Pushing the (Generative) Envelope: Measuring the Effect of Prompt Technique and Temperature on the Generation of Model-based Systems Engineering Artifacts

Erin Smith Crabb, Cedric Bernard, Matthew T. Jones, Daniel Dakota
Leidos Holdings, Inc.
{erin.a.smithcrabb,cerdric.bernard,
matthew.t.jones,daniel.d.dakota}@leidos.com

Abstract

System engineers use Model-based systems engineering (MBSE) approaches to help design and model system requirements. This manually intensive process requires expertise in both the domain of artifact creation (e.g., the requirements for a vacuum), and how to encode that information in a machine readable form (e.g., SysML). We investigated leveraging local LLMs to generate initial draft artifacts using a variety of prompt techniques and temperatures. Our experiments showed promise for generating certain types of artifacts, suggesting that even smaller, local models possesses enough MBSE knowledge to support system engineers. We observed however that while scores for artifacts remain stable across different temperature settings, this is potentially misleading as significantly different, though semantically equivalent, generations can be produced.

1 Introduction

Model-based systems engineering (MBSE), as defined by the International Council on Systems Engineering, is the *formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases* (Campo et al., 2022). In practice, it is the modeling of system requirements and components that have formalized technical design requirements (e.g., user requirements list) by a systems engineer, with applications to aerospace, medical fields, and software development, among many others. Usage of MBSE by an organization enables understanding and oversight of large, complex systems requiring support from a potentially diverse array of disciplines, and provides a supporting framework for simulation and testing of those systems (McGrath and Jonker, 2023).

To facilitate digital use and interaction, MBSE relies on SysML (and more recently SysML v2), which is based on Unified Modeling Language (UML). This encodes system design specifications into a machine readable form (see Figure 1) that can then be subsequently visualized (see Figure 2, examples taken from The Object Management Group (2023)). The manually intensive process of creating SysML models is a current bottleneck (Fuchs et al., 2024) and motivated our evaluation of how effectively LLMs can be leveraged to generate complete and consistent initial design item drafts.

We investigate the effectiveness of some currently available open-sourced models in generating initial SysML v2 artifacts (formal requirements list and state machine diagrams) and explore variability when applying different temperatures. We hypothesized that the model must have some internal knowledge of both MBSE and SysML to be effective, and that external knowledge injection (via prompt engineering) would generate better artifacts.

2 Related Work

Little work exists wrt. the application of LLMs and prompt engineering to MBSE, with most focusing on the use of prominent API-based models (e.g., GPT4). Smith Crabb and Jones (2024) performed user-based experiments incorporating Q&A with ChatGPT-3.5. Results showed that that users with access to models had their system engineering models ranked higher by a subject matter expert (SME) both quantitatively and qualitatively. Experiments by Longshore et al. (2024) comparing GPT-3.5 and ChatGPT-4 ability to generate system models show a noticeable improvement with the latter. Additionally, to overcome the inherent deficiency of SysML v2 knowledge, they implemented a RAG architecture that ingested SysML v2 docu-

Proceedings of Recent Advances in Natural Language Processing,pages 1188–1194
Varna, Sep 8–10, 2025
https://doi.org/10.26615/978-954-452-098-4-137

```
requirement def MassRequirement{
    doc /*The actual mass shall be less than the required mass*/
    attribute massRequired:>ISQ::mass;
    attribute massActual:>ISQ::mass;
    require constraint {massActual<=massRequired}
}
```

Figure 1: Example of SysML v2 for Mass Requirement.

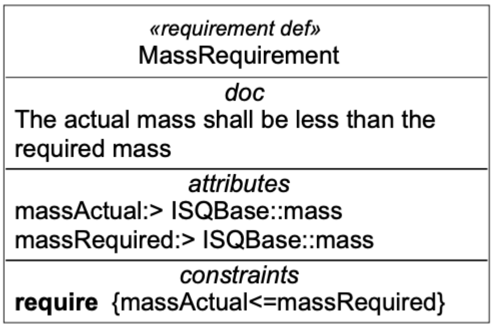

Figure 2: Visualization of SysML v2 for Mass Requirement in Fig. 1.

mentation, finding that one-shot prompting using ChatGPT-4 obtained valid SysML v2 models. Similar findings have been noted using UML directly (Bader et al., 2024) and MBSE relevant Object-Process Methodology (Alarcia et al., 2024). In line with this, Fuchs et al. (2024) found that including an LLM in the MBSE workflow can substantially improve costs and speed by factors of more than ten. We are not aware however of research that focuses on the use local, smaller models for MBSE, which is problematic for systems which encode data that may be unsuitable for analysis or incorporation into commercial LLMs, whether for legal, ethical, or business-preferential reasons.

