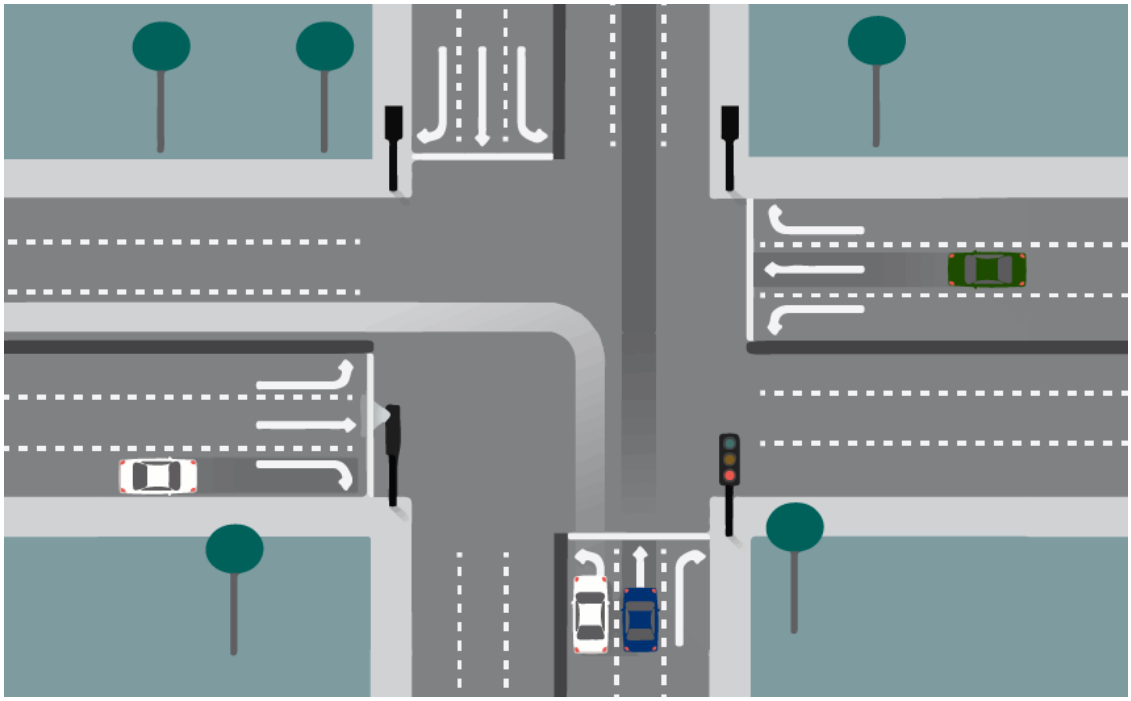
CHALMERS
UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



A New Approach to AD/ADAS Test Scenario Generation Using Open-Source Intelligence and Large Language Models

Bachelor of Science Thesis in Software Engineering and Management

ALEKSEY ZORIN

LOUIS MERCIER

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A New Approach to AD/ADAS Test Scenario Generation Using Open-Source Intelligence and Large Language Models

Supervisor: MUHAMMED ÇAĞRI KAYA
Examiner: CHRISTIAN BERGER

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

[Cover: illustration of a traffic scenario on an intersection involving 4 entities. Image created by Louis Mercier on Adobe Illustrator.]

Department of Computer Science and Engineering
University of Gothenburg
Chalmers University of Technology
Gothenburg, Sweden 2024

A New Approach to AD/ADAS Test Scenario Generation Using Open-Source Intelligence and Large Language Models

Aleksey Zorin
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden

Louis Mercier
Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden

***Abstract*—In the evolving field of Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS), the growing necessity for realistic simulation data coupled with the dynamic nature of real-world driving situations challenges traditional test scenario approaches. This is heightened by the increased complexity of AD/ADAS which necessitates rigorous testing to ensure safety and reliability.**

This paper explores a new practical approach to AD/ADAS test scenario generation using Open-Source Intelligence (OSINT), generative Artificial Intelligence (AI) and Scenario Description Languages (SDL). This new approach utilises a vast array of openly accessible data on the Internet and holds potential for future research and applications.

The study was performed following the framework of Design Science Research (DSR). In two DSR cycles, an artefact that generates dynamic scenarios by scraping internet sources and completing the scenarios with Large Language Models (LLMs) has been developed.

The developed artefact successfully generated scenarios in an automated manner using collected traffic incident data, demonstrating its practicability for AD/ADAS test scenario generation.

In conclusion, the combination of OSINT with generative AI for scenario generation holds the potential to be a viable approach for test scenario generation for AD/ADAS systems, potentially extending the existing scenario generation methods with a novel approach based on openly accessible online data. We have shown that the method can generate scenarios using keywords from online texts and adhere to a given scenario format, yet future research is encouraged to explore further and validate this approach to ensure its applicability in the automotive industry.

***Index Terms*—Artificial Intelligence, Autonomous Driving (AD), Advanced Driver Assistance Systems (ADAS), Design Science Research (DSR), Large Language Model (LLM), OpenSCENARIO, Open-source Intelligence, Scenario Generation, Web Mining**

I. Introduction

Software engineering plays a crucial role in the modern automotive industry. During the last decades, the amount of programmable embedded components in vehicles has grown exponentially, bringing along both new technological innovations and challenges for the professionals involved in car manufacturing [1]. Development of new car models requires not only successful adaptation of the latest development strategies from the software engineering domain but also a careful consideration of domain-specific requirements and constraints of the automotive industry, often demanding individual solutions.

Noticeably, even the software components that control safety-critical applications have become more complex, making the analysis and design phases of Autonomous driving (AD) and Advanced Driver Assistance Systems (ADAS) development an even more essential task. According to Lachmann and Schaefer, creating test cases is crucial to lay the foundation for the testing process and ensure that the new embedded systems are safe and reliable for their users [2].

Designing safe and dependable solutions for AD/ADAS requires a deep understanding of the real-world traffic environment. The complexity of the real world means a near-infinite number of potential situations and incidents that can happen to a user whose vehicle depends on its embedded AD/ADAS features. Constructing test scenarios helps developers predict these situations and understand the requirements of new AD/ADAS features. However, scenario generation is often a big challenge in the automotive industry, as it requires designing and implementing highly complex scenario elicitation methods. For instance, Alnaser et al. [3] discuss some existing methods for test scenario generation in the field and highlight their inherent complexity as a “big challenge”.

The purpose of this research has been to extract real-life driving situations from accessible internet resources

using web scraping, and to restructure the scraped contents into complete scenarios using Large Language Models (LLMs). The choice of using web scraping is motivated by the complexity of the real world mentioned above. Our new scenario generation method is specifically aimed at using the sheer scope of traffic incidents described on the internet, while also considering the challenges associated with existing generation methods. Ultimately, the aim of this study was to introduce a new approach built upon real, documented traffic incidents, and explore whether this approach can generate scenarios based on a specific Scenario Description Language (SDL) standard.

Web scraping has been utilized in various research capacities within the automotive industry. It has been employed to predict the prices of second-hand vehicles based on their features using the open-source web scraping tool Python Selenium [4], and to analyse trends in the number of vehicle recalls worldwide [5], to name a few. These studies underscore the versatility of web scraping applications yet generating test scenarios for AD/ADAS using the technology has not yet been researched. This highlights the gap that our work is meant to bridge.

Additionally, we have looked at some research that deals specifically with the safety features of vehicles, such as to analyse the safety of autonomous vehicles using scenario-based assessment [6] or to estimate the safety impact of lane-keeping assistant (LKA) systems [7]. We believe that incident information on the internet holds potential to contribute to this area of research. While assessing existing research articles, we have not identified any study that simultaneously deals with AD/ADAS test scenario generation and utilizes web scraping techniques to achieve their research objectives. This is another indication of the gap that we aimed to bridge and contribute with new theoretical and practical knowledge.

We have used the design science research methodology. Hence, our emphasis has been on the artefact itself, the web scraper. We have identified suitable web resources with traffic incident data and customized the web scraper to retrieve relevant information for potential AD/ADAS test scenarios. After all, the design science methodology is a way to address a relevant existing business issue by designing and evaluating a new technology or tool [8]. The choice of the design science methodology has also influenced our research questions for this study:

RQ1: *What public web resources can be used to extract data on AD/ADAS-related incidents and scenarios?*

RQ2: *How can scenario-relevant data from these resources be collected in a semi-automated fashion?*

RQ3: *To what extent can LLMs be used to structure and complete an AD/ADAS test scenario based on the scraped data?*

This approach not only introduces a novel application of web scraping in the automotive industry but also holds the potential to contribute to data collection for AD/ADAS development. From a theoretical perspective, this study might introduce a new AD/ADAS test scenario elicitation strategy which accounts for the intricate interplay between software design and real-world traffic scenarios. By conducting this research, we hope to advance and broaden the discussion about which methods can be used to design safer AD/ADAS systems for future car models. For practitioners, the purpose of this study is to provide a new practical process for gathering incident data, potentially transforming how test scenarios are created with the help of web scraping tools and Artificial Intelligence (AI) capabilities. Additionally, this research can help lay a foundation for the development of new web scraping tools, specifically designed to account for the domain-specific needs of the automotive industry. These hypothetical web scraping tools could help companies deal with the constant emergence of new real-world data, paving the way to developing self-reliant and automatically updated real-world incident databases, thus further expanding the capabilities of software engineering within the automotive industry.

II. Background and Related Work

A. Background - Automotive Software Development and Testing

Our study dwells upon the crucial aspect of testing in automotive software development, where the increasing importance of efficient testing is emphasised because of ever more complex software-driven features [2], as evidenced by Lachmann and Schaefer, whose key findings include the importance of systematic test case specification, the identification and removal of possible redundancies in test cases, as well as selecting the right test cases for execution.

Building upon this foundation, Broy conducted a comprehensive survey of automated test case generation techniques [1]. While Lachmann and Schaefer focused on optimising existing testing practices, Broy's work explores various approaches for automating test case generation. This automated aspect aligns with the goals of efficient testing emphasised in Lachmann and Schaefer's work. Broy's research provides a valuable foundation for exploring alternative data collections for test scenario generation. However, these techniques may struggle to gather comprehensive real-world data for AD/ADAS testing and do not account for the unpredictable nature of real-world driving environments.

While the works of Lachmann and Schaefer, as well as Broy, lay a substantial foundation in the realm of automotive software testing by emphasising efficiency

and automation, there remains a significant research gap in the context of ADAS and AD. Specifically, the study by Tan et al. [7] discusses the possible safety benefits of utilising Lane Keeping Assistant (LKA) systems in practice and estimates a significant decline in China's death and injury toll by 2030. This highlights the need for more sophisticated AD/ADAS-specific testing frameworks that combine real-world data and scenarios indicative of the complex driving environments these systems navigate, in addition to utilising automated test case generation.

B. Background - Web Scraping and its Applications

Web scraping is a technology used for automatic data extraction from openly accessible web resources [9]. It relies on procedures such as HyperText Transfer Protocol (HTTP) programming, Document Object Model (DOM) parsing and others, to scan the HyperText Markup Language (HTML) contents of given websites and extract all or specified parts of the page contents. Some common use cases of web scraping include data mining, marketing, data combination, and many others. Automatic data extraction is much faster and more effective than manual extraction, making web scraping an indispensable tool in various applications.

Researchers have studied different implications of web scraping, including its aspects of legality and ethics [10]. Generally, web scraping is still considered a "legal grey area", as the technology lacks specific regulations. However, practitioners should review and adhere to different websites' terms of use and refrain from scraping copyrighted, private, or other sensitive information. By answering the questions specified by Krotov and Silva [10], academic researchers and commercial practitioners alike can use web scraping techniques in a sensible, fair, and honest way, respecting the owners of the web resources.

We have not been able to identify other research that has used web scraping specifically to extract traffic incident data in the context of AD/ADAS test scenario generation. However, there are numerous well-documented applications of web scraping techniques. For example, Kalra et al. [11] have used the open-source Python Selenium framework, along with Python libraries PythonRequests and BeautifulSoup, to classify several YouTube videos into specific categories. This was done by scraping the videos' titles and description metadata.

Web scraping has also found its use in research in the automotive domain. Cihan and Cerrahoglu [4] have combined web scraping and machine learning to predict the prices of second-hand vehicles. That purpose was achieved by scraping data about the features and prices of different car models from websites that sell second-hand vehicles. Like Kalra et al. [11], Cihan and Cerrahoglu [4] have also used Python Selenium for automated data extraction. Additionally, Python Selenium has been used by Maione et al. [5] to gather public data about past vehicle recalls and convert the data into a structured format that allowed the researchers to effectively analyse it. In their conclusions, Maione et al. have stated that the use of web scraping has helped address problems that do not have an intuitive relationship with the technology itself [5], further underscoring the wide range of potential applications of web scraping.

