Uncovering Unsafe Feature Interactions in Vehicle Control Using Generative AI and Digital Twins

Laure Millet – Critical Systems Labs Inc.
Justin Kernot – Critical Systems Labs Inc.
Arun Adiththan – General Motors Company
Ramesh S. – General Motors Company
Rami Debouk – General Motors Company
Jeff Joyce – Critical Systems Labs Inc.

Uncovering Unsafe Feature Interactions in Vehicle Control Using Generative AI and Digital Twins

Laure Millet[1], Justin Kernot[1], Arun Adiththan[2], Ramesh S[2], Rami Debouk[2], Jeffrey Joyce[1]

[1] Critical Systems Labs Inc., Vancouver BC, Canada
[2] General Motors, Warren MI, USA
laure.millet@cslabs.com

Abstract. Feature interaction analysis is essential for ensuring the safety of Advanced Driver Assistance Systems (ADAS), but it is often resource-intensive. Traditionally, this process relies on expert-driven brainstorming and scenario-based testing using digital twin simulators. Recent studies suggest that Large Language Models (LLMs) can enhance these efforts by providing diverse perspectives and rapid content generation. However, effective use of LLMs in domain-specific contexts often requires complex adaptations, posing challenges for teams with limited resources. This paper explores how general-purpose LLMs can support feature interaction analysis in ADAS without complex LLM modification techniques. Through a case study, we demonstrate how LLMs can identify feature interactions and generate simulation parameters for evaluation. Our findings highlight prompt engineering as a lightweight strategy for adapting LLMs to specialized tasks and discuss the challenges faced while providing recommendations to improve their effectiveness in safety-critical applications.

Keywords: scenario generation, prompt engineering, advanced driver assistance systems (ADAS), feature interaction, large language model (LLM), digital twin.

1 Introduction

Collective human intelligence (CHI) enables groups of individuals to generate ideas and solutions for complex problems. By pooling diverse perspectives and knowledge, CHI often produces more comprehensive and creative outcomes than individual efforts alone. This collaborative approach is especially valuable and used in complex, safety-critical domains such as automotive systems, where identifying potential risks and feature interactions can be difficult and time-consuming. One common risk-reduction process involves generating a set of challenge scenarios for a given system [1]. The process typically involves a team of engineers collaborating over many days, weeks, or even months to brainstorm, select, and refine a set of candidate scenarios. These scenarios are designed to reveal unsafe behaviors in a controlled, pre-deployment environment (such as a digital twin simulation platform) allowing designers to test mitigation strategies and improve overall system safety. While this collaborative intelligence approach is effective, it is also resource-intensive, error-prone, and susceptible to biases. As automotive systems become increasingly sophisticated, there is a growing need for more efficient methods of generating test scenarios.

The emergence of Large Language Models (LLMs) offers new opportunities to enhance CHI in such contexts [2]. LLMs, with their capacity to process vast amounts of information and generate contextually relevant outputs, are ideal candidates to support idea generation and support the human brainstorming process. By refining and expanding on initial ideas, LLMs can amplify the collective intelligence of a group, providing new possibilities for collaborative ideation in fields like automotive safety. Recent studies have explored the use of LLMs in systems safety engineering for automotive applications, where researchers investigate automating scenario and digital twin parameter generation [3], [4] for Advanced Driver Assistance Systems (ADAS) [5], [6]. By leveraging the creative abilities of an LLM, it has been demonstrated that LLMs are indeed capable of generating test scenarios and simulation parameters to challenge an automotive system for safety analyses [3], [4], [5], [6]. These efforts often involve relying on methods, such as Retrieval-Augmented Generation (RAG), that use domain-specific data to enhance the models' ability to generate relevant and accurate scenarios. These methods, while effective, come with significant challenges, as they require substantial computational resources, domain expertise, and access to large datasets, which may not always be feasible, especially for resource-limited or exploratory projects [7].

This paper investigates an alternative lightweight approach: using off-the-shelf, pretrained LLMs for scenario generation without the need for fine-tuning or complex customization. The ability to generate meaningful feature interaction scenarios with general-purpose models could significantly reduce the barriers to entry for smaller teams or resource-constrained projects. Specifically, we explore whether general purpose LLMs can support scenario generation for ADAS vehicles in ways that are both practical and contextually relevant, without the need for extensive fine-tuning.

