LLM-Based Safety Case Generation for Baidu Apollo: Are We There Yet?

Oluwafemi Odu
Lassonde School Of Engineering
York University
Toronto, Canada
olufemi2@yorku.ca

Alvine Boaye Belle
Lassonde School Of Engineering
York University
Toronto, Canada
alvine.belle@lassonde.yorku.ca

Song Wang
Lassonde School Of Engineering
York University
Toronto, Canada
wangsong@yorku.ca

***Abstract*—Justifying the correct implementation of the non-functional requirements of mission-critical systems is crucial to prevent system failure. The latter could have severe consequences such as the death of people, and financial losses. Assurance cases (e.g., safety cases, security cases) can be used to prevent system failure. They are structured sets of arguments supported by evidence and aiming at demonstrating that a system's non-functional requirements have been correctly implemented. However, although the availability of complete assurance cases is crucial to allow the research community to contribute to the system assurance field, it remains very challenging to access complete assurance cases due to several concerns such as confidentiality issues. Furthermore, assurance cases are usually very large documents. Still, their creation remains a manual, tedious, and error-prone process that heavily relies on domain expertise. Thus, exploring techniques to support their automatic instantiation becomes crucial. To fill these gaps, our experience paper first demonstrates the feasibility of an AMLAS-based design methodology on a case study aiming at manually creating a safety case for the ML-enabled trajectory prediction component of an open-source autonomous driving system i.e. Baidu Apollo. Our paper then reports our experience in using a Large Language Model (LLM) to automatically re-create the same safety case. The lessons we have drawn from this case study provide actionable insights that could benefit researchers and practitioners.**

***Index Terms*—Requirements engineering, Assurance Cases, Safety, Machine Learning, Autonomous Driving Systems**

I. Introduction

The rapid advancement of autonomous driving systems (ADSs) has captured significant attention from both industry and academia due to their potential to revolutionize transportation. ADSs leverage a combination of sensors along with Machine Learning (ML) components to autonomously navigate vehicles, minimizing the need for human intervention [1]. These systems combine a network of interconnected elements that integrate both traditional code logic and ML models to ensure vehicle operation across various environments [2], [3]. The complexity of ML-enabled ADSs and the unpredictable environments in which they operate present significant challenges such as the assurance of their safety [4], which is crucial to prevent severe outcomes (e.g., fatalities, financial losses).

Assurance cases (e.g., safety cases, security cases) are useful in demonstrating that mission-critical systems such as ADSs comply with industrial standards, thereby preventing system failures and justifying the correct implementation of ADSs non-functional requirements (e.g., safety, security) [5]–[7].

Significant research efforts have focused on enhancing the safety assurance of ML-enabled autonomous systems operating in complex environments (e.g., [8]–[10]). For example, Hawkins et al. [10] introduced a design methodology called AMLAS. It allows to manually create safety cases and to systematically integrate safety assurance throughout the development life cycle of the ML components of autonomous systems. Still, despite the existence of such design methodologies, the number of freely available safety cases is pretty scarce, which hinders the research on system assurance.

Furthermore, with the increasing deployment and adoption of ML-enabled ADSs, traditional design methodologies for manual safety case creation may struggle to keep pace with the continually evolving complexity of these systems and the unpredictable and dynamic environments in which they operate. Additionally, using these methodologies to manually create assurance cases can be time-consuming, tedious, and error-prone [11]. However, to the best of our knowledge, no study has explored using an LLM (Large Language Model)-based approach for the automatic creation of safety cases for open-source ML-enabled ADSs.

This experience paper aims to bridge the aforementioned gaps by conducting a case study on Baidu Apollo, one of the leading ML-enabled ADSs. Thus, we focus on building a safety case for the trajectory prediction component of Baidu Apollo. To achieve this, we initially used an adaptation of the AMLAS [10] design methodology to manually create a safety case for that component, generating sixteen artefacts in the process. Subsequently, we leveraged these artefacts together with additional software engineering (SE) knowledge to facilitate the automatic generation of an LLM-based safety case for the same component. Insights from our case study suggest that combining human expertise with LLM capabilities presents a promising approach for the semi-automatic creation of assurance cases. This method leverages the strengths of both manual and automated techniques and could benefit researchers and practitioners, including vehicle manufacturers, corporate safety analysts, and regulators.

Our paper aims to investigate two research questions (RQs) in the light of a case study focusing on Baidu Apollo:

RQ1: Can we rely on an adaptation of AMLAS for the manual creation of the safety case of an open-source ML-enabled ADS?

RQ2: Can an LLM outperform human experts when creating the safety case of an open-source ML-enabled ADS?

We further describe our work in the remainder of this paper.

II. Background

A. Assurance Cases

Assurance cases allow justifying that specific systems' requirements (e.g., safety, security) are met. An assurance case is defined as *"a reasoned and compelling argument, supported by a body of evidence, that a system, service, or organization will operate as intended for a defined application in a defined environment"* [12], There are several categories of assurance cases (e.g., safety cases, security cases), each focusing on a particular requirement (e.g, safety) it is designed to validate.

The use of assurance cases across various domains (e.g., automotive, avionics) is popular for demonstrating the correct implementation of safety-critical systems. This is crucial to prevent system failure, which could lead to catastrophic outcomes (e.g., death of people, financial losses). Industry standards (e.g., ISO 26262), as well as regulatory agencies (e.g., Food and Drug Administration), recommend using assurance cases to support the certification of safety-critical systems in compliance with industry standards [13], [14].

The structure of an assurance case consists of three main components [15]: (1) The top *claim* - this represents the ultimate assertion about a given system requirement. The top claim is broken down into several sub-claims as needed to satisfy it. (2) A body of *evidence* - these include domain knowledge about system artefacts that support the sub-claims and the top claim. (3) A set of structured *arguments* - They represent the spine of an assurance case [16]. They link the set of evidence to the associated sub-claims and connect all the sub-claims to the top claim of an assurance case.

Several notations allow representing assurance cases. These include textual notations (e.g., structured prose) and graphical notations [17]. The most popular graphical notation is GSN (Goal Structuring Notation) [12], [18]. GSN core concepts include [12]: *Goals*, *Strategies*, *Solutions*. In an assurance case, they respectively map to *Claims*, *Arguments*, and *Evidence*. *InContextOf* and *SupportedBy* are the main GSN relationships that link GSN elements. GSN elements can be decorated using decorators (e.g., *Undeveloped*, and *Uninstantiated*). Figure 1 shows an excerpt of a GSN-compliant safety case [19].

B. Assurance Case patterns (ACPs)

Assurance case patterns serve as abstract reusable templates for the creation of assurance cases [20]. They contain placeholders for specifying system-specific information [2], [21], [22]. ACPs promote reusability and well-formedness and ease the creation of new assurance cases [21]–[23]. They also help mitigate assurance deficits [24], [25] i.e. knowledge gaps undermining trust in the assurance case [24]. ACPs provide a framework for the automatic construction of model-based assurance cases from system design models [21], [26]–[28]. Figure 2 shows a sample ACP represented using GSN.