C. Related Work - AD/ADAS Test Scenario Generation

We are most interested in the existing research on scenario-based development in the context of AD/ADAS systems, as our central purpose is essentially to add to this process by exploring whether and how web scraping and LLMs can produce new scenarios. Therefore, topics such as the definition of a scenario, scenario ontology, relevance analysis, and scenario generation methods, to name a few, are crucial as they help us understand the current state of the scenario-based development approach, along with its limitations and needs for future improvements. Li [12] has summarised the process of scenario-based testing in the autonomous driving domain, outlining three kinds of scenario data:

- *Real Data*: real-life data, including but not limited to accident data, gathered from multi-sensor collection platforms installed on cars.
- *Simulation Data*: virtual data obtained in a simulation environment.
- *Expert Experience Data*: data gathered from the expertise and knowledge gained through previous tests.

In the context of autonomous driving, Li states that "With the level of autonomous driving increasing, the test scenarios become infinitely rich, extremely complex, unpredictable, and inexhaustible" [12]. This point further illustrates the need for new approaches to data collection to address these challenges. Web scraping has the additional potential of finding scenarios that would otherwise be missed by existing methods of data gathering, whether real-life, virtual, or expertise-based. After all, each situation or incident published on the web offers a prospective testing scenario.

Challenges associated with autonomous vehicle testing have also been investigated by Alnaser et al. [3], who highlight the importance of testing and verifying the vehicles before they are deployed. The researchers underline the criticality of safety in AD and propose a scenario-driven experimental framework for virtual AD testing. Pseudo-random test scenarios are generated and tested automatically in this framework. However, the process has to simulate millions of configurations, adhere to input constraints, and decide whether a specific test has been generated earlier. The study provides valuable

insights into the domain of AD systems testing and further highlights the importance of good test scenarios, as well as demonstrating the inherent complexity of virtual scenario generation.

According to Ma et al., the current practice of manually designing test scenarios limits the number of scenarios that can be generated. They suggest that integrating data-driven and knowledge-driven approaches, especially through artificial intelligence, could aid in automatically identifying edge and corner cases, thus enhancing the scenario generation process [13].

Lastly, Riedmaier et al. [6] provide a valuable overview of scenario-based safety assessment of AD vehicles by conducting a comprehensive literature review. In the context of deriving scenarios from real-world data, such as field operational tests, the authors underline the importance of extracting as comprehensive data sets as possible. Additionally, the authors state that considering the theoretically infinite number of possible traffic scenarios, an essential task for practitioners is to decide which scenarios are most representative and, hence, most suitable for test generation. We believe that web scraping techniques can address this challenge, as each documented traffic incident on the internet constitutes a prospective test case; each incident is naturally representative of the real world and can serve as a basis for generating suitable test scenarios by directly filling the requirements presented by Riedmaier et al.

In summary, reviewing existing research in domains that are relevant to this study, we have discovered that topics such as automotive software testing, web scraping, and test scenario generation for AD/ADAS systems have all been thoroughly explored by researchers. However, we have not found any research that uses the procedures of web scraping to find new prospective test scenarios for AD/ADAS development. This indicates a clear research gap, which our research has the potential to address. The mentioned articles have supported us in this research, as they provide valuable insights into their respective domains and highlight many important areas for improvement.

III. Research Methodology

A. Overview

To answer the stated research questions, we have used the design science research methodology. The purpose of design science is to develop a new and innovative artefact to support humans and organisations by solving a relevant issue [8]. In the context of this study, the issue is the complexity of existing test scenario generation methods. Thus, we have focused on building an artefact that combines web scraping with industry-relevant scenario description formats. The emphasis that design science places on the artefact itself makes for a logical and reasonable approach to addressing the identified issue, which has the potential to introduce a new way of working for practitioners. In this section, we have described the research context of our study, the qualitative data we have collected, the subjects and instruments of the study, and our data analysis approach.

B. Research Context

Our context has consisted of relevant academic knowledge in the area, ethics and other considerations related to web scraping, websites with valuable traffic incident data (including each website's terms of use), web scraping tools with their documented applications, available LLMs, and scenario-relevant conventions such as OpenSCENARIO and its various deliverables. This context has shaped the research progression and thus the artefact design, prompting us to continuously evaluate and improve the artefact to generate scenarios of higher quality, completeness, and compatibility with industry standards.

As our focus shifted towards more specific SDLs, we determined that the integration of the OpenSCENARIO XML standard, specifically version 1.2.0, developed by the Association for Standardization of Automation and Measuring Systems (ASAM), enabled the description of dynamic content in concrete scenarios. Specifically, these include the actions and behaviours of vehicles, conditions, manoeuvres, and weather specifics that are prevalent in our collected data and internet resources. Furthermore, the ASAM OpenSCENARIO standard specifies an XML schema to which any generated OpenSCENARIO description file should conform, enabling a consistent structure that can be easily read, modified, and generated. The ASAM OpenX standards are well-supported in the automotive industry, hence providing us with a wide array of resources, tools, and documentation for their integration into the artefact. Its use would equally open the possibility for the integration of other OpenX standards, such as OpenDRIVE and OpenCRG. Other explored alternatives included the ASAM OpenSCENARIO DSL with its Python-like syntax, which was discarded based on its challenging integration and scripting approach. Lastly, we examined CommonRoad, but despite its XML data format and extensive resources, it did not align with the scope of the artefact [13].

C. Data Collection

The focus of the initial data collection process has been on the extraction of qualitative data with the web scraper, particularly due to the inherently textual and descriptive nature of the online sources we have used. While the varied nature of online sources does not exclude the fact that the scraper may also extract quantitative data, our primary interest has been

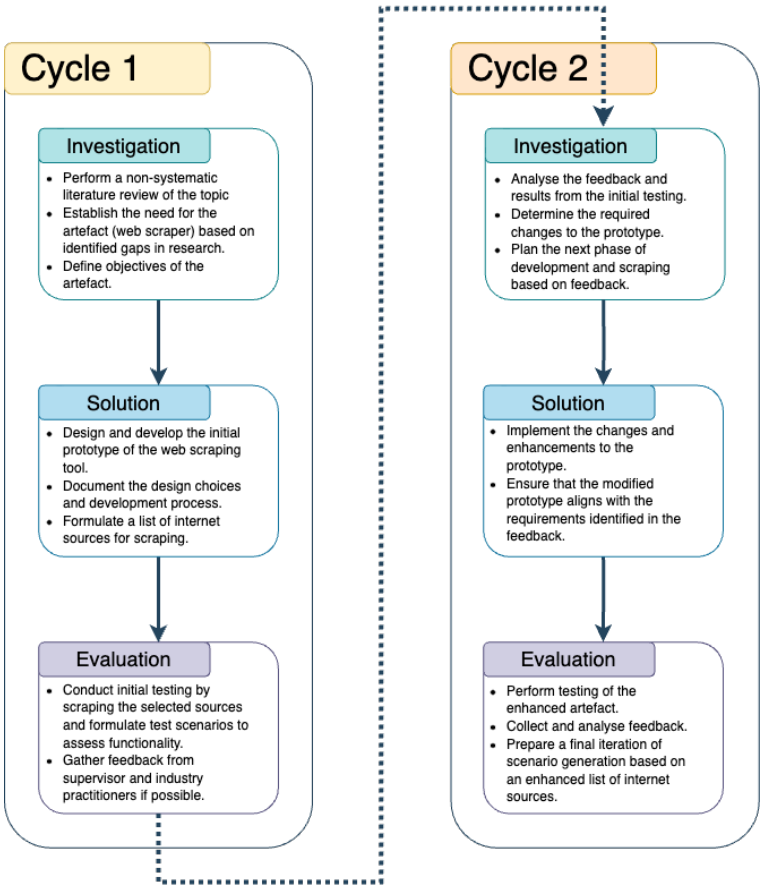


Fig. 1: Overview of design science cycles planned for this research

in scenario-relevant qualitative attributes. Primarily, the most valuable qualitative data we have obtained through scraping are words that can be included in an actual OpenSCENARIO Extensible Markup Language (XML) document, e.g., words describing the weather, ego car model, surrounding vehicles, and other details of the scenario environment.

Hence, the instrument for data collection is the web scraper itself. Websites that contain relevant qualitative incident descriptions while also directly or indirectly allowing automated extraction of their data have acted as subjects of the data collection. We consider the chosen LLMs, OpenAI ChatGPT 4.0 and Meta Llama 3, to be subjects as well, although it can be argued whether the subject in this case is the data on which each model was trained or simply the outputs produced by the model. When selecting which LLMs to use, we wanted to compare the performance of a strong proprietary model with a smaller, more customisable model in our research context. GPT 4.0 is a powerful model, surpassing its predecessor GPT 3.5 in multiple aspects [14]. Unlimited use of the model is not free, but one of us has had access to it from before, so we decided to proceed with it. Llama 3, on the other hand, is a newly launched model and has not yet been thoroughly evaluated in other research, but can be downloaded locally and tailored for specific needs [15]. The models differ strongly in size, and, as we saw later, GPT 4.0 performed better than Llama 3, indicating that the size of GPT 4.0 made it a better fit for our purpose compared to the customised Llama 3 models.

We have systematically evaluated the performance of our artefact as well as the overall research progression during weekly meetings with our academic supervisor. The assessment has been based on how well the artefact achieves our research objectives, i.e., its ability to collect relevant data and facilitate the identification of test scenarios for AD/ADAS systems. To support this process, we have planned and followed two design science cycles, each four weeks long:

*Cycle I: Emphasis on **RQ1**, the problem:* In this cycle, our primary focus has been on finding useful web resources that we can legally and ethically scrape from. The identification of such websites requires not only manually finding them and looking through their contents, but also the ethical and legal considerations

described previously in Related Work. Subsequently, we built the first scraper prototype and tested it on the identified websites. In order to generate real-life AD/ADAS test scenarios, we have looked into how such scenarios look in industry and incorporated this knowledge into the artefact. Lastly, we have assessed the potential risks of using LLMs and then constructed prompts to structure and complete the scenarios. The gathered insights from all the steps above have been used to support the following cycle.

*Cycle II: Emphasis on **RQ2**, the design:* The focus of this cycle has been to use the insights gathered from the previous cycle to improve the web scraper by refining its functionality and addressing identified issues. The main change in the artefact design has been the focus on dynamic scenario generation. Feedback from our supervisor has also been instrumental in evaluating the results of each task. Again, we have looked at potential new websites to use and validated the scenarios generated from the refined artefact by implementing a syntactic check and comparing the LLM-generated components with those of full scenarios from other sources. See figure 1 for a visualisation of the full process.

D. Data Analysis

Our analysis of the qualitative data involved assessing whether the fetched data contained enough relevant markers for their use in scenario templates. In the case of missing markers, the keyword "element_not_found" was generated into the data input. This step in the analysis was also integrated into the artefact to automate the process of generating prompts and iterating through the database for the first ten entries. The integration was made as part of the prompt formulation process in Cycle I and reused in Cycle II purely for data analysis. The markers were decided based on the main components of SDLs, and include the actor, in this case, the vehicle name, vehicle type, event or actions taken and location of the event with the date specified.

Naturally, the contents of any text we scraped were insufficient to fill the scenario XML file. Therefore, we resorted to comparing scraped text and determining their usability for the subsequent scenario generation process, based on the number of markers describing each incident. The complete scenarios, on the other hand, require more analysis as they consist of both real-world scraped details and LLM generations. We have proceeded with two steps to validate the scenarios. First, we have designed a syntactic check based on the official OpenSCENARIO schema that prevented faulty scenarios from being created, throwing an error. Second, we evaluated the LLM performance by taking complete scenarios from a different source, manually deleting some of their components and then asking an LLM to complete the gaps. Comparing this result with the original full scenario has helped determine the quality of missing gaps filled by the LLM. See Cycle II for more descriptions of this process.

IV. Cycle I

As planned, our emphasis in the first cycle was on the first research question. Having the ability to scrape the most comprehensive online resources is key to creating full or near-complete, plausible test scenarios for AD and ADAS. However, to conduct this study in a fair and responsible way, we began with a thorough overview of the state of the ethical and legal aspects of web scraping in general. We then proceeded to review each potential resource's own ToS and other rules, as well as their robots.txt files.