2 Relevant Work and Motivation

Given the complexity of operational design domains (ODDs) in autonomous driving, researchers in the automotive domain have begun exploring the use of LLMs to support the automated generation of diverse and realistic test scenarios aimed at uncovering problematic corner cases. Arora et al. [3] investigated the effectiveness of using LLMs to generate test scenarios for software quality assurance from a set of natural language requirements. They employed RAG to allow the LLM to retrieve relevant information from an external database before generating its response. The authors acknowledge that while this approach greatly enhances the relevance of the generated scenarios, it still struggles to capture precise action sequences and domain-specific nuances. Additionally, RAG may not always be accessible or could be challenging to implement in practice for groups lacking resources or expertise. Xu et al. [4] used LLMs to generate a diverse set of test scenarios for decision-making policies in robotic and automative applications by using mutations from a chosen seed scenario. While the researchers were successful in generating useful tests, their results indicate that the performance is highly dependent on the chosen LLM and hyperparameter tuning which adds an additional layer of complexity.

Some recent studies have extended the work further and attempted to address the challenge of translating high-level scenarios into executable test cases by instructing LLMs to generate testable code. Qiujing et al. [5] explore how LLMs can translate high-level challenge scenarios for automotive safety into simulation-ready code by using RAG in a feedback loop with a secondary LLM that critiques and proposes refinements to the generated scenarios. They found that 25% of the outputs were considered useful, suggesting some success in revealing critical behaviors. However, the implementation architecture is quite complex, and the authors do not investigate the causes of these failures or suggest ways to improve performance through framework modifications. Zhang et al. [6] also use LLMs to generate safety-critical scenarios for autonomous vehicles using RAG to augment the scenarios to match their specific simulation domain. However, while this method is more targeted to specific domain knowledge, it relies on a large database of simulation data.

It was shown in [3], [4], [5], and [6] that LLMs offer powerful natural language processing capabilities and have significant utility in generating domain-specific content. However, organizations with limited resources, such as insufficient data for fine-tuning or limited infrastructure, face challenges in adapting these models to complex, specialized tasks. The most accessible method of deploying an LLM remains its commercial-off-the-shelf (COTS) form, where users interact with the model through a browser interface or an open-source API, typically constrained to prompt input within token limits. Additionally, while prior work demonstrates the potential of LLMs in scenario generation and simulation for autonomous systems, none explicitly focus on identifying or analyzing unwanted feature interactions in automotive ADAS.

To address these constraints, our work explores strategies for leveraging LLMs effectively without resorting to resource-intensive approaches. Specifically, we investigate structured interaction methods in an applied case study aimed at identifying scenarios that may lead to unwanted feature interactions in an ADAS-equipped vehicle. These methods include task decomposition into smaller subtasks, assigning different models to specific task types to exploit their strengths, semantic prompt engineering, and various prompt-LLM interaction frameworks.

3 Background

This section provides a high-level overview of different elements and definitions that are key to understanding this work. Section 3.1 covers unwanted feature interactions in the automotive context, Section 3.2 covers digital twins in general, and Section 3.3 introduces general purpose LLMs.

3.1 Unwanted Feature Interactions

In the automotive context, a feature refers to a distinct unit of functionality that influences system behavior. These features could include various advanced driver assistance systems (ADAS); some examples of commonly implemented autonomous features include automated emergency braking (AEB) and adaptive cruise control (ACC). The term "unwanted feature interaction" refers to a situation when two or more features interact in a way that results in undesirable behaviours that could potentially put the vehicle in an unsafe state [8]. An example of this can be seen in **Fig. 1**, where a vehicle is receiving conflicting steering commands between the lateral collision avoidance (LCA) and the lane-changing (LX) feature.

