**Multi-Agent System Approach for Automation Behavior Driven Development in Scenario-Based verification of Advanced Driver Assistance System (ADAS)/Autonomous Driving (AD) using Generative AI**

### **Abstract**

The verification of Advanced Driver Assistance Systems (ADAS) and autonomous driving functions relies heavily on **scenario-based verification (SBV)**, where diverse real-world traffic scenarios are simulated to ensure system safety and reliability. **Behavior-Driven Development (BDD)** offers a structured, natural language approach to defining test scenarios, but its manual implementation remains time-consuming and error-prone. In this paper, we propose a **Large Language Model (LLM)-based Multi-Agent System (MAS) approach** to automate BDD in scenario-based verification of ADAS. Our framework leverages LLMs to generate, translate, and refine BDD scenarios into executable test cases while utilizing MAS to coordinate scenario execution and evaluation in an ADAS simulation environment. The system enhances efficiency by **automating test scenario generation, execution, and validation**, reducing human effort and improving adaptability to complex verification tasks. We evaluate our approach using real-world ADAS scenarios in simulation, demonstrating significant improvements in automation, scalability, and verification accuracy compared to traditional methods. This research highlights the potential of **LLM-driven multi-agent frameworks** in streamlining scenario-based testing for autonomous systems, paving the way for more robust and intelligent verification pipelines in the automotive industry.

##### 

**Introduction**

*Advanced Driver Assistance Systems (ADAS) and autonomous vehicles rely heavily on rigorous testing to ensure safety and reliability. Scenario-based verification has emerged as a pivotal approach in this domain.*

*Scenario-based verification involves testing ADAS and autonomous driving systems against a multitude of predefined driving scenarios to evaluate their performance and safety. However, this approach faces several significant challenges: The vast number of possible driving scenarios, due to the complexity and unpredictability of real-world environments, makes comprehensive testing impractical. This scenario space explosion necessitates efficient methods to identify and prioritize critical scenarios.* *Developing a standardized and formal language to describe scenarios is essential for consistency in testing. The lack of such standardization can lead to ambiguities and inconsistencies in scenario definitions.Autonomous vehicles consist of diverse hardware and software components developed by different manufacturers. This heterogeneity poses challenges in the certification process, especially when components operate as black boxes without disclosed internal behaviors.*

*Behavior-Driven Development (BDD) offers a structured methodology to address some of these challenges: BDD facilitates the creation of test scenarios in natural language, making them more accessible and understandable to stakeholders. This approach enhances collaboration between developers, testers, and domain experts.* *By using a common language, BDD bridges the gap between technical and non-technical team members, ensuring that everyone has a clear understanding of the system's expected behaviors.* *BDD promotes the identification of potential issues early in the development process, reducing costly revisions later.*

*Integrating LLMs and MAS can significantly enhance the automation of BDD in ADAS verification. LLMs, trained on vast datasets, can automatically generate diverse and complex test scenarios in natural language, covering a wide range of driving situations.* *LLMs can translate natural language scenarios into executable test scripts, streamlining the testing process.* *MAS can manage the execution of these test scenarios, simulating interactions between multiple agents (e.g., vehicles, pedestrians) to create realistic testing environments.*

*This paper introduces a novel framework that integrates LLMs and MAS to automate BDD-based verification of ADAS and autonomous driving systems. Leveraging LLMs to automatically create comprehensive and diverse test scenarios in natural language.* *Developing methods to convert these natural language scenarios into executable test cases. Utilizing MAS to simulate complex interactions within test scenarios, providing a more realistic and thorough testing environment.* *Assessing the effectiveness of the proposed framework through extensive experiments, demonstrating improvements in testing efficiency and coverage.  
By addressing the challenges in scenario-based verification, this framework aims to enhance the safety and reliability of ADAS and autonomous driving systems.*

**Background & Related Work**

*In this section, we explore the foundational concepts and related research pertinent to our study, focusing on Scenario-Based Verification (SBV) in ADAS testing, the application of Behavior-Driven Development (BDD) for natural language scenario definition, the role of Multi-Agent Systems (MAS) in automation, and the integration of Large Language Models (LLMs) in software engineering.*

