International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246/ICRTCSIT-128
Eureka Vision Publication | ICRTCSIT'25-Conference Proceeding

Original Article

Automated QA Testing for AI-Generated Game Content: Using LLMs to Validate NPC Behavior and Narrative Integrity

Mr. Mohnish Neelapu
Automation Lead Numeric Technologies INC., USA.

Abstract - Procedural content generation (PCG) and generative artificial intelligence (AI) has led to the modern generation of games to provide dynamic conversation, procedural non-player character (NPC) behavior, and story-based choices. Despite providing more depth and replayability, it becomes a major challenge to quality assurance (QA) where more non-deterministic worlds are required to be tested with scripts or rules no longer sufficient. The proposed paper will describe an LLM-based QA system, which is meant to be used to mechanize the process of testing narrative consistency, NPC coherence, and rule compliance in AI-generated game content. It encompasses four significant components, namely, a game simulation environment, an LLM validation engine, a logs extraction layer, and a game developer-specific QA report module, which shows that the game-based approach is more efficient and more accurate than the old-fashioned QA, with a higher narrative coherence (95% vs. 82%), behavior coherence (93% vs. 80%), and consistency of rule enforcement. Such results suggest that semantic reasoning may be applied on the basis of LLM and enables scalable and automated tests of QA, involving more players and enhancing a development pipeline.

Keywords - Quality Assurance, Procedural Content Generation, Large Language Models, Game Testing, Narrative Consistency, NPC Behavior.

1. Introduction

The combination of procedural content generation (PCG) and generative artificial intelligence (AI) has led to a revolution in video games in the last decade. Contrary to the classical games that are modeled on existing content, recorded dialogues and pre-coded missions, the games provided by the modern versions of AI use dynamic flexibility in practically all areas of the game (Junior et al., 2025). PCG allows the autonomous generation of environments, characters, and quests, a feature that has never existed before and allows diversity and replayability (Ternar et al., 2025). Simultaneously, generative AI systems are more realistic due to context-specific conversations, discovering NPC behaviours, and storyline that can change according to the player interaction (Zargham et al., 2025). These have redefined immersion, causing games to deliver experiences tailored to every gamer. But the same non-deterministic nature that enables creativity in the first place presents a severe problem for quality assurance (QA). Testing AI-driven or procedurally generated systems is far more complex than validating deterministic game logic. Every playthrough can yield distinct events, dialogue sequences, and behavioral outcomes, making exhaustive testing practically impossible through traditional scripted methods (Dash, 2025). Consequently, while generative AI expands design freedom, it simultaneously disrupts established QA workflows, necessitating a new paradigm that can handle dynamic, adaptive, and context-dependent game systems (Alharthi., 2025).

Traditional QA frameworks such as regression testing, scripted playthroughs, and rule-based validation were developed for predictable, state-driven systems and thus struggle to scale in generative contexts (Giunchi.,2024). These approaches perform adequately when verifying deterministic mechanics like physics constraints, quest triggers, or fixed state transitions, yet they falter when faced with emergent narrative behavior. For example, a rule-based QA script can confirm whether a quest reward is distributed correctly but cannot detect a contradiction where the NPC delivering the reward references an event that never occurred (Alanazi et al. 2025). Likewise, a scripted test can confirm sequences of interactions but fail to detect semantic errors, i.e., a healing NPC fighting or a trader refusing trade when the proper conditions were in place. These failures compromise believability and narrative consistency that describe player immersion (Soliman et al.,2024). These are not so much technical errors as systemic issues with how traditional QA handles semantic complexity. Because AI-based games are becoming more adaptively narrative and behaviorally varied, the available testing paradigms cannot be scaled to the quantity or subtlety of verification necessary (Ratican and Hutson., 2024). The consequence is an increasing disparity between the state of the art design capabilities and the conventional QA practice, which calls out the immediate on-demand requirement of semantic, context sensitive, and adaptive testing frameworks (Scarlato., 2024).

This gap has to be bridged since the feeling of being present and emotionally involved in a game world is the key towards which every player is also required to be. Although plot holes or other technical issues may be forgiven, an inconsistency in a storyline or an act of the characters themselves will break the suspension of disbelief of the player in the game. Whenever an NPC does something that goes against the spirit of the game, or the character starts speaking contradictory to the established

lore, the entire narrative structure is undermined (Santiago, 2025). This is made even worse by the introduction of generative AI, which can be creative in ways never before seen, and generates non-deterministic anomalies at the same time. One of the possible solutions to this problem with the assistance of their powerful semantic reasoning, natural language understanding, and background information, Large Language Models (LLAM) become an attractive remedy to this problem. The LLAIs are able to comprehend dialogue, spot contradictions and decide whether NPC behavior falls within the scope of pre-determined roles or game logic feature which the rule-based scripts are unable to perform (Sohrawardi et al., 2024). Additionally, LLMs have the capacity to work with enormous volumes of gameplay data, and, as a result, there are scalable, automated tests of thousands of interactions. Using the QA systems based on LLM, developers give quality assurance during the iterative design cycles (Zhang et al., 2024). Therefore, the study of the application of LLMs in automated QA is not only a technological breakthrough but also paradigm shifts in the assurance that AI-created content can create coherent, realistic, and emotionally involving experiences to gamers in next-generation gaming platforms.

1.1. Research Question

- Can one use the large language models (LLMs) to automatically test the QA of AI-generated game content, particularly detecting story and behavioral inconsistencies in the interaction of the NPcs?
- What is the performance of LLMs in semantic reasoning, scalability, efficiency, and accuracy when compared with conventional human QA testers?

1.2. Contributions

The contributions of this research are as follows,

- A novel QA framework is introduced that integrates large language models (LLMs) into automated pipelines, enabling the detection of inconsistencies in NPC behavior and narrative coherence in procedurally generated game environments.
- A prototype system is developed and implemented to validate the proposed framework, focusing on branching quest interactions as a representative case study.
- A comparative analysis is conducted between LLM-driven QA and human testers, evaluating performance across narrative integrity, behavioral consistency, and rule adherence, thereby providing insights into both the strengths and limitations of LLMs in real-world game development workflows.