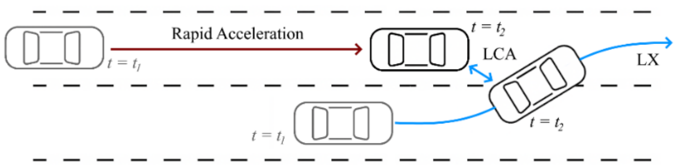

Fig. 1. A cartoon, bird's-eye view of a feature interaction occurring between the lane changing (LX) and lateral collision avoidance (LCA) of an ego vehicle, resulting in conflicting steering commands between the two features

In complex systems with multiple active features that can interact in numerous ways, detecting unwanted feature interactions is challenging due to the complexity of the system and its operational domain. One common way that the feature interactions are identified is through scenario testing in a digital twin, where challenge scenarios are executed to observe the behaviours and interactions of the ADAS features [9]. For example, the scenario that led to the unsafe state shown by **Fig. 1** can be described as follows: i) the ego vehicle LX system initiates a lane change when path is clear, ii) as the ego vehicle is crossing the centerline to change lanes, another vehicle rapidly accelerates from behind, iii) the ego vehicle detects the other vehicle and the LCA system activates.

To thoroughly challenge the system, an exhaustive set of well-defined scenarios must be established to ensure that all the possible unwanted feature interactions are identified early in the design process for the automotive system. However, creating an exhaustive set of test scenarios that lead to unwanted feature interactions remains a major challenge in systems safety engineering.

3.2 Digital Twins

A digital twin is a virtual representation of a physical system that mirrors its structure, behavior, and real-time operation. In the automotive domain, digital twins are used to simulate entire vehicles or subsystems by integrating sensor data, control logic, and environmental conditions. These models enable engineers to test system behavior in diverse scenarios without extensive physical trials. In safety-critical domains, they offer a powerful tool for identifying unwanted feature interactions. For example, frameworks like the Autonomous Driving Digital Twin (ADDT) have been used to explore such interactions under edge-case scenarios that are difficult to uncover through traditional testing methods [10]. Similarly, fault injection techniques within digital twins allow for the simulation of sensor failures or communication disruptions, providing insights into how such anomalies affect vehicle behavior and safety [11]. Despite their potential, configuring digital twins for meaningful scenario analysis remains a challenge. Parameterizing these simulations to explore relevant safety scenarios demands deep domain expertise and often lacks standardized methodologies.

3.3 General Purpose LLMs

General purpose LLMs such as OpenAI's GPT-4 [12], Anthropic's Claude [13], and Meta's LLaMa [14] are general-purpose AI systems trained on vast datasets that span books, articles, code, and web content. Their training enables them to generate coherent and contextually relevant text across a wide range of topics, making them useful for tasks like idea generation, translation, and conversational agents without requiring task-specific fine-tuning [12]. However, when applied to domain-specific applications, such as automotive safety engineering, LLMs often encounter some limitations due to their lack of grounding in the nuanced language and specialized reasoning required in these fields [15]. To address these limitations, practitioners increasingly rely on prompt engineering and LLM interaction techniques to incorporate domain-specific context into the model's inputs. By carefully crafting prompts, users can guide LLMs toward more accurate and useful outputs without modifying the underlying model.

4 LLMs and Prompt Engineering

Prompt engineering is the process of strategically designing task-specific instructions (prompts) to guide the LLM output without altering model parameters [16]. This is a beneficial process since it allows users to have highly tailored LLM responses to a specific task without the need for fine-tuning or other more complex methods. The term "prompt engineering" can refer to a number of methods, including semantic guidance on how to present information and structure sentences, or changing the number and type of prompt interactions that the user has with the LLM. When designing a prompt, there are many semantic strategies that are commonly used and many of them depend on the specific task that is being addressed by the LLM [16], [17], [18], [19], [20]. However, several strategies are emerging that are widely accepted as best practices. The following list presents principles for effective prompt design that are applicable to nearly all applications, and were applied in this work: 1) be clear and explicit with the task, 2) provide the domain specific context to frame the problem/task, 3) break down complex tasks into smaller and distinct tasks, 4), outline the output format, and 5) add constraints to keep the LLM on task and produce responses that meet an acceptable standard. While this list is not exhaustive, it provided a foundational set of principles that guided the semantic and content design of the prompts developed in this work. In addition to this semantic guidance, prompt engineering also refers to the methods by which an LLM is prompted, not the design of the content of the prompt. Examples of LLM interaction frameworks that are common include zero-shot prompting, prompt chaining, and active prompting [16], [21]. In this work, we focus on employing the following list of strategies across multiple LLM tasks:

