

LLMs in Debate: Does Arguing Make Them Better at Detecting Metamorphic Relations?

Dibyendu Brinto Bose
Department of Computer Science
Virginia Tech
 brintodibyendu@vt.edu

Yoseph Berhanu Alebachew
Department of Computer Science
Virginia Tech
 yoseph@vt.edu

Chris Brown
Department of Computer Science
Virginia Tech
 dcbrown@vt.edu

Abstract—Large Language Models (LLMs) are transforming software engineering, including mobile Augmented Reality (AR) applications. AR software behavior often depends on dynamic environmental factors, making it difficult to use conventional testing and verification approaches. Metamorphic Testing (MT) offers an alternative by assessing whether expected transformations hold across varied conditions. However, there is limited work exploring how well LLMs can detect these transformations—Metamorphic Relations (MRs)—in applications. We propose a *stability-driven evaluation framework* that examines whether LLMs consistently apply MRs across rephrasings. Our study finds that StarCoder and CodeLlama exhibit higher stability in MR identification compared to the general-purpose model Gemma. Additionally, we use a multi-agent debate framework to investigate whether combining multiple perspectives improves consistency in MR identification. The debate mechanism reduces MR inconsistencies, leading to more stable MR identification across all MRs. While debate helps stabilize MR identification, our evaluation against human-labeled ground truth reveals that stability alone does not always correlate with correctness. Some models maintain stable yet incorrect predictions (CodeLlama), whereas debate enhances both consistency and correctness alignment, making LLM reasoning more reliable. This work contributes a method to evaluate LLMs in the absence of ground truth, establishing stability as a metric for assessing model reliability. Applying a multi-agent debate framework offers a promising approach to enhancing LLM reliability, especially in contexts where the ground truth is elusive.

Index Terms—Large language models, Metamorphic testing, Augmented Reality (AR), Multi-Agent Debate, Stability Evaluation

I. INTRODUCTION

Advances in artificial intelligence (AI) have transformed software engineering, with Large Language Models (LLMs) assisting developers in code generation, debugging, maintenance, and automated testing [1], [2], [3], [4]. Recent surveys show over 70% of developers are adopting AI tools, primarily for writing and testing code [5]. Yet, the reliability of LLMs in reasoning about software behavior remains uncertain. Unlike static analysis or rule-based systems, LLMs generate probabilistic outputs [6], producing different answers to the same query. Such instability can negatively affect software quality [7], [8], especially in emerging domains where ground truth oracles are unavailable [9].

Augmented and virtual reality (AR/VR) exemplify this challenge. AR applications depend on real-world interactions and dynamic environments [10], making correctness difficult

to verify with conventional unit or functional testing [11]. For instance, a virtual object should remain within boundaries, scale with distance, or update visibility under occlusion. These behaviors require reasoning about spatial and physical properties rather than fixed input–output mappings [12], [13].

Metamorphic testing (MT) offers an alternative by defining *metamorphic relations* (MRs)—expected transformations that must hold when conditions change [14]. Prior work has applied MT to mobile AR software, outlining key MRs such as boundary checks, scaling, and orientation consistency [15]. However, whether LLMs can consistently recognize and apply these transformations when analyzing AR-related source code remains unclear.

We use these MR definitions to evaluate LLMs, focusing not on correctness but on *stability*: the ability for models to consistently identify the same MR across different rephrasings. Instability suggests sensitivity to phrasing rather than true semantic understanding. This lens is particularly important as AR/VR development itself is prone to instability, with “too many unknowns” complicating testing and debugging [12]. By emphasizing stability, we move beyond accuracy metrics toward assessing whether models demonstrate genuine comprehension of MRs, which is critical as LLMs are increasingly integrated into requirements analysis [16], test automation [17], and other reliability-sensitive tasks.

This study is guided by three research questions:

- RQ1: How stable are LLMs in identifying metamorphic relations across different rephrasings?
- RQ2: Can a multi-agent debate framework improve the stability of metamorphic relation identification in LLMs?
- RQ3: To what extent does stability in metamorphic relation detection correlate with correctness?

To answer these, we evaluate three open-source LLMs capable of code reasoning—StarCoder, CodeLlama, and Gemma—on MR identification. Beyond individual model performance, we explore a *multi-agent debate* framework, where models iteratively critique each other’s MR classifications. If debate reduces inconsistencies, it provides a pathway toward more reliable LLM-assisted testing.

Our contributions are:

- An evaluation framework for assessing how well code-focused LLMs detect established metamorphic relations in mobile AR applications.

- Empirical analysis of the stability of three LLMs (StarCoder, CodeLlama, and Gemma) in recognizing MRs across different formulations.
- An examination of multi-agent debate as a mechanism to refine LLM reasoning and enhance stability.