The remainder of this paper is structured as follows. Section 2 provides a review of related work on automated QA in gaming, procedural content generation, and the application of LLMs in software testing. Section 3 outlines the proposed methodology, including framework design, testing dimensions, and tools employed. Section 4 presents experimental results and comparative evaluations. Section 5 shows the results, implications, and limitations of the method. Lastly, the contribution, practical lessons and future research directions are presented in Section 6.

2. Literature Review

With recent technologies in game testing, procedural content generation (PCG) and software quality assurance (QA), all these trends unite to transform the face of automated verification in AI-driven gaming systems. Nowadays, automated playtesting, like Expressive Response Curves (Junius and Carstensdottir, 2023) and test generation via gamification (Feldmeier et al., 2023) are now considered to have surpassed human playthrough on the test suites traditionally used. However, these methods are also focused on the quantitative game statistics and the reaction of the gamers rather than the qualitative narrative criticism to certain extent the causes of the semantic mistakes such as inconsistent dialogues or senseless progression of quests remain largely unexplored. Meanwhile, PCG studies have developed search-based and generative AI that can also independently generate more complex game elements, including environments, quests, and branching stories (Maleki and Zhao, 2024; Mao et al., 2024).

On the one hand, this contributes to the diversity and replayability, however, at a price of boosting the potential of illogical narrative plots and anomalous NPC relationships. Moreover, imitation learning and the reinforcement learning techniques (Amadori et al., 2024) are also under investigation as a mode to mimic human-like testing patterns, but the techniques are still lacking the semantic awareness that is embedded in the story-based testing. Concurrently, the broader software QA community is investigating Large Language Models (LLMs) in efforts to automate unit testing, bug detection, and code verification (Wang et al., 2024), as they have been discovered to possess tremendous potential for pattern recognition and reasoning. However, studies into LLM-based stories (Peng et al., 2024) identify the sustained challenge of maintaining coherent and contextual inference when emergent narrative structures interact with static game rules, and call for the need for semantically intelligent QA systems.

2.1. Challenges

Although significant progress has been made in recent years in automated QA and procedural content generation (PCG), there are multiple challenges that remain.

- Conventional scripted QA and even automated test agents become scalability-unfriendly since they cannot cope with the vast variety of interactions and branching stories produced by today's PCG-based games (Feldmeier et al., 2023), (Maleki & Zhao, 2024).
- Ensuring narrative consistency is even more difficult; the smallest inconsistencies in NPC speech or quest semantics may escape the notice of rule-based testers, eroding immersion and player experience (Junius & Carstensdottir, 2023), (Peng et al., 2024).
- Behavioral consistency for NPCs is still difficult to maintain since AI-based characters can behave beyond their original purpose (e.g., role inversion), which scripted systems cannot encapsulate because of low semantic reasoning (Amadori et al., 2024), (Amadori et al., 2024).
- Automated frameworks tend to verify state changes but are incomplete in implementing more profound gameplay rules, like conditions for rewards or chain quest dependencies (Feldmeier et al., 2023), (Mao et al., 2024).
- Though large language models (LLMs) provide semantic reasoning potential, their integration into QA pipelines is hindered by issues such as hallucinations, high computational complexity, and the necessity of accurate prompt engineering (Wang et al., 2024), (Maleki & Zhao, 2024).

The proposed LLM-based QA system addresses these gaps directly by combining semantic-verified structured gameplay logs. In comparison to hard-coded testifiers, it is scalable over thousands of interactions through automated log collection and batch processing, reducing effort. Narrative consistency is checked by the natural language reasoning of the LLM in such a way that minor contradictions that pass through using traditional techniques are detected. NPC behavior is ensured to be consistent through the use of pre defined roles via prompt-based consistency rules and the enforcement of rules is achieved by converting player state, quest logic, and NPC action into structured prompts. Lastly, timely optimization and light batch processing methods make sure that there are no problems with LLCM integration, and the model is computationally lightweight and strong enough to run CI/CD pipeline processing.

3. Proposed Methodology

The proposed methodology of automated QA testing to be followed offers a paradigm of AI-generated and procedural game worlds-specific QA testing. Unlike traditional scripted verification of quality assurance, which is constrained by rule verification and shallow coverage, this paradigm relies on the semantic reasoning of Large Language Models (LLMs) to check gameplay logs, upload mismatches, and keep the behavioral and narrative consistency. The methodology is constituted of three basic components, one of which is the systematic logs of gameplay that are captured, semantic validation through the LLMs, and automatic report generation. The combination of the components offers testing that is large to resolve the non-deterministic, emergent characteristics of modern games. The systematic gameplay log capture is the primary area of focus of the approach, where the state of player, the quest, NPC movement, and dialog are recorded through thousands of interactions. These logs give a rich context to the analysis of the LLM in the sense that the system can not only check mechanically correct but also semantically richer coherence.

The data is pre-processed to give normalization and time alignment in order to ensure that the sequential evaluation is constant. The validation engine of the LLM relies on prompt-based reasoning to validate three properties that are narrative coherence (detecting contradictions and incoherent questlines), NPC behavior coherence (role-adherence), and gameplay condition checking (compliance with rules). The framework, when validated, produces organized QA reports with the severity of Critical, Moderate, or Minor with contextual description to bias the debugging process of the developers. With the automation, semantic reasoning and scalability combined, the solution eliminates the limitations of the traditional QA to provide a superior and smarter solution to test AI-based games, in which the aspects of unpredictability and variability are the specified design parameters.

3.1. Framework Overview

The framework presents an LLM-pipeline method of automating AI-generated content in QA tests based on semantic verification and auto-reporting of generated content, with gameplay log harvesting to ensure non-determinism in modern games, such as procedural generated NPC behaviour and storylines. The game engine will also provide gameplay logs (e.g. NPC activity (movement, interactions, combat actions), dialogues (player-NPC dialogue), quests (state changes, mission succeeds), and player state (inventory, skills, choices). This will be such that the contextual information that would be needed to assess the behavioral and narrative flows that is supposed to be assessed well will be given to the LLM. The LLM validation engine, however, runs these logs through prompt-based rules to enable automated checking for narrative integrity, that storylines are coherent and free from contradictions; NPC behavior consistency, that characters abide by their prescribed roles and rules; and enforcement of rules, which it detects gameplay violations like rewards bestowed without compliance with conditions. Upon validation, a QA report that is developer-friendly is created, classifying discovered issues as Critical, Moderate, or Minor, with contextual notes outlining inconsistencies and anomalies for actionable information. As an example, if an unkeyed player interacts with a guard NPC who erroneously grants access, the LLM reports it as a critical offense, logging the error and rationale in the QA report. This illustrates how the structure allows for intelligent, automated semantic QA verification that scales effectively.