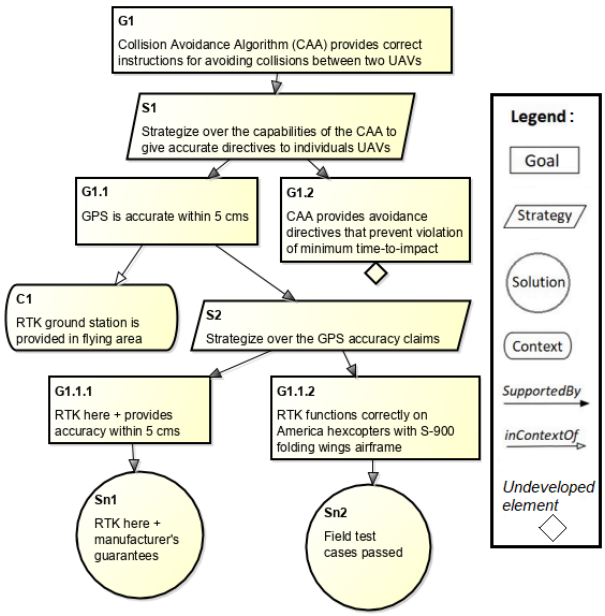

Fig. 1: A sample partial safety case adapted from [19]

C. Large Language Models (LLMs)

LLMs are advanced artificial intelligence systems that can both generate and process human language, creating the appearance of understanding these languages [29], [30]. Common examples of LLMs include the GPT series by OpenAI [31], LLaMA [32], T5 [33], and BERT [34].

LLMs support the automation of various SE tasks (e.g., code generation [35], software modeling [36], and test generation [37]). Various prompting techniques [38] have been introduced to converse and query LLMs. These include the Chain-of-Thought (CoT) technique [39], zero-shot prompting [40], one-shot prompting [41] and Few-shot prompting [42]. CoT prompting enhances the performance of LLMs in complex reasoning tasks by using a sequence of intermediate reasoning steps to demonstrate task completion. Zero-shot prompting involves querying an LLM without providing any examples for task completion, aiming to leverage the reasoning patterns it has learned. One-shot prompting provides the LLM with a single example to guide its response, assisting in illustrating the task at hand.

III. Related Work

A. Manual creation of assurance cases

Several studies focused on the manual creation of assurance cases (e.g., [8], [10], [43], [44]). They mostly target ML-enabled autonomous systems. For instance, Hawkins et al. [10] introduced a new design methodology called AMLAS. The latter aims at integrating safety assurance in the development of ML-enabled components. AMLAS comprises six iterative stages, each with argument patterns that can be instantiated

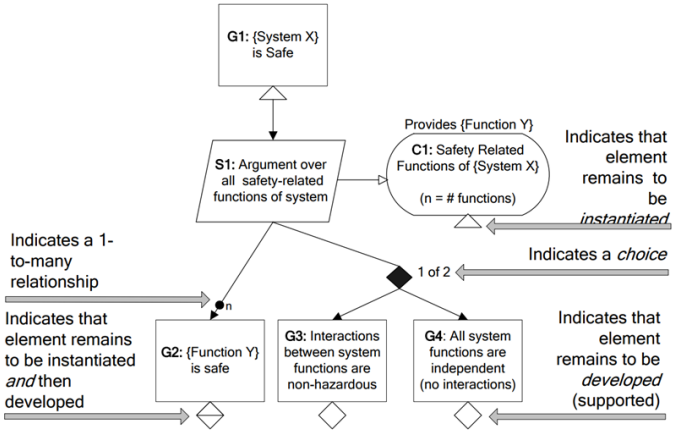

Fig. 2: A sample assurance case pattern adapted from [18]

to develop a safety case for the ML-enabled components in an autonomous system. The development of the safety case is done concurrently with the development of the system at hand. Borg et al. [44] followed AMLAS to manually create a safety case for SMIRK - a pedestrian automatic emergency braking (PAEB) system. Sivakumar et al. [8] proposed a methodology derived from AMLAS [10], aiming to simplify the creation of the safety cases of already deployed ML-enabled ADSs. They used their methodology to manually create a safety case for an ML-enabled component utilized by a Quanser Qcar [45].

B. Automatic creation of assurance cases

The manual creation and management of assurance cases are often laborious and error-prone [11], [46], especially for complex systems integrating ML components. To address this, automating assurance case creation has attracted significant interest, particularly through weaving techniques and model-based approaches [21], [26], [27]. These methods usually combine system models with ACPs to automate assurance case creation by extracting information from these models to form assurance cases. Ramakrishna et al. [46] explored a different aspect of automating assurance case creation by developing a workflow to ease the pattern selection process, viewed as a coverage problem, using ontology graphs and graph analytics. The tool called ACCELERATE [47] automates that approach. Recently, Odu et al. [20] proposed an approach that relies on LLMs to generate GSN-compliant assurance cases from ACPs. The tool called *SmartGSN* automates that approach [48].

C. Use of Assurance Cases in the Automotive Domain

Recent studies have explored the application of assurance cases to certify systems in the automotive domain. Wagner et al. [49] focused on constructing safety cases for automotive systems, such as cruise control components. Their findings indicate that the use of safety case patterns, modules, and automotive models—such as functional models, platform models, and environment models— guides the construction of safety arguments. Nearchou et al. [50] proposed incorporating assurance cases into the software development life-cycle of cyber-physical systems (CPS) to ensure adaptive safety behaviors at runtime. They demonstrated their approach with the F1TENTH racing car. Safety-critical systems like autonomous vehicles are designed to be interoperable and interconnected, leading to changes in their operating conditions at runtime [2], [26]. Hence, understanding operating conditions is vital for safety [51]. Thus, Weiss et al. [51], presented a safety case with eight heterogeneous sets of evidence to argue for the sufficient completeness and consistency of the Operational Design Domain (ODD) definition during the development of safety-critical AI-based ADS functionalities. They evaluated their approach on a driver-less train in the railway domain with a fixed route. Likewise, Burton et al. [52] demonstrated a safety assurance approach for ML-based road surface classification across various operational scenarios for chassis control functions. They illustrated this approach by creating an assurance case for a Tire Noise Recognition (TNR) system, which is essential in determining whether the road is dry.

IV. RQ1: Manual creation of the assurance case of an ML-enabled ADS

To address RQ1, we adopted Sivakumar et al. [8]'s design approach for creating a safety case for an ML-enabled ADS that is already developed and operational. We applied this approach to develop a safety case for an ML-enabled component within Baidu Apollo, an open-source ADS.

A. Steps to Manually Create an Assurance Case for an ML-Enabled ADS

Figure 3 illustrates the design methodology that Sivakumar et al. [8] proposed to create the safety cases of ML-enabled ADSs. We describe how we applied the phases of that methodology below.

