NEXUS: A Multi-Agent Agentic Workflow for Valid SysML v2 Model Generation from Unstructured Industrial Documentation

Oscar Artur Bernd Berg [1 2] **Christoph Klaassen** [3]

Abstract

The increasing complexity of modern cyber-physical systems necessitates Model-Based Systems Engineering, yet practical adoption of SysML v2 remains hindered by high manual effort and steep linguistic learning curves. Existing AI-assisted approaches address only clean natural-language prompts or fragmented model fragments rather than the unstructured multi-page technical PDFs prevalent in industrial practice. This paper presents NEXUS (Neural EXtraction for Unified Systems engineering), a multi-agent agentic workflow for semi-automated generation of formally valid SysML v2 models directly from unstructured technical documentation. The architecture comprises three specialized agents: an *Extraction Agent* that transforms legacy PDFs into structured JSON representations; a *Building Agent* that incrementally constructs SysML v2 code across three compiler-in-the-loop phases; and a *Verification Agent* that ensures semantic and structural integrity. Evaluation on eight industrial-grade documents shows that NEXUS achieves 100 % syntactic validity across all configurations (vs. 38 % for a one-shot baseline) while reducing hallucination by 2.4×. The approach generalizes across open-weight and proprietary backbone models with human-in-the-loop oversight at every stage.

1. Introduction

Modern engineered systems involve tightly coupled software, hardware, and cyber-physical components of increasing complexity. *Model-Based Systems Engineering* (MBSE) addresses this by shifting development from document-centric practices toward model-centric representations of system architecture, behavior, and requirements (Vaicenavičius et al., 2025). However, realizing this shift presupposes that the engineering knowledge currently bound to legacy documentation can be transferred into model-centric form. In industrial practice, requirements, specifications, and design rationale still reside predominantly in unstructured, multi-page technical documents, so the move toward MBSE hinges on transforming these document-bound artifacts into machine-readable data structures (Johnson & Williams, 2025). Such a representation is doubly advantageous: beyond enabling formal consistency checking, traceability, and automated verification, a machine-readable format renders system knowledge directly accessible to data-driven and AI-based methods, which can in turn assist, and increasingly automate, the modeling process itself. However, capturing these benefits requires a modeling language whose formal rigor supports both human comprehension and machine processing (Zhang et al.).

The *Systems Modeling Language version 2 (SysML v2)*, developed by the Object Management Group (OMG), improves upon SysML v1 with a more precise semantic foundation, better modularity, and both graphical and textual modeling. Built on the *Kernel Modeling Language (KerML)*, it enables rigorous definition of system structures, behaviors, and constraints (Zimmermann et al., 2024). However, practical adoption remains challenging: creating comprehensive models requires deep expertise in both systems engineering and formal syntax, and translating unstructured documentation into structured models is manual, time-consuming, and error-prone (Johnson & Williams, 2025).

Recent work has explored integrating LLMs into MBSE (Hanke et al., 2025), including agent-based SysML v2 generation (Cibrián et al., 2025) and LLM-assisted semantic alignment (Li et al., 2025). These approaches face limitations in reliability, traceability, and structured generation (Gesmann et al., 2026; Rafique et al., 2025), motivating the need for workflows combining structured extraction, automated construction, and systematic verification (Shi et al., 2025).

This paper proposes an agentic workflow for semi-automated SysML v2 generation from technical documentation, decomposing the process into specialized agents for

[1]TU Wien, Vienna, Austria [2]Mid Sweden University, Sundsvall, Sweden [3]AFRY Management Consulting Austria GmbH. Correspondence to: Oscar Artur Bernd Berg <oscar.berg@miun.se>.

Accepted to the ICML 2026 Workshop on Statistical Frameworks for Uncertainty in Agentic Systems, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

extraction, construction, and verification. The main contributions are: (i) a multi-agent architecture with human-in-the-loop oversight at every stage; (ii) an incremental three-phase building strategy with per-phase compiler-in-the-loop validation achieving 100 % syntactic validity; (iii) a structured JSON intermediate representation reducing hallucination by 2.4×; and (iv) empirical evidence of architectural generality across backbone models, including architecture-controlled comparisons isolating workflow contributions from model capability.

2. Related Work

Recent LLM advances have opened new possibilities for automating MBSE through SysML v2 generation. The four most relevant 2025 studies reveal consistent limitations in input handling, architectural transparency, validation depth, and behavioral completeness.