Subsequently, we have moved on to the remaining research questions. We have designed, built, and tested the first version of our web scraper. According to our plan, the last step before evaluating the results would be to complete the potential scenarios using available LLMs. We have realised that, to achieve this task, we first need a sufficient understanding of the AD/ADAS test scenario domain. After all, real-life test scenarios in the industry follow specific rules, syntaxes, and guidelines. Hence, we have reviewed the current state of the art to understand how a practical test scenario is structured, what guides its design, and which elements are usually included.

Lastly, we have implemented our idea of completing the scenarios by providing two subsequent prompts to an LLM, specifically ChatGPT 4.0 (developed by OpenAI). The first prompt structures the scraped natural text into a specific scenario format. The second prompt asks to fill the remaining scenario gaps with credible details. Both prompts were designed considering the potential limitations of using LLMs and the specific scenario format that we chose to follow. Though the results were quite promising, we discovered ways to further improve the artefact and use our learnings so far to implement new ideas in the next cycle.

A. Legality and Ethics of Web Scraping

Generally, it looks like web scraping itself is still considered a legal grey area [10]. However, it is being widely and publicly used to conduct research, build applications, or simply for personal needs. Big and small companies alike use and benefit from web scraping and similar techniques. Having looked at both factual information and various discussions about the use of it, we have come to the conclusion that web scraping itself, as a tool, is not a problem.

Subsequently, the questions of legality and ethics come down to reviewing the different websites' own

rules for scraping their data. A rule of thumb we propose is to avoid scraping personal data, as using that kind of data negligently can result in breaking data protection laws such as the General Data Protection Regulation (GDPR). Another guideline is to only scrape publicly available data. There is a consensus that all data on the internet that does not require a login or other authentication to access is generally considered publicly available. Additionally, information protected by copyrights should not be scraped either.

Before scraping any resource, it is crucial to consider how the scraped data will be used later. For instance, whether the data is intended to generate profits or not. Companies that conduct web scraping usually do not proceed before getting legal counsel from lawyers. Hence, we have an advantage as our goal is to collect the data for publicly available research rather than seeking to profit from it, if we make sure to adhere to each resource's own rules and strictly follow other considerations mentioned above.

B. Searching for Relevant Online Resources

In the context of Sweden, the government agencies Trafikverket and Transportstyrelsen possess the most detailed data about traffic incidents. Trafikverket provides an API for various interactions; hence, we could not apply web scraping [16]. Transportstyrelsen hosts the Strada database, which collects incident data from Swedish first responders [17]. The database is, however, not publicly accessible, and only a few government authorities are permitted to request access, as shown in Appendix A.

We continued by exploring the possibility of scraping videos on YouTube. The platform is of great interest to us thanks to its constant updating of a massive number of videos. Although scraping videos themselves is a problem on most video-hosting platforms because of their data policies, we entertained the idea of collecting video URLs based on different videos' metadata, such as title, tags, description, comments, and others. The envisioned result would be an automatically updated list of URLs leading to various user-uploaded videos of traffic incidents. Unfortunately, YouTube forbids scraping its services, [18], with only a few exceptions, as shown in Appendix A.

As a result, we investigated the websites of a few popular Swedish news agencies. Scraping news articles describing a variety of situations on the road has the potential to provide a basis for details for prospective test scenarios. However, we have discovered that news outlets tend to either limit or prohibit the scraping of their web pages. Additionally, all news stories are protected by copyright. Bonnier News, the owner of Dagens Nyheter, and other news agencies, for instance, do not allow any web scraping on their website data or metadata (see Appendix A). They make an exception for search engines, which may index the website's URLs and display parts of articles in the search results [19]. Aftonbladet, another large Swedish newspaper, does not prohibit web scraping specifically but specifies that they own the immaterial rights of their website, including text, images, and all other content [20]. Nothing is allowed outside the normal use of the service; see Appendix A.

For the purpose of testing our prototype, we have used trafiken.nu. It is an open online service containing relevant traffic information in the Stockholm and Gothenburg areas. The site itself receives its information from Västtrafik (a public transportation company in the Gothenburg region), Trafikverket, and purchased data from external providers, such as GPS data from vehicles. The site follows an "open data" policy and provides an API, although there is no prohibition of web scraping [21]. The only limitation of the website's rules is data related to traffic congestion. See Appendix A for the website's data rules and its robots.txt file.

C. LLMs Usage for Scenario Extraction and Completion

The complete result of our study is essentially intended to be a pipeline where relevant and accessible content is automatically scraped from selected internet resources and transformed into potential AD/ADAS test scenarios. For the last step, we want to explore the extent to which publicly available LLMs can be used to structure the scraped content into a given scenario format and whether the LLMs can be used to fill the remaining scenario gaps with realistic details.

The idea is to use LLMs in two iterations: first to enhance AD/ADAS test scenario descriptions when generated, and second, to fill the remaining gaps in the scenarios. This strategy is meant to mitigate some potential risks and downsides of using LLMs that we have identified below. The downsides are hardly limited to our specific research context, but rather apply to using LLMs in general.

1) Designing Prompts for LLMs: The prompts are formulated in a 3-step process, starting with the selection of a scenario template based on the outline of the scenario description. Moreover, data input is generated by iterating through the MongoDB database for scraped data relevant to the scenario; this step is specifically meant to reflect the use of scraped data in the description and allows for the introduction of specifications for elements. A finalised prompt is then generated, where both the data input and scenario template are inserted. This final step requires the manual insertion of the high-level scenario description for further contextualization. Overall, this process allows for the insertion of scraped data while introducing SDL concepts such as actors, locations,

events, ambient conditions, and activities, which are then contained in an XML description. The high-level description is meant to provide an overview of the scenario description while retaining an SDL sentence structure to distinguish scenario description elements. The process of prompt formulation is illustrated in Figure 2, where the final step consists of inserting the high-level scenario description and the XML template.

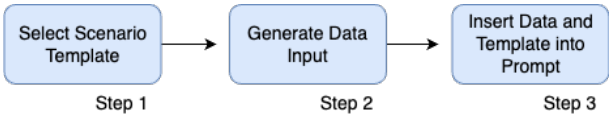


Fig. 2: The steps required to formulate a prompt in Cycle I.

These steps remain identical for the second step in the scenario formulation, which includes a second prompt for the final enhancement of the generated scenario description.

a) Prompt Structure and Content: Through the process of prompt formulation, both prompts remain consistent in terms of structure, composed of four main sections: the LLM instruction titled objective, followed by a section for the insertion of the data input, another section for the insertion of the XML scenario template, and finally a section for the insertion of the high-level scenario description. Both prompts are handled automatically through a Flask UI provided in the prototype, which allows for the generation of the prompts by simply following all the required steps as shown in Figure 2. An example of the prompts used during the scenario generation can be seen in Appendix B (Listings 1 and 2).

D. Limitations of Using LLMs

LLMs sometimes generate plausible but incorrect information. Further clarifications and prompts to oversee the generated data and fix mistakes can sometimes help, but this is not a guarantee. LLMs have a wide range of known problems, including the so-called "hallucinations". According to Liu et al., LLMs "hallucinate" when they generate nonsensical content yet present it confidently [22]. The authors state that the exact causes of this behaviour are still being researched, but the underlying training data may be one of the explanations. We need to be aware of this each time we submit a prompt and analyse the results.

Thus, it can be challenging to ensure the reliability of the generated data, which reduces its overall quality. The known problems associated with LLMs underline the importance of critically assessing whatever is being generated. A potential mitigation strategy is to compare the filled scenario gaps with traffic incidents in real life. In Cycle II, we have evaluated this step by comparing the LLM-generated gaps with actual test scenarios.

Additionally, results can be biassed because of the LLMs' training data and the lack of data that has not been properly described on the internet [22]. The risk of this would be to miss out on potential edge-case AD/ADAS scenarios that inherently lack many descriptions online. This is hard to mitigate in any way, as we cannot influence the training data of an LLM.

The overall quality of machine-generated content depends heavily on writing a good enough prompt. The key to working with LLMs in any capacity is to come up with controlled, structured, and well-thought-out prompts and to maintain the same prompt for subsequent requests to the model. Additionally, LLMs do not automatically understand the domain-specific context of what we're asking, including our research interests, which means we need to provide additional context information in the prompts. By including this information in the prompts as well, the model will hopefully yield more relevant and high-quality results. For each case of generating the content, we have evaluated the output considering the domain, research goals, realism, reliability, and logical cohesion. We have continuously questioned and problematized each generated result to spot potential faults.

To summarise, to use LLMs in a critical and controlled manner, we need to write good enough prompts and critically evaluate the results. A good prompt for the first iteration, where the LLM converts the scraped textual data into a scenario format, must explain our domain and research contexts and specify a concrete SDL structure. It must also specify that we do not expect the scenario to be complete given the scraped data and that we expect gaps where applicable. The prompt for the second iteration, in which the LLM fills out scenario gaps, must underline the importance of filling the gaps with realistic details, maintaining high quality, and avoiding breaking the previously generated scenario format or content in any way.

E. Scenario Description Languages

An SDL serves as a formalized syntax and set of protocols for defining and representing the dynamic behaviours of entities within simulation environments [23]. Furthermore, SDL is instrumental in developing virtual scenarios to test and validate ADAS functionalities under various traffic conditions, driver behaviours, and environmental factors.

In the domain of ADAS, SDL ensures that scenarios are not only standardized for consistency and reproducibility but also detailed enough to encapsulate complex interactions and edge cases that vehicles might encounter. One such instantiation of SDL in the automotive sector is OpenSCENARIO, which provides a structured framework for describing the entire lifecycle of a driving

Fig. 3: Example of a statement mapped to an SDL sentence structure.

scenario in a manner that is both understandable by humans and interpretable by simulation software.

Within the context of this study, SDL is essential for creating scenarios and provides a framework of standards that can be further incorporated into the workflow of the produced artefact. Consequently, this enables a consistent generation of scenarios relevant to AD/ADAS systems while outlining industry standards.

SDLs are substantial in providing a medium for the often-complex behaviours within automotive simulations. SDLs operate on a structured syntax that encapsulates the multifaceted requirement of traffic scenarios, essential for evaluating AD/ADAS functionalities [24]. At the core of SDLs are upper-level concepts that encapsulate common parameters in a constructed scenario, these are commonly defined as follows:

1) Actors – **Who** is involved in the scenario.
2) Activities – **What** activities are taking place.
3) Locations – **Where** is the scenario set.
4) Events – **When** do the events occur.
5) Ambient conditions – **What** ambient conditions are there to consider.

These parameters are applied to construct a scenario in a format that adheres to a natural language structure, therefore, providing a comprehensive description that is both human-readable and suitable for simulation interpretation. In this study, the integration of SDLs is necessary, as it not only guarantees the standardisation and detailing required for accurate scenario generation but also facilitates the identification of internet sources that can provide relevant parameters.

In practice, the first iteration of this study includes the formulation of scenarios that adhere to SDLs, following a defined sentence structure. This is complemented by using the OpenSCENARIO standard, part of the OpenX standards, as an attempt to formulate functional scenarios while adhering to both SDL concepts and OpenSCENARIO. Thus, providing deliverables that are in line with common practices in scenario generation and testing in the automotive industry.

The example provided in Figure 3 shows a simple sentence structure with mapped SDL parameters. The provided example shows in practice how the defined parameters can be utilised during a scenario formulation. This is implemented by structuring the artefact's NoSQL database through a predefined schema, selecting specific parameters with a query.

F. Prototype

The first iteration of the developed tool is a simple web scraper, built using the BeautifulSoup library, a popular tool for parsing HTML and XML documents. The web scraper is designed to generate HTTP requests using the Requests library, a Python package for sending HTTP requests. This functionality is triggered by manually providing a URL, which the scraper uses to fetch the corresponding webpage's content.

The BeautifulSoup library is then employed to parse the fetched content. It sifts through the webpage's source code and isolates specific elements based on the criteria defined in the code. In this first iteration, the scraper is configured to extract all paragraph ('p') elements from the webpage. This is achieved using BeautifulSoup's find_all method, which returns all instances of the specified element in the document.

Once the desired content has been extracted, it is structured into a list of dictionaries, aligning with MongoDB's document structure. Each dictionary represents a paragraph, with the key 'content' mapping to the text of the paragraph. This structured data is then ready for storage.

The storage is handled by PyMongo, a Python driver for MongoDB. The data is stored in a local MongoDB database, serving as a simple solution for data storage. The connection to the database is established using the MongoClient class. If the specified database or collection does not exist, they are automatically created, ensuring a smooth data storage process. The data is then inserted into the collection using the insert_many method, completing the web scraping process.