3 Experimental Setup

3.1 Models

Systems engineering requires both understanding of real-world concepts and domain knowledge (i.e., knowledge of MBSE, real-world objects, and encoding this into machine readable SysML). Models focused solely on code generation may be too limited with regard to general knowledge to act as an effective systems modeler. Instead, this work focuses on models that had access to code, real-world concepts, and potential domain knowledge during training. Additionally, due to requirements of operating in a closed environment (i.e., no external network connections), models requiring external API calls (e.g., GPT-4) are not applicable.

We selected `Mixtral-8x7B-Instruct-v0.1` (Mixtral; Jiang et al., 2024)[1], a 46.7 billion parameter sparse mixture of experts trained model, and `Llama-3-Smaug-8B` (Smaug; Pal et al., 2024)[2], a 8.03 billion parameter DPO-Positive trained model. Both had access to extensive sources of data (predominantly from the web) during training and have achieved good performances on a range of tasks (including code generation) when compared against both other open-source and older versions of proprietary models (e.g., GPT-3.5). Experiments for Mixtral were run on a p4d.24xlarge and Smaug on a `g4dn.12xlarge` EC2 instance respectively.

We recognize that we only perform experiments on two local models, because even this is both computationally (6-8 hours) and cost prohibitive (minimally $500 per 100 runs each) given we run them on GPUs of an appropriate size, meaning we must be selective of experimental choices due to cost and time constraints. More importantly however, the primary focus is not on obtaining the highest performance, as current MBSE research uses prominent models that require external APIs (see (Smith Crabb and Jones, 2024) for performance of commercial models on MBSE tasks), we are focused on assessing the feasibility and stability of deploying smaller, local models to support system engineers in more resource constrained and closed

[1]https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

[2]https://huggingface.co/abacusai/Smaug-72B-v0.1

Model-Based Systems Engineering (MBSE) is a systems engineering approach that uses graphical models to represent, analyze, and communicate the various aspects and components of a complex system. It is a shift away from traditional document-based systems engineering, where system requirements, design, and behavior are described in text documents.
In MBSE, a system is represented as a set of interconnected models that describe the system's structure, behavior, and performance. These models are created using a standardized modeling language, such as the Systems Modeling Language (SysML), which provides a common vocabulary and set of symbols for systems engineers to use.
The use of models in MBSE offers several advantages over traditional document-based systems engineering. Models allow systems engineers to:
* Represent complex systems in a more intuitive and visual way
Analyze the behavior and performance of a system before it is built
Identify and resolve design issues early in the development [INST] Give me an example of SysML. [/INST]Sure! Here is an example of a simple SysML block definition diagram (BDD) that represents a car system:

Figure 3: Mixtral 8x7B-Instruct's response to both "Define Model-Based Systems Engineering".

````
```
requirement WeightRequirement {
  doc "The maximum allowable weight for the system shall be 100 kg, with a tolerance of +/- 5 kg.";
  attribute weightRequired: ISQ::mass = 100 kg;
  attribute weightActual: ISQ::mass;
  attribute weightTolerance: ISQ::mass = 5 kg;
  require constraint { weightActual <= weightRequired + weightTolerance };
}
```
````

Figure 4: Mixtral 8x7B-Instruct Weight Requirement Definition SysML v2 Generated Example.

environments, as well as to address needs whereby use of any external APIs (whether commercial or otherwise) are prohibited.

3.2 Prompt Engineering

MBSE and SysML is a very specific set of domain knowledge. We first prompted models for internal knowledge. Each model was prompted with *"Define Model-Based Systems Engineering"* and a separate prompt of *"Give me an example of SysML."*. Both models provided accurate answers describing MBSE and also provided sufficient enough generations of SysML (e.g., generating high-level documentation to describe a weight requirement) to justify further investigation (see Figs. 3 and 4 respectively for examples of generated responses). However, both clearly showed limitation in knowledge of SysML v2.[3] Given the difficulty of generating perfectly-formatted SysML v2, we chose to instead generate dot notation versions of the SysML v2 diagrams, allowing us to evaluate quality of the model content separately from the formatting.

