A Dataset for SysML v2 as an LLM-Oriented Workflow Language

Yassine El Mouhi[1,*,†], Théo Le Calvar[1,†] and Massimo Tisi[1,†]

[1]IMT Atlantique, LS2N (UMR CNRS 6004), F-44307 Nantes, France

Abstract

Workflows are used in industry to organize LLM applications. Existing workflow representations are not well suited for LLM-based generation and reasoning. In this paper, we define requirements for an LLM-oriented workflow language. We propose SysML v2 action flows as a language satisfying these requirements. To support quantitative research in this direction, we build a public dataset of nearly 75k SysML v2 action flows by transforming existing datasets of workflow graphs, BPMN models, and standard operating procedures. The resulting dataset covers workflows with diverse complexity, providing a useful basis for future work on workflow analysis and generation with LLMs.

Keywords

SysML v2, LLM, Dataset, Workflow

1. Introduction

In LLM-based applications, human-made workflows are used to structure complex tasks as chains or graphs of computation steps, each one possibly including an LLM call. Such decomposition has been shown to improve the reliability and quality of outputs compared to single-step calls to LLMs [1, 2]. With LLM-based systems becoming more and more autonomous, we also need LLMs to analyze, generate or optimize such workflows.

Typically, popular LLM application frameworks like LangGraph[1] allow the user to programmatically define workflows as graphs of objects in memory. In related work, explicit formal models (hierarchical state machines) have been proposed for LLM workflows [2]. Standard operating procedures (SOPs) are also used in some LLM frameworks [3], mirroring human task decomposition.

In this paper we argue that existing formalisms do not fulfill all the requirements for a workflow language for LLMs, especially when LLMs themselves need to analyze or generate the workflow. We instead propose the use of action flows from SysML v2 as a formal workflow language that is well-suited for LLM-based generation.

To enable quantitative research on the application of SysML v2 to this domain, we construct a large-scale dataset of SysML v2 action flows, and make it publicly available on Zenodo.[2] We build this dataset by transforming and merging existing benchmarks and modeling datasets for workflows expressed as directed acyclic graphs (DAG) [4], BPMN models [5] and SOPs [6]. The code that performs the construction is also made available on GitHub.[3]

CEUR-WS.org/Vol-4338/LLM4SE-short1.pdf

Joint Proceedings of the STAF 2026 Workshops: AgileMDE, GCM, ICMM, LLM4SE, TTC. Rennes, France, June 29- July 3, 2026

*Corresponding author.

†These authors contributed equally.

yassine.elmouhi@imt-atlantique.fr (Y. El Mouhi); theo.le-calvar@imt-atlantique.fr (T. Le Calvar); massimo.tisi@imt-atlantique.fr (M. Tisi)

0009-0001-2736-2330 (Y. El Mouhi); 0000-0003-2273-2053 (T. Le Calvar); 0000-0001-7891-9138 (M. Tisi)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]https://www.langchain.com/langgraph

[2]https://zenodo.org/records/19346990

[3]https://github.com/ElmouhiYassine/sysmlv2-workflow-construction

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-26

2. SysML v2 as an LLM workflow language

We argue that a LLM-oriented workflow language should be expressive enough, while remaining interpretable by execution engines and LLMs. We define three requirements:

Expressiveness: The language should provide sufficient constructs to represent the essential aspects of LLM workflows. In particular, it should support: 1) **Actions**, to capture operational steps within a workflow, provided with a natural language description; 2) **Conditionals**, to express decision points and branching; 3) **Cycles**, to represent repeated activities; 4) **Data Flows**, to model data exchanges between inputs and outputs of actions; 5) **Parallel execution**, to express a flexible execution order among actions.

Textual notation: The language should be associated with a native textual notation, suitable for LLM processing. The notation should be popular enough to be sufficiently included in the training dataset of frontier LLMs. In alternative, the notation should be similar to another well-known notation, so that existing LLMs can be adapted with limited cost. Since LLMs have been found to struggle with generating graph-based structures [4], it is preferable to rely on different notations, e.g. as imperative procedures.

Formal semantics: The language should have a well-defined formal semantics to ensure precise interpretation by different workflow executors.

Existing modeling languages such as BPMN, UML Activities Diagrams and State Machines, only partially satisfy these requirements. They are expressive enough, and in some flavors they are provided with formal semantics. However their standard XML-based textual syntax is graph-oriented, and non-standard textual notations have limited popularity. Experimental evidence shows that state-of-the-art LLMs struggle in connecting edges in workflow graphs [4]. On the other side, approaches like SOPs have native textual formats structured as procedures, that is a structure more suitable for LLM-based processing (due to the great number of procedural languages LLMs are exposed to). However SOPs lack formal semantics, expressiveness and standardization. Finally, tool-specific workflow languages used in industry generally lack any formal semantics.