Bouamra et al. present SysTemp, a multi-agent template-driven framework converting natural-language specifications into SysML v2 code (Bouamra et al., 2025). The Jinja2 templates mitigate LLM difficulties with the sparse SysML v2 corpus (fewer than 150 public examples), but the method remains template-bound and produces models without behavioral state machines.

Cibrián et al. develop an industrial agent-based system processing natural-language prompts through a ReAct-style loop with RAG over 92 expert-annotated examples (Cibrián et al., 2025). While effective for curated prompts, RAG restricts generalization and the pipeline operates exclusively on clean typed text.

Li et al. apply prompt-driven LLM iterations to integrate fragmented SysML v2 models (Li et al., 2025) through a seven-stage alignment process with mandatory user confirmation, but do not address generation from raw documentation.

Hendricks and Cicirello build a knowledge graph from unstructured text, render SysML Block Definition Diagrams in PlantUML, and generate Python simulation code (Hendricks & Cicirello, 2025). Validation uses diagram similarity and simulation accuracy, but the output bypasses native SysML v2 textual modeling.

Unlike these approaches, which operate on concise prompts or curated specifications, NEXUS processes complete unstructured technical PDFs spanning multiple pages. Direct empirical comparison is infeasible: they use fundamentally different input modalities, none provide public implementations, and their evaluation corpora do not apply to the PDF-to-model task. Table 1 summarizes key characteristics.

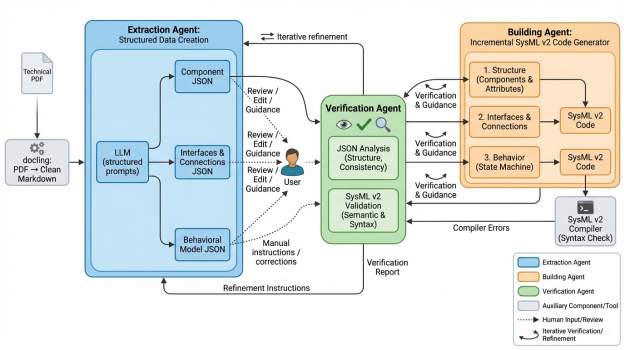

Figure 1. Overview of the NEXUS multi-agent architecture.

3. Methodology

3.1. Overview

The workflow comprises three specialized agents that collaboratively transform technical documentation into structured SysML v2 models (Figure 1). The *Extraction Agent* converts PDF content into structured JSON representations of components, interfaces, and behavior. The *Building Agent* incrementally generates SysML v2 code in three compiler-validated phases. The *Verification Agent* performs semantic validation at each stage. Human supervision is integrated between all stages: generated JSON structures and SysML v2 code can be reviewed, edited, and supplemented with corrective instructions before proceeding, enabling domain experts to incorporate knowledge not explicitly captured in the source documentation.

3.2. Extraction Agent

The Extraction Agent converts unstructured PDF content into machine-readable representations. The document is first processed using *docling*, which converts the PDF into clean Markdown while preserving semantic structure.

An LLM then performs structured extraction using chain-of-thought prompting in three steps, each producing a JSON representation conforming to a predefined schema: (1) system components with names, structural properties, and parameters; (2) interfaces and connections with port definitions, endpoints, communication media, and directionality; and (3) a behavioral model of states, actions, and transitions. This decomposition is motivated by the observation that extracting all aspects simultaneously degrades accuracy, as the model must attend to structurally heterogeneous information in a single pass. Each JSON schema enforces naming conventions and structural constraints facilitating downstream code generation. These artifacts provide an editable, human-readable checkpoint where domain experts can correct or augment extracted information.