1) Phase I: HARA (Hazard Analysis and Risk Assessment): Following Sivakumar et al. [8]'s methodology, we performed a HARA of the trajectory prediction component to identify its potential hazards and determine its safety requirements. HARA is a systematic approach used to identify and evaluate potential hazards associated with a system and to determine

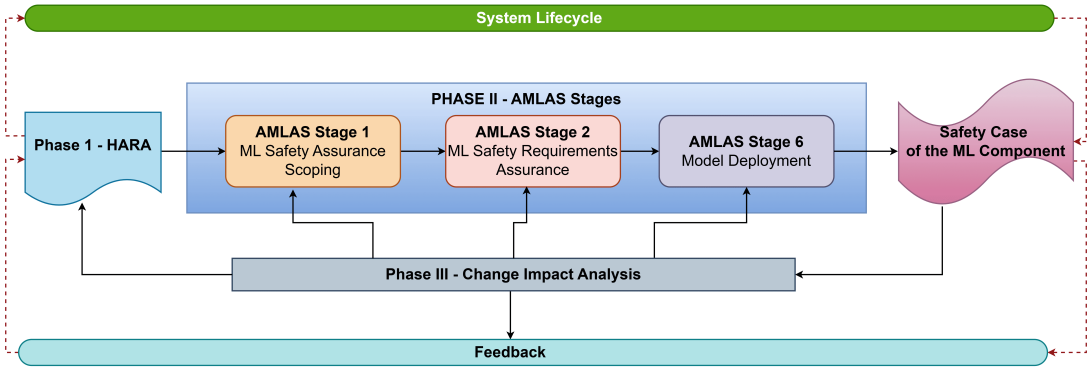

Fig. 3: RQ1 High-level Approach Overview adapted from [8]

the associated risks [8], [53]. The HARA process facilitates the creation of several artefacts used in Phase II for instantiating the selected argument patterns from AMLAS [10].

2) Phase II: Implementation of Selected AMLAS Stages: In this phase, following Sivakumar et al. [8], we implement the three stages from the AMLAS [10] that are relevant to the creation of our safety case. These three stages are stages 1,2 and 6 of AMLAS. We provide a brief description of each of these stages in the remainder of this section.

1) **Stage 1: ML Safety Assurance Scoping**
   In this stage, we aim to establish a clear framework for assuring the safety of the ML component. We define the scope of the safety assurance process for the ML component, determine the scope of the safety case, and create the top-level safety assurance claim while specifying relevant contextual information. We achieve this by instantiating the ML Safety Assurance Scoping Argument pattern shown in Figure 7 which is available online [1].
2) **Stage 2: ML Safety Requirements Assurance**
   In this stage, we focus on developing and validating the ML safety requirements. We derive these requirements from the allocated system safety requirements, ensuring they are aligned with the specific safety needs of the ML component. We validate these requirements against the allocated safety requirements, the system and software architecture, and the operational environment to ensure completeness. We achieve this process by instantiating the ML Safety Requirements Argument Pattern shown in Figure 8 which is available online [2].
3) **Stage 6: Model Deployment**
   This stage consists of providing the necessary arguments and evidence to demonstrate the successful integration of the ML component into the ADS. It aims to show that the system safety requirements continue to be satisfied post-integration. We achieve this by instantiating the ML Deployment Argument Pattern (see Figure 9 which is available online [3]). This ensures the ML model continues to meet its safety requirements within the ADS.

Each AMLAS stage consists of a list of input artefacts, output artefacts, and activities that must be performed in that stage when analyzing a specific system. We describe them in Section IV-B2, in the light of a real-world system.

3) Phase III: Change impact analysis: This phase addresses how modifications — such as changes to the ML component, other parts of the ADS, or the operating environment — affect system artefacts and lead to the update of the existing safety case. Carlan and Gallina [2] opined that *"in safety-critical domains, a change in the operating context triggers the need for impact analysis on the artefacts generated during the safety lifecycle"*. ADSs, being part of safety-critical domains, often encounter changes in their operating environment, such as transitioning between urban and rural environments. This highlights the importance of change impact analysis in ensuring the highest level of system safety and functionality. We can only focus on Phases I and II of the design methodology if we do not have access to real-time driving and simulation data needed to complete a comprehensive change impact analysis.

B. Case study: manual construction of a safety case for Baidu Apollo

The Baidu Apollo open-source ADS consists of several components, each containing smaller sub-components that utilize ML models to make critical decisions (e.g., traffic light detection, collision avoidance, and trajectory prediction). In this work, we focus on Baidu Apollo version 9.0. Baidu Apollo is developed using multiple programming languages, with C++ being the main one, accounting for nearly 76% of its codebase, including approximately 4.3k C++ files. The entire system consists of over 6k files and incorporates over 28 ML models across its various components [3], [54]. Baidu Apollo is recognized as one of the most advanced ADSs and is widely utilized in both academic and industrial settings.

[1]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure7.png

[2]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure8.png

[3]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure9.png

In this case study, we focus on manually creating a safety case for the ML-enabled trajectory prediction component of Baidu Apollo. To create that safety case, we first relied on several information sources freely available online and describing that ADS: Baidu Apollo GitHub project [54], and scientific papers (i.e. [3], [55], [56], [57]). Then we applied Sivakumar et al. [8]'s methodology. Thus, in this section, we describe all the activities and artefacts resulting from the application of that methodology to the trajectory prediction component of Baidu Apollo. The instantiated AMLAS patterns, along with the arguments, activities performed, and the evidence gathered, collectively form the safety case for that component.

Phase III of the design methodology [8] focuses on change impact analysis. In this paper, we do not complete this phase because we do not have access to the real-time driving and simulation data needed for a comprehensive change impact analysis. Hence, we generate a static safety case.

1) Phase I: Hazard Analysis and Risk Assessment (HARA): When completing the HARA process for our case study, we defined five key system functions of the trajectory prediction component as follows:

- SF1: Predicts the future position of an obstacle.
- SF2: Outputs the probability value for a predicted obstacle trajectory.
- SF3: Select the most appropriate ML model for prediction.
- SF4: Identifies the scenario of an obstacle.
- SF5: Assigns a priority to an obstacle.

To determine system malfunctions, we applied the guide word *"Wrongly"* combining it with each system function to derive potential malfunctions, which are listed below.

- MF1: Wrongly predicts the future position of an obstacle
- MF2: Wrongly outputs the probability value for a predicted obstacle trajectory
- MF3: Wrongly selects the most appropriate ML model for prediction
- MF4: Wrongly identifies the scenario of an obstacle.
- MF5: Wrongly assigns a priority to an obstacle.

Based on reference literature [3], [54], [55] on Baidu Apollo's trajectory prediction component, we defined several operational scenarios:

- OS1: A vehicle in Cruise scenario
- OS2: A vehicle in Junction scenario
- OS3: Obstacle is a Pedestrian
- OS4: Obstacle is a Vehicle
- OS5: Obstacle priority is set as Ignore
- OS6: Obstacle priority is set as Caution
- OS7: Obstacle priority is set as Normal

At the end of our HARA process, we defined the following four safety goals, which form the basis of our system safety requirements for the Baidu Apollo autonomous driving system:

- SG1: Rear-end collisions due to miscalculated obstacle halt or acceleration shall be prevented.
- SG2: Collision with obstacles at an intersection shall be prevented
- SG3: Collision due to wrong obstacle priority shall be prevented.
- SG4: Collision with obstacles wrongly classified shall be prevented.

The results of our HARA process are available on Github[4].