In this work, we adopt `SysML v2` action flows as a candidate language for LLM workflows, as it offers a formal semantics and a balanced trade-off between expressiveness and machine interpretability. Figure 1 shows an example of LLM workflow that we translate from WORFBENCH [4]. `SysML v2` textual syntax is concise and structured, making it suitable for both automated processing and generation. In particular we make extensive use of succession flows in `SysML v2` (first... then... then...), that give a procedural structure to the code, strongly reducing the graph-like connections. For instance in Figure 1 graph-like connections appear only to represent the output of a fork in line 4, and to the input of a join, in lines 6 and 8. In our early experiments the procedural structure improves the generation accuracy of LLMs.

3. Dataset construction

Due to the current lack of publicly available, large-scale, and formally structured workflow datasets encoded in SysML v2, we opted to transform existing ones into `SysML v2` representations. This limitation is primarily explained by the recent finalization of the `SysML v2` standard, which remains under gradual adoption within both academic and industrial communities.

To address this gap, we transform the models using fully deterministic pipelines, and only missing prompts are generated using LLMs after the transformation. The decision to avoid LLM transformations was motivated by the need to guarantee reproducibility, and prevent potential hallucinations during the conversion process. Our approach operates by systematically extracting workflow primitives, such as the list of actions representing nodes, along with the dependency relations defining the execution flows. More details about each transformation and evaluation are in the README file of the transformation in the paper repository.

```
package package1 {
   action def workflow1 {
      first start;
      then fork; then action1; then action2;
      action action1 { desc = "Find the total number of possible numbers."; }
         then joinNode3;
      action action2 { desc = "Find the number of perfect squares between 1 and 100.";}
         then joinNode3;
      join joinNode3;
      then action action3 { desc = "Calculate the probability that a randomly selected
          number is a perfect square.";}
      then done;
   }
}
```

Figure 1: Example of SysML v2 workflow: probability that a number between 1 and 100 is a perfect square

WORFBENCH transformation. **WORFBENCH** [4] introduces a unified benchmark for multiple scenarios, modeling workflows as DAGs. This dataset covers a diverse range of workflow generation tasks, including problem solving, function calling, embodied planning, and open-grounded planning. It comprises 18k training samples, 2146 test samples, and 723 held-out tasks intended to evaluate generalization. In addition, each workflow instance is paired with its corresponding user prompt, which serves as a natural language description of the target workflow. We implemented a deterministic transformation pipeline that converts WORFBENCH workflow graphs into SysML v2 code. Nodes of the original graph are translated to actions, dependency edges are mapped to succession flows, and fork nodes are introduced to represent multiple dependencies. We successfully transform 17.8k WORFBENCH models. The few hundred cases that could not be transformed either lacked valid actions or edges, or could not be parsed due to malformed syntax generated by the LLM.

SAP-SAM transformation. **SAP-SAM** [5] is a dataset that contains 1.02M manually produced diagrams, including UML and BPMN models. Here we focus on transforming BPMN diagrams (631.4k models) into SysML v2 code. We only consider BPMN diagrams written in English, and we restrict our selection to diagrams that use only a subset of BPMN elements that have a natural mapping to SysML v2 action flows. For instance, BPMN *activities* are directly mapped to SysML v2 actions and *exclusive gateways (XOR)* are represented using SysML v2 decision constructs with conditional branches. We successfully transform 56.2k SAP-SAM models.

SOP-Bench transformation. **SOP-Bench** [6] is a dataset designed to evaluate AI agents on realistic, multi-step workflows expressed as Standard Operating Procedures SOPs [6]. The transformation of this dataset is similar to what we performed for WORFBENCH, as both rely on an action-graph representation. However, SOP-Bench introduces OR nodes as an additional control structure to represent alternative execution paths. To handle this in our transformation, we adopt a sequential decision strategy. We first execute action A, then introduce a decision node that checks whether A succeeded. If it succeeds, the workflow continues; otherwise, the alternative path (action B) is executed. We successfully transform 903 SOPBench models.