Table 1. Comparison of AI-assisted SysML v2 generation approaches

Feature	Ours	SysTemp (Bouamra et al., 2025)	Cibrián et al. (Cibrián et al., 2025)	Li et al. (Li et al., 2025)	Hendricks et al. (Hendricks & Cicirello, 2025)
Input	Unstructured technical PDFs (Docling→Markdown)	NL specifications	Industrial NL requirements	Fragmented SysML v2 models	Unstructured NL corpus + system doc
Method	JSON extraction + 3-phase incremental build	Template-based multi-agent (Jinja2)	RAG + iterative repair	Prompt-driven semantic alignment	NLP + KG → PlantUML → Python simulation
Validation	Per-phase SysML v2 compiler + LLM report	Java grammar parser	ANTLR grammar	LLM consistency + user verification	Diagram similarity + simulation accuracy
Human Interaction	Editable JSON/code at every stage	Minimal (fully automated)	Minimal (RAG curation)	User confirmation per stage	Heavy (corpus, templates, refinement)
Native SysML v2 + Behavior	Yes (structure, interfaces, state machine)	Yes (textual, no behavior)	Yes (valid models)	Yes (aligned models)	No (PlantUML + Python)

3.3. Building Agent

The Building Agent translates extracted JSON representations into valid SysML v2 code while adhering to modeling rules defined in the system prompt. The JSON serves as the single source of truth for system structure and behavior.

Construction proceeds incrementally in three phases. Phase 1 generates a structural skeleton with component definitions and attributes, establishing all hierarchies before cross-cutting concerns. Phase 2 extends the skeleton with interface definitions, ports, and connections from the interface JSON and connection matrix, ensuring bindings reference only structurally defined components. Phase 3 generates state machines representing dynamic behavior, referencing elements established in preceding phases.

Each phase uses a minimal one-shot example in the system prompt. After each phase, the agent invokes a SysML v2 compiler to verify syntactic correctness; feedback is forwarded to verification.

3.4. Verification Agent

The Verification Agent identifies inconsistencies, omissions, and modeling errors before the workflow proceeds.

During extraction, it evaluates JSON completeness and consistency, checking for dangling interface references (ports referencing undefined components), orphaned components (without interface connectivity), and missing behavioral coverage (components with documented operational modes lacking state machine representations). Results feed into subsequent extraction runs.

During construction, it receives generated SysML v2 code and compiler error messages, producing a structured diagnostic report. When building is re-executed, the report and compiler feedback guide refinement. Domain experts can review all artifacts and apply manual corrections at every stage.

4. Results

4.1. Evaluation Methodology

Dataset. Evaluation uses a curated corpus of eight industrial-grade technical PDFs: one real-world vendor specification for a commercial hydrogen fuel-cell generator (S01), and seven domain-realistic synthesized documents spanning cyber-physical domains (S02–S08: micro-logistics factory, vertical farming station, water desalination module, smart traffic-light controller, EV battery management unit, robotic assembly cell, precision CNC machine). No public corpus of industrial SysML-ready PDFs exists, and genuine vendor specifications are typically under NDA; the synthesized documents reflect realistic structural diversity while enabling reproducible evaluation. This is consistent with evaluation strategies in the literature: Cibrián et al. (Cibrián et al., 2025) evaluate on 20 authored prompts, Bouamra et al. (Bouamra et al., 2025) use five crafted scenarios, and Li et al. (Li et al., 2025) operate on curated fragments. To our knowledge, NEXUS is the only approach including evaluation on a genuine vendor specification.[1]

Baseline and ablations. The full NEXUS pipeline is evaluated against:

1. Direct generation (Claude Opus 4.6): one-shot SysML v2 generation with a system prompt referencing the official specification, augmented with up to

[1]A video demonstration, prompt templates, JSON schemas, and a subset of the evaluation corpus are available at https://github.com/sysagentix-ops/NEXUS-Demo.

five syntax-error-feedback iterations in fully isolated sessions.

2. NEXUS without Verification Agent.
3. NEXUS without JSON intermediate: the Building Agent operates directly on Markdown-converted PDF.

All configurations use identical Markdown input from Docling preprocessing.

Evaluation metrics. Performance is measured on a 0–100 scale across three dimensions:

- Syntactic validity (s_{syn}): 100 if the model compiles error-free; otherwise LLM-judge-assigned based on error severity. All NEXUS configurations achieve compiler-verified $s_{\text{syn}} = 100$.
- Semantic coverage (s_{sem}): judge-assigned completeness and accuracy of captured components, interfaces, and behavior.
- Hallucination (s_{hal}): judge-assigned measure of ungrounded elements (lower is better).

Evaluation protocol. Semantic scores are assigned by Gemini 3.1 Pro as an independent judge using a fixed zero-shot prompt with the original PDF as reference. Each configuration is repeated five times under fixed temperature; mean$\pm$std are reported.

