Multi-dimensional Assessment of Crowdsourced Testing Reports via LLMs

Yue Wang
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
yue_wang@smail.nju.edu.cn

Yuan Zhao*
Laboratory of Data Intelligence and Interdisciplinary Innovation
Nanjing University
Nanjing, China
zhaoyuan@nju.edu.cn

Shengcheng Yu
School of Computation, Information and Technology
Technical University of Munich
Munich, Germany
shengcheng.yu@tum.de

Zhenyu Chen
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
zychen@nju.edu.cn

***Abstract*—Crowdsourced testing can markedly enhance test coverage and the discovery rate of potential defects compared to traditional software testing, making it increasingly popular. However, with the widespread use of crowdsourced testing, more and more crowdworkers from various backgrounds are submitting a large number of testing reports to crowdsourced testing platforms, which hinders developers from effectively reviewing the reports. Facing a vast amount of reports with varying quality, manual review is not only time-consuming and labor-intensive but also increases costs. Therefore, how to efficiently review crowdsourced testing reports has become a major challenge. To address this challenge, we propose a multi-dimensional assessment method for crowdsourced testing reports based on large language models. This method not only inherits the textuality dimension widely used in traditional report assessment but also innovatively introduces two new dimensions: adequacy and competitiveness. It comprehensively assesses the quality of crowdsourced testing reports from multiple perspectives, aiming to better screen for high-quality crowdsourced testing reports. Through experimental analysis conducted on three different applications, we have proven the consistency of our method with human raters across various dimensions, and we have also observed an enhancement in the efficiency of report assessment.**

***Index Terms*—Crowdsourced Testing Report Assessment, LLM-as-a-Judge, Large Language Models, Autograding**

I. INTRODUCTION

With the increasing scale and complexity of software systems, traditional software testing methods are facing challenges in achieving sufficient test coverage and effective defect detection. To address these limitations, crowdsourced testing has emerged as an effective approach to enhance test coverage and defect discovery by leveraging the collective intelligence of distributed testers [1], [2]. As a critical outcome of the crowdsourced testing process, the quality of crowdsourced testing reports, which are detailed records of test cases and discovered defects, directly determines the effectiveness of the entire testing process [3]–[5]. However, the explosive growth in report volume and varying quality across diverse tester backgrounds poses significant challenges for effective report assessment and filtering [6], [7].

*Yuan Zhao is the corresponding author.

Traditional assessment approaches face multiple limitations. Manual assessment suffers from subjectivity, inconsistency, and scalability issues when handling large volumes of reports [8]–[10]. Existing automated methods, while improving efficiency [11]–[13], often lack comprehensive assessment dimensions and effective feedback mechanisms. Specifically, current approaches overlook guidance from test requirement documentation for functional coverage assessment and neglect the competitive value of unique defect discoveries in crowdsourced environments [14].

The emergence of large language models (LLMs) and the LLM-as-a-Judge (LLM-J) paradigm [15] offers new opportunities for automated assessment. Unlike traditional rule-based approaches, LLM-J leverages sophisticated natural language understanding to perform nuanced assessments with detailed rationales [16]–[18]. Recent studies demonstrate LLM-J's effectiveness across various software engineering tasks, including code generation evaluation [19], [20], code summarization assessment [21], and automated grading [22]–[25]. The paradigm's ability to process heterogeneous information and generate structured feedback makes it particularly suitable for crowdsourced testing report assessment [26].

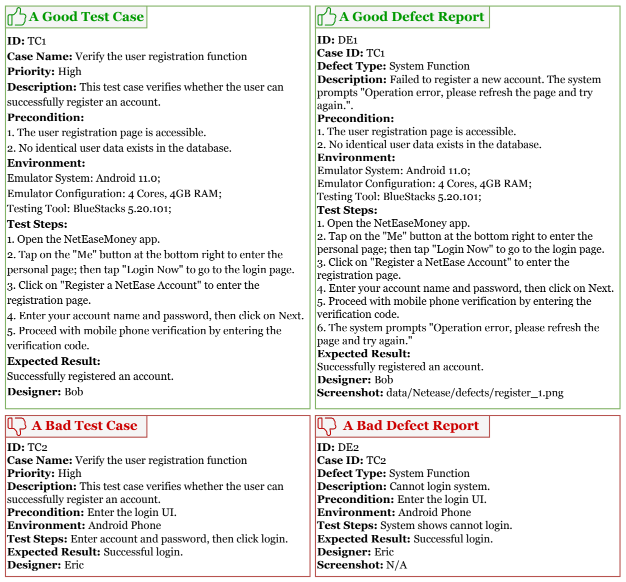
Inspired by the promising potential of LLM-J and motivated by the critical need for comprehensive assessment of crowdsourced testing reports, this paper proposes an LLM-based framework for crowdsourced testing reports assessment. Our approach introduces three complementary assessment dimensions—textuality, adequacy, and competitiveness—implemented through specialized agents. The textuality agent employs dual LLM assessment with checklist-based assessment across morphological, relational, and analytical categories. The adequacy agent constructs hierarchical requirement trees and uses semantic analysis to assess functional coverage. The competitiveness agent leverages LLM-based hierarchical clustering to assess defect discovery value through rarity-based scoring. Unlike traditional approaches that assess reports in isolation, our framework considers the competitive context of crowdsourced testing environments and provides comprehensive coverage analysis against requirement specifications. Our framework operates through parallel processing, aggregating specialized assessments into comprehensive quality scores with detailed feedback. Experiments across three crowdsourced testing tasks demonstrate substantial agreement with human assessments (QWK>70%) while achieving 5 to 90-fold efficiency improvements.