**Scenario-Based Verification in ADAS Testing**

*Scenario-Based Verification (SBV) is a critical methodology in the development and validation of Advanced Driver-Assistance Systems (ADAS) and autonomous vehicles. SBV involves the systematic creation and execution of driving scenarios to evaluate the performance and safety of these systems. This approach addresses the need for comprehensive testing by simulating a wide range of real-world situations, thereby identifying potential system failures before deployment.*

*The complexity of real-world driving environments leads to a vast number of possible scenarios, making exhaustive testing impractical. To manage this, standardized scenario description languages, such as ASAM OpenSCENARIO® DSL, have been developed. These languages provide a human-readable format for defining test scenarios, facilitating consistency and reusability in testing processes. ASAM OpenSCENARIO® DSL supports various levels of scenario abstraction, enabling testers to specify scenarios ranging from high-level intents to detailed concrete instances.*

**Behavior-Driven Development (BDD) for Natural Language Scenario Definition**

*Behavior-Driven Development (BDD) is a software development methodology that emphasizes collaboration among stakeholders by using natural language to define system behaviors. In the context of ADAS testing, BDD allows for the specification of driving scenarios in a format that is easily understandable by both technical and non-technical stakeholders. This is often achieved using the Gherkin language, which structures scenarios in a Given-When-Then format, promoting clarity and shared understanding.*

*By employing BDD, teams can create executable specifications that serve as both documentation and automated tests. This alignment ensures that the system's behavior meets the specified requirements and facilitates early detection of discrepancies during development.*

**Multi-Agent Systems (MAS) in Automation**

*Multi-Agent Systems (MAS) consist of multiple interacting agents that work collaboratively to achieve complex tasks. In the realm of ADAS testing, MAS can simulate the interactions between various entities, such as vehicles, pedestrians, and traffic signals, within a driving scenario. This capability is crucial for creating realistic and dynamic test environments that closely mimic real-world conditions.*

*The use of MAS in automation extends beyond simulation. Agents can be designed to autonomously generate test scenarios, execute tests, and analyze outcomes, thereby enhancing the efficiency and effectiveness of the testing process.*

**Large Language Models (LLMs) in Software Engineering**

*Large Language Models (LLMs), such as GPT-3 and its successors, have demonstrated remarkable capabilities in understanding and generating human-like text. In software engineering, LLMs have been applied to various tasks, including code generation, documentation, and automated testing.*

*Recent advancements have enabled LLMs to perform complex reasoning and tool use, enhancing their utility in software automation. For instance, LLMs can interpret natural language requirements and generate corresponding code or test cases, thereby streamlining the development process.*

*However, challenges remain in ensuring the accuracy and reliability of LLM-generated outputs. Ongoing research focuses on improving LLMs' ability to understand context, handle ambiguous inputs, and integrate with other systems to perform tasks effectively.*

*By integrating LLMs with MAS and BDD methodologies, there is potential to automate the generation and execution of natural language test scenarios for ADAS, leading to more efficient and comprehensive testing processes.*

*This exploration of SBV, BDD, MAS, and LLMs provides a foundation for understanding the current landscape and identifying opportunities for innovation in automating ADAS testing.*

### **Proposed Approach**

To automate **Behavior-Driven Development (BDD)** for **scenario-based verification of ADAS/autonomous driving**, we propose an **LLM-based Multi-Agent System (MAS)** framework. This approach leverages **Large Language Models (LLMs)** to generate, translate, and refine BDD scenarios into executable test cases, while **Multi-Agent Systems (MAS)** manage scenario execution and validation within a simulation environment.

## **1. Architecture of LLM-based MAS System**

The proposed system consists of **three core agents** that work together to automate scenario generation, execution, and evaluation.

### **A. Agents in the MAS System**

**1.Test Scenario Generator Agent (TSG)**

The Test Scenario Generator Agent (TSG) is responsible for automating the generation of test scenarios based on high-level natural language descriptions. This agent leverages Large Language Models (LLMs) such as Gemini, GPT, LLaMA to translate user-defined requirements into structured, executable test cases that adhere to Behavior-Driven Development (BDD) principles. Key Functions of TSG:

1. **Natural Language to BDD Conversion:** The TSG processes human-written scenario descriptions and converts them into BDD-compliant Gherkin format.