4.2. Main Comparison

Table 2 reports results across all eight systems. NEXUS achieves perfect syntactic validity ($s_{\text{syn}} = 100$) with zero variance, while the direct baseline averages only 38.4 ± 5.14 despite five compiler-feedback iterations, confirming that incremental phased construction is necessary for reliable generation from complex multi-page PDFs.

NEXUS achieves $s_{\text{sem}} = 81.3 \pm 5.45$, slightly below the baseline's 85.5 ± 3.60. The higher baseline coverage reflects unconstrained generation introducing plausible but undocumented elements, consistent with its substantially higher hallucination (66.0 ± 10.36 vs. 27.8 ± 5.06). In industrial MBSE, where traceability to source documentation is a primary correctness criterion, this trade-off favors NEXUS.

4.3. Ablation Study

We conduct a controlled ablation on the hydrogen fuel-cell generator specification S01 (Table 3). Removing the Verification Agent drops s_{syn} from 100 to 80 ± 4.5. Semantic coverage and hallucination are only slightly affected (s_{sem}: $70 \pm 4.2 \rightarrow 65 \pm 3.8$; s_{hal}: $35 \pm 2.8 \rightarrow 34 \pm 2.9$), confirming the Verification Agent primarily enforces structural consistency rather than grounding.

Bypassing the JSON intermediate causes the largest degradation: s_{sem} drops to 55 ± 4.0 and s_{hal} rises to 62 ± 5.6, approaching the direct baseline. Without grounding to document-derived elements, the Building Agent generates many unsupported elements, confirming the intermediate representation as the primary grounding mechanism.

4.4. Backbone Model Comparison

To assess whether gains stem from architecture rather than the underlying model, NEXUS is instantiated with three backbones (Table 4). DeepSeek-V3.2 is the default as the only fully open-weight model, enabling local deployment without API dependencies, which is critical for data-residency, IP protection, or air-gapped environments.

All configurations achieve $s_{\text{syn}} = 100$ and substantially lower hallucination than direct baselines. Notably, DeepSeek-V3.2 in direct mode achieves only 29 ± 5.4 syntactic validity yet reaches $s_{\text{syn}} = 100$ within NEXUS, demonstrating that the architecture compensates for weaker standalone capability. Claude Sonnet 3.7 and GPT o4-mini provide slightly higher semantic coverage and lower hallucination, consistent with stronger instruction-following, but differences remain within one to two standard deviations. Table 5 provides an architecture-controlled comparison isolating gains attributable to the workflow.

5. Discussion

The results confirm that NEXUS consistently produces formally valid SysML v2 models from unstructured industrial PDFs, whereas direct one-shot generation fails to do so.

Syntactic validity as a hard requirement. Perfect syntactic validity is a functional prerequisite: non-compiling models cannot be imported into modeling tools, analyzed, or exchanged across teams. The direct baseline reaches only 38.4 on average despite five compiler-feedback rounds. Incremental three-phase construction with per-phase compiler validation decomposes a globally complex task into locally verifiable steps. The $s_{\text{syn}} = 100$ across all NEXUS configurations is verified by the deterministic SysML v2 compiler, providing ground-truth confirmation.

The grounding role of the JSON intermediate. Ablation identifies the structured JSON intermediate as the most impactful component for hallucination suppression: its removal increases s_{hal} from 35 ± 2.8 to 62 ± 5.6, approaching the baseline. Without explicit grounding, the model fills gaps from parametric knowledge, producing plausible but unsupported content that may propagate into downstream analysis.

The Verification Agent as a structural consistency enforcer. The Verification Agent yields moderate but targeted

Table 2. NEXUS vs. direct one-shot baseline across eight systems (mean±std, n=5). s_{syn}: syntactic validity; s_{sem}: semantic coverage; s_{hal}: hallucination (lower is better). Scale: 0–100.

Configuration	Metric	S01	S02	S03	S04	S05	S06	S07	S08	Avg
Direct Gen (Opus 4.6)	s_{syn}	38±2.7	35±2.9	37±4.0	42±6.1	45±3.5	36±3.6	34±3.2	40±3.0	38.4±5.14
	s_{sem}	86±3.9	90±2.6	85±1.9	84±3.8	80±4.2	85±3.3	88±2.5	86±4.4	85.5±3.60
	s_{hal}	83±3.5	60±5.0	67±3.9	50±5.5	58±4.3	73±3.6	72±4.0	65±2.9	66.0±10.36
NEXUS (Ours)	s_{syn}	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0	100±0
	s_{sem}	70±4.2	85±2.8	83±2.6	87±2.6	82±4.0	78±3.5	83±4.5	82±1.8	81.3±5.45
	s_{hal}	35±2.8	32±3.6	22±4.0	29±4.6	23±2.9	28±3.9	29±4.3	24±2.7	27.8±5.06

Table 3. Ablation on the H2 fuel-cell specification S01 (mean±std, n=5). Metrics as in Table 2.

