Can Large Language Models Assist with SOTIF Scenario Generation?

Erin Cyffka, Critical Systems Labs Inc., Vancouver, BC, Canada
Simon Diemert, Critical Systems Labs Inc., Vancouver, BC, Canada
Arun Adiththan, General Motors, Warren, MI, USA
Rami Debouk, General Motors, Warren, MI, USA
Ramesh S., General Motors, Warren, MI, USA
Justin Kernot, Critical Systems Labs Inc., Vancouver, BC, Canada
Jeff Joyce, Critical Systems Labs Inc., Vancouver, BC, Canada

Can Large Language Models Assist with SOTIF Scenario Generation?

Erin Cyffka
Critical Systems Labs Inc.
Vancouver, Canada
erin.cyffka@cslabs.com

Simon Diemert
Critical Systems Labs Inc.
Vancouver, Canada
simon.diemert@cslabs.com

Arun Adiththan
General Motors
Warren, Michigan, USA
arun.adiththan@gm.com

Rami Debouk
General Motors
Warren, Michigan, USA
rami.debouk@gm.com

Ramesh S
General Motors
Warren, Michigan, USA
ramesh.s@gm.com

Justin Kernot
Critical Systems Labs Inc.
Vancouver, Canada
justin.kernot@cslabs.com

Jeffrey Joyce
Critical Systems Labs Inc.
Vancouver, Canada
jeff.joyce@cslabs.com

***Abstract*—Combinations of operating conditions can trigger a system to behave in a hazardous manner, even in the absence of malfunction. ISO 21448 - Road vehicles - Safety of the intended functionality (referred to as "SOTIF") describes strategies for managing this type of risk in automotive systems. One strategy includes the identification of operating scenarios that might lead to the occurrence of a hazard. Crafting scenarios is a technically challenging and labor-intensive task that requires sustained creative engagement, and the consequence of inadequate SOTIF analyses can be severe. This paper introduces Heraclitus, an engineering method and prototype software tool for performing SOTIF scenario generation with the support of a large language model. Large language models are a novel type of generative artificial intelligence targeted at natural language processing and generation that exhibit remarkable performance in a range of natural language applications that have historically been difficult for conventional artificial intelligence systems. As such, there is an opportunity to use these models, in collaboration with humans, to support SOTIF scenario creation. The goal of Heraclitus is to allow analysts to rapidly produce a comprehensive set of SOTIF scenarios that can be used as the basis for on-going SOTIF risk management. A preliminary control trial of Heraclitus was conducted, in which six system safety experts were asked to create SOTIF scenarios with and without the support of a large language model. Results indicate that these models show promise in supporting SOTIF analysis and are capable of generating useful SOTIF scenarios.**

I. Introduction

Modern automotive systems, including driver assistance or autonomous systems, depend on advanced technology to operate safely. For instance, the perception systems in autonomous vehicles depend heavily on machine learning models to interpret sensor data and generate (in real-time) model of the world near the vehicle which can be used to make motion planning decisions [1]. Even without a technical malfunction, such as failure of a sensing hardware or a software defect, it is possible for these systems to fail due to insufficiencies in processing a diverse range of conditions. Safety of the Intended Functionality ("SOTIF"), as contemplated by ISO 21448 [2], provides a framework and guidance on how to manage safety risk arising from a diverse range of known and unknown triggering conditions.

An important step in this process is identifying sequences of scenes, events, actions, and goals or "scenarios" that might lead to the occurrence of a hazard [2]. These scenarios form the basis of subsequent SOTIF risk management activities. Currently, identifying scenarios depends heavily on the capabilities of expert human analysts to craft scenarios, which is a technically challenging and labor-intensive task that requires sustained creative engagement. Further, the consequence of inadequate SOTIF analyses can be severe.

Large Language Models (LLMs), a novel type of machine learning model, are targeted at natural language processing and generation. Though their core function is to predict the next word in a sequence, based on the previous words and their statistical relationships, they have exhibited remarkable performance in a range of natural language applications. As such, there is an opportunity to deploy LLMs to support SOTIF analysis for Advanced Driver Assistance System (ADAS) and autonomous vehicles.

The contribution of this paper is *Heraclitus*, a human-LLM co-operative method and accompanying software tool that uses an LLM to enhance the productivity and quality of human-led SOTIF scenario generation. Three research questions regarding the feasibility and utility of the Heraclitus method are explored, with results indicating that LLMs, when used according to the Heraclitus method, can produce useful SOTIF scenarios and may have the potential to meaningfully support human analysts in SOTIF scenario generation.

Section 2 of this paper provides background information on SOTIF and LLM functionality. Section 3 introduces the Heraclitus method and software tool; Section 4 describes the evaluation methods used; Section 5 contains the results of the evaluation; Section 6 provides a summary of related work; and Section 7 concludes with discussion and future work.

II. Background

This section provides information on two key components of this work: SOTIF analysis and LLM functionality.