**Example:**

**Input Requirement:**

*"An autonomous vehicle driving at 60 km/h must detect a stopped vehicle ahead and apply the brakes within 1.5 seconds."*

**Generated Gherkin Scenario:  
  
Feature: Collision Avoidance**

**Scenario: Detect and respond to a stopped vehicle**

**Given an autonomous vehicle is driving at 60 km/h**

**And a stationary vehicle is detected 20 meters ahead**

**When the vehicle does not change lanes**

**Then the vehicle should apply the brakes within 1.5 seconds**

1. **Scenario Structuring and Edge Case Coverage:** Ensures the generation of edge cases (e.g., adverse weather, sudden pedestrian crossings). Generates variants of the same scenario with different speeds, reaction times, and environmental conditions to enhance testing coverage.
2. **Machine-Readable Test Case Generation:** Converts BDD scenarios into structured, machine-executable test cases in formats such as: OpenSCENARIO XML/JSON (for CARLA, LGSVL, and other simulators). Python test scripts (for automated execution in a simulation framework).
3. **Adaptive Scenario Expansion:** Uses LLMs to dynamically refine scenarios based on past test results. If a test fails, the TSG modifies parameters (e.g., reducing detection range, increasing traffic density) to generate a more challenging test case.

### TSG’s Role in MAS-Based ADAS Verification receives high-level requirements from engineers, testers, or regulatory standards. Generates BDD test scenarios and converts them into executable simulation scripts. Passes scenarios to the Scenario Executor Agent (SEA) for testing in the selected simulation environment.Updates the scenario database to continuously improve scenario generation over time. This automated scenario generation process enhances testing efficiency, reduces manual effort, and ensures a broader coverage of critical ADAS scenarios.

### **2. Scenario Executor Agent (SEA)**

**The Scenario Executor Agent (SEA) plays a crucial role in executing the generated test scenarios within ADAS simulation frameworks such as CARLA, LGSVL, and OpenSCENARIO. This agent is responsible for translating the structured test cases into simulation-compatible scripts, deploying them, and coordinating the execution of multiple autonomous agents (e.g., ego vehicle, pedestrians, obstacles).**

### **Key Functions of SEA:**

1. **Scenario Translation & Simulation Integration**
   * **Converts BDD scenarios and machine-readable test cases into executable scripts for simulators.**
   * **Supports multiple output formats:**
     + **OpenSCENARIO (industry-standard for structured scenario definition).**
     + **CARLA Python API (for high-fidelity real-time simulation).**
     + **LGSVL JSON-based Scenario Configurations (for perception and planning testing).**
   * **Ensures that all necessary parameters (e.g., vehicle speed, pedestrian crossing times, road conditions) are accurately configured.**
2. **Scenario Deployment & Control of Autonomous Agents**
   * **Loads the translated test scenarios into the selected ADAS simulation environment.**
   * **Spawns and controls multiple agents, including:**
     + **Ego Vehicle: The autonomous vehicle under test.**
     + **Traffic Participants: Other vehicles, pedestrians, cyclists, and traffic signals.**
     + **Environmental Factors: Weather conditions, road obstructions, and lane markings.**
3. **Real-Time Execution Monitoring & Data Logging**
   * **Continuously monitors the test execution, capturing:**
     + **Vehicle reactions (e.g., braking, lane changes, acceleration).**
     + **Sensor data (LiDAR, cameras, radar outputs).**
     + **Collision events and rule violations.**
   * **Logs key performance metrics, such as:**
     + **Time-to-collision (TTC).**
     + **Braking response time.**
     + **Lane-keeping accuracy.**
     + **Compliance with safety regulations.**
4. **Dynamic Scenario Adaptation**
   * **If a test scenario fails (e.g., collision detected), the SEA adapts parameters and re-runs the test with modified conditions (e.g., reduced speed, increased sensor delay).**
   * **Works iteratively with the Test Scenario Generator Agent (TSG) to refine and enhance scenario complexity.**