Our findings provide empirical evidence on LLM stability in AR-related code analysis. By focusing on consistency across MR formulations, we propose a stability-based evaluation approach that can generalize to domains where semantic preservation under transformation is critical, such as compiler optimization or refactoring. This complements accuracy metrics with robustness measures. Moreover, results show that multi-agent debate not only reduces inconsistencies but also improves alignment with correctness, suggesting a practical strategy for enhancing the reliability of LLMs in software engineering. Our code and responses are available in an anonymized repository.¹

II. RELATED WORK

A. LLM Evaluation Methodologies

Evaluating LLMs remains challenging as traditional benchmarks capture limited facets of capability [18], [?]. Recent directions include mixed-initiative evaluation (e.g., EvalGen) to align criteria with human feedback [19], using LLMs as annotators with human-comparable agreement and model-model agreement as a suitability signal [20], and reference-free trust metrics such as TrustScore and curated truth sets like TruthEval [21]. We add to this landscape by treating *stability across rephrasings* as an evaluation signal for metamorphic relation (MR) detection in code.

B. Metamorphic Testing for Software Systems

Metamorphic Testing (MT) addresses the oracle problem by defining MRs—expected relationships between inputs and outputs under systematic transformations [14]. Hybrid strategies and randomized source test generation improve fault detection [22], while algorithmic selection and input partitioning guide effective test design [23]. MT has also proven effective in visual computing, and search engines [24], [25]. Recently, MT has been adapted to AR software, where dynamic context complicates conventional testing [15].

C. Multi-Agent Systems in Software Engineering

Multi-agent LLM frameworks enhance software engineering (SE) tasks via role specialization, intermediate verification, and collaboration. Systems such as MAGIS for repository-level issue resolution [26], agentic pipelines for qualitative analysis [27], and agile-process simulations [28] show gains over single-agent baselines. We examine whether a *multi-agent debate* mechanism similarly improves stability for MR identification.

¹https://anonymous.4open.science/r/MR_LLM_MULTIAGENT-8269/README.md

D. Testing AR/VR Applications

Testing AR/VR is challenging due to reliance on real-world interaction, spatial reasoning, and physics [11], [12]. Proposed solutions span oracle prediction [29], generative exploration [30], and MT [11]. We adopt the AR MRs from Bose et al. [15] as our evaluation target (Table I).

TABLE I: Metamorphic Relations and Their Explanations

MR Name	Explanation
Raycast within Boundary	Ensures objects spawned within a boundary remain consistently detectable within that boundary even as the AR camera changes position across multiple planes.
Overlapping with Game Object	Verifies that new game objects properly detect existing objects and prevent spawning overlapping items when simulating touch events.
Visibility and Occlusion	Tests that partial occlusion of objects behind other objects adjusts correctly as the camera perspective changes to different viewing angles.
Object Scaling with Distance	Confirms that objects automatically scale larger or smaller in proper proportion as the camera moves closer to or further from them.
Orientation Consistency Relative to Gravity	Validates that objects maintain their correct orientation relative to gravity even when the AR device is rotated or tilted.
Varying Rotation	Ensures that an object’s rotation angle remains within acceptable threshold parameters when rotation is simulated at different angles and speeds.
Check Correct Object Instantiation	Verifies that objects instantiated through touch input match the expected object type and properties.

III. DATASET

We collected a dataset of open-source mobile AR projects from GitHub using the GitHub API. We filtered repositories using the following criteria. Our search targeted repositories containing AR-related terms such as “AR Foundation”, “ARCore”, “ARKit”, and “Augmented Reality”. The dataset includes projects written in C#, Java, Kotlin, and Swift. To ensure that the projects were actively maintained, we included only those that had at least 10 stars and had commits pushed after January 2023. The data collection process continued until February 2025. Additionally, we filtered for repositories licensed under widely recognized open-source licenses, including MIT², Apache-2.0³, and BSD-3-Clause⁴.

A. Data Filtering

After collecting an initial set of 97,510 source code files, we applied further validation to ensure quality. We verified the build status of projects using *Unity Test Runner*⁵ for Unity-based applications, *Gradle Build Scanner*⁶ for Android

²<https://opensource.org/licenses/mit>

³<https://www.apache.org/licenses/LICENSE-2.0>

⁴<https://opensource.org/licenses/bsd-3-clause>

⁵<https://docs.unity3d.com/2017.4/Documentation/Manual/testing-editor-test-runner.html>

⁶https://docs.gradle.org/current/userguide/build_scans.html

projects, and *Xcodebuild*⁷ for iOS projects. Projects that failed to compile were excluded. We also performed a project activity check, removing repositories with minimal commits or those maintained by a single contributor without a significant development history. After these filtering steps, the dataset was refined to 33,687 verified, compilable source code files.

B. Dataset Statistics

Table II summarizes the verified files per programming language. The dataset includes 253,630 import statements,

TABLE II: Final Dataset Overview

Language	Verified Files
C#	12,681
Java	7,322
Kotlin	9,448
Swift	4,236
Total	33,687

12,920,490 external dependencies, and 2,048,121 comment lines, with an average file length of 5,601 characters. This dataset provides a structured benchmark for evaluating LLM stability in mobile AR development, ensuring that only compilable and actively maintained projects are considered.

IV. METHODOLOGY

Our methodology employs pre-defined MRs (see Table I) to evaluate the response behavior of LLMs in source code transformation understanding. Given the absence of definitive ground truth, we hypothesize that an ideal model should produce responses that remain consistent across rephrased variations of an MR definition. This assumption is based on the premise that if a model genuinely understands the underlying transformation logic of an MR, minor linguistic variations should not alter its judgment. To assess this, we analyze how StarCoder, CodeLlama, and Gemma respond to different formulations of the same MR.