A. Safety of the Intended Functionality (SOTIF)

Combinations of operating conditions can trigger a system to behave in a hazardous manner, even in the absence of malfunction. SOTIF provides guidance on how to manage safety risk arising from a diverse range of known and unknown triggering conditions, formally defining SOTIF as "absence of unreasonable risk due to hazards resulting from functional insufficiencies of the intended functionality or its implementation" [2]. Importantly, SOTIF risk arises from the inability ("insufficiency") of the technology to handle the triggering conditions, not malfunctioning system components. Additionally, SOTIF includes the concept of reasonably foreseeable misuse by persons operating or near the vehicle.

Sequences of scenes, events, actions, and goals or "scenarios" are an important concept in SOTIF and are the basis of risk-management activities [2]. A scenario might contain a "triggering condition" that initiates potentially hazardous behavior of the system, though the occurrence of a trigger does not guarantee that hazard will occur. A trigger might reveal an insufficiency of a system resulting in a hazardous event. It is also possible, however, that the vehicle might not be in a situation where the hazardous event will cause harm or the operator of the vehicle might be able to intervene and prevent the hazard. The guidance in ISO 21448 is framed around the categorization of scenarios as known versus unknown and hazardous versus not hazardous. The objective of SOTIF activities is two-fold: 1) identify as many (hazardous) scenarios as possible, such that the set of unknown scenarios is reduced, and 2) mitigate risk arising from known (and possibly unknown) hazardous scenarios. A simple example of a SOTIF scenario for a vehicle perception system might be: *The vehicle is driving at night time. A pedestrian wearing dark clothing crosses the street in front of the vehicle. Due to the low lighting conditions, the camera-based perception system does not detect the pedestrian until it is too close the vehicle.*

B. Large Language Models

Despite their impressive capabilities, the core function of an LLM is to predict the next word in a sequence, based on the previous words and their statistical relationships [3]. Relationships between word fragments are encoded as numerical parameters in the LLM's internal neural network(s), which are learned by processing large volumes of textual training data. The number of numerical parameters (on the scale of billions for the largest models) in the LLM is often correlated with the language processing capabilities of the model [4].

An important consequence of the word prediction approach outlined above is that LLMs do not perform reasoning tasks in the same way humans do, e.g., making abstractions of reality, applying inferences, arriving at conclusions. Rather, the fact that the statistical structure of LLMs reflects, to at least some extent, human reasoning is a convenient reality that allows the output of LLMs to be used in a wide range of applications. Despite their utility, LLMs are ultimately "stochastic parrots," and their output should be treated with skepticism [3].

LLMs occasionally exhibit a phenomenon referred to as "hallucination", in which they produce outputs that appear to be plausible, but are, in fact, inaccurate [5]. The potential risks introduced by LLM hallucination should be carefully considered when designing applications that depend on LLMs [6]. In some cases, it might be advantageous for LLMs to hallucinate, as this might cause human users to pause and examine a situation more carefully which could ultimately lead to new insights [7].

Requests, along with any necessary instructions or background information, are sent to LLMs using a "prompt". As prompt content and style can have a significant impact on the quality of the results produced, "prompt engineering" is used for prompt refinements. Prompt length is limited by the size of the LLM's context window, i.e., the number of word fragments that it can use to predict the next word in the sequence. Commercially available LLMs have context windows on the order of 128,000 word fragments (about 96,000 words).

III. Heraclitus

This section introduces the Heraclitus[1] method and tool, which aim to identify and prioritize SOTIF scenarios for a system operating within a defined environment. It provides LLM support for a co-operative, human-led scenario generation, aiming to improve productivity and enrich results.

A. The Heraclitus Method

The Herclitus method has four steps: 1) model the system, 2) model the environment, 3) generate scenarios, and 4) rank scenarios. These steps are depicted in Fig. 1 and discussed in detail below.

To illustrate the method, this section will refer to a fictitious Adaptive Cruise Control (ACC) ADAS feature as a running example. The ACC uses a forward radar sensor to measure the relative speed and distance to a vehicle (the "forward vehicle") in front of the ego vehicle. If a forward vehicle is present, the ACC adjusts the speed of the ego vehicle to maintain a minimum longitudinal separation. Additionally, the ACC maintains the speed of the ego vehicle at or below a set point provided by the vehicle operator.

Step 1: Model the System

The first step of the Heraclitus method is to provide a model of the system that is composed of components, interactions, hazards, and a free text description.

Components correspond to functional elements of the real system such as sensors or control units. In the ACC example, components include a forward radar, ACC controller, and propulsion controller.

Interactions describe relationships between a source and target component and typically also include a condition expressed in natural language. In the ACC example, an Interaction from

[1]This method and software tool are named after the ancient Greek philosopher, Heraclitus, who is remembered for the observation: you cannot step into the same river twice. This references the fact that a river is in a constant state of flux and transformation. Like its namesake, this method and software tool are focused on the unbounded variability of scenarios that can be generated from a set of elements in the operating environment of a system.