1) Implementing SDL upper-level concepts: The development of the artefact involved outlining the main concepts that SDLs comprise and their selection based on feasibility and applicability. As a result, we have identified the following as relevant concepts for the development of the web scraper:

- Dynamic elements: These include components that are susceptible to change during a simulation. Typically, this includes moving vehicles, pedestrians, and any other items that may interact with the ego vehicle in unexpected ways.

- Scenery elements: Static elements are those that do not change throughout a simulation. Road signs, traffic signals, buildings, and stationary items along the route.
- Environment elements: Weather conditions, lighting (time of day), and road conditions are examples of environmental attributes.
- Ego Vehicle: Exclusively refers to the car equipped with AD/ADAS systems that is being tested. In this specific context, the ego vehicle is considered an actor.
- Manoeuvre Matrix: This is an outline that describes possible manoeuvres the ego vehicle can execute, often in response to events or conditions within the simulated environment.
- Events and conditions: Specific incidents within the scenario that require the ego vehicle to react, such as an unexpected pedestrian crossing or a sudden stop of the vehicle ahead.

Following the identification of SDL concepts relevant to AD/ADAS systems, the following approach was followed for the development of the artefact on its first iteration:

2) Scenario Template Structure: The principal role of scenario templates within our framework for scenario creation is to establish a structure for scenario generation, aligning with the main concepts of SDLs. The templates employed are XML files, delineated into two main components: a concrete-level XML schema for scenario construction, and a high-level description with mapped placeholder values for attributes. This approach facilitates the integration of data collected via web scraping into our scenario creation workflow. The XML schema employed aims to reflect the OpenSCENARIO standard, capturing specific actions, behaviours, and conditions with more precision. Consequently, this supports a direct translation of data collected through the web scraper into the structured format required for a scenario output.

Figure 4 is an example of a commonly used scenario template for our scenario generation process. The showcased template consists of the XML schema formulated with emphasis on a concrete scenario. In this specific template, the placeholders function as anchors for data input, which delineates the context in which scenario description attributes can be inserted and used. Essentially, each template used in the scenario formulation process adheres to distinct specifications at a concrete level of abstraction, where a varied array of data inputs is utilised to aid in replicating real-world conditions.

The second component of the template includes a high-level description of the scenario. The sentence structure emulates SDL concepts and follows the ordering of the XML schema accordingly:

> In the scenario titled [Scenario_Title], which involves [Actor_Type] with attributes such as a [Vehicle_Model] in [Vehicle_Color], the following events occur: During [Weather_Condition] at [Time_Of_Day] with the roads being [Road_Condition], the [Actor_Type] encounters a situation where [Event_Description]. The trigger for this event is [Event_Trigger], prompting the actor to [Event_Action]. As a result, the manoeuvre undertaken by the actor leads to [Maneuver_Action], which results in [Maneuver_Outcome].

All templates are systematically structured to incorporate specified attributes from defined SDL elements to varying degrees. The arrangement for the identified attributes is outlined as follows:

1) **Actors:**
   - [Actor_Type]: Represents the type of entity involved in the scenario, such as a vehicle, pedestrian, or any other interactive object within the simulation environment.
     - [Vehicle_Model]: Specific model of the vehicle involved.
     - [Vehicle_Color]: Colour of the vehicle, adding detail to the actor's description.
2) **Events:**
   - [Event_Description]: Describes what specific event or incident occurs, detailing the interaction or significant moment in the scenario.
   - [Event_Trigger]: Condition or factor that causes the event to occur, initiating the action.
   - [Event_Action]: The action taken by the actor in response to the event trigger.
3) **Locations:**
   - [Road_Condition]: Describes the condition of the road, which can affect the scenario's dynamics, such as wet, icy, or dry conditions.
4) **Ambient Conditions:**
   - [Weather_Condition]: The weather setting during the scenario, such as rainy, sunny, snowy, etc.
   - [Time_Of_Day]: The time of day, which could influence visibility and the behaviour of the actors, such as dawn, dusk, daytime, or nighttime.
5) **Activities:**
   - [Maneuver_Action]: Describes the specific manoeuvre or action taken by the actor as part of the scenario progression.

```
<Scenario>
    <Title>[Scenario_Title]</Title>
    <Description>[Scenario_Description]</Description>
    <Actors>
        <Actor id="[Actor_ID]" type="[Actor_Type]">
            <Attribute name="Model">[Vehicle_Model]</Attribute>
            <Attribute name="Color">[Vehicle_Color]</Attribute>
        </Actor>
    </Actors>
    <Environment>
        <Weather>[Weather_Condition]</Weather>
        <TimeOfDay>[Time_Of_Day]</TimeOfDay>
        <RoadCondition>[Road_Condition]</RoadCondition>
    </Environment>
    <Events>
        <Event id="[Event_ID]">
            <Description>[Event_Description]</Description>
            <Trigger>[Event_Trigger]</Trigger>
            <Action>[Event_Action]</Action>
        </Event>
    </Events>
    <Maneuver id="[Maneuver_ID]">
        <Action>[Maneuver_Action]</Action>
        <Outcome>[Maneuver_Outcome]</Outcome>
    </Maneuver>
</Scenario>
```

Fig. 4: Example of an XML scenario template used in the scenario formulation process.

- [Maneuver_Outcome]: The result or consequence of the manoeuvre action, providing a result of the scenario's activity.

V. Cycle II: Artefact Refinement

The focus of Cycle II has been on the second research question, i.e., drawing from our results to refine the artefact, with the goal being to create valuable test scenarios in the openSCENARIO framework. Having understood what scenarios might look like in the industry, we have identified ways to further improve the whole pipeline to generate better, more complete, and more realistic results.

Our work in Cycle II, like Cycle I, has encompassed a continuous progression through our research questions in their original order. We began by exploring new possibilities to scrape more content for our artefact, looking both for new interesting websites and their respective rules and for new technical ways to scrape the content. Then, new functionalities were built into the artefact, introducing major implications. Essentially, we have incorporated the expected scenario format into the entire pipeline. In the refined artefact, both the web scraping functionality and the prompts to LLMs have been modified to create and withhold the openSCENARIO format from the start, during the generation of each new test scenario. We believe that this new approach has helped us to ensure a higher quality of result, while simultaneously introducing built-in core mechanics to validate each scenario.

A. Exploring New Places to Scrape Data

Following the results from Cycle I, we started by investigating new ways to gather usable data from the internet. There are many ways to set up a personalised Really Simple Syndication (RSS) news feed, including subscribing to various news feeds at once and receiving news stories as soon as they are released. The feeds are public, and we have found numerous news aggregation services that scrape RSS news feeds and monetize them. There are no limitations on scraping RSS except for their commercial usage, such as running ads on top of the scraped content. RSS are simply small XML files with a header, short content, and a link to the full story. Although RSS stories lack details from the full story, in theory, they provide a simple way to scrape multiple news stories at once, with links to full articles that can be accessed manually in case the story seems

relevant to scenario-driven development. The downside of scraping RSS would be their short content with less valuable information for the scenario, leaving more parts to be completed by an LLM. However, we believe that RSS can theoretically act as triggers in the web scraper architecture. Then, given consent from the hosts, the scraper could assess the criticality of the incident and scrape the full story on the selected media outlet's web page.

News aggregator services, such as the Swedish aggregator Omni.se [25], are interesting in theory because they contain news stories from multiple media outlets in one place. We were enthusiastic about accessing Omni's content. However, we didn't continue due to copyright issues. Omni is currently owned by the Schibsted Media Group, which is an international media corporation that owns multiple Swedish news outlets, including Aftonbladet and Svenska Dagbladet. Selected stories on the website, like most other modern media websites, are also locked behind a paywall and require authentication in order to access selected stories, which raises additional concerns as that data should no longer be considered public. For Schibsted's rules, see "Aftonbladet" in Appendix A.

Lastly, we have explored the idea of scraping the international forum Reddit [26]. Each user on Reddit is anonymous per definition, and there are many interesting channels, called *subreddits*, including those where users may share their personal stories about traffic situations. Unfortunately, Reddit's ToS includes a specific prohibition against scraping the service's contents without given consent (c.f. Appendix A). With the reservation that some user-posted stories may lack important details of a traffic event, include overstatements, or be simply made up from the start, we think that there is future potential to scrape Reddit for the means of AD/ADAS scenario generation and many other domains. Like with news agencies, the interested researcher would opt for getting explicit consent from Reddit before scraping.

B. Refinement of Scenario Generation Through Dynamic Templates

The approach of using dynamically generated template scenarios was chosen based on the rigidity constraints of pre-made templates, as demonstrated in Cycle I. This process enables the creation of scenarios that are in essence adjusted based on plain text descriptions, extracting specifications on elements that are relevant to an OpenSCENARIO description.

The use of pre-made scenario templates as outlined in Cycle I did not effectively capture the level of variability that plain text descriptions displayed; therefore, the process was enhanced by developing a mechanism that included a regular expression (regex) parser for the parsing of plain text descriptions. This was further extended with the creation of an OpenSCENARIO generator class that allowed the construction of XML documents based on the parsed plain text descriptions. The generated XML files adhere to an OpenSCENARIO schema and are ultimately validated to ensure conformity with the OpenSCENARIO standard. This approach has enabled the enhancement of scenarios by dynamically generating templates that are better adjusted to plain text scenario descriptions.

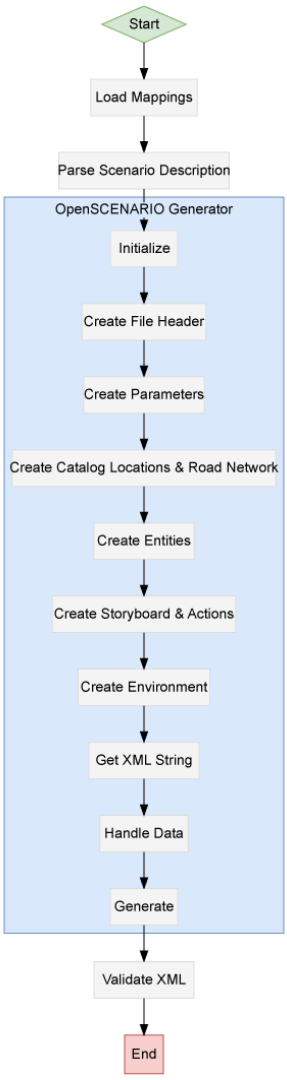


Fig. 5: Flowchart of the OpenSCENARIO Generator module.

The *parse_scenario_description* function parses plain text scenario descriptions given as input; this facilitates the capture of structured information about relevant aspects of the scenario description, including weather conditions, vehicle details, and the sequence of events. This specific function returns a dictionary with the structured data, which is then directly used in the OpenSCENARI-

OGenerator class to create XML elements. Subsequently, the OpenSCENARIOGenerator class generates an XML document while following the OpenSCENARIO schema. The class handles the creation of specific elements per the schema, including a file header, entities populated with details from the parsed description, and details about the conditions, such as weather or road type. The process ends with the compilation of the generated XML document, which is validated against the OpenSCENARIO schema. Figure 5 provides a visual representation of the generator module and its workflow.

1) Syntactic Scenario Validation: Results from the scenarios generated in Cycle I proved that solely relying on SDL concepts and transferring them into non-standard XML descriptions is an unreliable approach due to the lack of established elements transferable to SDL standards commonly used in AD/ADAS systems. Furthermore, the inconsistent nature of utilising LLMs via prompts and the diverse array of unstructured data meant to be scraped resulted in scenarios that lacked specifications for elements critical to scenario descriptions. Under these circumstances, the decision to use the ASAM OpenSCENARIO standard was considered a solution that aligned with the goals of Cycle II.

The implementation of the OpenSCENARIO standard required utilising an XML Schema Definition file (XSD) provided by the Association for Standardization of Automation and Measuring Systems (ASAM). In this specific instance, the schema specifications follow the ASAM OpenSCENARIO v1.2.0 standard and enforce a strict structure defined by four main components:

- **Road Network**: Comprised of mapping relevant information such as driving surfaces and specifics on road infrastructure.
- **Entities**: Road users such as vehicles, pedestrians, and other relevant objects interacting in a scenario.
- **Actions**: A set of dynamic behaviours for the entities. Including modifications in simulation states or components of the simulated world.
- **Triggers**: Mechanisms determining when an action starts or stops. Defined by logical expressions tied to the states of entities.