A. Model Selection

We selected StarCoder-7B⁸, CodeLlama-7B⁹, and Gemma-7B¹⁰ due to their relevance in source code understanding tasks. StarCoder is trained on a diverse set of programming languages and optimized for code generation, making it an appropriate baseline for assessing MR applicability. CodeLlama, a fine-tuned variant of Llama-2, is specifically optimized for code-related tasks, offering reasoning about source code structures robustness. Gemma, a general-purpose language model, was included to explore how well models not explicitly trained for programming tasks can generalize MR-based reasoning. This diverse selection allows us to compare models with different architectural biases and training paradigms.

⁷<https://developer.apple.com/documentation/xcode/build-system>

⁸<https://huggingface.co/blog/starcoder>

⁹<https://github.com/meta-llama/codellama>

¹⁰<https://huggingface.co/google/gemma-7b>

B. Generating MR Rephrasings

Prior work suggests rephrasing can enhance the capabilities of LLMs [31]. However, distinctly rephrasing MRs to assess the best working for models may not be feasible as constructing MRs is challenging for developers [32]—involving considerable manual effort [33], and programs can incorporate hundreds of MRs [34]. To evaluate how rephrased MR statements influence model responses, we first generated three rephrasings per MR: the original definition and two alternative formulations. We used the FLAN-T5-XXL (11B parameters)¹¹ model for rephrasing due to its strong instruction-following capabilities and state-of-the-art performance in controlled text generation tasks [35]. This model’s ability to maintain semantic consistency while introducing syntactic variation makes it ideal for creating meaningful rephrasings [36]. We applied cosine similarity scoring using the all-MiniLM-L6-v2 embedding model¹² to filter out redundant or excessively altered rephrasings because it efficiently captures semantic similarities between sentences while being computationally efficient [37]. Figure 1 shows an example of rephrasing of MR definition. The similarity score thresholds (0.4-0.9) were established based on prior research in semantic similarity measurement, demonstrating that scores below 0.4 typically indicate significant semantic drift, while scores above 0.9 suggest near-duplicate content [38]. This range ensures that our rephrasings maintain semantic fidelity while providing sufficient linguistic variation to test model robustness.

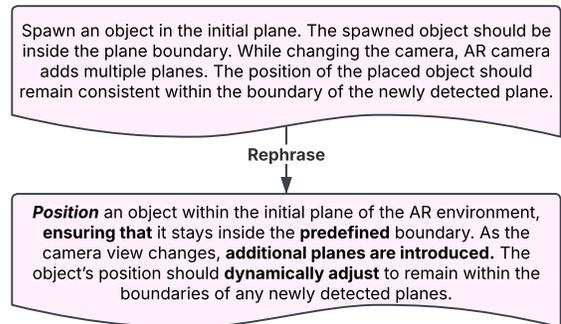


Fig. 1: Example of a rephrase of MR definition

After generating and filtering these initial rephrasings, we analyzed the variance in model responses across them. The observed variance was 0.0860 for CodeLlama, 0.0401 for Gemma, 0.0631 for StarCoder, and 0.1548 for the multi-agent debate system. Based on these variances, we conducted a power analysis to determine the necessary number of additional rephrasings required for statistical robustness. The analysis yielded sample size estimates of 15 for CodeLlama, 7 for Gemma, 11 for StarCoder, and 27 for the multi-agent debate framework. Given the number of model groups—2245

¹¹<https://huggingface.co/google/flan-t5-xxl>

¹²<https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

for CodeLlama, 4812 for Gemma, 3062 for StarCoder, and 1247 for the debate system—we ultimately generated 4812 rephrasings, ensuring that the largest model group had adequate coverage.

C. Prompting for Response Generation

To generate structured responses from LLMs for MR detection, we employ *structured instruction-based prompting* [39]. This prompt template ensures the LLM receives a well-defined task with clear input and output expectations. The prompt (shown below) presents the model with a mobile AR code snippet and asks whether a specified MR applies. We employed standard source code cleaning techniques, such as removing comments, redundant whitespace, and formatting inconsistencies [40], to ensure that the LLM focuses on meaningful code structures rather than extraneous noise. It follows a structured three-part format:

MR Classification Prompt:

Prompt Template
You are analyzing a mobile AR code snippet for applicability of the Metamorphic Relation (MR): {MR Name}.
Given Cleaned Code: {cleaned_code}
Question: Does this code have scope for applying the MR '{MR Name}'?
Answer Format:
<ul style="list-style-type: none"> • Yes or No • Brief explanation • Key elements from the code that support your answer

D. Assessing Response Consistency

To systematically measure whether models provide similar responses across rephrased MR variations, we introduced a formal metric. Instead of comparing each response to a single reference, we evaluate consistency within groups of responses generated for the same MR using pairwise cosine similarity. The consistency score for each sample group g is computed as:

$$S_g = \frac{1}{M} \sum_{i=1}^M \sum_{j=i+1}^M \text{sim}(R_{g_i}, R_{g_j}) \quad (1)$$

where M represents the number of MR variations within a group, and $\text{sim}(R_{g_i}, R_{g_j})$ is the cosine similarity between response pairs. The overall consistency across all sample groups is then calculated as:

$$S_{\text{final}} = \frac{1}{G} \sum_{g=1}^G S_g \quad (2)$$

where G is the total number of groups. This approach ensures that intra-group response agreement is assessed in a structured and statistically reliable manner.