[3]There are significant differences between SysML and SysML v2 which can be found here: `https://github.com/Systems-Modeling/SysML-v2-Release`

MBSE Artifacts here cover two different items (air purifier and vacuum), two different engineering artifacts (requirements lists and state machine diagrams), and assess three temperatures settings (0.2, 0.6, and 0.95). While air purifiers and vacuums are both general items, the quantity of information available on each differs, which we expect to also be reflected in the model's behavior. Requirements lists were chosen as they provide good insights into a model's general domain knowledge of SysML requirements, while state machine diagrams are problematic for system engineers and accelerating their creation would be highly beneficial. The same prompt to generate the desired SysML v2 for each experimental setting (e.g., vacuum - requirements list - temperature 0.2) was run 100 times.

3.2.1 N-Shot Prompting

Initial prompting techniques include zero-shot, one-shot, and few-shot. The zero-shot prompt contained a simple instruction, *"Generate the requirements necessary for a vacuum. Include only the requirements, in list format, with no introductory or concluding text."* which relied solely on the model's internal knowledge. One-shot and few-shot differed only in that model was supplied example artifacts

	Requirements List		
Technique	Temp	Air Purifier	Vacuum
zero-shot	0.20	0.1447	0.1935
	0.60	0.1402	0.1827
	0.95	0.1480	0.1828
one-shot	0.20	0.4527	0.3235
	0.60	0.4496	0.3379
	0.95	0.4489	0.3446
few-shot	0.20	0.2131	0.3667
	0.60	0.2609	0.3564
	0.95	0.2762	0.3422

	State Machine		
Technique	Temp	Air Purifier	Vacuum
zero-shot	0.20	0.0917	0.1642
	0.60	0.0964	0.1446
	0.95	0.1006	0.1287
one-shot	0.20	0.1052	0.3364
	0.60	0.1162	0.3397
	0.95	0.1164	0.3226
few-shot	0.20	0.1493	0.2863
	0.60	0.1132	0.3315
	0.95	0.1014	0.3399

Table 1: Mixtral zero-shot, one-shot, and few-shot METEOR Scores for air purifier and vacuum.

Requirements List		
Temp	Air Purifier	Vacuum
0.20	0.4507	0.4091
0.60	0.4512	0.3943
0.95	0.4544	0.3799

State Machine		
Temp	Air Purifier	Vacuum
0.20	0.2677	0.3539
0.60	0.2697	0.3522
0.95	0.2752	0.3516

Table 2: Smaug one-shot METEOR Scores for air Purifier and vacuum.

	State Machine		
	Temp	Air Purifier	Vacuum
Mixtral	0.20	0.1983	0.4639
	0.60	0.1928	0.4384
	0.95	0.2012	0.4349
Smaug	0.20	0.1761	0.4272
	0.60	0.1969	0.4434
	0.95	0.1985	0.4259

Table 3: Chain-of-Thought METEOR Scores for air purifier and vacuum.

in SysML v2 (see Fig. 1 for an example of the type of artifact provided). Importantly, each example provided to the prompt was chosen to ensure difference from the domain of the target item.

3.3 Evaluation

While standardized evaluation metrics for systems engineering have been proposed (Henderson et al., 2022), these are not particularly relevant for automatic assessment, and currently, most MBSE model analysis remains qualitative in nature. We also recognize that the best metrics for code generation is a debated topic of research, but here we focus here on an initial proof point and limit our evaluation for analysis purposes to focus more on feasibility and MBSE practitioner feedback.

For each chosen item (e.g., vacuum, air purifier), a gold-standard exemplar produced by a systems engineering expert exists. Each exemplar was translated into dot notation in order to account for the fact that SysML v2 is still, as previously noted, largely unrepresented in the LLMs, and this could artificially lower scores by virtue of errors in the generated formatting, rather than the model contents. The gold dot notation models were then compared to each generated response, obtained in a corresponding dot notation form, using METEOR (Banerjee and Lavie, 2005). We chose METEOR as multiple studies have shown strong correlations with human judgments (Hu et al., 2022; Takaichi et al., 2022). Given our focus is human-computer teaming for MBSE system design, we are particularly interesting in alignment with a systems engineer's expectations. We report the average METEOR score over the 100 runs.

4 Results

4.1 Mixtral

Table 1 presents results for Mixtral. For requirements lists, both air purifier and vacuum benefit from one-shot prompting. However, we saw that air purifiers decreases in performance with few-shot, while for vacuum performance remains similar. For state machine diagrams, we saw that air purifier fails to have any substantial increases. For vacuum however, a similar pattern as with requirements list emerged. Mixtral also showed a great deal of stability across temperature settings within each type of prompt.