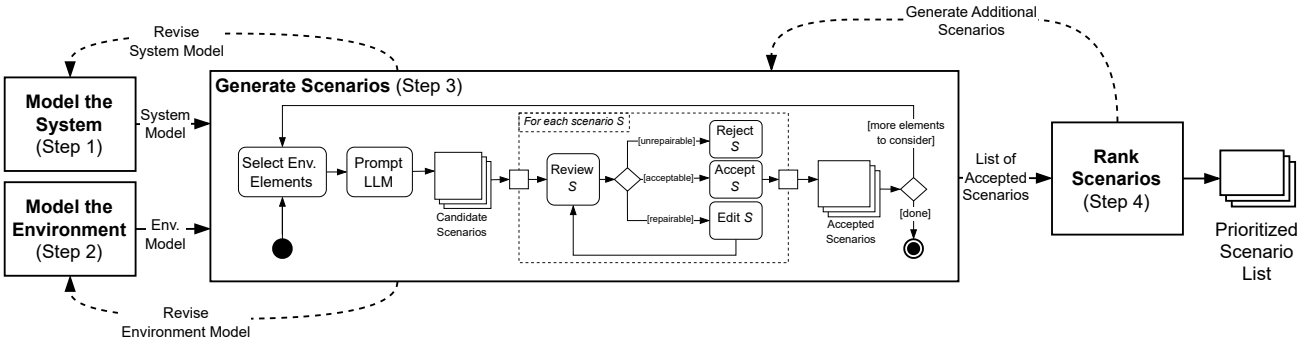

Fig. 1. Overview of the Heraclitus method.

the feature's ACC controller component and to the propulsion controller might have the description: "when the ego vehicle's speed falls below the provided speed setpoint, then the ACC controller requests acceleration from the propulsion controller."

Hazards describe conditions that, if they occur, might contribute to harm. The system model captures a simplified expression of these hazards that includes a name and definition. In the ACC example, a hazard might be "loss of longitudinal separation" with the description "this hazard occurs when the longitudinal separation between the ego vehicle and forward vehicle is less than 1 second (in terms of time-to-collision)".

Step 2: Model the Environment

The second step of the Heraclitus method is to provide a model of the system's operating conditions or environment composed of elements arranged in a recursive structure where each element can have child elements that are more specific. For instance, a top-level Element might be "precipitation" with child elements "rain", "snow", "fog", and so on.

Elements optionally have attributes with predefined values that describe how the element's manifestation might vary in a scenario. For example, the element "precipitation" might have an attribute "intensity" with values: "none", "light", "moderate", "heavy", and "extreme". In subsequent analysis steps, a list of elements and their corresponding attributes (including potential values) is sent to the LLM, which may incorporate the attribute values into the scenarios it generates. Child elements inherit their parent's attributes.

Step 3: Generate Scenarios

The third step of the Heraclitus method is to generate SOTIF scenarios with the aid of an LLM. A scenario consists of a description, trigger, hazard, rationale, and guideword. The description contains a sequence of events or conditions leading to the occurrence of a hazard. The trigger is the triggering condition in the scenario (e.g., "heavy rain reduces the detection performance of the forward radar"). The hazard is a condition of the system that occurs in the scenario that is likely to precede harm. The rationale contains additional narrative about why the trigger might result in an occurrence of the hazard. The guideword is a word or phrase that helps to interpret the trigger (e.g., "not enough").

To generate scenarios, the analyst first selects one or more environmental elements, which are provided to an LLM along with a description of the system. The LLM returns a "batch" of one or more generated scenarios that are referred to as "candidates". The human analyst must then review the candidate scenarios and accept only those that they deem useful. Analysts can edit candidate scenarios to correct small mistakes prior to acceptance or reject scenarios that are not repairable. Analysts may also manually craft scenarios during this step. This workflow is repeated for different combinations of environmental elements, using the analyst's intuition to guide element selection.

For simplicity, Heraclitus uses a "zero-shot" prompting approach, where all the information about the system and environment is provided in a single prompt without any examples or additional information provided, and where the analyst is unable to interact with the LLM to refine results. Prompts are formulated by composing information from the system model and the selected environmental elements in a prompt template as shown in Fig. 2.

Step 4: Rank Scenarios

The final step of the Heraclitus method is to rank the accepted list of scenarios according to some predefined criteria, such that the highest priority scenarios appear at the top of the list. Given many scenarios and limited engineering resources, this ranking makes it easier to focus on the highest priority scenarios when designing mitigations. In the prototype implementation of the Heraclitus software tool, two criteria were chosen for consideration: 1) likelihood - a subjective assessment of how frequently the scenario is likely to occur for the real system; and 2) novelty - an assessment of whether the scenario challenges the system in a unique or unexpected manner that reveals a functional insufficiency. While severity is also an important criterion, it is already associated with the scenario's hazard (assuming the hazards are assessed using the method from Part 3 of ISO 26262) [8], and is thus already accounted for in the ranking procedure.