2) Phase II: Implementation of Selected AMLAS Stages:

1) **Stage 1: ML Safety Assurance Scoping**:
We now provide and describe each input artefacts, output artefacts, and activities we performed in this stage.
   a) Input Artefacts:
      - Artefact [A] – System safety requirements:
        We derived the system safety requirements from the safety goals established when completing the HARA process. These requirements are as follows:
        – SR1: The Ego vehicle will automatically initiate deceleration to prevent rear-end collisions upon detecting sudden stops or unexpected changes in acceleration from nearby obstacles.
        – SR2: The Ego vehicle will halt at intersections if the predicted trajectory indicates potential collision with cross-traffic or obstacles.
        – SR3: The Ego vehicle will dynamically adjust the priority of obstacles using real-time environmental data to prevent collisions that may result from incorrect obstacle priority settings.
        – SR4: The Ego vehicle will utilize appropriate ML models to accurately detect and classify obstacles, preventing collisions due to misclassification.
      - Artefact [B] - Description of Operating Environment of System
        The Baidu Apollo ADS operates within a complex environment composed of several critical elements. The road and traffic infrastructure are defined by High Definition (HD) maps, which provide details of road layouts, lane markings, traffic signs, traffic lights, and crosswalks. To achieve accurate perception of the surrounding environment, the system employs a diverse array of sensors and localization tools, including cameras, LiDAR, and radars. These are complemented by the Global Navigation Satellite System (GNSS), which ensures precise localization capabilities.
        The operating environment includes a variety of traffic situations (e.g., highway cruising and intersection navigation) while adhering to traffic laws and regulations. We make three assumptions regarding the operating environment: 1)

[4]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Hazard_Analysis_Risk_Assessment.xlsx

perception and localization data used for trajectory prediction remain accurate and current; 2) obstacles within the environment are classified as either vehicles or pedestrians, and 3) all obstacles act independently.

- Artefact [C] - System Description
  The Baidu Apollo ADS is structured around five key components: Localization, Perception, Prediction, Planning, and Control. The Localization component automatically determines the exact position of the Ego vehicle, gathering relevant data about the road and surrounding environment. It employs vehicle cameras, Global Navigation Satellite Systems (GNSS), and High Definition (HD) Maps, which provide road details within a designated area to ensure safe navigation. Once the Ego vehicle's location is identified on the HD map, the ADS leverages the Perception component, with sensors like cameras, LiDAR, and radars, to detect surrounding obstacles and dynamic objects, measuring their distance from the Ego vehicle. It employs ML models for the processing and classification of these detected objects. Based on the perception information, the ADS uses the Trajectory prediction component to predict the future trajectories of all detected obstacles. The motion planning component then uses the compiled information from the localization, perception, and prediction components to generate a smooth path according to different driving scenarios, and sends this to the control component, which then controls the Ego vehicle's movement relative to its environment to avoid collisions.
- Artefact [D] - ML Component Description
  The trajectory prediction component of the Baidu Apollo ADS predicts the behavior and future positions of all obstacles (e.g., vehicles, pedestrians) identified by the Perception component. It relies on information such as road boundaries, lane markings, traffic light locations and statuses, crosswalks, distances, velocities, and accelerations to generate predicted trajectories with probability estimates for these obstacles. The Trajectory Prediction component consists of four sub-components: Container, Scenario, Evaluator, and Predictor. These sub-components contain various ML models designed for specific tasks or unique scenarios to predict obstacle trajectory probabilities. The trajectory prediction component's configuration file specifies these ML models and automatically selects the most suitable model for a task based on the perception information. The *Container* sub-component stores all perception and localization information, including data on the surrounding environment and obstacle history. The *Scenario* sub-component analyzes all possible scenarios involving the Ego vehicle, which include cruise or junction scenarios. The *Evaluator* sub-component evaluates a path and speed for any given obstacle by outputting a probability for it using a suitable model depending on the scenario type and obstacle priority. The *Predictor* sub-component is used to generate the predicted trajectories for all obstacles.
- Artefact [F] - the ML Safety Assurance Scoping Argument Pattern in Figure 7 [5] shows the ML assurance scoping argument pattern we adapted from the design methodology [8] for our case study.

b) Activities Performed:
We utilized the input artefacts ([A], [B], [C], [D]) to determine the safety requirements that are allocated to the trajectory prediction component and instantiate the ML assurance scoping argument pattern ([F]).

c) Output Artefacts: The outputs for this stage include

- Artefact [E] - Safety Requirements Allocated to ML Component
- Artefact [G] - Instantiated ML Safety Assurance Scoping Pattern
  The instantiated ML safety assurance scoping pattern is presented in Figure 10 [6], available online.

2) **Stage 2: ML Safety Requirements Assurance**

a) Input Artefacts:

- Artefact [E] - Safety Requirements Allocated to ML Component. Following the completion of the HARA process for the trajectory prediction component, we identified four safety goals specific to that component. These four safety goals outlined below now serve as the safety requirements allocated to the ML component.
  - SG1: Rear-end collisions due to miscalculated obstacle halt or acceleration shall be prevented.
  - SG2: Collision with obstacles at an intersection shall be prevented
  - SG3: Collision due to wrong obstacle priority shall be prevented.
  - SG4: Collision with obstacles wrongly classified shall be prevented.
- Artefact [I] - ML Safety Requirements Argu-

[5] https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure7.png

[6] https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure10.png

ment Pattern. Figure 8 [7] presents the ML Safety Requirements Argument Pattern we adapted from the design methodology [8] for the system at hand.

b) Activities Performed: In this stage, we use the output from the previous stage (Stage 1), namely the safety requirement allocated to the ML component, and include it as an input in Stage 2 to develop the actual ML safety requirement. Subsequently, we validate this ML safety requirement against the system safety requirements. We use the results from this validation process, along with other artefacts, to instantiate the ML safety requirement argument pattern. The artefacts used in this stage are listed below.

c) Output Artefact:

- Artefact [H] - ML Safety Requirements
  We have developed the safety requirements for our trajectory prediction component (ML safety requirements) based on the safety requirements allocated to the ML component and its operating environment. Consequently, we have identified the following specific ML safety requirements:
  - ML-SR1: The trajectory prediction component will accurately predict the path, velocity, and acceleration of obstacles to prevent rear-end collisions.
  - ML-SR2: The trajectory prediction component will provide reliable predictions of obstacle movements at intersections.
  - ML-SR3: The trajectory prediction component will correctly assign priorities to obstacles based on perceived information from the environment.
  - ML-SR4: The trajectory prediction component will continuously re-evaluate obstacle trajectories and priorities based on new information to ensure predictions remain accurate.

  Following the approach outlined in AMLAS [8], [10] and implemented in [8], [44], we have derived additional safety requirements focusing on the performance and robustness of the ML component.

  ML Performance Requirement:

  i) The trajectory prediction component will accurately predict the future position of an obstacle, maintaining an Average Displacement Error (ADE) threshold of 0.001 meters.

  ML Robustness Requirement:

  i) The Trajectory Prediction component will accurately assign priorities to obstacles under various weather conditions.

  ii) The trajectory prediction component will ensure high accuracy for both short-term and long-term prediction horizons.

- Artefact [J] - ML Safety Requirements Validation Results. This artefact presents the outcomes of domain expert reviews conducted to verify the validity of established ML safety requirements ([H]).
- Artefact [K] - ML Safety Requirements Argument:
  This output artefact represents the instantiated ML Safety Requirements Argument Pattern [I] specific to our system. The instantiated pattern is illustrated in Figure 11 which is available online [8]. We instantiated this pattern using artefacts [E], [H], and [J].