Configuration	s_{syn}	s_{sem}	s_{hal}
NEXUS (full)	100±0	70±4.2	35±2.8
w/o Verification Agent	80±4.5	65±3.8	34±2.9
w/o JSON intermediate	63±5.6	55±4.0	62±5.6
Direct Gen (Opus 4.6)	38±2.7	86±3.9	83±3.5

Table 4. Backbone comparison within NEXUS (mean±std, eight systems, n=5). Metrics as in Table 2.

Backbone	s_{syn}	s_{sem}	s_{hal}
DeepSeek-V3.2 (default)	100±0	81.3±5.45	27.8±5.06
Claude Sonnet 3.7	100±0	85.5±3.7	24.2±2.6
GPT o4-mini	100±0	84.2±3.5	22.8±4.2

improvement in syntactic validity, especially for structurally complex systems. Its limited impact on semantic coverage and hallucination confirms its role as a structural consistency enforcer complementing the JSON grounding function.

Semantic coverage and the precision–recall trade-off. The baseline's higher semantic coverage (85.5 ± 3.60 vs. 81.3 ± 5.45) reflects a precision–recall trade-off. Direct generation achieves higher recall by including plausible elements from parametric knowledge, but its hallucination (66.0 ± 10.36) indicates a large fraction of ungrounded content. NEXUS prioritizes precision by constraining generation to document-derived elements. In safety-critical domains where every model element must be traceable for certification and audit, precision is the dominant criterion.

Architectural generality. All backbones achieve identical syntactic validity. Differences in semantic coverage and hallucination remain within one to two standard deviations, confirming architecture-driven rather than model-specific gains. The architecture-controlled comparison (Table 5) further isolates this, showing strictly superior validity and reduced hallucination when the same backbone is used within NEXUS. This independence supports deployment under data-residency or confidentiality constraints.

Table 5. Architecture-controlled comparison: NEXUS vs. direct generation with identical backbones (mean±std, eight systems, n=5). Direct generation uses up to five compiler-feedback iterations. Metrics as in Table 2.

Backbone	Mode	s_{syn}	s_{sem}	s_{hal}
DeepSeek-V3.2	NEXUS	100±0	81.3±5.45	27.8±5.06
	Direct	29±5.4	74±3.8	64±3.0
Claude Opus 4.6	NEXUS	100±0	85±5.4	25±4.6
	Direct	38.4±5.14	85.5±3.60	66.0±10.36

Practical deployment. The NEXUS pipeline with locally hosted DeepSeek-V3.2 completes in approximately 81 % of the wall-clock time of direct generation with Claude Opus 4.6 extended thinking. The entire pipeline can be deployed on-premises without external cloud dependencies.

Expert assessment. The author team assessed NEXUS-generated models as substantially more useful initial prototypes, exhibiting coherent structure, consistent interfaces, and well-formed state machines. Directly generated models frequently contained syntax errors preventing tool import.

Limitations and future directions. The LLM-as-a-judge protocol for s_{sem} and s_{hal} provides a silver standard; however, $s_{\text{syn}} = 100$ is compiler-verified. The pipeline processes each PDF as a single Markdown artifact within a fixed context window, limiting scalability to very large documents. Future work should incorporate retrieval-augmented extraction with a structured knowledge graph for selective cross-section access.

6. Conclusion

This paper introduced NEXUS, a multi-agent workflow for semi-automated generation of formally valid SysML v2 models from unstructured industrial PDFs. By combining structured extraction, incremental three-phase construction, and dedicated verification, the approach addresses reliability and hallucination challenges in direct LLM-based generation. Evaluation across eight industrial documents shows that NEXUS achieves 100 % compiler-verified syntactic validity, reduces hallucination by 2.4× relative to a

one-shot baseline, and generalizes across diverse backbone models. Architecture-controlled comparisons confirm that gains are attributable to the workflow design rather than individual model capability. Expert evaluation corroborates that NEXUS-generated models are substantially more useful as initial prototypes for industrial MBSE workflows.