Ranking is performed with a pairwise comparison method where a systematic procedure is used to select the next pair of scenarios to compare, and a human analyst indicates which

You are an expert in autonomous vehicle safety engineering. Given the system description below, please generate exactly $$NUM_SCENARIOS$$ scenarios [...]
Please format your response as a syntactically correct array of $$NUM_SCENARIOS$$ JSON objects [...]
The system description and environmental conditions are as follows:
Section 1) System:
  Section 1.1) Description Summary: $$SYSTEM_DESCRIPTION$$
  Section 1.2) Components: $$COMPONENT_LIST$$
  Section 1.3) Allowable Interactions Between Components: $$INTERACTION_LIST$$
  Section 1.4) Potential Hazards: $$HAZARD_LIST$$
Section 2) Environment:
  Section 2.1) Environmental Elements: $$CONDITION_LIST$$
Section 3) Guide Phrases: $$GUIDE_WORD_LIST$$

Fig. 2. Prompt template. The $$ notation represents a placeholder for parameters or information from the system model and selected environmental elements.

of the two scenarios in the pair is preferred for each of the criteria.

B. The Heraclitus Tool

So far, this section has described the Heraclitus method in general terms without regard for how a tool might support the work. In practice, SOTIF analysis for any real-world ADAS or autonomous system is a demanding task involving many components, interactions, and environmental conditions which will produce a large number of SOTIF scenarios. A prototype software tool, one view of which is shown in Fig. 3 below, was developed to help analysts apply the Heraclitus method and to provide an experimental platform with which to evaluate the method. Users can input components, interactions, and hazards to model a system; add and select environmental elements; and generate, edit, and rank scenarios.

IV. Evaluation

A small control trial was conducted as a preliminary evaluation of the Heraclitus method, with a focus on the scenario generation capability of an LLM. Participants completed a SOTIF scenario generation task without LLM support (control) and with LLM support (intervention). Participants were divided into two groups, each performing a scenario generation task with and without the LLM. This experiment aims to answer the following research questions:

- **RQ1:** Can an LLM produce a useful proportion (33%) of acceptable SOTIF scenarios that would assist an analyst during a SOTIF analysis?
- **RQ2:** Does an LLM help analysts generate a larger number of acceptable SOTIF scenarios, compared to doing the same task manually?
- **RQ3:** Do the SOTIF scenarios produced by the LLM provide better coverage of the system environment, compared to doing the same task manually?

For RQ1 threshold of 33% was selected because it represents a situation where approximately one in every three scenarios generated by the LLM is meaningful for the purpose of SOTIF analysis. That is, in a co-operating analysis setting, not every scenario generated by an LLM must be meaningful. We reason that an analyst might be willing to tolerate two out of every three generated scenarios being unusable (even after light editing).

Six participants were recruited from the team of engineers at Critical Systems Labs and divided into two equal groups (Group A and Group B). Most participants had prior experience with SOTIF analysis, and none had prior experience using Heraclitus.

A. Experimental Procedure

The experiment was conducted over three sessions. In the introductory session, participants completed a short exercise designed to familiarize them with the tool, both with and without LLM support (Llama 3.2 11B Instruct running on Amazon Bedrock). Two versions of the Heraclitus tool were prepared, one with both manual scenario generation and LLM-based scenario generation enabled and a second version with only manual scenario generation enabled. Both versions were configured to track user actions. All data collected from Heraclitus was de-identified. Beyond basic functionality, minimal guidance was provided to participants regarding how to use the LLM aspect of the tool. Participants were told that the objective was to evaluate the LLM's "usefulness", and that they should utilize it in a manner they found useful to create SOTIF scenarios. An example system was presented in each session, with Adaptive Cruise Control used in session two and Parking Assist used in session three. The example systems were chosen such that the two exercises would be similar enough to be comparable in difficulty, but different enough that completing one before the other would be unlikely to introduce bias or influence the outcome of the second. Participants were given a model file containing a pre-defined system and environment which they loaded into Heraclitus. Participants were able to modify the model but were not required to do so, allowing them to focus exclusively on the task of scenario generation. Both groups were given one hour to generate SOTIF scenarios using Heraclitus. Participants were not required to rank the results, only generate a list of acceptable scenarios. After the final session was complete, participants completed a short online questionnaire, providing feedback on their overall experience, how useful they found the LLM support, and how they might improve Heraclitus.