E. Multi-Agent Debate Framework

Beyond evaluating models in isolation, we implemented a multi-agent debate system to assess whether structured deliberation improves consistency in identifying MRs. Instead

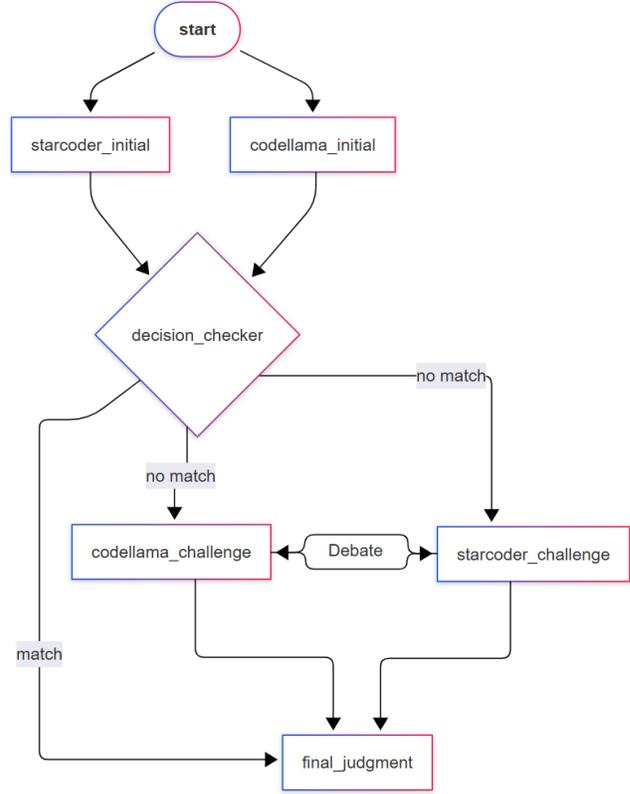


Fig. 2: Multi-Agent Debate Workflow

of relying solely on independent model responses, this approach introduces an iterative reasoning process where models challenge and refine their conclusions based on opposing viewpoints. We used StarCoder and CodeLlama for the multi-agent system, as these models show higher stability scores in our results for RQ1 than Gemma. The debate framework is designed to enhance the robustness of MR identification by incorporating multiple perspectives before reaching a final decision.

The debate process follows a structured sequence, illustrated in Figure 2. The debate begins with an initial assessment phase, where StarCoder and CodeLlama independently analyze the code. If their assessments align, the workflow proceeds directly to the final judgment. Otherwise, the models enter the challenge phase, exchanging arguments to reach a consensus. Each challenge cycle refines the reasoning of both models, potentially leading to improved consistency in MR identification. The process continues until either convergence is achieved or the debate reaches a predefined limit ($n = 3$). We selected a three-round structure for our multi-agent debate framework based on prior research [41], which shows that debates typically converge to stable conclusions within three iterations. Empirical studies in science debates and medical discussions have demonstrated that three rounds are optimal for

TABLE III: Example of Multi-agent debate for a given source code

Prompt: Analyze if Varying Rotation MR applies to the given AR scene rendering code.	
Round 1	
StarCoder	CodeLlama
The code handles AR object placement based on camera position. Varying Rotation MR does not apply because the rendering depends on absolute device orientation rather than relative rotation.	I believe Varying Rotation MR applies because the code uses quaternion-based transformations that preserve relative spatial relationships regardless of device orientation.
Round 2	
StarCoder	CodeLlama
After reviewing CodeLlama’s analysis, I see that while the rendering uses device orientation, the transformations maintain invariance under rotation as quaternion operations preserve relative positions. Varying Rotation MR does apply.	I maintain that Varying Rotation MR applies. The code properly accounts for device rotation by transforming coordinates within the AR space, ensuring consistent object placement regardless of how the user holds the device.
Final Judgment	
Decision Agent: After evaluating both arguments, I conclude that Varying Rotation MR applies to this AR rendering code. The quaternion-based coordinate system properly handles device rotation, maintaining consistent spatial relationships between virtual objects and the real world regardless of device orientation.	

balancing reasoning depth and computational efficiency [42]. By integrating independent model assessments with structured multi-agent debate, we aim to determine whether collaborative reasoning enhances the stability of LLM-based MR detection. To illustrate how this debate mechanism operates in practice, we provide an example in Table III, where the models assess whether Varying Rotation MR applies to a given AR scene rendering code. The example demonstrates how StarCoder initially disagrees with CodeLlama, refines its reasoning after a challenge, and ultimately aligns with the final decision made by the decision agent.

V. RESULTS

In this section, we answer each research question mentioned in the introduction.