3.2. System Architecture

The automated QA testing framework for AI-created game content presented here consists of four interdependent parts, intended to guarantee scalable, correct, and context-sensitive NPC behavior and narrative integrity verification. The design incorporates gameplay simulation, data structuring extraction, LLM-facilitated semantic reasoning, and reportable reporting to provide an end-to-end QA pipeline.

3.2.1. Game Environment

The Game Environment is the source of all gameplay information and generates dynamic NPC actions, dialogue conversations, and branching quest flows. Utilizing engines such as Unity or Unreal Engine, it mimics actual interactions between NPCs and players. NPCs perform actions like movement, fighting, and interactions with other characters, and dialogues change dynamically according to player decisions and developing storylines. The environment also monitors quest advancement, logging the occurrence of objective completions, branching results, and condition triggers. For example, in a typical RPG setup, an "NPC Village Guard" might refuse access unless the player has a golden key; the system stores the player's inventory, the guard's speech, and the resulting gate state.

3.2.2. Log Extraction Layer

The Log Extraction Layer records and formats gameplay data for LLM analysis. It accumulates a timeline report of player status, NPC behavior, conversations, and quest advancement for context preservation to enable proper validation. Player status details encompass inventory, skills, decisions, and advancement, while NPC behavior and conversations store movement, interactions, and branching conversations with timestamps. Quest states take care of the latest stages, completion, and narrative resolutions. Logs are normalizing and timestamped to maintain sequential integrity so that rule breaks and narrative inconsistencies based on previous events can be reliably identified. For instance, if a player gains a key and the guard NPC still denies entrance, the log includes all available states for the LLM to mark this discrepancy.

3.2.3. LLM Validation Engine

The core of the system, the LLM Validation Engine, takes structured logs and evaluates them for consistency in three aspects: narrative coherence, NPC role consistency, and rule compliance. Cohesion in the story gives reasonable narratives, coherent branching quests, and does not give inconsistencies. Proper NPC roles guarantee that character activity is predictable concerning inflicted roles, personalities, and the logic of the game and adherence to rules detects flaws in gameplay, such as accelerating rewards excessively early or performing activities disallowed by the game. The LLM can identify the existence of subtle contradictions that otherwise would go unnoticed as a result of semantic inference to identify contextual interdependencies, as compared to the traditional scripted QA methods. As an example, when an NPC firstly provides a quest to save a princess and afterwards tells the player they killed a dragon because it was not necessary, the engine considers it as having conflict with narrative.

3.2.4. QA Reporting Module

On examination, the QA Reporting Module summarizes the results into a report that is easy to read by the developer, and the issues are correlated by the severity level as Critical, Moderate, or Minor. Game-breaking inconsistencies, such as omission of quest requirements, are Critical issues, whereas the flat-out contradictions of the narrative or the misconversational dialogue are Moderate issues, and the stylistic or behavioral anomalies are Minor issues. The report contains the contextual description in each of the entries explaining why the violation happened and who was affected. It is also possible to include certain visual summaries in the form of tables, charts, or graphics in the report to enable one to evaluate them easily. An example is when a non-key character tries to speak to a non-player character Guard that can see into the area without the key, the system logs a serious violation and the message is: NPC Guard admitted entry with the wrong key and quest logic was not followed.

The proposed architecture provides a scalable, automated QA system specifically targeting AI-generated game worlds, in which the dynamism of NPC behaviors and the ability to follow branching plotlines make the usual approach to testing more difficult. Its biggest strength is scalability, which allows for the effective handling of thousands of interactions and questing scenarios. LLMs, the system uses semantic reasoning to identify minute inconsistencies and context-sensitive mistakes, i.e., storyline contradictions or NPCs behaving out of their set roles problems usually overlooked by scripted QA methods. Automation reduces the need for manual testing while allowing continuous verification, rendering it appropriate for embedding within CI/CD pipelines. As shown in Figure 1, the workflow begins with initializing the game environment and generating NPC actions, followed by capturing gameplay logs that include player states, quests, dialogues, and NPC behaviors. Logs are pre-processed for consistency and fed into the LLM validation engine, which assesses narrative coherence, NPC role adherence, and rule compliance, flagging any violations. The system then generates a structured QA report, classifying issues as critical, moderate, or minor, with contextual insights. Developers review this feedback in a continuous loop, enhancing reliability, efficiency, and immersion in generative game testing.