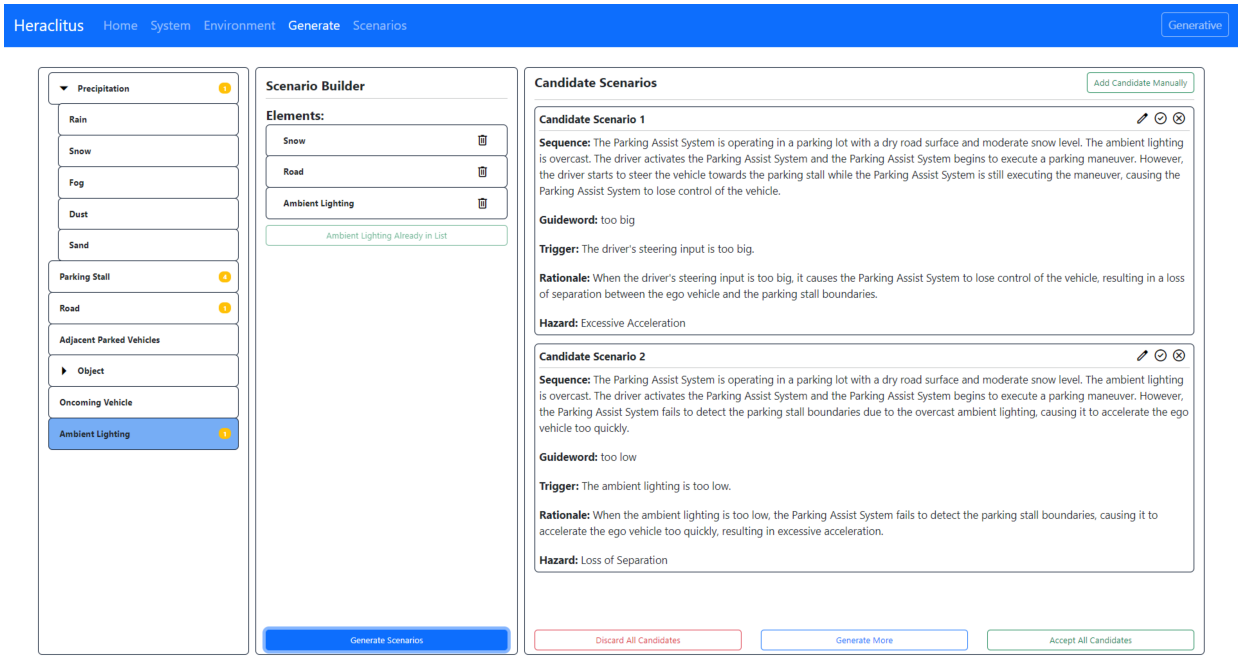

Fig. 3. Overview of the Heraclitus tool.

B. Analysis

The final scenarios were analyzed for quantity, quality, diversity, and coverage. For the purpose of answering RQ1 and RQ2, all scenarios accepted by the participants were assumed to be credible or useful. Scenario quantity was assessed by comparing the number of accepted scenarios each participant generated with and without LLM support. Quality was measured by inspecting a random sample of ten manually generated scenarios and ten LLM-generated scenarios and extracting general observations. Diversity and coverage scores were generated for each participant for both manual and LLM-generated scenarios, reflecting the average number of environment elements included in each scenario and the number of unique environment elements addressed in all scenarios.

V. Results

This section presents the results of the experiment and addresses the research questions described above.

A. Experimental Results

The results of the control trial are presented in Table 1. On average, participants sent 27 queries to the LLM (one query every 2.2 minutes). The LLM responded with a syntactically valid response that could be parsed by Heraclitus 88% of the time. Responses containing a batch of one or more scenarios were returned to the user within 12.9 seconds (95% CI [9.6, 16.3]), on average.

While using the LLM, each participant generated an average of 39.8 (95% CI [28.8, 50.9]) candidate scenarios, which is 1.47 candidates per query or 0.65 candidates per minute. On average, only 15.7 (95% CI [10.9, 20.4]) or 39% of the candidates were ultimately accepted by participants as credible, useful, and non-duplicate SOTIF scenarios, which corresponds to participants accepting a new scenario every 3.82 minutes. By contrast, when working manually participants created an average of 16 (95% CI [10.1, 21.9]) scenarios each.

The participants generated a total of 239 scenarios with LLM support. Of these, 94 scenarios were accepted by the participants, 53 of these scenarios were edited by participants before they were accepted (average of 6.6 scenarios per participant). Acceptance of a scenario by a participant was used as a proxy for whether the scenario was meaningful. A statistical test showed that the finding of 39% of scenarios being usable is statistically significant ($p < 0.05$) compared to the stated target threshold of 33% (from RQ1) [9]. That is, if this experiment were to be repeated with a similar set of participants, there is a high probability the number of accepted scenarios would be larger than 33%.

Only two scenarios generated with the support of an LLM were edited but were not accepted, otherwise all edited LLM generated scenarios were accepted. All scenarios that were manually created and edited were also accepted.

Participants explicitly rejected an average of 22.3 (56%) of the LLM generated scenarios. Note that in the Heraclitus tool, a user can explicitly reject a scenario (by clicking the reject button) but also has the option to discard it without making an explicit decision (by moving on to generate more scenarios). No manually generated scenarios were rejected by participants.

Answer to RQ2: Does an LLM help analysts generate a larger number of acceptable SOTIF scenarios, compared to doing the same task manually? Overall, with an LLM, participants generated an average of 0.3 fewer meaningful scenarios than when working manually. However, given the small sample size for this study, this difference is within the margin of error for this study, so no conclusion can be drawn as to whether the participants were more productive (in terms of quantity of scenarios generated) with or without the LLM.