A. RQ1: Evaluating Stability in LLM MR Detection

TABLE IV: Stability Score Metrics for CodeLlama, StarCoder, and Gemma

Model	Mean Stability Score	Std. Dev.	IQR
CodeLlama	0.7988	0.1199	0.1752
StarCoder	0.6684	0.1446	0.1941
Gemma	0.3309	0.0984	0.1106

Table IV presents the stability score values computed using Equation (2) for CodeLlama, StarCoder, and Gemma. We found CodeLlama achieved the highest stability (0.7988), followed by StarCoder (0.6684), while Gemma had the lowest

TABLE V: Stability Scores per Metamorphic Relation

Metamorphic Relation	StarCoder	Gemma	CodeLlama
Raycast within Boundary	0.6246	0.2221	0.7876
Overlapping with Game Object	0.6008	0.2766	0.7342
Visibility and Occlusion	0.6509	0.2051	0.7559
Object Scaling with Distance	0.6525	0.2936	0.7386
Orientation Consistency	0.6262	0.2677	0.7646
Relative to Gravity	0.6000	0.2669	0.7287
Varying Rotation	0.6267	0.3073	0.7596
Check Correct			
Object Instantiation			

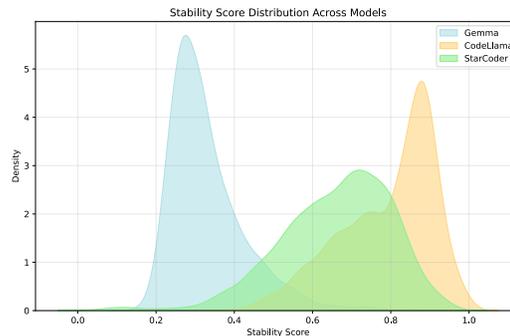


Fig. 3: Group Stability Score Distribution of our three experimented model

(0.3309), revealing noticeable differences in consistency. A breakdown of stability across different metamorphic relations is shown in Table V. CodeLlama performed better on *Raycast within Boundary*, while StarCoder had higher stability in *Visibility and Occlusion*. Some relations, such as *Overlapping with Game Object*, had lower stability scores across all models, indicating they are more challenging for LLMs.

We analyzed the distribution of group stability scores computed using Equation (1) across models to determine whether MR rephrasings significantly impact response consistency. Ideally, different variations of the same MR should yield stable responses; any substantial instability would indicate sensitivity to superficial linguistic modifications rather than the underlying transformation logic. Figure 3 illustrates the distribution of stability scores, demonstrating variations across models and MRs. To formally test this, we established the following hypotheses:

Null Hypothesis (H_0): Variations in MR rephrasings do not significantly impact stability scores.

Alternative Hypothesis (H_1): Variations in MR rephrasings significantly impact stability scores.

A Lilliefors test confirmed that the distribution of group stability scores was non-normal ($p = 0.0010$), necessitating the use of non-parametric statistical methods. Given the variance differences observed across models, we applied the Kruskal-

Wallis test to assess whether linguistic variations in MR definitions influenced stability. The test results ($p = 0.4973$ for Gemma, $p = 0.4982$ for StarCoder, and $p = 0.4969$ for CodeLlama) indicate no statistically significant difference in stability scores across different MR rephrasings. Since all p -values exceed the 0.05 significance threshold, we fail to reject the null hypothesis, suggesting that MR variations do not meaningfully alter model response behavior.

These findings suggest that, while different LLMs exhibit varying levels of stability, their responses remain consistent regardless of how MRs are phrased. This further supports the stability score as a reliable measure for evaluating LLM behavior in source code analysis. The fact that stability remains unaffected by linguistic rephrasings implies that models primarily rely on their underlying learned patterns rather than superficial textual variations when identifying metamorphic relations.

B. RQ2: Evaluating the Impact of Multi-Agent Debate on Stability

TABLE VI: Per-Metamorphic Relation Stability Scores

Metamorphic Relation	Stability Score
Overlapping with Game Object	0.8818
Visibility and Occlusion	0.8707
Check Correct Object Instantiation	0.8511
Raycast within Boundary	0.8311
Object Scaling with Distance	0.8210
Orientation Consistency Relative to Gravity	0.8209
Varying Rotation	0.7402
Average Stability Score	0.8309

TABLE VII: Decision Type Distribution Across Multi-Agent Debate. Each decision is made per metamorphic relation, leading to a total of $33,687 \times 7$ evaluations.

Decision Type	Count
Yes-Yes	51,353
No-No	120,058
Inconclusive-Inconclusive	28,780
Mismatches	36,618

The multi-agent debate framework was introduced to examine whether collaboration between models improves stability in metamorphic relation identification. Table VI shows the stability scores for each metamorphic relation after the debate process. The overall stability increased to 0.8309, with notable improvements in certain relations. *Overlapping with Game Object* achieved the highest stability (0.8818), while *Varying Rotation* remained the least stable (0.7402). This suggests that some relations benefit more from the debate mechanism, while others remain more sensitive to variations.

The debate framework also influenced how models reached decisions. Table VII presents the distribution of decision types across all evaluated cases. The system produced 51,353 *yes-yes* and 120,058 *no-no* agreements, along with 28,780 inconclusive cases. However, 36,618 instances showed mismatches, indicating points of persistent disagreement.