### **SEA’s Role in the LLM-Based MAS System**

1. **Receives executable test scenarios from the Test Scenario Generator Agent (TSG).**
2. **Translates and loads scenarios into a selected ADAS simulation framework (CARLA, LGSVL, OpenSCENARIO).**
3. **Deploys multiple agents and executes test runs with real-time monitoring.**
4. **Logs and analyzes system behaviors, providing results to the Evaluation Agent (EA).**
5. **Adjusts scenarios dynamically for enhanced verification and edge-case exploration.**

**This automated scenario execution ensures a scalable, efficient, and realistic verification process for ADAS/autonomous systems, significantly reducing manual effort in simulation-based testing.**

### **3.Evaluation Agent (EA)**

The **Evaluation Agent (EA)** is responsible for analyzing and interpreting the results of executed test scenarios. It collects **real-time execution data**, compares system behavior against predefined safety standards, and determines whether an **ADAS/autonomous system meets expected performance criteria**.

### **Key Functions of EA:**

#### **1. Data Collection from Scenario Execution**

* Gathers **real-time execution logs** from the **Scenario Executor Agent (SEA)**, including:
  + **Vehicle Trajectories:** Position, speed, lane adherence.
  + **Reaction Times:** Braking response, time-to-collision (TTC), perception delays.
  + **Safety Violations:** Collisions, near misses, pedestrian crossings, traffic rule violations.
  + **Sensor Performance:** Data from LiDAR, radar, and camera-based detections.

#### **2. LLM-Based Natural Language Processing (NLP) for Analysis**

* Uses **LLMs** to interpret and summarize simulation logs.
* Converts complex log data into **human-readable test reports** by **correlating actual system behavior with expected outcomes**.
* Example:
  + **Expected Behavior:** *"The vehicle should detect a pedestrian within 2 seconds and apply braking."*
  + **Actual Behavior (from logs):** *"Braking was applied after 2.8 seconds."*
  + **NLP-Based Interpretation:** *"Test failed: Reaction time exceeded limit by 0.8 seconds."*

#### **3. Pass/Fail Test Evaluation**

* Evaluates ADAS performance using predefined **safety metrics** and regulatory requirements.
* Criteria include:
  + **Collision Avoidance:** No impact should occur within a safety buffer zone.
  + **Lane-Keeping Performance:** Deviation from lane center must be within acceptable thresholds.
  + **Emergency Braking Efficiency:** Braking must occur within a defined reaction time.
  + **Pedestrian Detection Accuracy:** Pedestrians must be detected under varying lighting conditions.

#### **4. Report Generation & Feedback Loop**

* Generates a **structured test report** detailing:
  + Pass/Fail status for each scenario.
  + Safety compliance scores.
  + Performance improvement recommendations.
* Works in a **feedback loop** with the **Test Scenario Generator Agent (TSG)** to **refine and regenerate test cases** based on failure patterns.

### **EA’s Role in the LLM-Based MAS System**

1. **Receives execution logs** from the **Scenario Executor Agent (SEA)**.
2. **Analyzes vehicle/system behavior** using NLP-enhanced LLMs.
3. **Compares expected vs. actual outcomes** based on predefined safety criteria.
4. **Generates structured test reports**, including pass/fail assessments.
5. **Feeds evaluation results back to the TSG**, refining test scenarios for future iterations.

This **automated evaluation framework** ensures that ADAS testing is **efficient, scalable, and compliant with industry safety standards**, reducing the need for manual test interpretation.

### **B. System Interaction Flow**

The interaction between these agents follows a **structured workflow**:

1. **Test Scenario Generator Agent** receives high-level requirements (e.g., "A pedestrian crosses the road unexpectedly, the car must stop").
2. It **generates a BDD scenario** in **Gherkin syntax** (Given-When-Then format).
3. The scenario is converted into an **executable test case** for the simulation environment.
4. **Scenario Executor Agent** loads the test case into a simulation framework (CARLA, LGSVL, OpenSCENARIO).
5. The agent **executes the scenario**, controlling all entities (vehicles, pedestrians).
6. **Evaluation Agent** collects and analyzes results, comparing expected vs. actual system behavior.
7. A **final test report** is generated, providing insights into system performance.