- **Single Prompt (Zero-shot)**: Providing instructions or task requirements (without examples) to the LLM in a single prompt and expecting the appropriate result in a single response.
- **Multi-shot History (Negative Constraints)**: Similar to the previous framework, but with the prompt containing previous outputs from the LLM for the same task performed in prior sessions. These outputs serve as both few-shot examples and exclusion constraints.
- **Prompt Chaining (Recursive Query):** Providing follow-up prompts to the LLM once it has provided a response, using the information produced in the session as part of the next result.
- **Chain-of-Verification (Internal Review):** Two prompts are created: the first contains the main instructions, and the follow-up contains a list of verification questions for the LLM to check its work.
- **Multi-Model Verification (External Review):** In this framework, two independent LLM sessions are active; one generates information, while the second verifies the response and suggests corrections. This method intentionally separates the generation and verification tasks to mitigate token bias from previous responses.

5 Case Study Definition

To evaluate the strategies outlined in Section 4, we conducted a targeted case study within an automotive application, aiming to enhance the system safety engineering process. The system chosen for this study is an autonomous automotive system incorporating several unique ADAS features. Using a distinct system design encourages the model to generate scenarios based on proprietary knowledge and novel features, rather than relying on real-world data from the model's training set. This strategy promotes the generation of relevant scenarios tailored to the unique design of the system without the need for an additional database or finetuning.

The system includes five ADAS features: lane centering (LC), autonomous lane changing (LX), lateral collision avoidance (LCA), adaptive cruise control (ACC), and automated emergency braking (AEB). Each feature has been implemented in the digital twin simulator given the system's specific characteristics, with each feature also exhibiting an intentionally unconventional design/behaviours. For example, the LCA system and its rules for combining LCA and LC signals are designed specifically to contrast typical implementations in real automotive applications. In doing so, the system challenges the LLM's pseudo-reasoning capabilities by presenting new information that may not be part of its existing knowledge base.

Using the open-source Modelica language, this automotive system was modeled as a digital twin (DT) simulation and integrated into a custom library. The scope of the DT environment and scenario capabilities is better described by **Fig. 2**. The DT is restricted to a specific set of scenarios which simulate an ego vehicle controlled by the previously described ADAS features interacting on a flat road with up to three other vehicles. The interaction of the ADAS features, the environment, and the other agents on the road create an emergent behaviour of the ego vehicle.

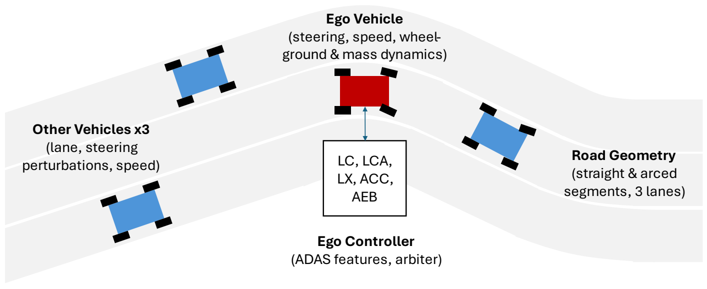
Ego Vehicle
(steering, speed, wheel-ground & mass dynamics)
Other Vehicles x3
(lane, steering perturbations, speed)
LC, LCA, LX, ACC, AEB
Road Geometry
(straight & arced segments, 3 lanes)
Ego Controller
(ADAS features, arbiter)

Fig. 2. Visual representation of the scope and limitations of the DT simulator, highlighting the maximum number of vehicles, lanes, and ADAS features.

6 Approach

In this case study, the high-level task is to generate a set of scenarios and scenario parameters that result in an unsafe feature interaction between the ADAS features described for the automotive system in **Fig. 2**. These scenarios must be unique, logically lead to an unsafe vehicle state due to feature interactions, and generate parameters that represent the scenario to be simulated in the DT described in Section 5. As outlined in Section 4, a key strategy for developing a robust prompt is to break down complex LLM tasks into smaller, dedicated sub-problems that can be solved individually. From the larger task as defined previously, three key sub-tasks can be identified, each targeting a distinct problem that can be addressed with specific goals for the LLM.