C. RQ3: Does Stability in LLM Predictions Indicate Correctness?

While RQ1 showed that LLMs can be stable across rephrasings, stability does not guarantee correctness—a model may consistently repeat the wrong answer. To examine this relationship, we compared stability with correctness against human-labeled responses. Two authors independently annotated 140 predictions (20 samples across 7 MRs), achieving substantial agreement ($\kappa = 0.73$).

Accuracy results show CodeLlama at 52.3%, Gemma at 61.5%, StarCoder at 64.7%, and the multi-agent debate at 81.2%, indicating that debate improves correctness over single models. Spearman’s rank correlation further quantifies the stability–correctness link: CodeLlama shows only a weak correlation ($\rho = 0.32$), reflecting that it can be consistently wrong; Gemma ($\rho = 0.45$) and StarCoder ($\rho = 0.51$) exhibit moderate correlation; and the debate framework achieves the strongest alignment ($\rho = 0.73$).

These results confirm that stability alone is insufficient as a correctness signal. However, structured debate not only enhances stability but also produces outputs more consistent with human-labeled correctness, suggesting a more reliable reasoning process.

VI. DISCUSSION

A. Stability Trends Across Metamorphic Relations

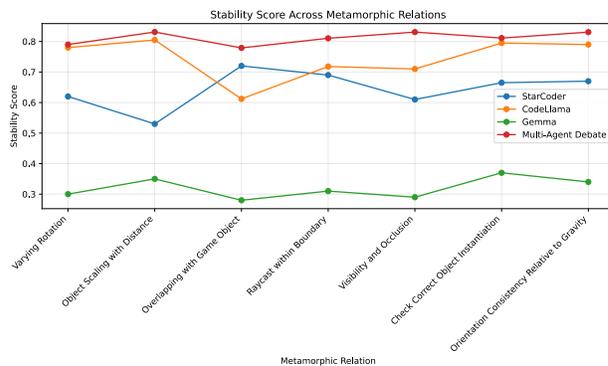


Fig. 4: Group Stability Score Distribution of our multi-agent debate system

We compared individual model stability with the multi-agent debate framework across metamorphic relations. As shown in Figure 4, StarCoder and CodeLlama exhibit fluctuations, with StarCoder dropping on *Object Scaling with Distance* (0.5300) and CodeLlama on *Overlapping with Game Object* (0.6120). Gemma remains unstable across all MRs.

In contrast, the debate framework achieves balanced scores (0.7790–0.8310) without sharp declines for any MR, mitigating weaknesses of individual models. Notably, debate improves stability on the very relations where single models struggle—object scaling and overlapping—demonstrating the benefit of collaborative reasoning.

These results suggest practical value for AR developers: instead of depending on a single LLM prone to blind spots, multi-agent debate provides more consistent validation of complex behaviors such as scaling, distance handling, and collision detection, leading to more robust AR testing.

VII. FUTURE WORK

This study establishes a stability-based framework for evaluating LLMs in MR identification, suggesting several avenues for extension. While we focused on predefined MRs in AR applications, future work can explore *MR discovery*, where multi-agent debate enables LLMs to propose and refine transformations across domains such as debugging, compiler optimization, and AI-generated code validation—contexts where correctness depends on behavioral transformations rather than absolute ground truth.

Beyond MR evaluation, stability scores provide a general metric for assessing LLM consistency in reasoning-based software tasks. Applying stability to benchmark AI-assisted tools (e.g., automated testing, static analysis, code reviews) could yield insights into reliability. Integrating multi-agent debate into these workflows may also improve interpretability by structuring LLM reasoning and justifications. As LLMs become more embedded in development [43], refining debate-driven methods and stability-based evaluation will be critical for building trustworthy software tools. Finally, further studies should examine how these methods influence developer decision-making [44] and the resulting quality of software products.

VIII. THREATS TO VALIDITY

A. Internal Validity

A primary internal threat concerns the selection of MRs. Our study relies on predefined transformations from prior work [15], raising the risk of missing relevant relations. To mitigate this, we reviewed prior MT literature and validated definitions with domain experts.

Another risk lies in computing stability scores, which depend on embedding-based similarity. The choice of embeddings and thresholds could bias results. We used `all-MiniLM-L6-v2`, a widely adopted model, and tuned thresholds against established NLP benchmarks to reduce this effect.

The multi-agent debate framework also introduces potential bias, as stronger models may dominate consensus [45]. To counter this, we enforced iterative self-reasoning and analyzed individual responses before integration.

Finally, reproducibility is threatened by stochastic LLM variation [46] and prompt interpretation inconsistencies [47]. We controlled randomness through fixed prompts and sampling settings, though ensemble-based methods could further improve robustness.

B. External Validity

External validity threats stem from model selection. We studied three open-source LLMs (`StarCoder-7B`,

`CodeLlama-7B`, `Gemma-7B`), which may not represent larger or proprietary models (e.g., GPT-4¹³). We chose these for computational feasibility but deliberately included both code-specialized and general-purpose models. Future work should extend to multi-modal and instruction-tuned architectures.

Domain specificity poses another limitation. Our focus on mobile AR MRs may not generalize across software domains [48]. While stability offers a generalizable metric, its effectiveness should be validated in additional SE contexts.