3) **Stage 6: Model Deployment**

a) Input Artefacts: The input artefacts needed for this stage include the following:

- Artefact [A] - System Safety Requirements
- Artefact [B] - Environment Description
- Artefact [C] - System Description.

We previously described these three artefacts in stage 1. The additional input artefacts for this stage include:

- Artefact [V] - ML Model. This artefact describes the list of all ML models (e.g., the MLP, RNN, and LSTM models) used within the trajectory prediction component of the Baidu Apollo ADS. A complete list of these models, and their specific use cases for various scenarios, obstacle types, and obstacle priority types, can be found in [3], [54].
- Artefact [EE] - Operational scenarios
  The operational scenario describes the set of real scenarios that may be encountered during the operation of the ADS [10]. In our case study, we have narrowed down the operational scenarios to:
  - OS1: A vehicle in Cruise scenario - when a vehicle is moving along a lane
  - OS2: A vehicle in Junction scenario - when a vehicle approaches a road junction
  - OS3: Obstacle is a Pedestrian
  - OS4: Obstacle is a Vehicle
  - OS5: Obstacle priority is set as Ignore (i.e., Obstacles that do not affect the Ego vehicle's trajectory and can be disregarded)
  - OS6: Obstacle priority is set as Caution (i.e., Obstacles with a high possibility of interacting with the Ego vehicle)

[7] https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure8.png

[8] https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure11.png

– OS7: Obstacle priority is set as Normal (i.e. when an obstacle does not fit into the "ignore" or "caution" categories).

- Artefact [GG] - ML Deployment Argument Pattern. Figure 9 which is available online[9] presents the ML Deployment Argument Pattern adapted from the design methodology [8] for our system.

b) Activities Performed: In this stage, we instantiate the ML Deployment Argument Pattern [GG] to generate the instantiated ML Deployment Argument Pattern [HH].

c) Output Artefacts:

- Artefact [DD] - Erroneous Behaviour Log. In our case, we assume that no erroneous behavior is detected in the operation of the trajectory prediction component or its integration with the ADS. Therefore, we do not reference this artefact in the instantiated ML Deployment Argument Pattern.
- Artefact [FF] - Integration Testing Results. This includes integration test case results indicating satisfactory system safety in the defined operational scenarios after integration with the ML component.
- Artefact [HH] - ML Deployment Argument. This represents the instantiated ML Safety Requirements Argument Pattern [GG] specific to our system, demonstrating that the safety requirements assigned to the trajectory prediction component are still met when the component is deployed within the ADS. The instantiated pattern is illustrated in Figure 12 which is available online [10].

Finally, the three instantiated argument patterns derived from the AMLAS stages, along with all the described and referenced artefacts, collectively form the first draft of our safety case for the trajectory prediction component of the Baidu Apollo autonomous driving system.

3) Manual evaluation and refinement of the safety case: To review the first draft of the safety case, two co-authors (raters) with at least six years of experience in system assurance and/or SE followed the assurance case review guidelines that the literature (e.g., [12], [58]–[60]) recommends. This allowed them to review that safety case by assessing nine review criteria. These criteria are: 1) Argument comprehension; 2) Well-formedness; 3) Expressive sufficiency; 4) Presence of Defeaters and Counter-evidence; 5) Clarity; 6) Atomicity; 7) Ambiguity; 8) Vagueness; and 9) Appropriate use of elements.

During the review process, the two raters independently rated each of these criteria on a linear scale going from 1 to 5, with 1 = Very High, 2 = High, 3 = Average, 4 = Low, and, 5 = Very Low. For each of these criteria, when the associated rating was suboptimal (i.e. higher than 1), the rater also commented on how to refine the safety case by improving that criteria. The table reporting the nine assessment criteria, the short explanation of these criteria, and the independent ratings made by each of the two co-authors are available online[11].

To assess the consistency of the ratings made by the two co-authors, we relied on Kendall's Tau [61]. The latter is a correlation coefficient that varies between - 1 (lack of agreement between raters) and 1 (strong agreement between raters). We used an online tool [12] to automatically compute the value of Kendall's Tau with a 95% confidence interval, using the ratings from the table reporting the assessment criteria the two raters used for the review. Hence, the value of the Kendall's Tau is **0.44**. This indicates a moderate agreement between the two raters. Besides, the averages of their ratings are respectively 1.33 and 1.22. This results in a combined overall average of **1.28**, which is close to 1 i.e. Very High. Thus, the rating results indicate that the manually created safety case meets most of the assessment criteria. To account for their moderate agreement, the two raters then discussed their ratings and reached a consensus. They then used their so-finalized comments to refine the safety case of the trajectory prediction component of Baidu Apollo. Figure 14 which is available online [13] depicts this refined safety case. It notably consists of a total of 38 GSN elements, including 15 goals, 6 strategies, and 5 solutions.

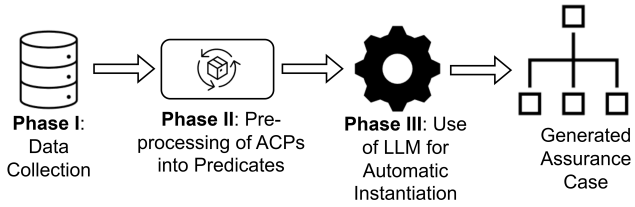

Fig. 4: RQ2 High-level Approach Overview adapted from [20]

V. RQ2: Automatic creation of assurance cases using an LLM

To facilitate the automatic creation of assurance cases using LLMs, we rely on the approach that Odu et al. [20] proposed. We provide a brief description of this approach in Section V-A.

A. Description of the LLM-powered approach

We follow the LLM-based approach proposed in [20] to instantiate assurance cases from ACPs (i.e. AMLAS argument patterns). Figure 4 depicts this approach. We describe it below.

[9]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure9.png

[10]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure12.png

[11]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Safety_Case_Assessment.zip

[12]https://www.gigacalculator.com/calculators/correlation-coefficient-calculator.php

[13]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure14.png

1) Phase I: Data Collection: To collect relevant data for the automatic instantiation of assurance cases from patterns, we can focus on safety-critical systems that include specialized components, such as those employing ML technologies. These systems present unique assurance requirements due to their complexity making them ideal candidates for analysis. We can then identify from peer-reviewed studies (e.g,. [62]), a set of well-established and validated ACPs which have been used to structure the assurance cases of the target systems. Thus, our dataset may comprise these ACPs and the assurance cases developed from them. We can divide the dataset into two parts: one for use as a one-shot example and the other (i.e. the ground truth) for evaluating our approach's performance. In our study, we identified and extracted the ACP and the resulting assurance case from the pattern referenced in [62], which we used as our one shot example.

2) Phase II: Pre-processing of ACPs into Predicates: In this phase, we utilized the predicate-based rules Odu et al. introduced in [20] for the formalization of assurance cases and ACPs. Thus, we convert each ACP from our dataset into this predicate-based format. The latter is an advanced structured prose format that complies with GSN. This conversion enables our LLM to use it as input for generating an assurance case that complies with this pattern.

3) Phase III: Using LLM to Automatically Generate Assurance Cases: In this phase, we utilize an LLM to automatically generate an assurance case for the analyzed component of Baidu Apollo, using the formalized ACPs provided as input to that model. These ACPs are the AMLAS argument patterns.