The first task (Task 1) involves producing a diverse set of high-level scenarios that could potentially lead to unwanted feature interactions. This task demands creativity, adherence to system constraints, and the ability to generate unique yet relevant situations. Several prompt strategies were considered for this task. Firstly, the zero-shot approach was applied and involved asking the LLM to generate *N* scenarios in a single output. Secondly, prompt-chaining was attempted, where the first prompt provided the task description and context, and follow-up prompts requested additional unique scenarios, each building on the previous responses. Finally, a multi-shot history approach was employed, where scenarios from prior outputs were included as examples with negative constraints to prompt the LLM to generate scenarios that were not part of the set.

Once scenarios are generated, they must be structured into logically coherent sequences of causal events. This task (Task 2) aims to ensure continuity, causality, and logical correctness while maintaining consistency with the original high-level scenario and the defined system behavior. The first prompting strategy considered for this task involved a zero-shot approach, where the LLM was simply asked to generate a sequence of events from a given high-level scenario and system description. The second prompt strategy tested was the chain-of-verification method, where the LLM first generated a sequence of events in one step, and a follow-up prompt requested the LLM to review and refine its output based on a set of verification criteria. Finally, the third prompt strategy involved the multi-model verification approach, in which two independent LLMs interacted with each other; one LLM was responsible for generating a sequence of events, and a second LLM reviewed the output based on a set of verification criteria and gave correction instructions.

Finally, Task 3 takes the structured event sequences as input and translates the scenario into a set of simulation-ready parameters that recreate the scenario in the DT. This process involves converting the natural language descriptions from the high-level scenario and sequences numerical values in the predefined code snippets template to ensure compatibility with the Modelica language. Only a zero-shot approach was employed here, with the LLM being asked to generate the required parameters in a single response from a detailed template that provides some specific formatting guidelines and the task description. The decision to test a single prompt method here was made because it is well known that LLMs do not internally propagate physical dynamics and, therefore, cannot provide deterministic parameters for a simulation [22]. Instead, they provide a reasonable approximation for value based on context and their internal knowledge. While it has been shown that more complex architectures can be employed to have an LLM review and correct simulation parameters based on errors [5], the objective was to employ a strategy that does not require substantial expertise or resources to implement.

For each task, a brief prompt engineering phase was performed using the semantic guidance provided in Section 4. In order to better understand prompt-to-output correlations, we iteratively made single adjustments using low temperature settings to reduce stochastic influences in the models. The prompt engineering phase reinforced the importance of precise definitions and clear instructions in guiding LLMs to generate relevant, accurate, and consistent outputs. Furthermore, it was observed that certain LLMs yielded better results based on the outlined tasks. Ultimately, Task 1 used OpenAI's GPT-4o-mini and Tasks 2-3 used Meta's LLaMa3-2-90b. Following this, an evaluation was performed where each of these prompt strategies was evaluated against predefined criteria.

7 Results

In Task 1, the three prompt strategies described in Section 6 were evaluated on a set of 10 generated scenarios. The evaluation of these scenarios was based on several criteria, including uniqueness, complexity, variance, quality, and novelty. Uniqueness was measured as the percentage of responses containing unique scenarios, while complexity evaluated the percentage of total possible scenario elements utilized. Variance assessed the categorical variance of potential scenario elements across all generated scenarios. Quality was a subjective score (normalized from a 1-5 scale) based on predefined criteria: 5 – no issues, 4 – system misunderstanding, 3 – system/task misunderstanding, 2 – multiple system/task misunderstandings, 1 – no feature interactions in scenario. Novelty was measured by the percentage of scenarios that were substantially different from the 16 scenarios generated by human experts during a one-hour working session. The evaluation results for the three prompt strategies are shown in **Table 1**.

Table 1. Task 1 evaluation results.