Rephrasing strategies also present risks. We used `FLAN-T5-XXL` for paraphrasing with embedding-based filtering to remove extreme variants, yet subtle linguistic artifacts may remain. Future work could explore adversarial or human-in-the-loop paraphrasing for stronger control.

Lastly, the practical impact of stability-based evaluation requires validation in real-world workflows. Adoption depends on integration into developer pipelines [43]. Future research should explore embedding stability scores and debate frameworks into verification processes to support developer decision-making.

IX. CONCLUSION

LLMs are increasingly used for source code analysis, yet domains like mobile AR lack ground truth for traditional evaluation. We introduced a stability-driven framework that measures consistency in MR identification across rephrasings, offering an alternative to correctness-based assessment. Our results show that code-focused models are more stable than general-purpose ones, and that multi-agent debate further improves both stability and correctness by mitigating individual weaknesses. These findings suggest that debate-based evaluation can make LLM reasoning more reliable for software testing. Future work should refine debate structures, extend this approach to other domains, and explore automated consensus to strengthen AI-assisted software engineering.

REFERENCES

- [1] T. Coignon, C. Quinton, and R. Rouvoy, "A performance study of llm-generated code on leetcode," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, pp. 79–89, 2024.
- [2] Y. Zhu and M. Pan, "Automatic code summarization: A systematic literature review," *arXiv preprint arXiv:1909.04352*, 2019.
- [3] C. Spiess, D. Gros, K. S. Pai, M. Pradel, M. Rafiqul Islam Rabin, S. Jha, P. Devanbu, and T. Ahmed, "Quality and trust in llm-generated code," *arXiv e-prints*, pp. arXiv-2402, 2024.
- [4] H. J. Kang, K. L. Aw, and D. Lo, "Detecting false alarms from automatic static analysis tools: How far are we?," in *Proceedings of the 44th International Conference on Software Engineering*, pp. 698–709, 2022.
- [5] "2024 Stack Overflow Developer Survey." <https://survey.stackoverflow.co/2024/ai#sentiment-and-usage-ai-select>.
- [6] M. Sclar, Y. Choi, Y. Tsvetkov, and A. Suhr, "Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting," *arXiv preprint arXiv:2310.11324*, 2023.
- [7] A. Mockus, "Organizational volatility and its effects on software defects," in *International Symposium on Foundations of software engineering*, pp. 117–126, 2010.