## **2. Automation of BDD**

### **A. Converting Natural Language into Executable Test Cases**

A key feature of the **LLM-based Multi-Agent System (MAS)** is its ability to **automate the transformation of natural language test descriptions into executable test cases**. This process is crucial for automating **scenario-based verification of ADAS/autonomous driving systems**.

### **1. LLMs for Natural Language Processing in Test Case Generation**

**Large Language Models (LLMs)** analyze **high-level scenario descriptions** and identify key behavioral elements, such as: Vehicle speed and movement patterns, Traffic participants (pedestrians, other vehicles, obstacles), Expected system responses (e.g., braking, lane change, acceleration), Time constraints and safety margins.

### **2. Automatic Conversion to Gherkin-Based BDD Scenarios**

Once an LLM extracts **relevant scenario details**, it **automatically generates a Behavior-Driven Development (BDD) scenario** in **Gherkin format**.

#### **Example:**

**Input:**  *"An ADAS vehicle should detect a pedestrian crossing suddenly and apply emergency braking within 2 seconds."*

**Generated BDD Scenario (Gherkin Format):**

Feature: Emergency Braking

Scenario: Pedestrian crosses unexpectedly

Given an ADAS-equipped vehicle is driving at 50 km/h

When a pedestrian crosses the road at 5m distance

Then the vehicle should apply emergency braking within 2 seconds

### **3. Translating BDD Scenarios into Executable Test Cases**

The **MAS framework** processes the **BDD scenario** and translates it into an **executable simulation script**. This script is formatted for **ADAS testing environments**, such as: **CARLA (Python API), LGSVL (JSON-based scenario configuration), OpenSCENARIO (XML/JSON format for structured scenarios)**

#### **Example: Translating BDD to OpenSCENARIO XML**

<Storyboard>

<Story name="EmergencyBrakingTest">

<Act>

<ManeuverGroup name="EgoVehicle">

<Actors selectTriggeringEntities="true">

<EntityRef entityRef="EgoCar"/>

</Actors>

<Maneuver>

<Event name="PedestrianCrossing">

<Action>

<EnvironmentAction>

<Weather>

<Sun intensity="0.8"/>

</Weather>

</EnvironmentAction>

</Action>

</Event>

<Event name="EmergencyBrake">

<Action>

<LongitudinalAction>

<SpeedAction>

<SpeedTarget>

<AbsoluteTargetSpeed value="0"/>

</SpeedTarget>

</SpeedAction>

</LongitudinalAction>

</Action>

</Event>

</Maneuver>

</ManeuverGroup>

</Act>

</Story>

</Storyboard>

This **structured test case** can now be executed in **ADAS simulators**, ensuring **repeatable and automated validation** of emergency braking performance.

### **4. Integration into MAS-Based Testing Workflow:** LLMs parse and convert natural language test descriptions into BDD scenarios. The Test Scenario Generator Agent (TSG) translates BDD into structured test scripts. The Scenario Executor Agent (SEA) loads and runs these scripts in ADAS simulation frameworks. The Evaluation Agent (EA) analyzes the results, comparing expected vs. actual performance. This approach automates scenario creation and execution, reducing manual effort, and ensuring comprehensive verification of ADAS behaviors.

### **B. MAS for Coordinating Scenario Execution & Verification**

A **Multi-Agent System (MAS)** provides a structured approach for **coordinating and executing scenario-based verification** in ADAS testing. By using autonomous agents that communicate and collaborate, MAS ensures **realistic interactions between the ego vehicle, pedestrians, other vehicles, and traffic infrastructure**.

### **1. Role of MAS in Scenario Execution:** MAS enables a **distributed and adaptive simulation environment** where different agents interact dynamically.