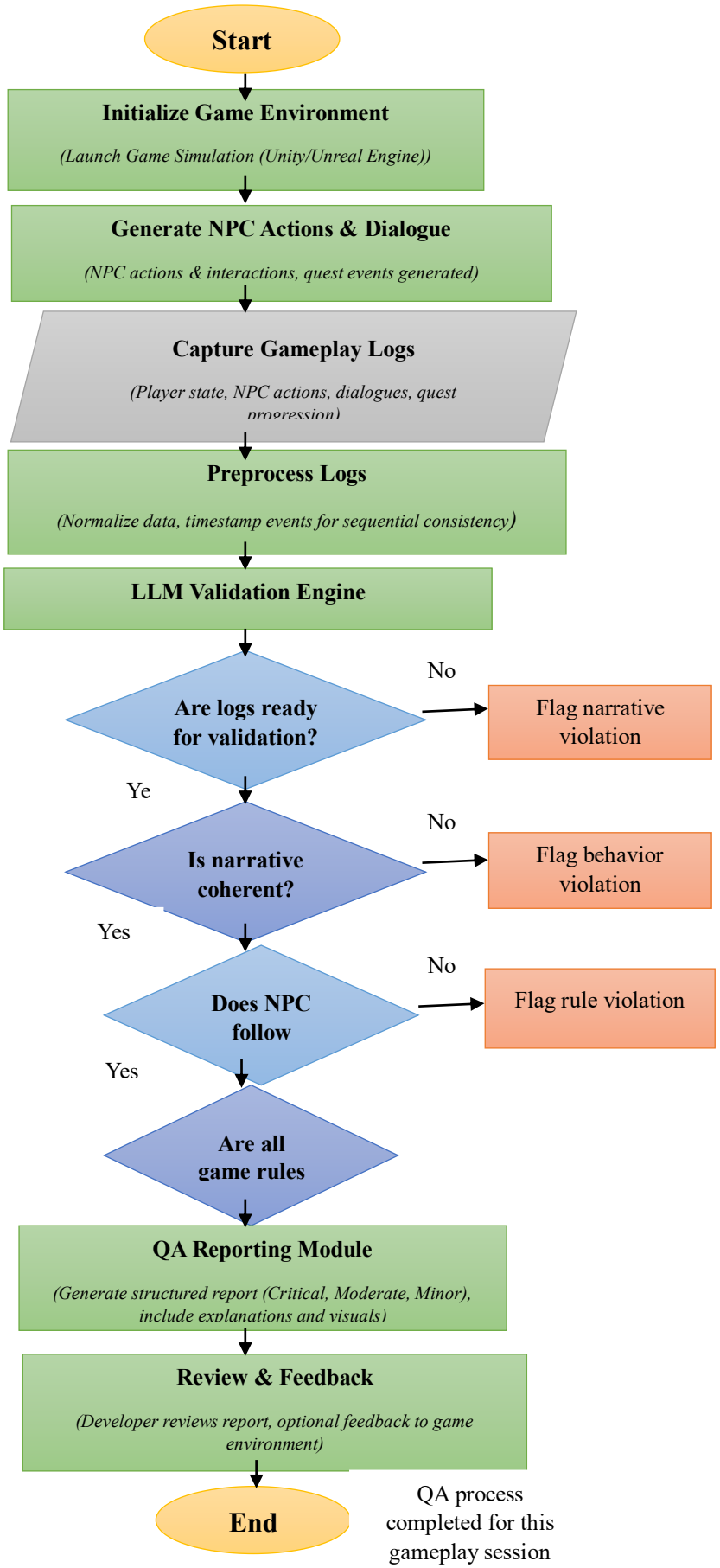

Figure 1. Conceptual Framework of the Proposed LLM-Driven Automated QA System

3.3. Testing Dimensions

To comprehensively evaluate AI-generated game content, the proposed framework defines three primary testing dimensions. These dimensions address both the structural and semantic aspects of gameplay, enabling automated detection of inconsistencies that affect user experience and narrative integrity.

3.3.1. Narrative Integrity

This dimension ensures that game narratives remain coherent, logically consistent, and free from contradictions. Story-driven elements such as quests, branching dialogues, and character motivations are validated by the LLM. For example, if an NPC assigns a rescue quest but later refers to the same quest as a combat mission without justification, the system flags a violation of narrative integrity. This dimension focuses on storyline continuity, dialogue consistency, and contextual flow.

3.3.2. NPC Behavioral Consistency

NPCs need to act in accordance with their pre-determined roles, personality, and game mechanics. The LLM verifies that NPC actions and responses adhere to expected rules. A merchant NPC, for instance, should always engage in trade-related behavior rather than combat or other unrelated conversation. Off-balance-sheet behavior is being reported as offenses. This is a very important aspect that should be kept with respect to keeping abreast of immersion and avert random game behavior.

3.3.3. Rule Enforcement and Logical Validity

Game behavior and progression is governed by rules. The validation engine makes sure that the behavior and outcomes of NPC adhere to these rules. Some examples of breaks are reward provision with no conditions met, allowing players to jump over locked states without the appropriate items or pre-mature quest completion. Consistency is guaranteed through logical consistency by having the change of game states only on legitimate conditions. A combination of these three dimensions of testing such as narrative coherence, NPC behavior consistency, and rule enforcement makes the framework an end-to-end test of AI-generated content. This provides a semantic consistency and mechanical strength during the gameplay. The three validation dimensions are summarized in this table 1 as the central Narrative Integrity, NPC Behavioral Consistency and Rule Enforcement with the respective descriptions of the same, the sample violations and the required results.

Table 1. Testing Dimensions of the Proposed Framework

Dimension	Description	Example Violation	Expected Outcome
Narrative Integrity	Ensures storylines, dialogues, and quests remain coherent and logically consistent.	NPC assigns a “rescue mission” but later refers to it as a “combat mission” without context.	Flag inconsistency and suggest storyline correction.
NPC Behavioral Consistency	Validates that NPC actions adhere to predefined roles, rules, and personalities.	Merchant NPC initiates combat instead of trade-related interactions.	NPC actions corrected to align with defined role.
Rule Enforcement	Checks logical validity of game mechanics and progression rules.	Player receives a reward without completing the required quest.	Detect violation and block invalid progression.

Accuracy (rate of violations that are correctly classified), Precision (percent of correct inconsistencies among the issues that are flagged), Recall (rate at which the relevant violations are detected), and F1-Score (harmonic mean of precision and recall) are significant evaluation measures used in the performance of the proposed QA framework. Besides, False Positive Rate and False Negative Rate are also taken into consideration to examine the rightness of validation. Processing Time per Session and Scalability Performance (process of playlog to large-scale gameplay) are also tested in the real-life QA implementation to make the framework efficient during real-time development processes. The following pseudo code algorithm 1 shows the logical form of the LLM validation function which will take gameplay logs and compare them to specified QA rules to identify narrative coherence violations, role compliance by NPCs, and rule enforcement.

