Can LLM Support Formal Requirements Engineers in ASMETA?

Andrea Bombarda[1 [0000−0003−4244−9319]], Silvia Bonfanti[1 [0000−0001−9679−4551]], Angelo Gargantini[1 [0000−0002−4035−0131]], and Nico Pellegrinelli[1 [0009−0000−4944−6845]]

University of Bergamo, Bergamo, Italy
{andrea.bombarda, silvia.bonfanti, angelo.gargantini, nico.pellegrinelli}@unibg.it

Abstract. Bridging natural-language requirements and analyzable formal artifacts remains a major obstacle in model-based development. This paper reports recent results on the use of Large Language Models (LLMs) within the ASMETA toolset to support requirements-driven modeling, temporal-property formalization, validation, and explanation. The proposed workflow is explicitly human in the loop: the LLM drafts ASMETA models, CTL/LTL properties, and scenarios or explanations, while ASMETA tools provide feedback that guides iterative repair and semantic checking. Evidence from a case study and property-oriented examples shows that LLMs can accelerate the production of analyzable first drafts and stakeholder-readable explanations. At the same time, compilation, simulation, scenario-based validation, and manual review remain essential to detect syntax errors, vocabulary mismatches, misunderstandings of ASM semantics, and incomplete behavioral coverage.

Keywords: Formal Requirements Engineering · ASMETA· Abstract State Machines · Large Language Models

1 Introduction

Formal requirements engineering aims to reduce ambiguity before design and implementation choices make errors costly. In the ASMETA toolset, executable Abstract State Machine models support simulation, scenario-based validation, and model checking, turning requirements into analyzable artifacts rather than static documentation [1, 4]. In practice, however, two bottlenecks remain recurrent: deriving a consistent executable model from textual requirements, and expressing behavioral requirements as temporal properties over the model signature. Recent work on LLM-assisted formalization has shown that language models can help bridge natural-language requirements and temporal logic [7, 8], while complementary work on explainable formal methods suggests that formal artifacts can also be translated back into stakeholder-readable language [5].

This paper presents a preliminary investigation of that bridge in the specific setting of ASMETA. We consider LLMs not as replacements for formal methods,

but as interactive assistants embedded in a tool-supported workflow. Their role is to draft models, properties, scenarios, and explanations; the role of ASMETA tools is to check whether these artifacts compile, execute, validate, and match the vocabulary of the specification; and the role of the engineer is to decide whether the resulting artifacts preserve stakeholder intent. We report our experience on using LLMs in supporting the modeling of a reduced planetary rover case study derived from the ABZ 2026 case study [2] and property-oriented examples over ASMETA models [3]. Across these settings, the same conclusion emerges: LLMs are useful at the interface between informal requirements and formal artifacts, but trustworthy use requires a closed loop of tool feedback and human review.

2 Using LLMs for ASMETA-Based Requirements Engineering

We envision leveraging LLMs to support requirements engineers in two activities: translating natural language requirements into formal ASMETA specifications, and validating and verifying these artifacts.

The LLM-supported modeling process introduced in [2] assists engineers in translating informal requirements into formal specifications. More specifically, in [2], we leveraged GPT 5.2 LLM, accessed via the ChatGPT interface and its *Projects* functionalities. Starting from a high-level description of the system of interest, the LLM is prompted to generate an initial AsmetaL model or a set of modules. The process is iterative and supported by the ASMETA toolset. Each generated ASMETA model is compiled using AsmetaC to verify whether the specification can be correctly parsed and compiled. When compilation errors occur, the feedback provided by AsmetaC is incorporated into subsequent interactions with the LLM to refine and correct the ASMETA specification. This iterative cycle can continue until a syntactically correct specification is obtained, or until engineers decide to intervene and manually revise the specification.

Once informal requirements have been encoded into a syntactically correct ASMETA model, the LLM can be employed to support validation and verification activities. During validation, an LLM can assist in generating Avalla scenarios (i.e., formal descriptions of system behavior) from textual descriptions of expected interactions. Additionally, the LLM can be used to explain scenarios automatically generated by the model checker or produced through random simulation, helping engineers better understand the system behavior captured by the model. During verification, the LLM can assist engineers in formalizing temporal properties [4] from textual descriptions and in validating that such properties correctly capture the intended system requirements, as introduced in [3]. If validation or verification activities discover undesired or wrong behaviors, system formalization can restart and the ASMETA model be fixed.

This workflow gives the LLM and engineers complementary responsibilities. The former contributes speed in drafting, reformulation, and explanation with the help of formal tools, which contribute precise feedback about syntax, exe-

cution, and property satisfaction. The latter remains responsible for semantic acceptance and supervising the entire process.

3 Results on LLM-Assisted Modeling, Properties, and Validation

We summarize the main results on the use of LLMs for modeling the rover case study [2] and for formalizing and validating properties [3] in ASMETA, highlighting both their potential and current limitations.