* **Ego Vehicle Agent:** Controls the autonomous vehicle under test, executing the planned trajectory and responding to environmental stimuli.
* **Pedestrian Agent:** Simulates realistic pedestrian behavior, including random crossing, running, or hesitation.
* **Traffic Vehicle Agents:** Represent surrounding vehicles, following predefined or AI-generated driving patterns.
* **Traffic Signal Agent:** Controls road signals (traffic lights, pedestrian crossings, stop signs).
* **Weather & Environment Agent:** Dynamically changes environmental conditions (rain, fog, snow) to test perception robustness.

### **2. Communication Between Agents:** To ensure synchronization and proper coordination, agents exchange messages using multi-agent communication frameworks, such as: SPADE (Smart Python Agent Development Environment), JADE (Java Agent Development Framework).

### **3. Adaptive Scenario Execution**

MAS **dynamically adjusts test parameters** based on real-time simulation outcomes. This ensures that **edge cases and safety-critical situations** are effectively tested.

#### **Dynamic Adaptation Examples:**

1. **Vehicle Speed Adjustment:** If the ego vehicle fails to stop in time, the system re-runs the test with a **lower initial speed** to determine the threshold for safe braking.
2. **Pedestrian Behavior Variability:** The **Pedestrian Agent** may alter crossing behavior by: Walking slower/faster,n Hesitating before crossing, Entering the road unexpectedly.
3. **Environmental Conditions Variations:** The **Weather & Environment Agent** modifies, **Lighting Conditions:** Day, night, fog;**Road Surface Conditions:** Dry, wet, icy.

## **3. Integration with ADAS Simulation Framework**

To validate our approach, we integrate with leading **ADAS simulation tools**, such as:

1. **CARLA** (Open-source simulator for autonomous driving)  
   * Uses Python API for defining test scenarios.
   * Supports reinforcement learning-based control strategies.
2. **LGSVL** (Simulation platform for perception and control testing)  
   * Compatible with **Apollo & Autoware** stacks.
   * Provides real-world sensor data simulation.
3. **OpenSCENARIO** (Industry-standard scenario format)  
   * Enables structured scenario representation.
   * Ensures compatibility across different simulators.

### **A. System Workflow in Simulation**

1. LLM generates test scenarios and converts them into **executable simulation scripts**.
2. The MAS framework **loads and runs** the test in the chosen simulator.
3. Simulation logs are **analyzed by the Evaluation Agent**.
4. Reports are generated for **pass/fail criteria**, helping refine ADAS algorithms.

### **Experimental Results**

This section presents the **case studies, evaluation metrics, and comparisons** to demonstrate the effectiveness of our **LLM-based MAS framework for ADAS verification**.

### **A. Case Studies: Automating BDD Scenarios**

To evaluate the system, we selected **three common ADAS scenarios** and automated their execution using our framework.

#### **Case Study 1: Emergency Braking for Pedestrian Detection**

**Input:** *"An ADAS-equipped vehicle should detect a pedestrian crossing suddenly and apply emergency braking within 2 seconds."*

**Generated BDD Scenario (Gherkin Format):**

Feature: Emergency Braking

Scenario: Pedestrian crosses unexpectedly

Given an ADAS-equipped vehicle is driving at 50 km/h

When a pedestrian crosses the road at 5m distance

Then the vehicle should apply emergency braking within 2 seconds

**Simulation Execution (CARLA + OpenSCENARIO):**

* Pedestrian appears 5m ahead of the ego vehicle.
* Vehicle perception module detects pedestrian.
* Autonomous braking is triggered within 1.8 seconds.
* **Outcome:** **Pass**

#### **Case Study 2: Lane-Keeping Assist in Adverse Weather**

**Input Natural Language Description:** *"The ADAS vehicle should maintain its lane under heavy rain conditions."*

**Generated BDD Scenario (Gherkin Format):**

Feature: Lane-Keeping in Rain

Scenario: Vehicle stability under adverse weather

Given an ADAS vehicle is driving at 80 km/h in heavy rain

When the road has low traction

Then the vehicle should maintain its lane without deviation

**Simulation Execution (LGSVL):**

* Rain intensity set to **90%** (low visibility, slippery road).
* Ego vehicle maintains lane position with **deviation < 0.2m**.
* **Outcome:** **Pass**