¹³<https://openai.com/index/gpt-4/>

- [8] A. E. Hassan and R. C. Holt, "The chaos of software development," in *Sixth International Workshop on Principles of Software Evolution, 2003. Proceedings.*, pp. 84–94, IEEE, 2003.
- [9] E. T. Barr, M. Harman, P. McMinin, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE transactions on software engineering*, vol. 41, no. 5, pp. 507–525, 2014.
- [10] V. Krauß, A. Boden, L. Oppermann, and R. Reiners, "Current practices, challenges, and design implications for collaborative ar/vr application development," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, (New York, NY, USA), Association for Computing Machinery, 2021.
- [11] D. B. Bose and C. Brown, "An empirical study on current practices and challenges of core ar/vr developers," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering Workshops*, ASEW '24, (New York, NY, USA), p. 233–238, Association for Computing Machinery, 2024.
- [12] N. Ashtari, A. Bunt, J. McGrenere, M. Nebeling, and P. K. Chilana, "Creating augmented and virtual reality applications: Current practices, challenges, and opportunities," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, (New York, NY, USA), p. 1–13, Association for Computing Machinery, 2020.
- [13] M. Eswaran and M. V. A. R. Bahubalendruni, "Challenges and opportunities on ar/vr technologies for manufacturing systems in the context of industry 4.0: A state of the art review," *Journal of Manufacturing Systems*, vol. 65, pp. 260–278, 2022.
- [14] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: a new approach for generating next test cases," *Technical Report HKUST-CS98-01, Department of Computer Science The Hong Kong University of Science and Technology*, 1998.
- [15] D. B. Bose, B. David-John, and C. Brown, "Optimizing ar application testing: Integrating metamorphic testing to address developer and end-user challenges," in *International Conference on Human-Computer Interaction*, pp. 20–33, Springer, 2025.
- [16] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through generative ai: Assessing the role of llms," in *Generative AI for Effective Software Development*, pp. 129–148, Springer, 2024.
- [17] D. B. Bose, "From prompts to properties: Rethinking llm code generation with property-based testing," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, FSE Companion '25*, (New York, NY, USA), p. 1660–1665, Association for Computing Machinery, 2025.
- [18] J. Liu, C. S. Xia, Y. Wang, and L. Zhang, "Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation," *Advances in Neural Information Processing Systems*, vol. 36, pp. 21558–21572, 2023.
- [19] S. Shankar, J. Zamfirescu-Pereira, B. Hartmann, A. Parameswaran, and I. Arawjo, "Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences," in *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–14, 2024.
- [20] T. Ahmed, P. Devanbu, C. Treude, and M. Pradel, "Can llms replace manual annotation of software engineering artifacts?," *arXiv preprint arXiv:2408.05534*, 2024.
- [21] A. Khatun and D. G. Brown, "Truthval: A dataset to evaluate llm truthfulness and reliability," *arXiv preprint arXiv:2406.01855*, 2024.
- [22] T. Chen, D. Huang, T. Tse, and Z. Q. Zhou, "Case studies on the selection of useful relations in metamorphic testing," 11 2004.
- [23] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic testing: A review of challenges and opportunities," *ACM Comput. Surv.*, vol. 51, jan 2018.
- [24] U. Kanewala and J. M. Bieman, "Using machine learning techniques to detect metamorphic relations for programs without test oracles," in *2013 IEEE 24th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 1–10, IEEE, 2013.
- [25] Z. Q. Zhou, S. Xiang, and T. Y. Chen, "Metamorphic testing for software quality assessment: A study of search engines," *IEEE Transactions on Software Engineering*, vol. 42, no. 3, pp. 264–284, 2015.
- [26] W. Tao, Y. Zhou, Y. Wang, W. Zhang, H. Zhang, and Y. Cheng, "Magis: Llm-based multi-agent framework for github issue resolution," *Advances in Neural Information Processing Systems*, vol. 37, pp. 51963–51993, 2024.
- [27] Z. Rasheed, M. Waseem, A. Ahmad, K.-K. Kemell, W. Xiaofeng, A. N. Duc, and P. Abrahamsson, "Can large language models serve as data analysts? a multi-agent assisted approach for qualitative data analysis," *arXiv preprint arXiv:2402.01386*, 2024.
- [28] S. Manish, "An autonomous multi-agent llm framework for agile software development," *International Journal of Trend in Scientific Research and Development*, vol. 8, no. 5, pp. 892–898, 2024.
- [29] X. Yang, Y. Wang, T. Rafi, D. Liu, X. Wang, and X. Zhang, "Towards automatic oracle prediction for ar testing: Assessing virtual object placement quality under real-world scenes," in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 717–729, 2024.
- [30] X. Qin and G. Weaver, "Utilizing generative ai for vr exploration testing: A case study," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering Workshops*, pp. 228–232, 2024.
- [31] H. Yang, H. Lu, and W. Lam, "Rephrasing invokes better generations for large language models," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop)*, pp. 8–15, 2024.
- [32] B. J. Li, J. Peña, and Y. Jung, "Vr/ar and wellbeing: The use of immersive technologies in promoting health outcomes," 2023.
- [33] A. Duque-Torres, "Selecting and constraining metamorphic relations," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pp. 212–216, 2024.
- [34] S. Hyun, M. Guo, and M. A. Babar, "Metal: Metamorphic testing framework for analyzing large-language model qualities," in *2024 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pp. 117–128, IEEE, 2024.
- [35] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al., "Scaling instruction-finetuned language models," *Journal of Machine Learning Research*, vol. 25, no. 70, pp. 1–53, 2024.
- [36] Y. Deng, W. Zhang, Z. Chen, and Q. Gu, "Rephrase and respond: Let large language models ask better questions for themselves," *arXiv preprint arXiv:2311.04205*, 2023.
- [37] E. Vergou, I. Pagouni, M. Nanos, and K. L. Keramidis, "Readability classification with wikipedia data and all-minilm embeddings," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pp. 369–380, Springer, 2023.
- [38] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.
- [39] A. Prasad, P. Hase, X. Zhou, and M. Bansal, "Grips: Gradient-free, edited instruction search for prompting large language models," *arXiv preprint arXiv:2203.07281*, 2022.
- [40] M. Musleh, M. Ouzzani, N. Tang, and A. Doan, "Coclean: Collaborative data cleaning," in *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, pp. 2757–2760, 2020.
- [41] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch, "Improving factuality and reasoning in language models through multiagent debate," in *Forty-first International Conference on Machine Learning*, 2023.
- [42] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," *arXiv preprint arXiv:2402.01680*, 2024.
- [43] D. Russo, "Navigating the complexity of generative ai adoption in software engineering," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 5, pp. 1–50, 2024.
- [44] A. Khurana, H. Subramonyam, and P. K. Chilana, "Why and when llm-based assistants can go wrong: Investigating the effectiveness of prompt-based interactions for software help-seeking," in *Proceedings of the 29th International Conference on Intelligent User Interfaces*, pp. 288–303, 2024.
- [45] A. Taubenfeld, Y. Dover, R. Reichart, and A. Goldstein, "Systematic biases in llm simulations of debates," *arXiv preprint arXiv:2402.04049*, 2024.
- [46] E. Evstafev, "The paradox of stochasticity: Limited creativity and computational decoupling in temperature-varied llm outputs of structured fictional data," *arXiv preprint arXiv:2502.08515*, 2025.
- [47] D. Garigliotti, "Explaining llm-based question answering via the self-interpretations of a model," *Advances in Interpretable Machine Learning and Artificial Intelligence, co-located with ECML-PKDD*, 2024.
- [48] P. Aggarwal, "Exploring different domains in software: A beginner's guide," *Medium*, 2023. <https://medium.com/@pranshu1902/exploring-different-domains-in-software-a-beginners-guide-f5b67e1887a8>.