B. Case study: automatic generation of a safety case for Baidu Apollo

1) Dataset: It consists of the following:

1) **ACP and Derived Assurance Case for the *DeepMind* System**: In [20], the authors relied on peer-reviewed studies to identify ACPs and instantiated assurance cases for many systems across various domains. Among these, *DeepMind* [62] is the only ML-enabled system. Thus, we selected the ACP and the derived assurance case focusing on the interpretability of the ML component in the *DeepMind* system as part of our dataset. Our one-shot experiment uses the two of them as a single example.
2) **Safety Argument Pattern and Derived Safety Case for Baidu Apollo ADS**: Our dataset also comprises the safety case we manually developed in Section IV-B and the ACP resulting from the combination of the three argument patterns we used in Section IV-B to create that safety case. To validate the experiment results associated with RQ2, we use that safety case as the ground truth.

2) LLM Description and settings: In our experiments, we chose GPT-4o as our LLM due to its robust capabilities and the advanced features it offers as a recent addition to the OpenAI GPT series. To address the non-deterministic behavior of our LLM, we conducted each experiment five times (K=5) following the literature (e.g., [36], [63], [64]). We interacted with our LLM using the OpenAI API [65], while maintaining default settings for the different parameters used in our interaction with GPT-4o. Hence, we set its *temperature* to **1** and its *maximum number of tokens* to **4096**.

3) Description of the experiments: To investigate RQ2, we conducted two different experiments to evaluate the effectiveness of GPT-4o in generating an assurance case for Baidu Apollo. Our experiments utilized the CoT prompting technique [39] combined with SE knowledge. As in existing work [20], that knowledge includes: 1) Predicate-based rules, 2) Domain Information, 3) Context Information, and 4) One-shot Example. Domain information is the knowledge and facts specific to the application domain or system for which the assurance case is being automatically generated [20]. Context information is the background details about the structure and representation of various elements and decorators used in an assurance case and ACP as represented in GSN [12].

- **Experiment 1: One-shot, with context information, with domain information, and with predicate rules** - This experiment evaluates our LLM ability to instantiate an assurance case using SE knowledge i.e. a one-shot example, domain information about the system, contextual information, and predicate-based rules.
- **Experiment 2: Zero-shot, without context information, without domain information, and without predicate rules** - It evaluates our LLM ability to instantiate an assurance case without SE knowledge in its prompts.

4) Description of the LLM prompts: In our experiments, we interact with our LLM by specifying two types of input prompts: the system prompt and the user prompt. The system prompt consists of instructions and guidelines provided to our LLM to ensure it understands the task and responds appropriately. This prompt contains our domain information, context information, and a one-shot example, all structured using CoT. The user prompt represents our direct input to the model, requesting it to complete the generation of an assurance case complying with the given pattern in the user prompt. The LLM generates the safety case in a textual format, and more specifically, a structured prose format complying with GSN.

Figure 15 which is available online [14] illustrates the generic structure of the system prompt used in Experiment 1, while Figure 16 [15] presents an excerpt from the user prompt for the same experiment. The user prompt includes an excerpt of the formalized safety case pattern. Figure 13 [16] illustrates that pattern using GSN. The system prompt in Experiment 1 incorporates all four categories of our SE knowledge. Specifically, the area outlined by blue dotted lines denotes the context information, the area with orange dotted lines represents our

[14]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure15.png

[15]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure16.png

[16]https://github.com/Oluwafemi17/LLM-Based-Safety-Case-Generation-for-Baidu-Apollo-Are-We-There-Yet-/blob/f61775743d03e8368e3c086471c80cfe6a51dc67/Figure13.png

TABLE I: Experiment Results

Exp	Exact Match		BLEU Score		Semantic Similarity	
	Mean	Median	Mean	Median	Mean	Median
Exp 1	0.386	0.37	0.47	0.48	0.82	0.81
Exp 2	0.014	0.01	0.01	0.01	0.212	0.22

predicate rules, the area with purple dotted lines showcases our one-shot example, and the area with the red dotted lines contains our domain information. That pattern is also provided as input to the LLM for its automatic instantiation of a safety case. In Experiment 2, we used the system and user prompts that Figures 5 and 6, respectively depict.

You are an assistant who assists in developing an assurance case in a tree structure using Goal Structuring Notation (GSN). Your role is to create an assurance case.

Fig. 5: A Sample System Prompt for Experiment 2

Create a safety case for the trajectory prediction component of the Baidu Apollo autonomous driving system and display this safety case in a hierarchical tree format using dashes (-) to denote different levels.

Fig. 6: A Sample User Prompt for Experiment 2

5) Description of the assessment measures: To assess the quality of our experiment results, we focus on well-established metrics designed to measure the similarity between machine-generated text (i.e. LLM-generated safety case) and reference human text (i.e. manually generated safety case). These metrics are: **Exact Match**, **BLEU Score**, **Semantic Similarity**. To evaluate the latter, we use the cosine similarity metric.

6) Result analysis: Table I presents the Exact Match, BLEU score, and Semantic Similarity results from our Experiments. The mean and median values across the five runs for both experiments are remarkably close, indicating an absence of extreme outliers and consistent results across different runs.

As Table I shows, Experiment 1 consistently outperformed Experiment 2 across all three similarity metrics. Hence Experiment 1 yields an LLM-generated safety case that is significantly closer to the ground-truth (i.e. manually generated safety case for the trajectory prediction component). One probable reason for this outcome could be the integration of comprehensive SE knowledge, particularly context information, predicate-based rules, one-shot example, and domain information containing system artefacts generated during our manual safety case creation in section IV-B2.

VI. Discussion and implications for future work

A. Reflections on the manual creation of assurance cases

Manually creating assurance cases from scratch can be tedious and time-consuming. Despite the use of an AMLAS-based approach to support that task, we found that gathering the necessary artefacts and performing the required activities required significant domain expertise across multiple disciplines. This expertise encompasses understanding domain-specific jargon, international standards, safety and compliance processes within that domain, and a deep knowledge of the Baidu Apollo ADS, its architecture, and its operating environment. This timeline highlights the complex and time-intensive nature of developing a robust safety case for the trajectory prediction component of Baidu Apollo.

In addition, we relied on several guidelines from the literature to review and refine the manually created safety case. This review process was very useful and informed the improvement of that safety case. Thus, we think it would greatly enhance AMLAS to incorporate an explicit phase for reviewing generated assurance cases. This phase could focus on criteria relevant to ML-enabled components in Autonomous Systems, thereby further improving that methodology's robustness.

B. Reflections on the use of an LLM-based approach to automatically create assurance cases

Our case study suggests that the use of an LLM can provide significant speed and efficiency as it can quickly process large volumes of data and generate content, allowing for rapid generation of safety cases compared to traditional manual methods. This is advantageous for complex systems like Baidu Apollo's ADS, where the manual creation of an assurance case can be a lengthy process, often taking several months.

Our case study also shows that an LLM may produce inaccurate or inconsistent outputs if not provided with sufficient or relevant information. The generation of a reliable assurance case requires detailed domain-specific knowledge that LLMs may not inherently possess. This point is highlighted in our experiments: Experiment 1, which included key SE inputs outperformed Experiment 2, which lacked them.