An important part of SOTIF analysis is exploring a wide range of triggering conditions and their potential impact on safety and having methods that push users to consider a wider range of conditions can be advantageous. The level of coverage of the modelled environment between the manual and LLM generated scenarios was calculated by matching keywords from the environment model to the scenarios produced during the experiment. For example, if the environment model included "rain", then scenarios that included the word "rain" were counted as covering that element. A single scenario was permitted to cover more than one element if multiple unique elements were found. The measure was validated by manually checking the measured results for a subset of the scenarios to confirm it did not produce false positive or negative matches.

When using the LLM to create scenarios, each scenario covered an average of 13% of the available elements from the environment model. Without the LLM, each scenario only covered an average of 8% of the elements. Across all scenarios generated with the LLM, 72% of available elements were covered compared to only 14% across all manually created scenarios.

While the LLM appears to increase the coverage of the environmental model, some participants noted that the LLM added elements to scenarios that did not impact the outcome. We have called this phenomenon "scenario packing". For instance, a scenario for the adaptive cruise control example might involve a rear vehicle, but the presence of the rear vehicle did not contribute to the functional insufficiency described. Despite limitations in the measure used for this study, the possibility of an LLM improving the environmental coverage of a SOTIF analysis is important to consider.

Answer to RQ3: Do the SOTIF scenarios produced by the LLM provide better coverage of the system environment, compared to doing the same task manually? Using an LLM, participants generated scenarios that cover a wider breadth of environmental conditions than when manually creating scenarios. However, it is possible that some of environmental conditions in the LLM-generated scenarios were not directly related to the functional insufficiency being explored in the scenario.

B. Questionnaire Results

Several questions on the questionnaire asked participants about the perceived utility of LLM-supported scenario creation. Questions covered topics such as speed of scenario creation, coverage of environmental conditions, novelty of scenarios, the role of the LLM for brainstorming, and whether an LLM could be used in future SOTIF analysis work. A majority of participants felt that using an LLM allowed them to create scenarios faster, compared to manual-only creation. Further, a majority felt that the LLM provided useful scenarios that covered a wider range of environmental conditions, even if they required editing to fix problems. However, participants were split as to whether the resulting scenarios described interesting or novel scenarios, compared to ones created manually.

Five out of six participants felt that even if the LLM-provided scenarios were not immediately usable, they gave them ideas for additional scenarios to record manually. When asked how they would use an LLM to support SOTIF analysis for a real-world analysis, only one out of six participants said they would not use an LLM at all. The remaining five indicated they would use an LLM for some manner of brainstorming to augment or enrich scenarios created manually. Therefore, findings suggest that LLMs are capable of producing credible SOTIF scenarios.

C. Observations on Quality

After the experiment, a random sample of twenty accepted scenarios (ten manually created, ten created with LLM support) were reviewed for quality. The following observations summarize the findings of this review.

Incorrect or Incomplete Scenarios: LLM-provided hazards and guidewords for each scenario were often incorrect, however, questionnaire results indicated that participants felt these fields were usually consistent with the given scenario. Incorrect guidewords were also observed in the manually generated scenarios. In some scenarios the LLM split key pieces of information between the description, rationale, and trigger fields, leading to descriptions that did not fully describe a sequence of events leading to the occurrence of a hazard.

Shallow World Knowledge: In some scenarios, the LLM displayed inaccuracies in physical and logical reasoning and a shallow understanding of the system (e.g., how radar sensors behave under various environmental conditions).

Imprecision in Manual Scenarios: Several forms of imprecision were noted amongst the manually generated scenarios. These included abstractions that detracted from scenario specificity; deviations from the requested "sequence of events" style, instead providing "if-then" hypotheses; and "scenario packing", in which descriptions included multiple scenarios. Scenarios generated with LLM support tended to focus on one scenario per description and were more likely to have precise descriptions, despite being prone to wordiness and inclusion of unnecessary elements.

TABLE I
INTERACTIONS BETWEEN STUDY PARTICIPANTS AND HERACLITUS.

Participants		**LLM Interaction**			**Candidates**		**Rejected**		**Edited**		**Accepted**	
#	Group	# Queries Sent	# Valid Resp.	Avg. Resp. Time[s]	Man.	LLM	Man.	LLM	Man.	LLM	Man.	LLM
P1	B	24	19 (79%)	14.5	5	37	-	20 (54%)	1	0	5	17 (46%)
P2	B	24	24 (100%)	15.5	27	45	-	29 (64%)	15	12	27	13 (29%)
P3	B	31	26 (84%)	14.2	14	46	-	26 (56%)	1	6	14	20 (43%)
P4	A	29	26 (90%)	14.0	13	43	-	20 (47%)	10	16	13	23 (53%)
P5	A	25	20 (80%)	4.4	17	14	-	6 (43%)	8	6	17	6 (43%)
P6	A	29	28 (97%)	15.0	20	54	-	33 (61%)	0	13	20	15 (28%)
TOTAL		**162**	**143**	**-**	**96**	**239**	**-**	**134**	**35**	**53**	**96**	**94**
AVERAGE		**27**	**23.8 (88%)**	**12.9**	**16**	**39.8**	**-**	**22.3**	**5.8**	**6.6**	**16**	**15.7**