Utilising the OpenSCENARIO standard enabled the enforcement of consistency in how OpenSCENARIO descriptions were created; mainly, it provided an established structure for how relevant information in plain text descriptions could be translated. The resulting scenarios showed considerable improvement in relation to how specific elements, actors, events, actions, and environmental conditions were defined, thereby minimizing the variability prevalent in unstandardized LLM-generated scenarios. By relying on the XSD file for validation, an early identification and correction of discrepancies in how certain elements were created was made possible, ensuring that only well-structured and abiding scenarios were generated as output during the scenario generation process.

The validation process was included as part of the OpenSCENARIO generator module, specifically developed for the dynamic generation of scenarios. By subsequently validating generated scenarios against the XML schema definition, the module ensures that the output is syntactically correct and semantically consistent with the defined OpenSCENARIO modelling standards. The validation is handled automatically as part of the scenario generation process, removing the need for correction by an LLM during the final steps of the pipeline.

It is important to note that this validation process places greater emphasis on the syntactic aspects of the generated scenarios. While validated to some extent, this approach still requires a separate procedure for the semantic validation of the generated scenarios. Lastly, this is the only validation mechanism that has been fully implemented into the artefact, partly automating the validation process by covering the syntactic aspect in generated scenarios.

2) Semantic Scenario Validation: The semantic validation process covers aspects of consistency, completeness, and logical coherence while conforming to the provided scenario description. Mainly, this section of the validation process focuses on how the data is translated, its consistency, and its clarity, as opposed to a syntactic validation process, which instead focuses on the structure of the scenario. These criteria were chosen based on established practices in scenario-based testing and the convenience of researchers. We acknowledge that further research is needed to validate our approach and refine the criteria selection to create a comprehensive process grounded in practical application and dynamic scenario generation.

Semantic Aspects for Validation

- **Consistency**: Ensure the scenario elements align with realistic driving conditions. Consistency is crucial for reliable validation when testing AD/ADAS features, this is mainly emphasised in simulation-based testing methodologies [27], [28].
- **Clarity**: Check that all required elements of the scenario are present and well-defined. Clear test cases are crucial for ensuring comprehensive coverage of potential driving scenarios [28], [29].
- **Logical Coherence**: Ensure that the sequence of events and actors are consistent and logical. This aspect is crucial for the testing and adjustment of parameters [30].
- **Conformity**: Ensure that the generated OpenSCENARIO description aligns with its plain text scenario description. This aspect validates whether all

elements from the text scenario description are accurately included in the generated scenario.

Overall, the semantic validation process is not meant to be limited to the scenario itself but can equally validate the LLM's capacity to effectively complete a scenario without compromising its structure or specifications.

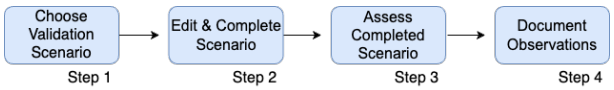


Fig. 6: The steps required for semantic validation on a scenario.

In practice, the validation process for the semantics of a scenario is done through a 4-step process. Step 1 consists of simply choosing a scenario for validation. Step 2 includes partial removal of elements from the scenario and ends by completing it through an LLM. Step 3 involves evaluating the completed scenario with the above-mentioned four semantic aspects. This step can be complemented by comparing the LLM-completed scenario with the original scenario selected for validation. Finally, Step 4 concludes with a brief observation based on the four semantic aspects to consider and any notable discrepancies. Figure 6 illustrates the 4-step process for validation.

A thorough example of an OpenSCENARIO description fully generated through the pipeline of Cycle II is found attached in listings 11 and 12). The showcased scenario has undergone the entire validation process and has been fully generated from a plain text description.

3) Parsing Textual Scenarios: The *parse_scenario_description* function in the scenario generator module is designed to extract structured data from text-based scenario descriptions using regular expressions. The regex pattern is defined through a string to match and extract specific portions of data from the input description, including various named capturing groups to extract parts of the processed string into a dictionary with keys named according to the group names. The groups follow the outline of OpenSCENARIO and are defined as follows:

- 'description': Extracts the full scenario description.
- 'weather': Extracts weather information.
- 'precipitation': Extracts the precipitation type.
- 'wind_speed': Extracts the wind speed in [m/s].
- 'time_of_day': Extracts the DateTime value as per the schema.
- 'traffic': Extracts the traffic intensity.
- 'road_type': Extracts the type of road.
- 'vehicles': Extracts the vehicles involved, type and name.
- 'pedestrians': Extracts the intensity of pedestrians.
- 'events': Extracts the sequence of events.

The function can search through the input description by utilising the 're.search' with the 're.DOTALL' and 're.VERBOSE' flags to apply the regex pattern to the input description.

C. Enhanced Prompt Formulation

The process of formulating prompts was improved in Cycle II by introducing dynamically generated templates. These changes were made to better address formatting and syntactic issues related to the use of static templates. Overall, the enhancement included the removal of the data input as it was made obsolete by introducing plain text descriptions as a starting point for the generation of templates.

Figure 7 showcases the enhanced process, augmented by the incorporation of dynamic templates. The instructions included in the finalized prompts had to be revised to better adjust the changes as well as the objectives provided to the LLMs. The new prompts can be seen in Appendix B in listings 3 and 4, where templates for the Cycle II prompts are included.

Step 1: A plain text description is meant to be formulated, outlining specifications for the entities involved, including weather conditions and any relevant actions. The emphasis is given to the dynamic elements of the description, which will provide the necessary values for the generation of the OpenSCENARIO file. The plain text description is ideally provided through web scraping by filtering and selecting relevant scenarios. However, the current prototype does not provide any mechanism to enable this task, mainly due to limitations in scraping internet sources.

Step 2: The OpenSCENARIO template is generated by parsing the provided plain text description and mapping out relevant values that conform with the OpenSCENARIO standard. This process aids in outlining the entities involved and any parameters that will play a crucial role in the scenario. The process is done automatically after the description is provided as input through the prototype's interface.

Step 3: The prototype's interface provides the option to automatically insert the generated template into a prompt. This is enabled after selecting the template and completing step 2. The finalised prompt will also include the scenario's text description.

Step 4: The finalised prompt can then be copied and pasted and used to generate a scenario as instructed. This process varies depending on whether Llama3 is used or ChatGPT, as the former is the only LLM integrated into the prototype. Steps 3 to 4 need to be repeated to include the second prompt.

Due to constraints related to the Llama3 integration, the process mentioned earlier has only been thoroughly

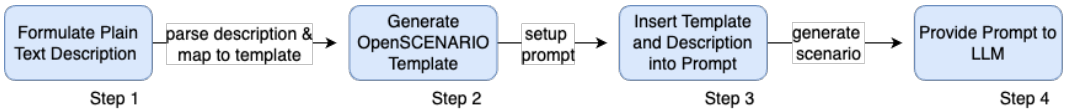


Fig. 7: Refined process for prompt formulation in Cycle II.

applied when using ChatGPT and thus requires further development to integrate it into the pipeline.

D. Integration of Llama3

We wanted to explore how different LLMs perform given the exact same tasks and if there would be large differences in quality, logic, and realism in the generated scenarios. Our original plan consisted of comparing the popular services of OpenAI ChatGPT 4.0, Microsoft 365 Copilot, and Google Gemini. However, all these three models are proprietary, and we have instead opted to compare how a proprietary model (GPT) compares against a more open model. We have downloaded the weights of the 8B version of Meta Llama3. Using Ollama, an open-source platform for running LLMs locally, we were able to customise the "clean" Llama3 8B version to fit our needs using so-called modelfiles. Modelfiles are essentially blueprints for customised models in Ollama.

Following the 2-prompts strategy, we wrote 2 separate modelfiles, each a custom model built on top of Llama3 8B. The first model, which we titled "scenario-structure", takes any natural text—in our case, a short description of a traffic incident scraped from the web—and refactors it into the specific OpenSCENARIO XML structure. The second model, "scenario-completion", takes the generated result from "scenario-structure" as input and specifically looks for missing elements to replace them with credible, realistic scenario details. In other words, though there are two models, only one "prompt" is provided (i.e., the scraped natural text), while our full "prompts" from before are instead built into the models from the start.

We have included both modelfiles along with the Python script that runs them. Although part of the artefact, the integration is not fully implemented; as Llama3 is a local model, it requires downloading the model weights and creating the modelfiles locally for the Python script to compile. Hence, the Llama functionality is separate from the scraper module and the scenario generator module. The parts included in the artefact have been tested separately.

Generally, the result of this approach has been that Llama is less coherent in its results and allows itself to freely move parameters around or even introduce new parameters while deleting some necessary ones. We believe this result indicates a need to further experiment with the modelfiles. For instance, the "temperature" argument, which controls the level of creativity of a customised model, could be changed with subsequent comparisons of results.

VI. Discussion

A. Interpretation of Findings

As evidenced by the steps we took to evaluate and validate the results, our pipeline can produce scenarios very similar to full scenarios from other sources. Our interpretation is that, generally, all parts of the pipeline function as expected and demonstrate sufficient levels of applicability in "real-life", i.e., industry. The artefact can use scraped texts to dynamically create test scenarios for AD/ADAS testing. Moreover, ChatGPT has been shown to complete the scenarios with reasonable components.

B. Comparison With Existing Literature

It is a challenge to compare our findings with the research we have identified in related work. After all, combining web scraping with generative AI is a novel approach in AD/ADAS scenario generation, limiting the number of direct comparisons we can possibly make.

Referring to Krotov and Silva's overview of the legality and ethics of web scraping, we have been thorough in following the guiding principles outlined by the authors [10]. We would add, however, that although a legislature may lack rules specifically addressed to web scraping, using this technology is still indirectly associated with a multitude of various legislative documents, such as GDPR in the case of the EU. Hence, identifying local laws that may indirectly address the use of web scraping is an important first step in any research with the technology. Krotov and Silva's guiding principles, however, remain important in this context and should be considered in any possible use case.

Riedmaier et al. have underlined the importance of finding the most representative scenarios in the scenario-based safety assessment of AD vehicles [6]. We can argue that the representativeness of our own scenarios is heavily influenced by the incorporation of LLMs in the generation process. Although the LLMs merely build upon scraped components of real-life traffic incidents, the added details can vary in both realism and logical cohesion. At the same time, drawing inspiration from the framework by Alnaser et al. [3], LLMs could theoretically be used to create countless variations of the same

scraped incident, like how the researchers have used their seed and constraints matrix to generate pseudo-random experiments for the ego vehicle. In other words, a single scraped article holds the potential to be the foundation for many test scenarios. The additional versions with varying ego car models, traffic, environment, and other settings can help find the most representative test case for a specific scraped text.

C. Implications of Findings

At the core of this research is the proposition of a new approach to construct useful test cases for AD/ADAS. AI continues to grow on a massive scale, and more companies are building various AI-driven capabilities into a wider range of services. According to Lachmann and Schaefer, the complexity of software components in new vehicles keeps rising, posing new challenges in the industry, especially regarding autonomous driving [2]. The utmost criticality of the safety of new vehicles requires exploring new solutions and testing new approaches. Thus, we believe that the combination of web scraping and generative AI technology could propel the industry in new directions and help address these challenges from an entirely new vantage point.

Combining the powers of AI and web scraping to generate new test scenarios in the automotive industry has theoretical implications for the field. The tandem of the two technologies holds certain promises. With access to more resources, researchers can apply this tandem on a larger scale, incorporating further elements in the pipeline to build a more comprehensive scenario-generation tool. Such elements, for example, could be the integration of existing simulation programs with the artefact or the incorporation of mathematical and statistical models to further support scenario creation and evaluation.

Therefore, further expanded and improved variations of our proposed method hold practical applications as well. Practitioners in the industry can apply it as an additional strategy for AD/ADAS testing. This artefact can be customised or reproduced in different formats, tailored to the needs of specific companies, or used for testing specific vehicle components.

D. Threats to Validity

We have planned to interview experts in the field of scenario-based testing in the automotive industry to validate the generated scenarios and understand their value to practitioners. Unfortunately, we have not found available interviewees. The lack of a semantic analysis in the context of the domain constitutes a threat to the construct validity of our research. Although we have seen that our artefact generates plausible test scenarios, the consequence of this threat to validity is that we do not know to which extent the tool can generate value for practitioners. Thus, it also becomes challenging to suggest a viable plan for future additions to the artefact.

Another noticeable threat to construct validity has been the generally restrictive policies of most identified web resources. Our time constraints have not allowed us to seek explicit consent agreements to scrape content from specific websites. We believe that such consent agreements, in theory, could have resulted in scenarios with a higher percentage of actual real-life incident components. By having access to scrape a larger number of texts, and more detailed texts, we could have identified both more advantages and disadvantages of the proposed approach. The exclusion of valuable sources due to limitations in their rules is also a threat to internal validity since it resulted in a less representative sample.