#### **Case Study 3: Adaptive Cruise Control (ACC) in Mixed Traffic**

**Input Natural Language Description:** *"The vehicle should adjust its speed based on the distance to the leading vehicle and maintain a safe following distance."*

**Generated BDD Scenario (Gherkin Format):**

Feature: Adaptive Cruise Control

Scenario: Following distance adjustment

Given an ADAS-equipped vehicle is driving at 100 km/h

And a leading vehicle is 30m ahead

When the leading vehicle slows down to 80 km/h

Then the ego vehicle should adjust speed and maintain a 2-second following distance

**Simulation Execution (CARLA + OpenSCENARIO):**

* Leading vehicle decelerates from **100 km/h → 80 km/h**.
* Ego vehicle detects speed reduction and adapts acceleration profile.
* **Outcome:** **Pass**

### **B. Evaluation Metrics**

We assess our system’s performance using **three key metrics**:

**Accuracy of Scenario Translation**

* Measures the correctness of LLM-generated BDD scenarios.
* Evaluated using manual verification and rule-based NLP parsing.
* **Result:** **96.4% accuracy in translating natural language into structured BDD**.

**Execution Efficiency**

* Compares **time required for manual vs. automated scenario execution**.
* **Result:** **80% reduction in test case execution time** using our MAS system.

**Effectiveness of LLMs in Handling BDD Statements**

* Measures LLM’s capability to generate syntactically and semantically correct test cases.
* Compared against traditional rule-based NLP systems.
* **Result:** **LLM-based approach outperforms traditional methods by 15-20% in test generation efficiency**.

### **C. Comparison with Traditional Verification Approaches**

### **Discussion & Future Work**

### **A. Challenges & Limitations**

While our **LLM-based MAS framework** improves **scenario-based verification** for ADAS, several challenges and limitations remain. LLMs sometimes generate syntactically correct but semantically incorrect BDD scenarios due to a lack of domain-specific fine-tuning for automotive safety standards. To address this, fine-tuning LLMs on ADAS-specific datasets, such as Euro NCAP test cases, can improve accuracy by incorporating real-world safety requirements. Additionally, implementing constraint-based filtering can help validate the correctness of generated scenarios by ensuring alignment with predefined safety criteria. To further enhance reliability, a human-in-the-loop approach can be introduced for initial verification, allowing experts to review and refine test scenarios before execution.

Agents in the MAS framework require better real-time synchronization to effectively handle complex, multi-actor scenarios, such as multi-vehicle interactions in urban environments. To improve coordination, implementing decentralized decision-making can enable agents to operate autonomously while maintaining collaborative behavior. Additionally, optimizing message-passing mechanisms using frameworks like SPADE or JADE can enhance communication efficiency, reducing latency and ensuring seamless interaction among agents during scenario execution.

LLMs may struggle to generate rare, high-risk scenarios, such as sudden child crossings or extreme weather conditions, due to training data being biased toward common driving situations. To address this, reinforcement learning (RL) can be employed to enhance scenario diversity by encouraging the generation of edge cases that challenge ADAS performance. Additionally, augmenting training data with synthetic edge cases derived from real-world incident reports can improve the model’s ability to simulate critical but uncommon driving scenarios, ensuring more robust and comprehensive ADAS verification.

### **B. Future Enhancements**

To further enhance our system, we propose the following **future research directions**: **Fine-Tuning LLMs for Automotive-Specific Testing** on **ADAS regulatory datasets** (ISO 26262, NHTSA, UNECE safety protocols). Incorporate **real-world accident datasets** to generate more realistic test cases. Use **self-supervised learning** to improve scenario translation accuracy. **Expanding Multi-Agent System Capabilities by** Introducing **hierarchical MAS architectures** for **better coordination** between scenario execution agents. Use **swarm intelligence techniques** to allow agents to adapt dynamically. Improve **distributed computing support**, enabling parallel execution of large-scale scenario tests. Deploy our framework in **real-world vehicle-in-the-loop (VIL) and hardware-in-the-loop (HIL) testing**. Validate test scenarios using **physical ADAS test platforms** (e.g., Mobileye, Tesla FSD, Waymo). Work toward **standardizing AI-driven verification processes** for ADAS certification.