2018). Integrating SysML and Agent-Based Modeling for Rapid Architecture Evaluation. *Insight, 21*(2), 47-51.

Marchiori, L. H., Oliveira, A. H., Reis, P. A., Júnior, F. S., Cavalcante, M. S., de Lima, J. V., & Soares, L. F. (2024). Impact Analysis of using Natural Language Processing and Large Language Model on Automated Correction of Systems Engineering Requirements. *INCOSE International Symposium.* INCOSE.

Mour, A., Kenley, C. R., Davendralingam, N., & DeLaurentis, D. (2013). Agent-Based Modeling for Systems of Systems. Purdue University.

Pierce, G. J., Nicoli, P. E., Hill, T. R., & Cornford, S. L. (2024). How the INCOSE Model-Based Capability Matrix Has Steered Model-Based Systems Engineering Transformation at NASA. Dublin, Ireland: INCOSE.

Ryseff, J., Bruhl, B. D., & Newberry, S. J. (2024). *The Root Causes of Failure for Artificial Intelligence Projects and How They Can Succeed.* RAND.

Sadik, A. R., & Goerick, C. (2024). Multi-Robot System Architecture Design in SysML and BPMN. *Advances in Science, Technology and Engineering Systems Journal (ASTESJ)*.

Sprockhoff, J., Lukic, B., Janson, V., Ahlbrecht, A., Durak, U., Gupta , S., & Krueger, T. (2023). Model-Based Systems Engineering for AI-Based Systems. AIAA Scitech Forum.

Stefani, T., Christensen, J. M., Hoemann, E., Girija, A. A., Köster, F., Krueger, T., & Hallerbach, S. (2024). Applying Model-Based System Engineering and DevOps on the Implementation of an AI-Based Collision Avoidance System. ICAS.

Sycara, K. P. (1998, June). Multiagent Systems. *AI Magazine*, pp. 79-92.

Biography

Ir. Habibi Husain Arifin, B.Eng, M.Sc. is an MBSE, SysML/UML, & CATIA Magic/Cameo Champion, HW/SW System Integrator, and a Research & Development professional, has over 15 years of experience as a Java, Python, C/C++/C#, JavaScript, and Assembly software engineer, MBSE consultant, and solution architect, contributing to various private and government projects. His expertise spans Digital Engineering Strategy, MBSE (SysML/UML, UAF/DoDAF/NAF), Document Management Systems (DMS), Data Dictionaries, Business Process Management (BPM), Enterprise Resource Planning (ERP)/Point of Sale (POS), Relational/Graph Database Architecture, and Business Intelligence. Currently, at Dassault Systèmes, he focuses on Digital Engineering, Model-Based Systems Engineering (MBSE), Artificial Intelligence (AI), and Product Lifecycle Management (PLM). Certified in OMG Systems Modeling Language (SysML), CESAMES Associate, INCOSE Certified Systems Engineering Professional (CSEP), and Graph Database (Neo4J). Helping to adopt Digital Engineering, MBSE, and/or PLM at major companies: Lockheed Martin, BAE Systems, KBR, Leidos, SAAB, Australian Missile Corporation (NIOA), Clarion, Hyundai, Nissan, etc. He actively publishes academic papers in AI/ML and Systems Engineering, which have been presented at IEEE, ACM, and INCOSE conferences. He has received awards from prestigious international innovation competitions such as ENVEX, ITEX, i-Inova, and NRIC.

Asst. Prof. Dr. Jirapun Daengdej was awarded a Ph.D. in Computer Science in the area of Artificial Intelligence from University of New England, Australia in 1999. He received IBM Faculty Award in 2009 and again in 2012 for his research, which dealt with problems of software development in real-world environment. He used to serve as Executive Technical Advisor for No Magic Asia Ltd., a company that developed software and system modeling tools for more than 15 years. He also used to have a role as CTO of Merlin's Solution International, a company that specializes in providing consulting for Digital Transformation and Design and Development of airport systems. He is the founder of Innocop Co., Ltd, a company that focuses mainly on designing and developing AI solutions for various companies such as financial institutes and insurance companies.

Muhammad Khalik Humar, B.Eng received a Bachelor of Engineering degree, majoring in Mechatronic Engineering from the University of New South Wales, Australia in 2023. He is currently pursuing a Master's

degree in Engineering, majoring in Robotics at the University of Technology Sydney, Australia. He has 5 years of experience in 3D designing and Programming languages (Python, C++, and C). His recent research interests include the development of vehicle attachments for traffic cones.

Saulius Pavalkis, PhD has 20 years of experience working on modeling solutions. Working as a chief MBSE solutions architect, consultant, trainer, PLM products integration manager. Former analyst in the R&D department's core MagicDraw team with 10 years of experience. Major expertise areas are MBSE, system simulation, requirements, data interchange, SysML, UML, traceability, modeling solutions, change and configuration management. Helping to adopt MBSE at major companies: Ford, Boeing, NASA, Orbital ATK, Draper, Raytheon, UTAS, BAH, MITRE, L3Harris. Experience working on projects (including European Union funded for different OMG standards integration) guiding team to create model-based prototypes and working solutions. Multiple professional certificates: OMG-Certified Systems Modelling Professional, OMG-Certified UML Professional, OMG-Certified Expert in BPM, and ITIL V3. Multiple research and practical articles in model-based design. Founder of "MBSE Execution" YouTube channel on model simulation and chief editor of a modeling community blog (blog.nomagic.com) dedicated to sharing practical model-based engineering experience. 40+ papers, tutorials, and presentations in top system engineering conferences: INCOSE IS, NDIA, NEFWS, WSRC, GLRC, NMWS. PhD in Software Engineering – model traceability, BS and MS in Telecommunication and Electronics Engineering (from Kaunas University of Technology). CATIA | No Magic, Inc. representative at INCOSE CAB. In 2018 he received a lifetime achievement / award in MBSE and Modeling from No Magic, Inc. "Cameo Award for Modeling, Simulation & Analysis Excellence."

Firas Mahmoud is the Asia Pacific Senior Technical Sales Manager at Dassault Systèmes, bringing 10 years of experience driving innovation and digital transformation across industries. Specializing in Data Science and Artificial Intelligence, Firas helps organizations unlock the full value of their data enabling smarter decision-making, operational excellence, and sustainable growth. Throughout his career, he has held diverse roles spanning Research & Development, the Center of Excellence, and the Sales Organization, giving him a 360° perspective on how advanced analytics and AI can be applied to real-world challenges. In his current leadership role, he empowers enterprises across Asia Pacific to transform insights into business impact by leveraging cutting-edge technologies.