To enhance the effectiveness of LLMs in generating high-quality assurance cases, we can consider several approaches. These include incorporating additional categories of relevant knowledge such as results of HARA, details on international standards and compliance with these standards, ethics, and operational context relevant to the system being assessed.

While it may appear advantageous to directly "*teach*" AMLAS-based methodologies to an LLM, doing so would necessitate extensive training datasets specifically tailored to these methodologies. Unfortunately, such datasets are often unavailable due to proprietary and sensitive data concerns, posing a significant challenge to this automation approach.

C. Semi-automatic creation of assurance cases: Human ft. Machine

Our case study (see experiment 1 results) suggests that combining human expertise with ML capabilities may offer

a promising approach for the semi-automatic creation of assurance cases, leveraging the strengths of both manual and automated approaches. The manual approach which starts with a detailed HARA, followed by identifying necessary artefacts, activities, and information, ensures thorough extraction of domain information. This can be effectively integrated with LLMs, which provide speed and efficiency, enabling swift generation of reliable assurance cases.

VII. Threats to Validity

In our work, we focused on a single case study: Baidu Apollo. This hinders the generalization of our results to the automotive domain. Still, Baidu Apollo is a globally recognized commercial ADS and a popular open-source platform. Furthermore, its features are common to most ML-enabled ADSs and it is widely utilized in both academia and industry.

To gauge the ability of an LLM to generate a correct safety case, we relied on a safety case that we created ourselves. Even though our safety case has been created by an experienced team of researchers, it has not yet been reviewed by the experts who developed Baidu Apollo. This may hinder the verification of the completeness and usefulness of that safety case. Still, we have used the adaptation of a well-established design methodology to manually create that safety case and followed a very stringent process to review and refine that safety case. Still, in the future, we plan to work closely with Baidu Apollo experts to further review our safety case.

VIII. Conclusion and Future Work

In this experience paper, we have explored two approaches — manual and automated approaches for creating a safety case for the trajectory prediction component of an autonomous driving system. Our study not only demonstrates the potential of each approach independently but also reveals the specific strengths and limitations inherent to each approach.

In future work, we aim to expand our work by focusing on additional case studies and engaging in close collaboration with industry partners to further validate our work. Also, we plan to conduct a user study to manually assess the LLM-generated safety cases using several assessment criteria including the ones outlined in section IV-B3. This will help to further evaluate the equivalence of LLM-generated assurance cases to those developed by assurance case experts.

References

[1] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. in 2018 33rd ieee," in *ACM International Conference on Automated Software Engineering (ASE)*, pp. 132–142.

[2] C. Carlan and B. Gallina, "Enhancing state-of-the-art safety case patterns to support change impact analysis," in *30th European Safety and Reliability Conference*, 2020.

[3] Z. Peng, J. Yang, T.-H. Chen, and L. Ma, "A first look at the integration of machine learning models in complex autonomous driving systems: a case study on apollo," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1240–1250, 2020.

[4] S. Burton, L. Gauerhof, B. B. Sethy, I. Habli, and R. Hawkins, "Confidence arguments for evidence of performance in machine learning for highly automated driving functions," in *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings 38*, pp. 365–377, Springer, 2019.

[5] P. Koopman, "Autonomous vehicles and software safety engineering," *ICSE keynote*, 2022.

[6] E. Wozniak, C. Cârlan, E. Acar-Celik, and H. J. Putzer, "A safety case pattern for systems with machine learning components," in *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops: DECSoS 2020, DepDevOps 2020, USDAI 2020, and WAISE 2020, Lisbon, Portugal, September 15, 2020, Proceedings 39*, pp. 370–382, Springer, 2020.

[7] R. Salay, K. Czarnecki, H. Kuwajima, H. Yasuoka, T. Nakae, V. Abdelzad, C. Huang, M. Kahn, and V. D. Nguyen, "The missing link: Developing a safety case for perception components in automated driving," *arXiv preprint arXiv:2108.13294*, 2021.

[8] M. Sivakumar, A. B. Belle, J. Shan, O. Odu, and M. Yuan, "Design of the safety case of the reinforcement learning-enabled component of a quanser autonomous vehicle," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, pp. 57–67, IEEE, 2024.

[9] R. Hawkins, M. Osborne, M. Parsons, M. Nicholson, J. McDermid, and I. Habli, "Guidance on the safety assurance of autonomous systems in complex environments (sace)," *arXiv preprint arXiv:2208.00853*, 2022.

[10] R. Hawkins, C. Paterson, C. Picardi, Y. Jia, R. Calinescu, and I. Habli, "Guidance on the assurance of machine learning in autonomous systems (amlas)," *arXiv preprint arXiv:2102.01564*, 2021.

[11] C. Menghi, T. Viger, A. Di Sandro, C. Rees, J. Joyce, and M. Chechik, "Assurance case development as data: A manifesto," in *2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 135–139, IEEE, 2023.

[12] G. S. N. S. W. Group, "Gsn (version 3)," 2023. Accessed on November 30, 2023.

[13] A. Finnegan and F. McCaffery, "A security argument pattern for medical device assurance cases," in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pp. 220–225, IEEE, 2014.

[14] E. Denney and G. Pai, "Tool support for assurance case development," *Automated Software Engineering*, vol. 25, no. 3, pp. 435–499, 2018.

[15] P. J. Graydon, J. C. Knight, and E. A. Strunk, "Assurance based development of critical systems," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pp. 347–357, IEEE, 2007.

[16] I. Sljivo, B. Gallina, J. Carlson, H. Hansson, and S. Puri, "Tool-supported safety-relevant component reuse: From specification to argumentation," in *Reliable Software Technologies–Ada-Europe 2018: 23rd Ada-Europe International Conference on Reliable Software Technologies, Lisbon, Portugal, June 18-22, 2018, Proceedings 23*, pp. 19–33, Springer, 2018.

[17] C. M. Holloway, "Safety case notations: Alternatives for the non-graphically inclined?," in *2008 3rd IET International Conference on System Safety*, pp. 1–6, IET, 2008.

[18] R. Alexander, T. Kelly, Z. Kurd, and J. McDermid, "Safety cases for advanced control software: Safety case patterns," *Final Report, NASA Contract FA8655-07-1-3025, Univ. of York (October 2007)*, 2007.

[19] M. Vierhauser, S. Bayley, J. Wyngaard, W. Xiong, J. Cheng, J. Huseman, R. Lutz, and J. Cleland-Huang, "Interlocking safety cases for unmanned autonomous systems in shared airspaces," *IEEE transactions on software engineering*, vol. 47, no. 5, pp. 899–918, 2019.

[20] O. Odu, A. B. Belle, S. Wang, S. Kpodjedo, T. C. Lethbridge, and H. Hemmati, "Automatic instantiation of assurance cases from patterns using large language models," *Journal of Systems and Software*, p. 112353, 2025.

[21] C. Hartsell, N. Mahadevan, A. Dubey, and G. Karsai, "Automated method for assurance case construction from system design models," in *2021 5th International Conference on System Reliability and Safety (ICSRS)*, pp. 230–239, IEEE, 2021.

[22] O. Odu, A. B. Belle, S. Wang, and K. K. Shahandashti, "A prisma-driven bibliometric analysis of the scientific literature on assurance case patterns," *arXiv preprint arXiv:2407.04961*, 2024.