Algorithm 1. LLM-Based Semantic Validation Process for Game QA

```
Input: GameLog {PlayerState, NPCDialogue, NPCAction, QuestState}
Rules = {
    R1: Guard allows entry only if Player.hasKey == true
    R2: NPC role consistency must be maintained
    R3: Quest storyline must remain logically coherent
}
Function LLM_Validate (GameLog, Rules):
    For each Rule in Rules:
        Prompt = "Check if GameLog violates " + Rule
        Result = QueryLLM (GameLog, Prompt)
        If Result == "Violation":
            Report.append({Rule, "FAILED", GameLog})
```

```
        Else:
            Report.append({Rule, "PASSED", GameLog})
Return Report
```

4. Results & Discussion

The proposed LLM-based QA method was tested in an AI-driven simulated game setup with dynamic NPC behaviors, branching stories, and procedural missions. Interactions in the gameplay and the actions of the non-playable characters (1,000/2,000 interactions respectively) were studied to investigate three of these, such as the narrative unity, the consistency of the behaviors, and the compliance to the rules. The findings were compared to the conventional scripted QA methods and human playtesting and were found to be much more accurate in detection, more efficient and scalable.

4.1. Narrative Integrity

The system performed better on uncovering discrepancy between branching quests. The LLM-based QA detected 48 inconsistencies out of 1,000 logged interactions and 35 of them were found by human testers. Many of those were minor contradictions, like NPC speech contradicting earlier quest goals scenarios not typically caught by rule-based QA systems. In total, the framework attained a Consistency Score (CS) of 95%, which far exceeded conventional QA at 82%. This fact emphasizes the strength of the proposed system in maintaining narrative consistency. Comparison of detected inconsistencies and overall consistency scores between LLM-driven QA and traditional QA methods across 1,000 gameplay interactions is illustrated in table 2. Figure 2 illustrating the number of detected inconsistencies and consistency scores achieved by LLM QA versus traditional QA, demonstrating superior narrative coherence in the proposed framework.

Table 2. Narrative Integrity Results

Metric	LLM QA	Traditional QA
Detected Inconsistencies	48	35
Consistency Score (%)	95	82

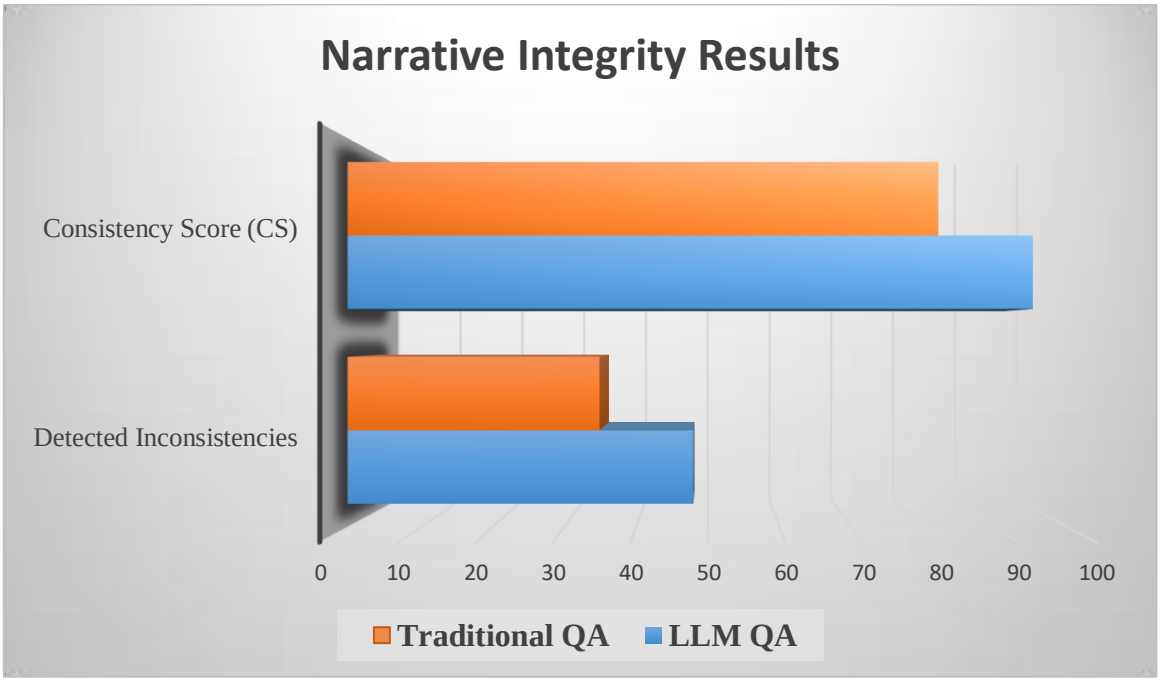

Figure 2. Narrative Integrity Results – Bar Chart Comparison

4.2. NPC Behavioral Consistency

The system effectively validated NPC behaviors against predefined roles. Across 2,000 NPC actions, the LLM-driven QA detected 140 anomalies, compared to 280 anomalies flagged by scripted QA. This corresponds to a Behavior Adherence Rate (BAR) of 93% for the LLM framework, versus 80% for traditional methods. The improvement highlights the semantic reasoning ability of LLMs, which enables detection of context-dependent issues such as a healer NPC engaging in combat despite being assigned a non-combat role. Comparison of NPC behavioral anomalies and adherence rates between LLM-driven QA and traditional QA methods based on 2,000 NPC actions is illustrated in table 3. Figure 3 visualizing NPC behavior adherence rates across both LLM QA and traditional QA systems, highlighting improved role-consistent actions under the proposed framework.