4.2 Smaug

We only investigate one-shot experiments using Smaug as results from Mixtral suggested this was a good prompting technique[4] and these results are

[4]This was also to align with the goal of reducing computational resources and financial costs.

Technique	Item	Artifact	Temp Pair
one-shot	air purifier	state machine	0.20 & 0.60*
			0.60 & 0.95
			0.20 & 0.95*
		requirements list	0.20 & 0.60
			0.60 & 0.95
			0.20 & 0.95
	vacuum	state machine	0.20 & 0.60
			0.60 & 0.95*
			0.20 & 0.95*
		requirements list	0.20 & 0.60*
			0.60 & 0.95*
			0.20 & 0.95*
CoT	air purifier	state machine	0.20 & 0.60*
			0.60 & 0.95*
			0.20 & 0.95
		requirements list	0.20 & 0.60
			0.60 & 0.95*
			0.20 & 0.95*
	vacuum	state machine	0.20 & 0.60*
			0.60 & 0.95
			0.20 & 0.95*
		requirements list	0.20 & 0.60
			0.60 & 0.95*
			0.20 & 0.95

Table 4: Mixtral Temperature t-tests. Pairs marked with * indicate significance at .05 p-value.

Technique	Item	Artifact	Temp Pair
one-shot	air purifier	state machine	0.20 & 0.60*
			0.60 & 0.95*
			0.20 & 0.95*
		requirements list	0.20 & 0.60
			0.60 & 0.95
			0.20 & 0.95
	vacuum	state machine	0.20 & 0.60
			0.60 & 0.95
			0.20 & 0.95
		requirements list	0.20 & 0.60*
			0.60 & 0.95*
			0.20 & 0.95*
CoT	air purifier	state machine	0.20 & 0.60*
			0.60 & 0.95
			0.20 & 0.95*
		requirements list	0.20 & 0.60*
			0.60 & 0.95*
			0.20 & 0.95
	vacuum	state machine	0.20 & 0.60*
			0.60 & 0.95
			0.20 & 0.95
		requirements list	0.20 & 0.60*
			0.60 & 0.95*
			0.20 & 0.95*

Table 5: Smaug Temperature t-tests. Pairs marked with * indicate significance at .05 p-value.

presented in Table 2. Interestingly, we saw that the results of requirements lists for air purifier were nearly identical as that of Mixtral, while vacuum actually increased in performance. For state machine diagrams, the reverse was true: air purifier increased compared to its Mixtral counterpart and vacuum remained relatively the same. Similar to Mixtral, we also saw a lack of variation between results wrt. temperature settings. These results demonstrated that smaller models with sufficient base MBSE knowledge could be leveraged to achieve performance on par with larger ones.

5 Chain-of-Thought

SME reviews noted that part of the variation in performance between one and few-shot results was due to increased LLM hallucinations as overly long examples seemed to encourage longer responses by the LLMs, which in turn increased generations. The LLMs seemed to be attempting to fill gaps where they did not include enough realistic components. To address this we experimented with simple Chain-of-Thought (CoT) experiments where the model was instructed to first generate its own requirements list and then used this generated list to support state machine design. As state machines are supposed to document all the possible states and their relationships, we hypothesized this may be challenging when the model does not know exactly what those could be.

Table 3 shows results using both Mixtral and Smaug, where we saw that different trends emerge. Air purifier using Mixtral increased in performance compared to few-shot, but decreased compared to its one-shot Smaug performance. Vacuum received noticeable boosts in performance in both models, with each reaching a similar plateau. Again scores remained highly similar across temperature settings for both models.

6 Temperature Significance Testing

To further investigate the seemingly stable behavior of temperature within an artifact, we performed t-tests between the 100 runs for each temperature pair (results for Mixtral are presented in Table 4) and Table 5 for Smaug results). Tests indicated that the surface stability was misleading, as many temperature pairs for the same artifact were statistically significant, suggesting underlying distributions differences between temperatures. Stability of generations is of great concern to users, especially in technical designs. However, given the subjective nature of individual systems engineers preferences in our conversations with SMEs, this behavior actually mimics the ambiguity that many system engineers would themselves exhibit.