Furthermore, the semantic validation of scenarios, specifically the first and second steps involving the parallel completion and comparison of complete scenarios from different sources by an LLM, mainly serves as an approach to assess the LLM's capacity to generate coherent, plausible, and contextually accurate scenarios based on partial information, serving as a crucial aspect of the validation by handling sources with missing details and testing the LLM's ability to complete these details accurately. However, the lack of established practices around LLM-generated scenarios poses a threat to construct validity since there are no standardised conventions for ensuring that generated scenarios remain accurate to their source when enhanced by LLMs.

In addition, the inherent variability of LLM outputs can introduce factors that cannot be controlled, leading to a threat to internal validity. This variability is exacerbated by the inconsistent outputs given by LLMs, rendering the reliability and consistency of the generated scenarios a challenging process. While this step of validation reinforces the pipeline by providing an approach to evaluating how an LLM performs in generating complete scenarios, it does not fully validate the accuracy and applicability of these scenarios in a practical setting.

Lastly, we lacked access to industry-standard virtual scenario generation and testing environments. Hence, we could not import our own scenarios into such simulation programs to build complete, full test cases and evaluate how the scenarios would look in a virtual environment. This is a threat to external validity, as it limits the generalizability of our findings to industrial settings. Importing our scenarios into a suitable virtual testing framework could reveal further limitations of our approach and underline crucial areas for improvement, paving the way for potential future implementation in automotive companies.

VII. Conclusions

A. Summary of Key Findings

The purpose of this study has been to advance the state-of-the-art of AD/ADAS development by introducing a novel approach to scenario-based testing. Various websites on the internet have been explored, with a focus on traffic incident data. Resources possessing such data have been thoroughly assessed for their data accessibility rules and policies. Although it is not possible to access the content through an automated script considering the permissions of the owners in most cases, we have shown the applicability of the proposed methodology by generating test scenarios in a specific format. In this study, a rigorous step-by-step process for accessing related content and transforming natural language text into scenarios using LLMs was demonstrated. Further studies might explore obtaining explicit consent to scrape the most detail-rich websites.

We identified a list of usable resources and included their specific rules related to web scraping. Then, we explored how data can be effectively collected using a web scraper. To address the gaps in the discovered scenario, a 2-prompt approach was utilised with ChatGPT 4.0 in Cycle I, prompting it to first structure and then completing a prospective test scenario. Drawing from the insights from Cycle I, we have improved the process in Cycle II by introducing dynamic scenario generation using the openSCENARIO schema. Both a syntactic and semantic check have been used to validate the resulting dynamic scenarios. Finally, a comparison of the results of ChatGPT with customised instances of Llama 3 was conducted, concluding that ChatGPT is a more capable model for this task.

The findings underline the vitality of the proposed method of scenario generation. Although some technical challenges persist in both the scraping aspect and using LLMs, we assess that the fundamental idea has been demonstrated. The importance of these findings lies in their potential for future developments in the field of AD/ADAS testing. The vastly expanding scope of available data online, together with the rapid development of generative AI, holds additional promise for further improvements and application of the proposed way of working.

B. Suggestions for Future Research

The new pipeline for AD/ADAS scenario generation presented in this study has the potential for future refinement. First and foremost, it can be recommended to gain explicit access to the contents of more websites. We also encourage potential future researchers to look beyond the scope of the sources we have identified here. Considering sources such as news aggregators, social media, and personal blogs hold potentially valuable incident information, the finished scenarios will benefit from having more qualitative data inputs from the web scraper. Most importantly, this could help elicit new edge-case scenarios, which would help AD/ADAS developers locate vital areas for improvement in new vehicle components and systems.

Furthermore, in this study, the exploration of alternative generative AI models is recommended for future developers. With the rapid development of AI-powered technology, we assume that its capability to produce useful test scenarios will only expand. A concrete suggestion for future research is to acquire models with the highest degree of allowed customisation and tailor these models for scenario generation. For instance, specific schemas, guidelines, and practices of a selected SDL could be directly embedded into a generative model, covering the multiple case-specific quality and functional requirements of the test cases. A vigorous evaluation of scenarios, however, is strongly recommended for the purpose of ensuring compatibility with real life. Different models can be compared to achieve this task. As a footnote, we recommend using the most capable and big models, as "heavier" models have more parameters and are reasonably better suited for the complexity of scenario generation.

Lastly, the technical prowess of the web scraper can be expanded, such as by allowing fully automatic finding and scraping of relevant articles on a given website. A powerful, automated web scraper, in conjunction with all necessary scenario frameworks and rigorously crafted AI models, has endless potential for future development and experimentation. We believe that continued research in this direction has the potential to transform the process of test scenario elicitation in the automotive domain, propelling businesses and the safety of passengers.

VIII. Acknowledgement

We would like to warmly thank our supervisor, Muhammed Çağrı Kaya, for his continued support and valuable feedback throughout our research. We deeply appreciate his dedication and patience, which have substantially contributed to the successful completion of this thesis. Additionally, we thank our examiner, Christian Berger, for his insightful feedback on our research, which helped us improve throughout the many iterations of our paper.

References

[1] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06. New York, NY, USA: Association for Computing Machinery, May 2006, p. 33–42.

[2] R. Lachmann and I. Schaefer, "Towards efficient and effective testing in automotive software development," in *Informatik 2014*. Bonn: Gesellschaft für Informatik e.V., 2014, pp. 2181–2192.

[3] A. J. Alnaser, M. I. Akbas, A. Sargolzaei, and R. Razdan, "Autonomous vehicles scenario testing framework and model of computation," *SAE International Journal of Connected and Automated Vehicles*, vol. 2, no. 4, Dec 2019.

[4] P. Cihan and E. Cerrahoğlu, "Web scraping and machine learning techniques to prediction of secondhand car prices," pp. 410–419, Jun 2023, accessed: 2024-04-25. [Online]. Available: https://www.researchgate.net/publication/374915042_Web_Scraping_and_Machine_Learning_Techniques_to_Prediction_of_Secondhand_Car_Prices

[5] B. F. Maione, P. C. Kaminski, and E. C. Baraldi, "The automotive recall data search and its analysis applying machine learning," *Production/Produção*, vol. 33, p. e20220117, Jun 2023, issue Date: 02 June 2023.

[6] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, May 2020, received March 19, 2020, accepted April 23, 2020, date of publication May 11, 2020, date of current version May 21, 2020.

[7] H. H. Hong Tan, Fuquan Zhao and Z. Liu, "Estimate of safety impact of lane keeping assistant system on fatalities and injuries reduction for china: Scenarios through 2030," *Traffic Injury Prevention*, vol. 21, no. 2, pp. 156–162, Feb 2020, received 07 Apr 2019, Accepted 30 Dec 2019, Published online: 05 Feb 2020.

[8] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, Mar 2004. [Online]. Available: https://dl.acm.org/doi/10.5555/2017212.2017217

[9] V. Singrodia, A. Mitra, and S. Paul, "A review on web scrapping and its applications," in *2019 International Conference on Computer Communication and Informatics (ICCCI)*, Coimbatore, India, Jan 2019, pp. 1–6.

[10] V. Krotov and L. Silva, "Legality and ethics of web scraping," in *Twenty-fourth Americas Conference on Information Systems*. New Orleans, LA, USA: Murray State University; University of Houston, Sep 2018.

[11] G. S. Kalra, R. S. Kathuria, and A. Kumar, "Youtube video classification based on title and description text," in *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*. Greater Noida, India: IEEE, Oct 2019, pp. 74–79.

[12] X. Li, "A scenario-based development framework for autonomous driving," 2020, arXiv:2011.01439v2 [cs.DC]. [Online]. Available: https://arxiv.org/abs/2011.01439

[13] J. Ma, X. Che, Y. Li, and E. M.-K. Lai, "Traffic scenarios for automated vehicle testing: A review of description languages and systems," *Machines*, vol. 9, no. 12, 2021. [Online]. Available: https://www.mdpi.com/2075-1702/9/12/342

[14] A. Koubaa, "Gpt-4 vs. gpt-3.5: A concise showdown," Mar 2023, accessed: 2024-04-25. [Online]. Available: https://www.researchgate.net/publication/369897711_GPT-4_vs_GPT-35_A_Concise_Showdown

[15] "Llama 3 model card," https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md, 2024, accessed: 2024-06-07.

[16] "Trafikverket's api," https://www.trafikverket.se/e-tjanster/trafikverkets-oppna-api-for-trafikinformation/, 2024, accessed: 2024-05-23.

[17] "Transportstyrelsen's strada database access," https://www.transportstyrelsen.se/sv/vagtrafik/statistik/olycksstatistik/om-strada/, 2024, accessed: 2024-05-23.

[18] "Youtube's terms of service," https://www.youtube.com/static?template=terms, 2024, accessed: 2024-05-23.

[19] "Bonnier news' rules," https://www.bonniernews.se/info/upphovsratt-och-ai, 2024, accessed: 2024-05-23.

[20] "Aftonbladet's rules," https://www.aftonbladet.se/omaftonbladet/a/7lwLPv/anvandarvillkor-aftonbladet, 2024, accessed: 2024-05-23.

[21] "trafiken.nu's open data policy," https://trafiken.nu/goteborg/sidfot/om-oppna-data/, 2024, accessed: 2024-05-23.

[22] Y. Liu, Y. Yao, J.-F. Ton, X. Zhang, R. G. H. Cheng, Y. Klochkov, M. F. Taufiq, and H. Li, "Trustworthy llms: a survey and guideline for evaluating large language models' alignment," 2024.

[23] T. Braun, L. Ries, F. Körtke, L. Turner, S. Otten, and E. Sax, "Collection of requirements and model-based approach for scenario description," in *7th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2021)*, May 2021, pp. 634–645.

[24] W. U. of Warwick, "Scenario description language (sdl) - version 8.1," accessed: 2024-04-19. [Online]. Available: https://cdn.document360.io/d9bf0084-0d5b-468a-971b-537dce7153b6/Images/Documentation/WMG_SDL_8.1.pdf

[25] "Omni homepage," https://www.omni.se, 2024, accessed: 2024-05-16.

[26] "Reddit's user agreement," https://www.redditinc.com/policies/user-agreement, 2024, accessed: 2024-05-23.

[27] K. Shibuya, A. Hyodo, A. Akai, and T. Yamada, "Automatic scenario generation for simulation-based testing of ad/adas," Hitachi, Ltd., Hitachi Astemo, Ltd., Tech. Rep., 2023. [Online]. Available: https://doi.org/10.4271/2023-01-0825

[28] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," pp. 1821–1827, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1801.08598

[29] Siemens, "Generating scenarios for adas and avs," 2023, accessed: 2024-04-25. [Online]. Available: https://static.sw.cdn.siemens.com/siemens-disw-assets/public/22hX0xqsUK02zZPytpjluK/en-US/Siemens%20SW%20Generating%20scenarios%20for%20ADAS%20and%20AVs.pdf

[30] M. Markofsky, M. Schäfer, and D. Schramm, "Use cases and methods of virtual adas/ads calibration in simulation," *Vehicles*, vol. 5, no. 3, pp. 802–829, 2023. [Online]. Available: https://www.mdpi.com/2624-8921/5/3/44

APPENDIX A
WEBSITES' RULES ON WEB SCRAPING

Here, we have included copies of the rules of the mentioned websites, describing their policies related to web scraping.

A. *Transportstyrelsen's Strada Database [17]*

> "Tillgången till Strada uttagswebb styrs av lagen om Transportstyrelsens olycksdatabas (2021:319)
> Enligt 15 § får följande organisationer tillgång till Strada Uttagswebb; Myndigheten för samhällsskydd och beredskap, Polismyndigheten, Statens väg- och transportforskningsinstitut, Trafikanalys, Trafikverket, kommunala myndigheter och länsstyrelser."

B. *YouTube [18]*

> "The following restrictions apply to your use of the Service. You are not allowed to:
> [...] access, reproduce, download, distribute, transmit, broadcast, display, sell, license, alter, modify or otherwise use any part of the Service or any Content except: (a) as specifically permitted by the Service; (b) with prior written permission from YouTube and, if applicable, the respective rights holders; or (c) as permitted by applicable law; circumvent, disable, fraudulently engage, or otherwise interfere with the Service (or attempt to do any of these things), including security-related features or features that: (a) prevent or restrict the copying or other use of Content; or (b) limit the use of the Service or Content; access the Service using any automated means (such as robots, botnets or scrapers) except: (a) in the case of public search engines, in accordance with YouTube's robots.txt file; (b) with YouTube's prior written permission; or (c) as permitted by applicable law; ..."