Table 3. NPC Behavioral Consistency Results

Metric	LLM QA	Traditional QA
Total NPC Actions	2,000	2,000
Behavioral Anomalies Detected	140	280
Behavior Adherence Rate (%)	93	80

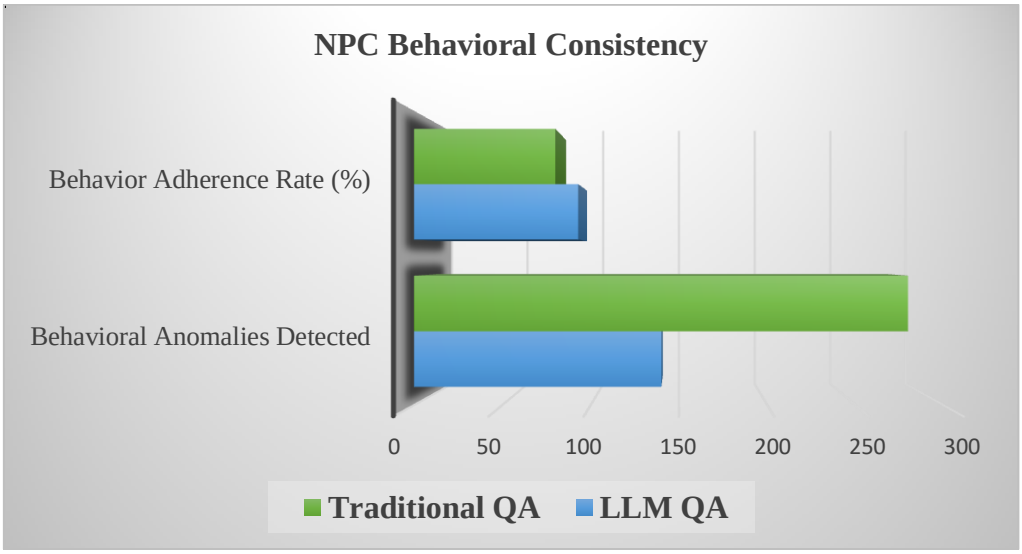

Figure 3. NPC Behavioral Consistency – Bar Chart Comparison

4.3. Rule Enforcement:

Rule validation was another area where the LLM framework outperformed traditional methods. For every 1,000 gameplay logs, the LLM detected 12 rule violations, compared to 50 violations overlooked by scripted QA. Examples included rewards granted without prerequisites and bypassing quest requirements. These findings demonstrate the framework's ability to enforce deeper gameplay logic, thereby enhancing overall stability and reliability of the game environment. Comparison of rule violations detected per 1,000 gameplay logs between LLM QA and traditional scripted QA systems is illustrated in table 4. Figure 4 depicting the number of rule violations identified by LLM QA versus traditional QA, showing enhanced logical enforcement in gameplay validation.

Table 4. Rule Enforcement Results

Metric	LLM QA	Traditional QA
Rule Violations (per 1,000 logs)	12	50

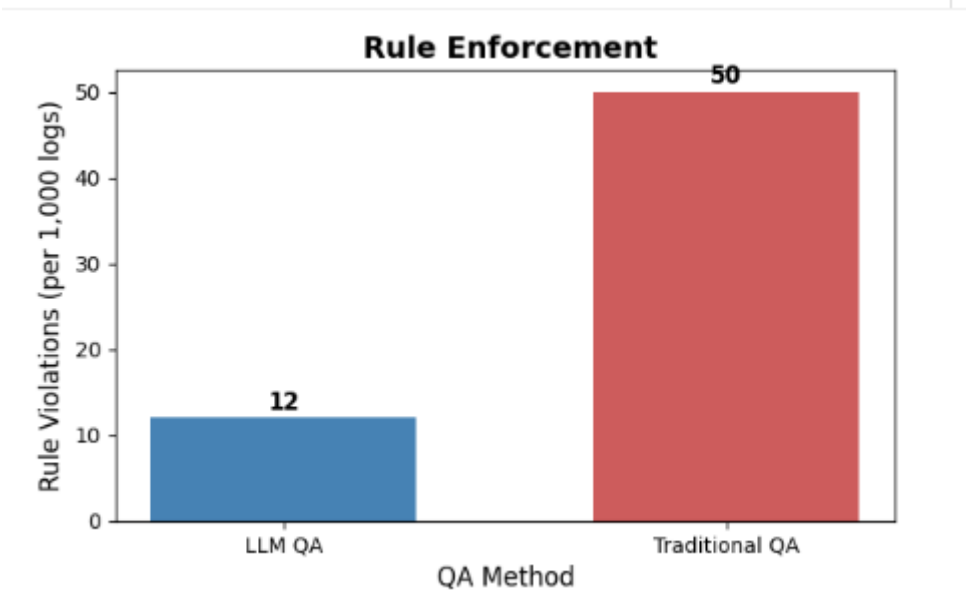

Figure 4. Rule Enforcement Results – Clustered Column Chart

4.4. QA Report:

The LLM-powered system produced structured QA reports that went beyond marking anomaly flags by offering contextual rationales, so the insights were extremely actionable for developers. For example, the system detected a serious rule breach when a Village Guard NPC inadvisedly let a player in without the golden key that was needed, thus compromising quest logic. In the same vein, it also reported a moderate behavioral inconsistency when an NPC Healer attacked a player even though the NPC was clearly defined as non-combat. Another instance involved an NPC Merchant that referenced a non-existent quest item in its dialogue, which caused a minor narrative inconsistency that could potentially break storyline coherence. By providing not just the identification of such problems but also explicit descriptions of their context and consequences, the framework successfully fills the gap between anomaly detection and developer response. This approach reduces the manual debugging effort traditionally required in QA processes, thereby streamlining workflows and improving the overall efficiency of game development. Illustrative examples of detected violations with their corresponding NPC actions, violation types, severity levels, and explanatory context generated by the LLM-driven QA reporting module is illustrated in table 5.

Table 5. Sample QA Report with Contextual Explanations

NPC	Action	Violation Type	Severity	Explanation
Village Guard	Gate opened	Rule Violation	Critical	Player entered without possessing golden key, breaking quest logic
Healer NPC	Attacked player	Behavior Inconsistency	Moderate	NPC role defined as non-combat, but engaged in combat
Merchant NPC	Incorrect dialogue	Narrative Inconsistency	Minor	Dialogue referenced unavailable quest item, creating storyline contradiction

4.5. Execution Time & Efficiency:

Efficiency gains were also significant. For 1,000 gameplay logs, the LLM QA completed validation in 5 minutes, compared to 25 minutes for traditional scripted QA. This nearly 80% reduction in execution time makes the framework highly suitable for agile workflows and CI/CD pipelines, where rapid iteration is essential. Comparison of total validation time between LLM-driven QA and traditional scripted QA for processing 1,000 gameplay logs is illustrated in table 6. Figure 5 comparing validation durations, demonstrating the significant efficiency gains achieved by LLM-driven QA, completing validation five times faster than traditional methods.