Examining various temperature output pairs indicates semantically similar generations for code were quite common. For example, a state machine for air purifier showed the state “Purifying” with a transition to another state “FilterChange”. This can be represented as:

```
Purifying -> FilterChange
[label=``Filter Replacement
Needed'']
```

in SysML. However the label within “FilterChange” could be realized by many different (but semantically equivalent) options: “Filter Replacement Needed”, “Filter Replacement Required”, “Filter Replacement Necessary” which METEOR better handled. This behavior however was an expected parallel of reality, as individual systems engineers may also choose such semantically equivalent terms and would expect them to be easily understood and accepted.

7 Limitations

Due to the structure of chain-of-thought prompting, our chosen prompt chain only yielded use case and state machine diagram examples for that technique; the 900 requirements lists, while they were generated, were not scored as part of the final analysis.

We recognize that we only perform experiments on two local models and selected a sub-set of prompting techniques, and acknowledge that additional prompting experiments and models may result in a different analysis. However, we view our findings here as providing evidence that local models are a viable path forward.

8 Conclusions

We reported our investigation of the ability of local, open-source LLMs to generate system engineering artifacts, showing even smaller models to be viable sources of first draft MBSE models. However, success depended on several factors: the complexity of the artifact, the availability of real-world knowledge of the item, and the prompt technique, with one-shot and CoT obtaining best results. Interestingly, while results showed that temperature variation within artifacts yielded little differences in score changes, significance testing indicated there was substantial underlying variation. This was potentially due to providing guidance in structured generations and different semantically equivalent generations. Future research will investigate adding additional external knowledge to support in the creation of more complex domain specific design requirements (e.g., aerospace).

Ethical Statement

We are not aware of any ethical concerns.

Acknowledgments

This document is export approved by Leidos for release under identification number **24-LEIDOS-1004-28406**.

References

Ramón María García Alarcia, Pietro Russo, Alfredo Renga, and Alessandro Golkar. 2024. Bringing systems engineering models to large language models: An integration of opm with an llm for design assistants. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2024)*, pages 334–345.

Elias Bader, Dominik Vereno, and Christian Neureiter. 2024. Facilitating user-centric model-based systems engineering using generative ai. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2024)*, pages 371–377.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Kelly X. Campo, Thomas H. Teper, Casey E. Eaton, Anna M. Shipman, Garima Bhatia, and Bryan L. Mesmer. 2022. Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature. *Systems Engineering*, 26:104 – 129.

Jared Fuchs, Christopher Helmerich, and Steven Holland. 2024. Transforming system modeling with declarative methods and generative ai. *AIAA SCITECH 2024 Forum*.

Kaitlin Henderson, Thomas A. McDermott, Eileen M. Van Aken, and Alejandro Salado. 2022. Towards Developing Metrics to Evaluate Digital Engineering. *Systems Engineering*, 26:3–31.

Xing Hu, Qiuyuan Chen, Haoye Wang, Xin Xia, David Lo, and Thomas Zimmermann. 2022. Correlating

Automated and Human Evaluation of Code Documentation Generation Quality. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(4):1–28.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. *ArXiv*, abs/2401.04088.

Ryan Longshore, Ryan Bell, and Ray Madachy. 2024. Leveraging Generative AI to Create, Modify, and Query MBSE Models. In *Proceedings of the Twenty-First Annual Acquisition Research Symposium*, pages 311–327, Monterey, CA.

Amanda McGrath and Alexandra Jonker. 2023. What is model-based systems engineering (MBSE)? IBM. https://www.ibm.com/think/topics/model-based-systems-engineering.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing Failure Modes of Preference Optimisation with DPO-Positive. *ArXiv*, abs/2402.13228.

Erin A. Smith Crabb and Matthew T. Jones. 2024. Accelerating Model-Based Systems Engineering by Harnessing Generative AI. In *2024 19th Annual System of Systems Engineering Conference (SoSE)*, pages 110–115.

Riku Takaichi, Yoshiki Higo, Shinsuke Matsumoto, Shinji Kusumoto, Toshiyuki Kurabayashi, Hiroyuki Kirinuki, and Haruto Tanno. 2022. Are NLP Metrics Suitable fornbsp;Evaluating Generated Code? In *Product-Focused Software Process Improvement: 23rd International Conference, PROFES 2022, Jyväskylä, Finland, November 21–23, 2022, Proceedings*, page 531–537, Berlin, Heidelberg. Springer-Verlag.

The Object Management Group. 2023. ”omg systems modeling language (sysml) version 2.0 beta 1”. Technical report, ”OMG Systems Modeling Language”.