C. *Bonnier News [19]*

> "Bonnier News tillåter inte "crawling" eller "scraping" eller liknande användning av det innehåll som görs tillgängligt via Tjänsten, vare sig det sker manuellt eller automatiserat, eller på annat sätt med hjälp av automatiserade metoder (inklusive robotar, skrapor och spindlar) för att visa, komma åt eller samla in information, av vilket slag det än må vara. Vidare är det inte tillåtet att använda innehållet eller vår data (inklusive tillhörande metadata) för att träna en maskininlärnings- och/eller AI-modell eller på annat sätt föra in innehåll från Tjänsten i en maskininlärnings- och/ eller AI-modell."

D. *Aftonbladet [20]*

> "Aftonbladet, eller Aftonbladets licensgivare, innehar de immateriella rättigheterna till text, bild, design och det övriga material och information som görs tillgängligt för dig genom din användning av Tjänsten. Samma sak gäller den bakomliggande programkoden för Tjänsten. Sådant material och information får inte användas på annat sätt än inom ramen för normal användning av Tjänsten."

E. *trafiken.nu [21]*

> "Mycket av informationen på Trafiken.nu bygger på öppna data, digital information som är fritt tillgänglig för alla.
> På http://goteborg.se/psidata kan du via ett API hämta öppna data från Göteborgs Stad att använda i dina applikationer eller analysera med valfritt verktyg. Det går också bra att hämta data direkt till Excel. Även data till Trafikverket och Västtrafiks respektive API:er finns att hitta via länkar."

Robots.txt:

> User-agent: * Disallow: /driftmeddelanden/ Disallow: /*?currentRoad=*

F. *Reddit [26]*

> "In addition to what is prohibited in the Content Policy, you may not do any of the following: [...]
> Access, search, or collect data from the Services by any means (automated or otherwise) except as permitted in these Terms or in a separate agreement with Reddit (we conditionally grant permission to crawl the Services in accordance with the parameters set forth in our robots.txt file, but scraping the Services without Reddit's prior written consent is prohibited); ..."

APPENDIX B
PROMPTS FOR SCENARIO FORMULATION

```
**Objective:** Given unstructured data scraped from various web sources, your task is to
    organize the information into both an XML formatted scenario and its equivalent high-level
    scenario description sentence structure based on the provided SDL concepts. If certain
    elements are missing in the scraped data, use 'element_not_found' as a placeholder.

**Data Input:**
- Time of Day: {time_of_day}
- Actor: {actor} (if empty, use 'element_not_found')
- Vehicle Type: {vehicle_type}
- Event: {event}
- Location: {location}
- Weather Conditions: {weather_conditions}
- Visibility Distance: {visibility_distance} (if empty, use 'element_not_found')
- Traffic Conditions: {traffic_conditions}
- Activity: {activity}

**Scenario Template:**
- XML scenario template:
```INSERT XML SCENARIO TEMPLATE HERE```

- High-level scenario description sentence
```INSERT SDL SENTENCE FROM TEMPLATE HERE```
```

Listing 1: First prompt used for scenario formulation in Cycle I

```
**Objective:** Refine the previously generated scenario by filling in the missing parameters
    where 'element_not_found' is indicated. Your task is to ensure that each added parameter is
     plausible, coherent, and aligns with typical driving environments and standard SDL
    concepts.

**Given Scenario:**
- XML format:
```INSERT XML SCENARIO GIVEN IN FIRST PROMPT HERE```

- High-level scenario description sentence:
```INSERT HIGH LEVEL DESCRIPTION SENTENCE HERE```
```

Listing 2: Second prompt used for scenario generation in Cycle I

```
**Objective:** Given a scenario description, your task is to enhance it by filling in elements
    and missing values all whilst respecting the OpenSCENARIO XML v1.2.0 standard. Your
    response should only include the enhanced OpenSCENARIO description and a high-level
    scenario description sentence.

**Scenario Template:**
- XML scenario template:
```INSERT XML SCENARIO TEMPLATE HERE```

- High-level scenario description sentence
```INSERT SDL SENTENCE FROM TEMPLATE HERE```
```

Listing 3: First prompt used for scenario formulation in Cycle II

```
**Objective:** Refine the previously generated scenario by correcting inconsistent parameters.
    Your task is to ensure that each parameter is plausible, coherent, and aligns with the
    specifications of this scenario while adhering to the OpenSCENARIO XML v1.2.0 standard.
    Your response should only include the refined OpenSCENARIO description and a high-level
    scenario description sentence.

**Given Scenario:**
- XML format:
```INSERT XML SCENARIO GIVEN IN FIRST PROMPT HERE```

- High-level scenario description sentence:
```INSERT HIGH LEVEL DESCRIPTION SENTENCE HERE```
```

Listing 4: Second prompt used for scenario generation in Cycle II

APPENDIX C
EXAMPLES OF GENERATED SCENARIOS

An example of a generated scenario from the pipeline is provided here. The dynamic scenario has been generated out of a plain text description and has been enhanced with the use of LLMs.

```
<?xml version='1.0' encoding='UTF-8'?>
<OpenSCENARIO>
  <FileHeader revMajor="1" revMinor="2" date="2024-05-17T00:01:26" description="Enhanced
    Intersection Collision Scenario" author="GU">
    <License name="ASAM OpenSCENARIO" resource="https://www.asam.net/standards/detail/
    openscenario/" spdxId="ASAM-1.2"/>
  </FileHeader>
  <ParameterDeclarations/>
  <CatalogLocations/>
  <RoadNetwork/>
  <Entities>
    <ScenarioObject name="Sedan">
      <Vehicle name="Sedan" vehicleCategory="car">
        <BoundingBox>
          <Center x="1.5" y="0" z="0.5"/>
          <Dimensions width="2.0" length="4.5" height="1.4"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.6" positionX="3.2"
    positionZ="0.35"/>
          <RearAxle maxSteering="0" wheelDiameter="0.7" trackWidth="1.6" positionX="1.0"
    positionZ="0.35"/>
        </Axles>
        <Properties>
          <Property name="type" value="sedan"/>
        </Properties>
        <Performance maxSpeed="180" maxAcceleration="3.5" maxDeceleration="8.0"/>
      </Vehicle>
    </ScenarioObject>
    <ScenarioObject name="Pickup Truck">
      <Vehicle name="Pickup Truck" vehicleCategory="truck">
        <BoundingBox>
          <Center x="1.5" y="0" z="0.5"/>
          <Dimensions width="2.5" length="5.0" height="1.8"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.8" positionX="3.7"
    positionZ="0.45"/>
          <RearAxle maxSteering="0" wheelDiameter="0.7" trackWidth="1.8" positionX="1.1"
    positionZ="0.45"/>
        </Axles>
        <Properties>
          <Property name="type" value="pickup"/>
        </Properties>
        <Performance maxSpeed="160" maxAcceleration="3.2" maxDeceleration="7.5"/>
      </Vehicle>
    </ScenarioObject>
  </Entities>
```

Listing 5: Enhanced Intersection Collision Scenario (Part 1)

```
  <Storyboard>
    <Init>
      <Actions>
        <GlobalAction>
          <EnvironmentAction>
            <Environment name="ClearSunnyAfternoon">
              <TimeOfDay animation="false" dateTime="2024-05-17T15:00:00Z"/>
              <Weather>
                <Precipitation precipitationType="dry" precipitationIntensity="0"/>
                <Wind speed="5" direction="90"/>
              </Weather>
              <RoadCondition frictionScaleFactor="0.9" wetness="dry">
                <Properties>
                  <Property name="surfaceType" value="asphalt"/>
                </Properties>
              </RoadCondition>
            </Environment>
          </EnvironmentAction>
        </GlobalAction>
      </Actions>
    </Init>
    <Story name="IntersectionCollisionStory">
      <Act name="CollisionAct">
        <ManeuverGroup name="CollisionManeuverGroup" maximumExecutionCount="1">
          <Actors selectTriggeringEntities="true">
            <EntityRef entityRef="Sedan"/>
            <EntityRef entityRef="Pickup Truck"/>
          </Actors>
          <Maneuver name="ImpactManeuver">
            <Event name="CollisionEvent" priority="overwrite">
              <Action name="ImpactAction">
                <PrivateAction>
                  <LongitudinalAction>
                    <SpeedAction>
                      <SpeedActionDynamics dynamicsShape="linear" dynamicsDimension="time"
    value="2.0"/>
                      <SpeedActionTarget>
                        <AbsoluteTargetSpeed value="80"/>
                      </SpeedActionTarget>
                    </SpeedAction>
                  </LongitudinalAction>
                </PrivateAction>
              </Action>
            </Event>
          </Maneuver>
        </ManeuverGroup>
        <StartTrigger>
          <ConditionGroup>
            <Condition name="collisionStartCondition" conditionEdge="rising" delay="0">
              <ByValueCondition>
                <ParameterCondition parameterRef="VehicleSpeed" rule="greaterThan" value="50"/>
              </ByValueCondition>
            </Condition>
          </ConditionGroup>
        </StartTrigger>
      </Act>
    </Story>
    <StopTrigger/>
  </Storyboard>
</OpenSCENARIO>
```

Listing 6: Continued: Enhanced Intersection Collision Scenario (Part 2)

```xml
<?xml version='1.0' encoding='UTF-8'?>
<OpenSCENARIO>
  <FileHeader revMajor="1" revMinor="2" date="2024-06-07T13:49:13" description="Refined Dynamic Driving Scenario" author="GU">
    <License name="ASAM OpenSCENARIO" resource="https://www.asam.net/standards/detail/openscenario/" spdxId="ASAM-1.2"/>
  </FileHeader>
  <ParameterDeclarations/>
  <CatalogLocations/>
  <RoadNetwork/>
  <Entities>
    <ScenarioObject name="Honda Civic">
      <Vehicle name="Honda Civic" vehicleCategory="car">
        <BoundingBox>
          <Center x="1.5" y="0" z="0.5"/>
          <Dimensions width="1.8" length="4.6" height="1.4"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.6" trackWidth="1.5" positionX="3.5" positionZ="0.4"/>
          <RearAxle maxSteering="0" wheelDiameter="0.6" trackWidth="1.5" positionX="1.0" positionZ="0.4"/>
        </Axles>
        <Properties>
          <Property name="type" value="sedan"/>
        </Properties>
        <Performance maxSpeed="180" maxAcceleration="5.0" maxDeceleration="10.0"/>
      </Vehicle>
    </ScenarioObject>
    <ScenarioObject name="Van">
      <Vehicle name="Van" vehicleCategory="van">
        <BoundingBox>
          <Center x="1.6" y="0" z="0.6"/>
          <Dimensions width="2.0" length="4.8" height="2.0"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.8" trackWidth="1.8" positionX="3.4" positionZ="0.5"/>
          <RearAxle maxSteering="0" wheelDiameter="0.8" trackWidth="1.8" positionX="1.2" positionZ="0.5"/>
        </Axles>
        <Properties>
          <Property name="type" value="van"/>
        </Properties>
        <Performance maxSpeed="140" maxAcceleration="3.0" maxDeceleration="7.0"/>
      </Vehicle>
    </ScenarioObject>
    <ScenarioObject name="Sports Car">
      <Vehicle name="Sports Car" vehicleCategory="car">
        <BoundingBox>
          <Center x="1.4" y="0" z="0.4"/>
          <Dimensions width="1.9" length="4.3" height="1.2"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="45" wheelDiameter="0.5" trackWidth="1.4" positionX="3.7" positionZ="0.3"/>
          <RearAxle maxSteering="0" wheelDiameter="0.5" trackWidth="1.4" positionX="1.1" positionZ="0.3"/>
        </Axles>
        <Properties>
          <Property name="type" value="sports car"/>
        </Properties>
        <Performance maxSpeed="240" maxAcceleration="6.5" maxDeceleration="12.0"/>
      </Vehicle>
    </ScenarioObject>
  </Entities>
```