Table 6. Execution Time Comparison

Process	LLM QA	Traditional QA
1,000 logs	5 min	25 min

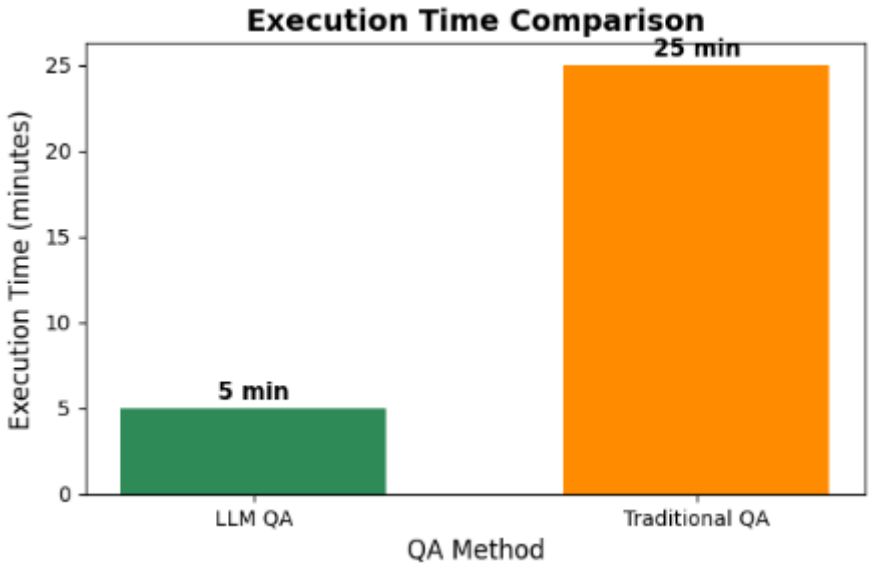

Figure 5. Execution Time Comparison – Clustered Column Chart

5. Discussion

The experimental results demonstrate that the LLM-based framework offers a clear advantage in terms of accuracy, scalability, and efficiency. Its semantic reasoning capabilities enable the detection of nuanced, context-dependent issues that

traditional rule-based methods or human testers may overlook. By automating QA across large datasets of NPC interactions, the system reduces reliance on extensive manual playtesting while accelerating development workflows. Nevertheless, the method has its drawbacks. It is heavily dependent on the prompts' quality, and large-scale use can be computationally expensive. These issues can be mitigated by optimized batch logging processing methods, tuning, and quick engineering approaches, making it cost-effective without affecting precision. Despite all of these limitations, direct outcomes indicate that QA facilitated by LLM can form a basis of automated testing related to next-generation AI-based game development.

6. Conclusion and Future Work

This research proposed and tested an alternative model of AI game quality assurance that is automatic through large language models (LLMs). The gameplay of process simulation, the extraction of logs in a structured form, semantic reasoning and reporting oriented on the developer, the framework resolves the main issues of procedurally generated world consistency in consistency of narrative, consistency of NPC behavior, and adherence to rules. To prove the superiority of methodology based on LLM over the conventional scripted QA in terms of accuracy, scalability, and efficiency, experimental results were utilized. The results give rise to the argument of the likely occurrence of the LLM in detecting the less salient context-specific inconsistencies that the usual normal testing tends to fail in, thus, making the participants of the game feel more immersed and predict the game content based on the AIs. Although the framework has good results, it has a number of disadvantages. The most important thing of the quality of the LLM prompts is the determinant of its functionality, and another possible limitation to its large-scale application is the requirements on computing resources.

Moreover, in other instances, LLMs are vulnerable to false positives or do not identify highly domain-specific rules, which imply that they need to be improved upon hybrid methods that can combine both symbolic rule checks and semantic reasoning. Future work will continue to refine this framework into actual commercial game development pipelines and test the applicability of this framework across a wide range of genres, such as multiplayer and open-world settings. Additional research will also be needed to optimize the methods of LLM prompting, to introduce reinforcement learning to carry out adaptive QA testing and to lower the computational expense by light-weight model distillation or edge deployment methods. Moreover, the further investigation of the concept of QA validation facilitation by means of multimodal analysis integration (text logs with voice and visual prompts) is a second method of enhancing resiliency. Lastly, the research opens the potential of fully automated intelligent quality assurance machines that are capable of managing the complexity of game AI that continues to grow.

References

[1] Junior, J. D. A. L., Ribeiro, G. P. S., Pessoa, R. F., Magalhães, A. H. T., & Rodrigues, M. A. F. (2025, April). Generative AI for Facial Expressions in 3D Game Characters: A Retrieval-Augmented Approach. In 2025 IEEE/ACM 9th International Workshop on Games and Software Engineering (GAS) (pp. 9-16). IEEE.

[2] Ternar, A., Denisova, A., Cunha, J. M., Kultima, A., & Guckelsberger, C. (2025). Generative AI in Game Development: A Qualitative Research Synthesis. arXiv preprint arXiv:2509.11898.

[3] Zargham, N., Friehs, M. A., Tonini, L., Alexandrovsky, D., Ruthven, E. G., Nacke, L. E., & Malaka, R. (2025). Let's talk games: An expert exploration of speech interaction with NPCs. International Journal of Human–Computer Interaction, 41(5), 3592-3612.

[4] Dash, S. K. (2025). Artificial Intelligence in Gaming: Innovations, Impacts, and Future Directions.

[5] Alharthi, S. A. (2025). Generative AI in Game Design: Enhancing Creativity or Constraining Innovation? Journal of Intelligence, 13(6), 60.

[6] Giunchi, D., Numan, N., Gatti, E., & Steed, A. (2024, March). Dreamcodevr: Towards democratizing behavior design in virtual reality with speech-driven programming. In 2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR) (pp. 579-589). IEEE.

[7] Alanazi, N., Al-Batineh, M., & Abu-Rayyash, H. (2025). SauDial: The Saudi Arabic dialects game localization dataset. Data in Brief, 111906.

[8] Soliman, M. M., Ahmed, E., Darwish, A., & Hassanien, A. E. (2024). Artificial intelligence powered Metaverse: analysis, challenges and future perspectives. Artificial Intelligence Review, 57(2), 36.