Software and Data

A video demonstration, prompt templates, JSON schemas, and a subset of the evaluation corpus are available at https://github.com/sysagentix-ops/NEXUS-Demo.

Acknowledgements

The NEXUS pipeline uses large language models (DeepSeek-V3.2, Claude Sonnet 3.7, GPT o4-mini) as backbone agents for structured extraction, SysML v2 code generation, and semantic verification, as described in Section 3. Gemini 3.1 Pro was used as an independent evaluation judge. During manuscript preparation, an AI writing assistant was used for grammar and style refinement; all technical content, experimental design, and analysis were conducted by the authors.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning applied to Model-Based Systems Engineering. The proposed system assists engineers in generating formal system models from technical documentation, potentially reducing manual modeling effort and improving consistency. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bouamra, Y., Yun, B., Poisson, A., and Armetta, F. Systemp: A multi-agent system for template-based generation of sysml v2, 2025. URL https://arxiv.org/abs/2506.21608.

Cibrián, E., Olivert-Iserte, J., Llorens, J., and Álvarez Rodríguez, J. M. An agent-based approach for the automatic generation of valid sysmlv2 models in industrial contexts. *Comput. Ind.*, 172(C), December 2025. ISSN 0166-3615. doi: 10.1016/j.compind.2025.104350. URL https://doi.org/10.1016/j.compind.2025.104350.

Gesmann, L., Fischer, M., Lauff, A., Völk, T., Düser, T., and Albers, A. Challenges in the Development of Cyber-Physical Systems – Insights from Research and Industrial Practice. *Procedia Computer Science*, 277:1963–1972, January 2026. ISSN 1877-0509. doi: 10.1016/j.procs.2026.02.234.

Hanke, F., Bita, I. M., von Heißen, O., Julian, W., Aschot, H., and Roman, D. AI-augmented systems engineering: Conceptual application of retrieval-augmented generation for model-based systems engineering graph. *Proceedings of the Design Society*, 5:439–448, August 2025. ISSN 2732-527X. doi: 10.1017/pds.2025.10058.

Hendricks, M. A. and Cicirello, A. Text to model via sysml: Automated generation of dynamical system computational models from unstructured natural language text via enhanced system modeling language diagrams, 2025. URL https://arxiv.org/abs/2507.06803.

Johnson, T. and Williams, A. Automated Legacy Documentation to SysML Conversion. *INCOSE International Symposium*, 35(1):1544–1563, 2025. ISSN 2334-5837. doi: 10.1002/iis2.70064.

Li, Z., Husung, S., and Wang, H. Llm-assisted semantic alignment and integration in collaborative model-based systems engineering using sysml v2. In *2025 IEEE International Symposium on Systems Engineering (ISSE)*, pp. 1–8. IEEE, October 2025. doi: 10.1109/isse65546.2025.11369983. URL http://dx.doi.org/10.1109/ISSE65546.2025.11369983.

Rafique, K. A., Shah, S., Dalecke, Š., and Grimm, C. Enhancing Model-Based Systems Engineering with Large Language Models. *INCOSE International Symposium*, 35(1):1523–1543, 2025. ISSN 2334-5837. doi: 10.1002/iis2.70067.

Shi, Y., Wang, M., Cao, Y., Lai, H., Lan, J., Han, X., Wang, Y., Geng, J., Li, Z., Xia, Z., Chen, X., Li, C., Xu, J., Duan, W., and Zhu, Y. Aime: Towards Fully-Autonomous Multi-Agent Framework, July 2025.

Vaicenavičius, J., Wiklund, T., Kavolis, D., Draukšas, S., Kalkauskas, A., and Vaicenavičius, R. SysIDE: SysML v2 textual editing and analysis system: Overview and applications. *CEAS Space Journal*, February 2025. ISSN 1868-2510. doi: 10.1007/s12567-025-00595-x.

Zhang, M., Li, Y., and Yue, T. Uncertainty Modeling for SysML v2. URL http://arxiv.org/abs/2602.21641.

Zimmermann, T. C., Manoury, M., and Lindow, K. SysML v2 for Automated Co-simulation from Systems Architecture Models. July 2024. doi: 10.1007/978-3-031-62554-1_4.