Criteria	Zero-Shot	Chain Prompt	Multi-Shot	Human 1	Human 2
Uniqueness	0.90	0.80	0.80	1.0	1.0
Variance	0.14	0.50	0.66	0.03	0.08
Complexity	0.40	0.82	0.70	0.45	0.49
Quality	0.78	0.47	0.78	0.60	0.98
Novelty	0.67	1.0	1.0	-	-

Among the different prompt and model combinations, the zero-shot prompt outperformed the other approaches. This performance difference can likely be attributed to LLM limitations with negation and negative constraints [23]. Additionally, when testing the prompt chaining and multi-shot history methods, it was observed that the models often misinterpreted the "uniqueness" requirement, which seemed to be interpreted as semantic uniqueness rather than content uniqueness. Furthermore, it is hypothesized that self-consistency is more easily maintained when LLMs are tasked with producing a single output as opposed to multiple outputs with regards to having unique responses. When comparing the LLM performance to human-generated results, it is notable that while human participants (who had one hour to generate scenarios) created 7-9 scenarios each, the LLM generated 10 scenarios in mere seconds. Despite the speed, the complexity of the generated scenarios matched human output, demonstrating a suitable level of complexity, novelty, and uniqueness in the generated scenarios. This result indicates that despite using significantly fewer resources, LLM-assisted scenario generation can approach the performance of a human executing the same task.

For Task 2, the three prompt strategies were tested on a set of six high-level scenarios of varying quality to evaluate the models' ability to correct logical inconsistencies and maintain causality. The evaluation criteria included causality correction, which was rated on a scale of 0-1 based on how well the model identified and corrected logical inconsistencies; causality quality, which measured the percentage of steps that were logically connected to the following steps; the quality of initial conditions, rated on a scale of 0-1; and continuity, which was measured by the accuracy of the model's event sequence compared to the original high-level scenario. While these criteria were mostly based on a subjective scoring system, a rubric was established a priori, and the evaluation was carried out by a competent team of system engineers with the necessary domain knowledge and expertise. The evaluation results for the three prompt strategies applied to Task 2 are shown in **Table 2**.

In this task, the chain-of-verification method performed the best compared to the zero-shot and multi-model verification approaches when considering causality performance and quality. Surprisingly, the zero-shot approach was of a comparable performance to the chain-of-verification method with a better continuity score. This is because if the scenario had a fundamental causality issue, the chain-of-verification method would identify and alter the generated scenario, resulting in a deviation from the reference scenario. Despite having the worst overall performance, the multi-model verification method performed the verification and review to a high degree of quality and could be a valid approach to review, however, the corrections made to the generated scenarios were often too extreme and lead to poor output quality. Although the results were close with respect to the different interaction frameworks, a larger sample size might have provided more definitive insights into the relative effectiveness of each method.

Table 2. Task 2 evaluation results.

Criteria	Zero-Shot	Chain-of-Verification	Multi-Model Verification
Causality Correction	0.72	0.72	0.67
Causality Quality	0.87	0.89	0.77
I.C. Quality	0.83	0.89	0.83
Continuity	1.0	0.50	0.50

In Task 3, only the zero-shot strategy was attempted, and the performance of all models was evaluated by running the simulations in the DT. The responses for Task 3 were tested within the DT framework to see how well the model's output matched the expected scenario outcomes. The results demonstrate that using the framework outlined in this paper, it is possible for an LLM to generate DT parameters for scenarios that lead to unwanted feature interactions, with minimal human intervention. In some instances, the LLM produced parameter sets compatible with the DT simulator, successfully recreating the scenario and predicting the unwanted feature interactions described in the parent scenario. However, this occurred only about 25% of the time. In most cases, the LLM-generated parameters did not align with the expected outcomes described by the scenario, and thus no feature interactions were observed in the simulation. In cases where the parameters did not lead to the intended outcome, it was found that roughly 2-3 manual parameter adjustments (which equated to roughly 10% of the total parameter count) per output were necessary to correct the scenario. For example, if two vehicles were intended to interact but did not due to a temporal or positional mismatch, modifying the initial relative position of the vehicles, given their speeds, was sufficient to resolve the issue. This implies that despite the LLM having imperfect responses, a considerable amount of the data returned could be used as a baseline for the digital twin.

These are expected outcomes for the performance of the LLM with respect to Task 3, since LLMs merely provide reasonable estimates for parameters, rather than verified values based on deterministic dynamics propagation. While the LLM's outputs cannot guarantee exact values, there is still a possibility that the LLM may "stumble" upon the correct answer without requiring further modifications to result in the expected outcomes in the simulation. These results indicate that even though using an LLM does not guarantee perfect accuracy for simulation parameter generation, it can still serve as a valuable tool. By assisting engineers, the LLM can perform much of the "heavy lifting" and provide a reasonable starting point for further corrections, rather than requiring the engineer to manually synthesize parameters from scratch.