From a minimal prompt based on selected requirements of the ABZ 2026 planetary rover [6], the LLM generated an initial model with six ASMETA modules and a main specification. While the decomposition provided useful insights, the resulting model was not directly usable. Compilation and manual inspection revealed recurring defects, including missing export statements, incorrect domain definitions, missing function initializations, and references to non-existing functions. A more constrained prompt fixed some of these issues, but manual intervention was still needed to obtain a compilable model. Indeed. we performed three iterations with the LLM before obtaining a usable model, which required manual fixes though.

Simulation and animation then exposed problems that compilation alone could not detect. Functions were duplicated across modules, certain rules reflected a sequential-programming intuition rather than ASM parallel semantics, and some subdomains would have required impractical explicit enumeration. These results show that syntactic correctness is not a sufficient acceptance criterion and that semantic alignment requires execution-based feedback.

During scenario-based evaluation of the rover model, the LLM proved useful both for explaining existing AVALLA scenarios and for generating new ones to check specific behaviors. It correctly recognized that randomly generated scenarios were plausible but did not cover important rover capabilities such as movement. We therefore prompted the LLM to generate a scenario that drives the rover to a charging position. Although the first version failed because of an incorrect use of controlled functions, it later helped reveal errors introduced during manual fixes. In this setting, LLM-generated scenarios were useful both for expanding behavioral coverage and for identifying manually introduced errors.

Complementary evidence comes from verification activities [3]. Given an ASMETA model, the LLM was able to suggest natural-language properties, translate requirements into temporal formulas (in both LTL and CTL), explain such formulas in plain language, and summarize counterexamples. On a simple clock model, for example, it derived range and carry-over properties and translated them into CTL formulas, although not in a form directly compatible with ASMETA. We also found that ASMETASMV feedback (e.g., error traces or counterexamples) can be incorporated into subsequent prompts to support understanding and repair.

These results suggest that LLMs can help users obtain analyzable first drafts, reformulate requirements as temporal properties, and interpret scenarios, formu-

las, and traces. However, they do not reliably produce syntactically and semantically consistent artifacts. The engineers, supported by formal tools, must retain full responsibility for supervising the process.

4 Conclusion

These results indicate that LLM assistance can be integrated meaningfully into ASMETA-based formal requirements engineering, provided that the interaction is organized as a closed loop between requirements, formal artifacts, tool feedback, and human judgment. In this role, the LLM acts as a drafting and explanation aid that can shorten the path to executable models, temporal properties, and interpretable validation evidence. The formal tools remain responsible for checking behavior, and the engineer remains responsible for intent. The next steps are to integrate this interaction more tightly into ASMETA, evaluate it on larger and more diverse case studies and requirement styles, and study whether richer contextual grounding or more specialized models can improve the quality of the generated artifacts without weakening semantic control.

References

1. Arcaini, P., Bombarda, A., Bonfanti, S., Gargantini, A., Riccobene, E., Scandurra, P.: The ASMETA Approach to Safety Assurance of Software Systems, pp. 215–238. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-76020-5_13
2. Bombarda, A., Bonfanti, S., Gargantini, A., Pellegrinelli, N.: Can Large Language Models Support Modeling Systems with ASMETA? A Case Study with a Planetary Rover. In: ABZ 2026: The 12th International Conference on Rigorous State Based Methods. Springer International Publishing, Cham (2026)
3. Bombarda, A., Bonfanti, S., Gargantini, A., Pellegrinelli, N.: Formalizing and validating properties in asmeta with large language models (extended abstract) (2026), https://arxiv.org/abs/2603.15375
4. Bombarda, A., Bonfanti, S., Gargantini, A., Riccobene, E., Scandurra, P.: ASMETA Tool Set for Rigorous System Design. In: Formal Methods. pp. 492–517. Springer Nature Switzerland, Cham (2025)
5. Cherukuri, H., Ferrari, A., Spoletini, P.: Towards explainable formal methods: From LTL to natural language with neural machine translation. In: Requirements Engineering: Foundation for Software Quality. pp. 79–86. Springer International Publishing, Cham (2022)
6. Farell, M., Kobayashi, T.: ABZ 2026 Case Study: A Planetary Rover, https://github.com/trarse-nii/ABZ2026-case-study/blob/main/doc/document_v2.pdf
7. Mendoza, D., Hahn, C., Trippel, C.: Translating Natural Language to Temporal Logics with Large Language Models and Model Checkers. In: 2024 Formal Methods in Computer-Aided Design (FMCAD). pp. 1–11 (2024). https://doi.org/10.34727/2024/isbn.978-3-85448-065-5_17
8. Zhao, M., Tao, R., et al.: NL2CTL: Automatic Generation of Formal Requirements Specifications via Large Language Models. In: Formal Methods and Software Engineering. pp. 1–17. Springer Nature Singapore, Singapore (2024). https://doi.org/10.1007/978-981-96-0617-7_1