Answer to RQ1: Can an LLM produce a useful proportion (33%) of acceptable SOTIF scenarios that would assist an analyst during a SOTIF analysis? A useful proportion (40%, $p < 0.05$) of the SOTIF scenarios generated by the LLM in this experiment were of sufficient quality to be accepted (possibly with editing) by human analysts. Further, a majority of participants in the experiment felt that LLM-assisted scenario generation could be useful for SOTIF scenario generation work. Taken together, these results indicate that it is possible for an LLM, used in combination with the Heraclitus method, to provide meaningful support for a SOTIF analysis. However, opportunities exist for improvement in the quality of the generated scenarios.

VI. RELATED WORK

Given the novelty of this topic, there are no existing works on LLM-supported SOTIF analysis. As such, a literature review was performed with a focus on the use of LLMs to support or perform system safety analysis. Some of the methods explored include Hazard and Operability Study (HAZOP) [10], System-Theoretic Process Analysis (STPA) [11], [12], Hazard and Risk Analysis (HARA) [13], [14], Failure Mode and Effects Analysis (FMEA) [15], pure hazard identification [16], and eliminative argumentation [17].

The common theme across the reviewed literature was that LLMs were found to show potential, but only if a human was involved to check and/or moderate the generated content. The technology has not yet been found effective for a fully automated pipeline due to limitations with its robustness and trustworthiness.

Qi et al. [11] evaluated how different human-LLM collaboration schemes affect the quality of an STPA. Three schemes were tested against two separate automotive baseline examples. The authors qualitatively assessed their performance based on a set of predefined attributes (e.g. ability to identify hazards, ability to identify causes) and found that the effective performance of the different schemes was commensurate with the cost and complexity of its execution, with one scheme ("Recurring Duplex") outperforming both expert generated baselines. This scheme facilitated the most collaboration between the users and LLM throughout the STPA process, resulting in a higher quality output. The authors' preliminary conclusion is that there appears to be an objective improvement of STPA quality using the collaboration schemes with LLMs. Concerns regarding the validity of their conclusions still exist regarding the subjectivity of the human-in-the-loop during the evaluation of the tool, and the relatively low number of experiments performed using the ChatGPT platform.

Nouri et al. [13], [14] developed a fully automated pipeline that integrates GPT-4.0 into the HARA process (defined as identifying hazards, assigning scenarios to each identified hazard, and defining safety goals). The pipeline consists of a multi-task framework, where a description of an autonomous driving function is input into a cascading structure of LLM modules, each dedicated to a task of the HARA. Each modules receives a structured prompt based on the input, other module outputs, and databases of scenarios/malfunctions, and the resulting HARA is output. To evaluate the performance of the tool, the researchers engaged nine industry experts to review an LLM generated HARA. The framework was found to underperform when compared to expert-crafted HARAs but was particularly ineffective at identifying and formulating the hazardous events. The researchers concluded that the current state of the tool could be useful in supplementing the HARA process, but the technology is not yet developed to a point where the process can be fully automated.

Slivis-Cividjian et al. [15] developed a web-based application called "i-SART" to assist radiation therapy (RT) practitioners in performing an effective proactive safety analysis by augmenting the FMEA method. A database of failure modes was generated, primarily by experts, but with additional "synthetic" failure modes developed using various natural language processing techniques. A model was trained on a data set consisting of failure modes from the i-SART database and from headlines of RT incident reports. Though 230 (53 duplicates) of the 640 failure modes produced were identified as useful by an RT expert, the authors found that the LLM generated failure modes lacked syntactic accuracy and clarity.

VII. Discussion

This paper has introduced the Heraclitus method, which aims to facilitate human-led SOTIF scenario generation using an LLM. The method was operationalized by implementing it as part of a prototype software tool by the same name. A small control trial was performed to provide (preliminary) validation of the method and answer key research questions related to the feasibility and utility of the Heraclitus method.

The most important finding of this experiment is that LLMs, when used according to the Heraclitus method, can indeed produce meaningful or acceptable SOTIF scenarios, though some might require light editing. Further, the proportion of meaningful or acceptable scenarios is at least 33% (i.e., one in three scenarios are acceptable). Our interpretation of this result is that LLM-assisted SOTIF scenario generation has the potential to meaningfully support human analysts.

The small sample size for our study prevents any conclusion about utility of LLM-supported SOTIF scenario generation compared to a manual only approach, in terms of the quantity of scenarios created. However, it is interesting that the results between the control and intervention groups in the experiment are very similar. This suggests that additional improvements to Heraclitus could result in improvements to productivity. This interpretation is further supported by qualitative results from the post-experiment questionnaire and inspection of the generated scenarios.