8 Conclusion

For organizations without the resources to fine-tune LLMs, alternative strategies can still yield practical and effective results. This work demonstrated that LLMs can generate a set of candidate scenarios, decompose scenarios into a set of logical events, and generate digital twin parameters that correspond to a given scenario. Despite the results being imperfect, this method still offers a means to significantly reduce the human overhead required to generate concept scenarios and a strong baseline estimate for their respective simulation parameters. By leveraging structured prompting, decomposition, and verification techniques, domain-specific applications of LLMs become more feasible.

Additionally, this work supports the idea that not all LLMs are interchangeable when used for the same tasks, as some exhibit better performance in a given task type. It is therefore important to consider the application in which the LLM is being applied and consider the nature of the task when selecting an LLM. A notable aspect of our method is the use of a multi-LLM setup, where different models are selected for tasks based on empirical performance, paired with tailored prompt interaction strategies. This modularity enhances flexibility and enables more effective use of general-purpose models without complex infrastructure. Furthermore, the results of this work indicate that the readiness for using LLMs in a fully automated pipeline might not be feasible at the time of writing this paper. However, using LLMs with humans in the loop might help reduce overhead in brainstorming tasks and potentially open new avenues for discussion and idea generation.

The evaluation methods presented in this paper for assessing the performance of LLMs on Tasks 1-3 provide a lightweight yet effective metric for gauging their capabilities. While these methods may be unconventional, they are highly transferrable across various domains, offering valuable insights for LLM evaluation. This approach not only aids in assessing model effectiveness but also serves as a practical framework for guiding prompt engineering and design decisions for implementing LLMs in practical applications.

8.1 Limitations and Threats to Validity

The methods used to evaluate both the performance of the LLMs and the proposed approach rely on metrics that require expert knowledge and involve subjective judgment. As a result, the findings may not be consistently replicable across different evaluators and carry a degree of uncertainty. Additionally, the small sample size limits the statistical significance of the results, making it difficult to draw generalizable conclusions. This study should therefore be viewed as offering preliminary insights about the effectiveness of the proposed methods and the performance of LLMs in similar applications.

Beyond these methodological concerns, LLMs have known limitations that constrain the capability of this work. Firstly, LLMs are known to produce hallucinations or incorrect outputs that can be subtle and difficult for human reviewers to detect, potentially leading to incorrect information propagating through the proposed framework.

However, since scenarios are validated using a digital twin, hallucinations are likely to be caught during simulation.

In addition to hallucinations, the models used in this paper are known to not have the ability to propagate dynamics through time and can therefore not generate a mathematically proven set of parameters. Despite this, the LLM generated parameters for a given scenario perfectly 25% of the time; the remaining 75% of the time yielded a set of scenarios where only 10% of the parameters generated required manual adjustment. As system complexity increases, it is possible that the error rates will also increase and ultimately require more human oversight to correct the issues. However, as LLMs improve overtime, the frequency and severity of such risks (e.g., hallucinations and incorrect numerical generation) might be substantially improved and not be a source of significant risk of this work, though this is a speculative assumption.

8.2 Future Work and Opportunities

To enhance scenario diversity and coverage, future work will explore guiding the LLM to generate scenarios based on targeted parameter sets, using strategies inspired by combinatorial testing to expose more corner cases. We also aim to improve the scalability of our experiments to yield statistically significant insights. Additionally, we plan to investigate methods to automate the iterative review and adjustments required for Task 3. Currently, an expert is required to review the simulation outputs and identify incorrect parameters based on the output of the simulation. To mitigate this effort and minimize the amount of human involvement, we plan to include an iterative feedback loop where simulation results from the digital twin are fed back into an LLM along with the simulation errors and outcome descriptions, enabling it to refine scenario translations when discrepancies arise due to misaligned numerical representations.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. S. Diemert, A. Casey and J. Robertson, "Challenging Autonomy with Combinatorial Testing," in 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2023.
2. J. Burton, E. Lopez-Lopez, S. Hechtlinger, Z. Rahwan, S. Aeschbach, M. Bakker et al., "How large language models can reshape collective intelligence," Nature Human Behaviour, pp. 1643-1655, 01 09 2024.
3. C. Arora, T. Herda and V. Homm, "Generating test scenarios from NL requirements using retrieval-augmented LLMs: an industrial study," in Requirements Engineering, Vienna,Austria, 2024.
4. W. Xu, H. Pei, J. Yang, Y. Shi, Y. Zhang and Q. Zhao, "Exploring critical testing scenarios for decision-making policies: an LLM approach," in arXiv, 2024.
5. L. Qiujing, W. Xuanhan , J. Yiwei , Z. Guangming, . M. Mingyue and F. Shuo , "Multimodal large language model driven scenario testing for autonomous vehicles," in Artificial Intelligence, Automation and High Performance Computing, Zhuhai, China, 2024.