Resulting Dataset. Table 1 presents a summary of the transformed datasets. To assess the quality of the transformation, we applied two complementary validation steps. First, we validated the syntactic correctness of all the generated models using the SysML v2 pilot implementation[4] parser and PlantUML exporter. Then, we manually inspected a randomly selected subset of transformed instances by visualizing the generated SysML v2 models to ensure semantic consistency with the original workflows. In addition, to assess the richness of the generated dataset, we inspect the frequency of actions, decisions and forks. In particular number of decisions and forks can be use as a metric of the complexity and

[4] github.com/Systems-Modeling/SysML-v2-Pilot-Implementation/

Table 1
Summary of source datasets and their transformation into SysML v2

Language	Source Dataset	Initial Size	Generated Dataset Size (Transformation Rate)	Natural Language Description
Graph	WORFBENCH (Feb 2025)	18k	17.8k (98.84%)	User prompt and list of actions
BPMN	SAP-SAM (Aug 2022)	Total : 1.02M BPMN : 631.4k English : 367.4k Filtered : 56.6k	56.2k (99.36%)	Generated by providing an LLM with the list of BPMN elements (ongoing)
SOP	SOPBench (Jun 2025)	0.9k	0.9k (100%)	Process description
Total	-	75.1k	74.9k (99.73%)	-

Table 2
Frequency of SysML v2 elements in the dataset

	% of models containing elements			% of elements in the dataset		
Element	Action	Decision	Fork	Action	Decision	Fork
SOP-Bench	100	81	88	46	19	15
WORFBENCH	100	0	15	92	0	4
SAP-SAM	100	60	28	82	11	3

Table 3
Mean and maximum count of SysML v2 elements in the dataset

Element	Action		Decision		Fork	
	Mean	Max	Mean	Max	Mean	Max
SOP-Bench	5.321	10	2.268	9	1.767	4
WORFBENCH	3.821	15	0	0	0.168	12
SAP-SAM	8.955	273	1.248	109	0.35	30

non-linearity of the workflows. In Table 2, the metrics are divided into two sections. We first measure, for each generated dataset, the percentage of models that contain respectively actions, decisions and forks. Then we measure the percentage of each element type out of all elements; for instance, in SOP-Bench, 19% of the elements in all workflows are decision nodes. In Table 3, we present the minimum, maximum, and mean of each element per model. We observe that SAP-SAM contains much larger models, with up to 273 actions and 109 decisions at maximum. Overall, these results indicate that the generated dataset captures a diverse range of workflow structures, from simple linear sequences to more complex and non-linear patterns. This diversity provides a solid foundation for studying and improving workflow generation in LLM-based systems.

4. Conclusion and Future Work

In this paper we argued that action flows in SysML v2 satisfy the key requirements for expressing LLM workflows in an LLM-understandable way. We built a public dataset of action flows that enables future quantitative research on this application of SysML v2.

As future steps, we aim to extend the coverage of SysML v2 to support a broader range of workflow constructs and modeling patterns (e.g. explicit loops). In addition, we plan to fine-tune LLMs on the generated dataset to improve their ability to generate structured workflows from natural language descriptions.

Acknowledgments. This paper is supported by the European Union under the Grant Agreement No 101189664 (MOSAICO project).

Declaration on Generative AI. The authors have not employed any generative AI tools.

References

[1] B. Wang, S. Min, X. Deng, J. Shen, Y. Wu, L. Zettlemoyer, H. Sun, Towards understanding chain-of-thought prompting: An empirical study of what matters, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada, 2023, pp. 2717–2739. URL: https://aclanthology.org/2023.acl-long.153/. doi:10.18653/v1/2023.acl-long.153.

[2] B. Chen, K. Chene, J. A. H. López, G. Mussbacher, D. Varró, A. Feizpour, SHERPA: A model-driven framework for large language model execution, in: 2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems (MODELS), IEEE, 2025, pp. 197–208.

[3] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, J. Schmidhuber, MetaGPT: Meta programming for a multi-agent collaborative framework, in: The Twelfth International Conference on Learning Representations, 2024.

[4] S. Qiao, R. Fang, Z. Qiu, X. Wang, N. Zhang, Y. Jiang, P. Xie, F. Huang, H. Chen, Benchmarking agentic workflow generation, in: The Thirteenth International Conference on Learning Representations, 2025.

[5] D. Sola, C. Warmuth, B. Schäfer, P. Badakhshan, J.-R. Rehse, T. Kampik, Sap signavio academic models: A large process model dataset, in: International Conference on Process Mining, Springer, 2022, pp. 453–465.

[6] S. Nandi, A. Datta, N. Vichare, I. Bhattacharya, H. Raja, J. Xu, S. Ray, G. Carenini, A. Srivastava, A. Chan, et al., SOP-bench: Complex industrial SOPs for evaluating LLM agents, arXiv preprint arXiv:2506.08119, accepted in KDD 2026 (2025).