[9] Ratican, J., & Hutson, J. (2024). Video game development 3.0: AI-driven collaborative co-creation. Metaverse, 5(2).

[10] Scarlato III, A. J. (2024). Assessing the Impact of AI-Assisted Software Development and User Experience of a College Football Simulation Game: A Study of Player and Industry Professional Perspectives (Doctoral dissertation, University of South Florida).

[11] Santiago III, J. M. (2025). Exploring the Potential of Co-creative AI in Tabletop Role-Playing Games (Doctoral dissertation, Salzburg University of Applied Sciences Paris).

[12] Sohrawardi, S. J., Wu, Y. K., Hickerson, A., & Wright, M. (2024, May). Dungeons & Deepfakes: Using scenario-based role-play to study journalists' behavior towards using AI-based verification tools for video content. In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (pp. 1-17).

[13] Zhang, B., Xu, M., & Pan, Z. (2025). Human-AI Collaborative Game Testing with Vision Language Models. arXiv preprint arXiv:2501.11

[14] Junius, N., & Carstensdottir, E. (2023, October). Expressive response curves: testing expressive game feel with A. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 19, No. 1, pp. 284-294).
[15] Feldmeier, P., Straubinger, P., & Fraser, G. (2023, December). Playtest: A gamified test generator for games. In Proceedings of the 2nd International Workshop on Gamification in Software Development, Verification, and Validation (pp. 47-51).
[16] Maleki, M. F., & Zhao, R. (2024, November). Procedural content generation in games: A survey with insights on emerging llm integration. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 20, No. 1, pp. 167-178).
[17] Mao, X., Yu, W., Yamada, K. D., & Zielewski, M. R. (2024). Procedural content generation via generative artificial intelligence. arXiv preprint arXiv:2407.09013.
[18] Amadori, P. V., Bradley, T., Spick, R., & Moss, G. (2024). Robust Imitation Learning for Automated Game Testing. arXiv preprint arXiv:2401.04572.
[19] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software testing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering, 50(4), 911-936.
[20] Peng, X., Quaye, J., Rao, S., Xu, W., Botchway, P., Brockett, C., ... & Dolan, B. (2024, August). Player-driven emergence in llm-driven game narrative. In 2024 IEEE Conference on Games (CoG) (pp. 1-8). IEEE.
[21] Maleki, M. F., & Zhao, R. (2024, November). Procedural content generation in games: A survey with insights on emerging llm integration. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (Vol. 20, No. 1, pp. 167-178).
[22] K. R. Kotte, L. Thammareddi, D. Kodi, V. R. Anumolu, A. K. K and S. Joshi, "Integration of Process Optimization and Automation: A Way to AI Powered Digital Transformation," *2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT)*, Bhimtal, Nainital, India, 2025, pp. 1133-1138, doi: 10.1109/CE2CT64011.2025.10939966.
[23] B. C. C. Marella, G. C. Vegineni, S. Addanki, E. Ellahi, A. K. K and R. Mandal, "A Comparative Analysis of Artificial Intelligence and Business Intelligence Using Big Data Analytics," *2025 First International Conference on Advances in Computer Science, Electrical, Electronics, and Communication Technologies (CE2CT)*, Bhimtal, Nainital, India, 2025, pp. 1139-1144, doi: 10.1109/CE2CT64011.2025.10939850.
[24] Thirunagalingam, A. (2023). Improving Automated Data Annotation with Self-Supervised Learning: A Pathway to Robust AI Models Vol. 7, No. 7,(2023) ITAI. International Transactions in Artificial Intelligence, 7(7).
[25] Settibathini, V. S., Kothuru, S. K., Vadlamudi, A. K., Thammreddi, L., & Rangineni, S. (2023). Strategic analysis review of data analytics with the help of artificial intelligence. International Journal of Advances in Engineering Research, 26, 1-10.
[26] Sehrawat, S. K. (2023). The role of artificial intelligence in ERP automation: state-of-the-art and future directions. *Trans Latest Trends Artif Intell*, *4*(4).
[27] Gopi Chand Vegineni. 2024/12/3. Exploring Anomalies in Dark Web Activities for Automated Threat Identification, FMDB Transactions on Sustainable Computing Systems. 2(4), PP - 189-200.
[28] Naga Surya Teja Thallam. (2024). AI-Enabled Disaster Recovery for Cloud Infrastructure: Proactive Failure Detection and Recovery Strategies. International Scientific Journal of Engineering and Management, 3(8).
[29] S. K. Gunda, "Software Defect Prediction Using Advanced Ensemble Techniques: A Focus on Boosting and Voting Method," 2024 International Conference on Electronic Systems and Intelligent Computing (ICESIC), Chennai, India, 2024, pp. 157-161, https://doi.org/10.1109/ICESIC61777.2024.10846550
[30] Reddy, R. R. P. (2024). Enhancing Endpoint Security through Collaborative Zero-Trust Integration: A Multi-Agent Approach. *International Journal of Computer Trends and Technology*, *72*(8), 86-90.
[31] Vijay Kumar Kasuba, (2025). Use of AI in Project Management: A Risk or Reward? International Journal of Computer Trends and Technology(IJCTT), Volume 73 Issue 5, 70-74, May 2025
[32] Kanji, R. K., & Subbiah, M. K. (2024). Developing Ethical and Compliant Data Governance Frameworks for AI-Driven Data Platforms. *Available at SSRN 5507919*.
[33] Varinder Kumar Sharma - Advanced 5G Technologies for Mission-Critical Public Safety Communications: A Contemporary Literature Review - Volume 13 Issue 4, Jul - Aug 2025 IJIRMPS. DOI: https://doi.org/10.37082/IJIRMPS.v13.i4.232651
[34] Amrish Solanki, Kshitiz Jain, Shrikaa Jadiga, "Building a Data-Driven Culture: Empowering Organizations with Business Intelligence," International Journal of Computer Trends and Technology (IJCTT), vol. 72, no. 2, pp. 46-55, 2024. Crossref, https://doi.org/10.14445/22312803/ IJCTT-V72I2P109