6. J. Zhang, C. Xu and B. Li, "ChatScene: knowledge-enabled safety-critical scenario generation for autonomous vehicles," in Computer Vision and Pattern Recognition, Seatle WA, USA, 2024.
7. S. Zhao, Y. Yang, Z. Wang, Z. He, L. K. Qiu and L. Qiu, "Retrieval augmented generation (rag) and beyond: A comprehensive survey on how to make your LLMs use external data more wisely," arXiv preprint arXiv:2409.14924, 2024.
8. A. L. J. Dominguez, "Feature interaction detection in the automotive domain," in 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, 2008.
9. L. Birkemeyer, T. Pett, A. Vogelsang, C. Seidl and I. Schaefer, "Feature-interaction sampling for scenario-based testing of advanced driver assistance systems," in Proceedings of the 16th International Working Conference on Variability Modelling of Software-Intensive Systems, 2022.
10. B. Yu, C. Yuan, Z. Wan, J. Tang, F. Kurdahi and S. Liu, "ADDT - digital twin framework for proactive safety validation in autonomous driving systems," arXiv:2504.09461, 2025.
11. D. Bergin, W. L. Carden, K. Huynh, P. Parikh, P. Bounker, B. Gates and J. Whitt, "Tailoring the digital twin for autonomous systems development and testing," The ITEA Journal of Test and Evolution, vol. 44, no. 4, 2023.
12. OpenAI, "GPT-4 technical report," arXiv preprint arXiv:2303.08774, 2023.
13. Antrhopic, "The Claude 3 Model Family: Opus, Sonnet, Haiku," 2024.
14. A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, et al., "The LLaMa 3 herd of models," arXiv preprint arXiv:2407.21783, 2024.
15. R. Bommasani, D. A. Hudson, E. Adeli, R. B. Altman, S. Arora, S. von Arx, et al., "On the opportunities and risks of foundation models," arxiv:2108.07258, 2021.
16. P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal and A. Chadha, "A systematic survey of prompt engineering in large language models: techniques and applications," arXiv preprint arXiv:2402.07927, 2024.
17. J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert et al., "A prompt pattern catalog to enhance prompt engineering with ChatGPT," arXiv preprint arXiv:2302.11382, 2023.
18. J. D. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann and Q. Yang, "Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts," in Proceedings of the 2023 CHI conference on human factors in computing systems, 2023.
19. W. C. Choi and C. I. Chang, "A survey of techniques, key components, strategies, challenges, and student perspectives on prompt engineering for large language models (LLMs) in education," Preprint, 2025.
20. S. Maaz, J. C. Palaganas, G. Palaganas and M. Bajwa, "A guide to prompt design: foundations and applications for healthcare simulationists," Frontiers in Medicine, vol. 11, p. 1504532, 2025.
21. J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," in Advances in Neural Information Processing Systems, 2022.
22. M. Ali-Dib and K. Menou, "Physics simulation capabilities of LLMs," Physica Scripta, vol. 99, no. 11, p. 116003, 2024.
23. J. Jang, S. Ye and M. Seo, "Can large language models truly understand prompts? a case study with negated prompts," in Transfer learning for natural language processing workshop, 2023.
24. Y. Wang, Z. Zhang, H. Chen and H. Shen, "Reasoning with large language models on graph tasks: the influence of temperature," in 2024 5th International Conference on Computer Engineering and Application (ICCEA), 2024.