Listing 7: First iteration of the enhanced Highway cut-in incident Scenario. This scenario was generated with the first prompt following the 2-step process for enhancement by an LLM. It demonstrates how the scenario is improved through both iterations. (Part 1)

```
<Storyboard>
   <Init>
     <Actions>
       <GlobalAction>
         <EnvironmentAction>
           <Environment name="DynamicDriveEnvironment">
             <TimeOfDay animation="true" dateTime="2024-06-07T18:45:00Z"/>
             <Weather>
               <Sun intensity="0.4" azimuth="50" elevation="5"/>
               <Wind speed="5" direction="135"/>
               <Precipitation precipitationType="lightRain" precipitationIntensity="0.2"/>
             </Weather>
             <RoadCondition>
               <Surface frictionScaleFactor="0.6" roughness="0.5"/>
             </RoadCondition>
           </Environment>
         </EnvironmentAction>
       </GlobalAction>
     </Actions>
   </Init>
   <Story name="EveningCommuteStory">
     <Act name="CommuteAct">
       <ManeuverGroup name="MainManeuverGroup" maximumExecutionCount="1">
         <Actors selectTriggeringEntities="true">
           <EntityRef entityRef="Honda Civic"/>
           <EntityRef entityRef="Van"/>
           <EntityRef entityRef="Sports Car"/>
         </Actors>
         <Maneuver name="MainManeuver">
           <Event name="EveningCommuteEvent" priority="overwrite">
             <Action name="DriveAction">
               <PrivateAction>
                 <LongitudinalAction>
                   <SpeedAction>
                     <SpeedActionDynamics dynamicsShape="step" dynamicsDimension="time" value=
   "1.5"/>
                     <SpeedActionTarget>
                       <RelativeTargetSpeed value="85" continuous="false"/>
                     </SpeedActionTarget>
                   </SpeedAction>
                 </LongitudinalAction>
               </PrivateAction>
             </Action>
           </Event>
         </Maneuver>
       </ManeuverGroup>
       <StartTrigger>
         <ConditionGroup>
           <Condition name="startCondition" conditionEdge="rising" delay="0">
             <ByEntityCondition>
               <TriggeringEntities rule="any">
                 <EntityRef entityRef="Honda Civic"/>
               </TriggeringEntities>
               <EntityCondition>
                 <ReachPositionCondition tolerance="2.0" alongRoute="true" positionX="50"
   positionY="0" positionZ="10" roadId="1" laneId="2" s="200"/>
               </EntityCondition>
             </ByEntityCondition>
           </Condition>
         </ConditionGroup>
       </StartTrigger>
     </Act>
   </Story>
 </Storyboard>
</OpenSCENARIO>
```

Listing 8: Continued: First iteration of the enhanced Highway cut-in incident Scenario (Part 2)

```xml
<?xml version='1.0' encoding='UTF-8'?>
<OpenSCENARIO>
  <FileHeader revMajor="1" revMinor="2" date="2024-06-07T13:49:13" description="Refined Dynamic Driving Scenario" author="GU">
    <License name="ASAM OpenSCENARIO" resource="https://www.asam.net/standards/detail/openscenario/" spdxId="ASAM-1.2"/>
  </FileHeader>
  <ParameterDeclarations/>
  <CatalogLocations/>
  <RoadNetwork/>
  <Entities>
    <ScenarioObject name="Honda Civic">
      <Vehicle name="Honda Civic" vehicleCategory="car">
        <BoundingBox>
          <Center x="1.5" y="0" z="0.5"/>
          <Dimensions width="1.8" length="4.6" height="1.4"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.65" trackWidth="1.6" positionX="3.3" positionZ="0.35"/>
          <RearAxle maxSteering="0" wheelDiameter="0.65" trackWidth="1.6" positionX="1.3" positionZ="0.35"/>
        </Axles>
        <Properties>
          <Property name="type" value="sedan"/>
        </Properties>
        <Performance maxSpeed="180" maxAcceleration="5.0" maxDeceleration="10.0"/>
      </Vehicle>
    </ScenarioObject>
    <ScenarioObject name="Van">
      <Vehicle name="Van" vehicleCategory="van">
        <BoundingBox>
          <Center x="1.7" y="0" z="0.75"/>
          <Dimensions width="2.1" length="5.0" height="2.2"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="25" wheelDiameter="0.85" trackWidth="1.7" positionX="3.5" positionZ="0.5"/>
          <RearAxle maxSteering="0" wheelDiameter="0.85" trackWidth="1.7" positionX="1.2" positionZ="0.5"/>
        </Axles>
        <Properties>
          <Property name="type" value="van"/>
        </Properties>
        <Performance maxSpeed="140" maxAcceleration="3.0" maxDeceleration="7.0"/>
      </Vehicle>
    </ScenarioObject>
    <ScenarioObject name="Sports Car">
      <Vehicle name="Sports Car" vehicleCategory="sportsCar">
        <BoundingBox>
          <Center x="1.4" y="0" z="0.4"/>
          <Dimensions width="1.9" length="4.3" height="1.1"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="50" wheelDiameter="0.55" trackWidth="1.4" positionX="3.6" positionZ="0.25"/>
          <RearAxle maxSteering="0" wheelDiameter="0.55" trackWidth="1.4" positionX="1.0" positionZ="0.25"/>
        </Axles>
        <Properties>
          <Property name="type" value="sportsCar"/>
        </Properties>
        <Performance maxSpeed="250" maxAcceleration="7.0" maxDeceleration="12.0"/>
      </Vehicle>
    </ScenarioObject>
  </Entities>
```

Listing 9: Final iteration of the enhanced Highway cut-in incident Scenario. The scenario was improved through the second and final prompt as specified in the 2-step process for enhancement by an LLM. (Part 1)

```
<Storyboard>
   <Init>
     <Actions>
       <GlobalAction>
         <EnvironmentAction>
           <Environment name="DynamicDriveEnvironment">
             <TimeOfDay animation="true" dateTime="2024-06-07T18:45:00Z"/>
             <Weather>
               <Sun intensity="0.4" azimuth="50" elevation="10"/>
               <Wind speed="5" direction="135"/>
               <Precipitation precipitationType="lightRain" precipitationIntensity="0.3"/>
             </Weather>
             <RoadCondition>
               <Surface frictionScaleFactor="0.65" roughness="0.45"/>
             </RoadCondition>
           </Environment>
         </EnvironmentAction>
       </GlobalAction>
     </Actions>
   </Init>
   <Story name="EveningCommuteStory">
     <Act name="CommuteAct">
       <ManeuverGroup name="MainManeuverGroup" maximumExecutionCount="1">
         <Actors selectTriggeringEntities="true">
           <EntityRef entityRef="Honda Civic"/>
           <EntityRef entityRef="Van"/>
           <EntityRef entityRef="Sports Car"/>
         </Actors>
         <Maneuver name="MainManeuver">
           <Event name="EveningCommuteEvent" priority="overwrite">
             <Action name="DriveAction">
               <PrivateAction>
                 <LongitudinalAction>
                   <SpeedAction>
                     <SpeedActionDynamics dynamicsShape="step" dynamicsDimension="time" value=
   "1.5"/>
                     <SpeedActionTarget>
                       <RelativeTargetSpeed value="85" continuous="false"/>
                     </SpeedActionTarget>
                   </SpeedAction>
                 </LongitudinalAction>
               </PrivateAction>
             </Action>
           </Event>
         </Maneuver>
       </ManeuverGroup>
       <StartTrigger>
         <ConditionGroup>
           <Condition name="startCondition" conditionEdge="rising" delay="0">
             <ByEntityCondition>
               <TriggeringEntities rule="any">
                 <EntityRef entityRef="Honda Civic"/>
               </TriggeringEntities>
               <EntityCondition>
                 <ReachPositionCondition tolerance="1.5" alongRoute="true" positionX="55"
   positionY="0" positionZ="10" roadId="1" laneId="2" s="250"/>
               </EntityCondition>
             </ByEntityCondition>
           </Condition>
         </ConditionGroup>
       </StartTrigger>
     </Act>
   </Story>
 </Storyboard>
</OpenSCENARIO>
```

Listing 10: Continued: Final iteration of the enhanced Highway cut-in incident Scenario. (Part 2)

```xml
<?xml version='1.0' encoding='UTF-8'?>
<OpenSCENARIO>
  <FileHeader revMajor="1" revMinor="2" date="2024-06-07T13:49:13" description="Dynamic Scenario Template" author="GU">
    <License name="ASAM OpenSCENARIO" resource="https://www.asam.net/standards/detail/openscenario/" spdxId="ASAM-1.2"/>
  </FileHeader>
  <ParameterDeclarations/>
  <CatalogLocations/>
  <RoadNetwork/>
  <Entities>
    <ScenarioObject name="Honda Civic">
      <Vehicle name="Honda Civic" vehicleCategory="car">
        <BoundingBox>
          <Center x="1.5" y="0" z="0.5"/>
          <Dimensions width="2.0" length="4.5" height="1.4"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.6" positionX="3.2" positionZ="0.35"/>
          <RearAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.6" positionX="1.0" positionZ="0.35"/>
        </Axles>
        <Properties>
          <Property name="type" value="car"/>
        </Properties>
        <Performance maxSpeed="40.0" maxAcceleration="3.5" maxDeceleration="8.0"/>
      </Vehicle>
    </ScenarioObject>
    <ScenarioObject name="Van">
      <Vehicle name="Van" vehicleCategory="car">
        <BoundingBox>
          <Center x="1.5" y="0" z="0.5"/>
          <Dimensions width="2.0" length="4.5" height="1.4"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.6" positionX="3.2" positionZ="0.35"/>
          <RearAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.6" positionX="1.0" positionZ="0.35"/>
        </Axles>
        <Properties>
          <Property name="type" value="van"/>
        </Properties>
        <Performance maxSpeed="40.0" maxAcceleration="3.5" maxDeceleration="8.0"/>
      </Vehicle>
    </ScenarioObject>
    <ScenarioObject name="Sports Car">
       <Vehicle name="Sports Car" vehicleCategory="car">
        <BoundingBox>
          <Center x="1.5" y="0" z="0.5"/>
          <Dimensions width="2.0" length="4.5" height="1.4"/>
        </BoundingBox>
        <Axles>
          <FrontAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.6" positionX="3.2" positionZ="0.35"/>
          <RearAxle maxSteering="30" wheelDiameter="0.7" trackWidth="1.6" positionX="1.0" positionZ="0.35"/>
        </Axles>
        <Properties>
          <Property name="type" value="car"/>
        </Properties>
        <Performance maxSpeed="40.0" maxAcceleration="3.5" maxDeceleration="8.0"/>
      </Vehicle>
    </ScenarioObject>
  </Entities>
```

Listing 11: Dynamically generated Highway cut-in incident Scenario. This scenario was not enhanced by an LLM for demonstration purposes. (Part 1)

```
  <Storyboard>
    <Init>
      <Actions>
        <GlobalAction>
          <EnvironmentAction>
            <Environment name="DefaultEnvironment">
              <TimeOfDay animation="false" dateTime="2024-06-07T13:49:13Z"/>
              <Weather>
                <Precipitation precipitationType="rain" precipitationIntensity="0.1"/>
                <Wind speed="0" direction="0"/>
              </Weather>
              <RoadCondition frictionScaleFactor="0.7" wetness="moist">
                <Properties>
                  <Property name="surfaceType" value="asphalt"/>
                </Properties>
              </RoadCondition>
            </Environment>
          </EnvironmentAction>
        </GlobalAction>
      </Actions>
    </Init>
    <Story name="MainStory">
      <Act name="MainAct">
        <ManeuverGroup name="MainManeuverGroup" maximumExecutionCount="1">
          <Actors selectTriggeringEntities="true">
            <EntityRef entityRef="VehicleObject"/>
          </Actors>
          <Maneuver name="MainManeuver">
            <Event name="MainEvent" priority="overwrite">
              <Action name="MainAction">
                <PrivateAction>
                  <LongitudinalAction>
                    <SpeedAction>
                      <SpeedActionDynamics dynamicsShape="linear" dynamicsDimension="time"
    value="2.0"/>
                      <SpeedActionTarget>
                        <AbsoluteTargetSpeed value="30"/>
                      </SpeedActionTarget>
                    </SpeedAction>
                  </LongitudinalAction>
                </PrivateAction>
              </Action>
            </Event>
          </Maneuver>
        </ManeuverGroup>
        <StartTrigger>
          <ConditionGroup>
            <Condition name="startCondition" conditionEdge="rising" delay="0">
              <ByValueCondition>
                <ParameterCondition parameterRef="VehicleSpeed" rule="greaterThan" value="10"/>
              </ByValueCondition>
            </Condition>
          </ConditionGroup>
        </StartTrigger>
      </Act>
    </Story>
    <StopTrigger/>
  </Storyboard>
</OpenSCENARIO>
```

Listing 12: Continued: Dynamically generated Highway cut-in incident Scenario (Part 2)