The quantity of scenarios is only one measure of utility. It is also important to consider other ways that an LLM might support SOTIF analysis. For instance, some participants felt that the LLM helped guide their thinking, even if they did not use scenarios that were generated. Another measure of utility is whether using an LLM increased the breadth of environmental condition coverage. Our results indicate that it is possible that using an LLM increases environmental coverage, but further work is required to determine if the increased coverage is meaningful to the generated scenarios.

A. Threats to Validity

The small number of participants is a threat to external validity; the results do not provide enough statistical power to draw significant conclusions about the impact of an LLM on SOTIF scenario creation versus manual scenario creation. It is also possible that participant subjectivity may have influenced results. In this regard, the results presented should be interpreted observationally (i.e., only for this study group, using the Llama 3.2 11b model in the Fall of 2024). The absence of statistically significant conclusions, along with the qualitative nature of many of the stated findings also impacts internal validity. As such, results should be viewed as preliminary, but certainly promising.

B. Closing Remarks and Future Work

This paper has reported on the first step in a research project aimed at using LLMs for SOTIF scenario generation. The preliminary results are promising and have revealed several directions for future work, many of which are already being implemented. These include exploring prompt refinement, more sophisticated prompting techniques (e.g. iterative prompting), and newer and/or larger models to improve the LLM's physical and logical reasoning around a system's response to stimuli; improving system modelling to better capture the complexities of real-world systems; performing further validation of the Heraclitus method using case studies and other experiments with larger sample sizes and more diverse systems; and developing a quantitative approach to assessing coverage of LLM-generated scenarios.

References

[1] H.-H. Jebamikyous and R. Kashef, "Autonomous vehicles perception (avp) using deep learning: Modeling, assessment, and challenges," *IEEE Access*, vol. 10, pp. 10 523–10 535, 2022.

[2] "Road vehicles — Safety of the intended functionality," International Organization for Standardization, Standard, 2022.

[3] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21. Association for Computing Machinery, 2021.

[4] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling Laws for Neural Language Models," 2020.

[5] Z. Ji *et al.*, "Survey of Hallucination in Natural Language Generation," 2024.

[6] S. Diemert, E. Cyffka, N. Anwari, O. Foster, T. Viger, L. Millet, and J. Joyce, "Balancing the Risks and Benefits of Using Large Language Models to Support Assurance Case Development," in *SAFECOMP 2025, Computer Safety, Reliability, and Security.*, 2025.

[7] T. Viger, L. Murphy, S. Diemert, C. Menghi, J. Joyce, A. Di Sandro, and M. Chechik, "AI-Supported Eliminative Argumentation: Practical Experience Generating Defeaters to Increase Confidence in Assurance Cases," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2024.

[8] "Road vehicles — Functional safety," International Organization for Standardization, Standard, 2018.

[9] J. L. Devore, *Probability and Statistics for Engineering and the Sciences*, 7th ed. Belmont, CA: Duxbury Press, Jan. 2007.

[10] X. Feng, Y. Dai, X. Ji, L. Zhou, and Y. Dang, "Application of natural language processing in HAZOP reports," *Process Safety and Environmental Protection*, vol. 155, 2021.

[11] Y. Qi, X. Zhao, S. Khastgir, and X. Huang, "Safety Analysis in the Era of Large Language Models: A Case Study of STPA using ChatGPT," *Machine Learning with Applications*, vol. 19, 2025.

[12] S. Diemert and J. H. Weber, "Can Large Language Models Assist in Hazard Analysis?" in *Computer Safety, Reliability, and Security. SAFECOMP 2023 Workshops. SAFECOMP 2023. Lecture Notes in Computer Science, vol 14182*, Toulouse, France, 2023.

[13] A. Nouri, B. Cabrero-Daniel, F. Törner, H. Sivencrona, and C. Berger, "Engineering Safety Requirements for Autonomous Driving with Large Language Models," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, 2024.

[14] A. Nouri, B. Cabrero-Daniel, F. Töerner, H. Sivencrona, and C. Berger, "Welcome Your New AI Teammate: On Safety Analysis by Leashing Large Language Models," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, 2024.

[15] N. Silvis-Cividjian, Y. Zhou, A. Sarchosoglou, and E. Pappas, "i-SART: An Intelligent Assistant for Safety Analysis in Radiation Therapy:," in *Proceedings of the 17th International Joint Conference on Biomedical Engineering Systems and Technologies*. SCITEPRESS - Science and Technology Publications, 2024.

[16] S. M. J. Uddin, A. Albert, A. Ovid, and A. Alsharef, "Leveraging ChatGPT to Aid Construction Hazard Recognition and Support Safety Education and Training," *Sustainability*, vol. 15, no. 9, 2023.

[17] T. Viger, L. Murphy, S. Diemert, C. Menghi, A. Di, and M. Chechik, "Supporting Assurance Case Development Using Generative AI," in *SAFECOMP 2023, Position Paper*, 2023.