[23] T. A. Beyene and C. Carlan, "Cybergsn: a semi-formal language for specifying safety cases," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 63–66, IEEE, 2021.

[24] R. Hawkins, T. Kelly, J. Knight, and P. Graydon, "A new approach to creating clear safety arguments," in *Advances in Systems Safety: Proceedings of the Nineteenth Safety-Critical Systems Symposium, Southampton, UK, 8-10th February 2011*, pp. 3–23, Springer, 2011.

[25] K. K. Shahandashti, A. B. Belle, T. C. Lethbridge, O. Odu, and M. Sivakumar, "A prisma-driven systematic mapping study on system assurance weakeners," *Information and Software Technology*, p. 107526, 2024.

[26] R. Wei, T. P. Kelly, X. Dai, S. Zhao, and R. Hawkins, "Model based system assurance using the structured assurance case metamodel," *Journal of Systems and Software*, vol. 154, pp. 211–233, 2019.

[27] R. Hawkins, I. Habli, D. Kolovos, R. Paige, and T. Kelly, "Weaving an assurance case from design: a model-based approach," in *2015 IEEE 16th International Symposium on High Assurance Systems Engineering*, pp. 110–117, IEEE, 2015.

[28] M. Zeroual, B. Hamid, M. Adedjouma, and J. Jaskolka, "Formal model-based argument patterns for security cases," in *Proceedings of the 28th European Conference on Pattern Languages of Programs*, pp. 1–12, 2023.

[29] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, and H. Wang, "Large language models for software engineering: a systematic literature review (2023)," *arXiv preprint arXiv:2308.10620*, 2023.

[30] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Barnes, and A. Mian, "A comprehensive overview of large language models," *arXiv preprint arXiv:2307.06435*, 2023.

[31] T. Wu, S. He, J. Liu, S. Sun, K. Liu, Q.-L. Han, and Y. Tang, "A brief overview of chatgpt: The history, status quo and potential future development," *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 5, pp. 1122–1136, 2023.

[32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[33] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[34] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[35] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang, "Unified pre-training for program understanding and generation," *arXiv preprint arXiv:2103.06333*, 2021.

[36] K. Chen, Y. Yang, B. Chen, J. A. H. López, G. Mussbacher, and D. Varró, "Automated domain modeling with large language models: A comparative study," in *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pp. 162–172, IEEE, 2023.

[37] J. Shin, S. Hashtroudi, H. Hemmati, and S. Wang, "Domain adaptation for code model-based unit test case generation," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1211–1222, 2024.

[38] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

[39] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24824–24837, 2022.

[40] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *International conference on machine learning*, pp. 2152–2161, PMLR, 2015.

[41] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[42] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[43] E. A. Nguyen and A. G. Ellis, "Experiences with assurance cases for spacecraft safing," in *2011 IEEE 22nd International Symposium on Software Reliability Engineering*, pp. 50–59, IEEE, 2011.

[44] M. Borg, J. Henriksson, K. Socha, O. Lennartsson, E. Sonnsjö Lönegren, T. Bui, P. Tomaszewski, S. R. Sathyamoorthy, S. Brink, and M. Helali Moghadam, "Ergo, smirk is safe: a safety case for a machine learning component in a pedestrian automatic emergency brake system," *Software quality journal*, vol. 31, no. 2, pp. 335–403, 2023.

[45] Quanser, "Qcar." https://www.quanser.com/products/qcar/, 2024.

[46] S. Ramakrishna, H. Jin, A. Dubey, and A. Ramamurthy, "Automating pattern selection for assurance case development for cyber-physical systems," in *International Conference on Computer Safety, Reliability, and Security*, pp. 82–96, Springer, 2022.

[47] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[48] O. Odu, D. M. Beltrán, E. B. Gutiérrez, A. B. Belle, and M. Sherafat, "Smartgsn: a generative ai-powered online tool for the management of assurance cases," *arXiv preprint arXiv:2410.16675*, 2024.

[49] S. Wagner, B. Schätz, S. Puchner, and P. Kock, "A case study on safety cases in the automotive domain: Modules, patterns, and models," in *2010 IEEE 21st International Symposium on Software Reliability Engineering*, pp. 269–278, IEEE, 2010.

[50] I. Nearchou, L. Rafalko, R. Phillips, M. Anderson, W. Shen, and S. Drager, "An assurance case driven development paradigm for autonomous vehicles: An f1tenth racing car case study," in *2023 IEEE/ACIS 21st International Conference on Software Engineering Research, Management and Applications (SERA)*, pp. 156–161, IEEE, 2023.

[51] G. Weiss, M. Zeller, H. Schoenhaar, C. Drabek, and A. Kreutz, "Approach for argumenting safety on basis of an operational design domain," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pp. 184–193, 2024.

[52] S. Burton, I. Kurzidem, A. Schwaiger, P. Schleiss, M. Unterreiner, T. Graeber, and P. Becker, "Safety assurance of machine learning for chassis control functions," in *Computer Safety, Reliability, and Security: 40th International Conference, SAFECOMP 2021, York, UK, September 8–10, 2021, Proceedings 40*, pp. 149–162, Springer, 2021.

[53] F. Yan, S. D. Foster, I. Habli, and R. Wei, "Model-based generation of hazard-driven arguments and formal verification evidence for assurance cases," in *10th International Conference on Model-Driven Engineering and Software Development*, pp. 252–263, SciTePress, 2022.

[54] Baidu, "Apollo, https://github.com/apolloauto/apollo."

[55] K. Xu, X. Xiao, J. Miao, and Q. Luo, "Data driven prediction architecture for autonomous driving and its application on apollo platform," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 175–181, IEEE, 2020.

[56] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[57] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.

[58] T. Kelly, "Reviewing assurance arguments-a step-by-step approach," in *Workshop on assurance cases for security-the metrics challenge, dependable systems and networks (DSN)*, 2007.

[59] S. Yamamoto and S. Morisaki, "A system theoretic assurance case review," in *2016 11th International Conference on Computer Science & Education (ICCSE)*, pp. 992–996, IEEE, 2016.

[60] F. U. Muram and M. A. Javed, "Attest: Automating the review and update of assurance case arguments," *Journal of systems architecture*, vol. 134, p. 102781, 2023.

[61] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1-2, pp. 81–93, 1938.

[62] F. R. Ward and I. Habli, "An assurance case pattern for the interpretability of machine learning in safety-critical systems," in *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops: DECSoS 2020, DepDevOps 2020, USDAI 2020, and WAISE 2020, Lisbon, Portugal, September 15, 2020, Proceedings 39*, pp. 395–407, Springer, 2020.

[63] M. Sivakumar, A. B. Belle, J. Shan, and K. K. Shahandashti, "Prompting gpt–4 to support automatic safety case generation," *Expert Systems with Applications*, vol. 255, p. 124653, 2024.

[64] M. Sivakumar, A. B. Belle, J. Shan, and K. K. Shahandashti, "Exploring the capabilities of large language models for the generation of safety cases: the case of gpt-4," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, pp. 35–45, IEEE, 2024.

[65] OpenAI, "Openai api." https://openai.com/api/, 2023.