The main contributions of this paper include:

- We propose three novel assessment dimensions (textual quality, coverage adequacy, and competitive value) that comprehensively assess the quality of crowdsourced testing reports from multiple perspectives.
- We develop an LLM-based framework employing specialized assessment agents to automatically assess crowdsourced testing reports across the proposed dimensions.
- We conduct comprehensive experimental validation demonstrating high consistency with human expert assessments while significantly improving assessment efficiency compared to manual review.

II. BACKGROUND AND MOTIVATION

A. Crowdsourced Testing Report



A Good Test Case

ID: TC1
Case Name: Verify the user registration function
Priority: High
Description: This test case verifies whether the user can successfully register an account.
Precondition:
1. The user registration page is accessible.
2. No identical user data exists in the database.
Environment:
Emulator System: Android 11.0;
Emulator Configuration: 4 Cores, 4GB RAM;
Testing Tool: BlueStacks 5.20.101;
Test Steps:
1. Open the NetEaseMoney app.
2. Tap on the "Me" button at the bottom right to enter the personal page; then tap "Login Now" to go to the login page.
3. Click on "Register a NetEase Account" to enter the registration page.
4. Enter your account name and password, then click on Next.
5. Proceed with mobile phone verification by entering the verification code.
Expected Result:
Successfully registered an account.
Designer: Bob

A Good Defect Report

ID: DE1
Case ID: TC1
Defect Type: System Function
Description: Failed to register a new account. The system prompts "Operation error, please refresh the page and try again.".
Precondition:
1. The user registration page is accessible.
2. No identical user data exists in the database.
Environment:
Emulator System: Android 11.0;
Emulator Configuration: 4 Cores, 4GB RAM;
Testing Tool: BlueStacks 5.20.101;
Test Steps:
1. Open the NetEaseMoney app.
2. Tap on the "Me" button at the bottom right to enter the personal page; then tap "Login Now" to go to the login page.
3. Click on "Register a NetEase Account" to enter the registration page.
4. Enter your account name and password, then click on Next.
5. Proceed with mobile phone verification by entering the verification code.
6. The system prompts "Operation error, please refresh the page and try again."
Expected Result:
Successfully registered an account.
Designer: Bob
Screenshot: data/Netease/defects/register_1.png

A Bad Test Case

ID: TC2
Case Name: Verify the user registration function
Priority: High
Description: This test case verifies whether the user can successfully register an account.
Precondition: Enter the login UI.
Environment: Android Phone
Test Steps: Enter account and password, then click login.
Expected Result: Successful login.
Designer: Eric

A Bad Defect Report

ID: DE2
Case ID: TC2
Defect Type: System Function
Description: Cannot login system.
Precondition: Enter the login UI.
Environment: Android Phone
Test Steps: System shows cannot login.
Expected Result: Successful login.
Designer: Eric
Screenshot: N/A

Fig. 1: Example of Crowdsourced Testing Report.

Crowdsourced testing reports are essential deliverables generated by testers through crowdsourcing platforms [2]. Each report consists of two main components: test cases designed by testers and defects discovered during test execution. Although report formats may differ across platforms, certain core fields remain consistent, consisting mainly of testing environment specifications, step-by-step test procedures, and actual results [3], [6], [7]. For this study, we use a report template from a crowdsourced testing platform in China for quality assessment. Figure 1 illustrates the contrast between well-written and poorly-written test case and defect report through examples.

As shown on the left side of Figure 1, **TC1** exemplifies a high-quality test case with comprehensive attributes, including a descriptive case name, appropriate priority settings for the login functionality, well-defined preconditions, and clearly specified expected results. Its test steps are logically structured and easy to execute. In contrast, **TC2** represents a low-quality test case characterized by ambiguous descriptions, misassigned priority levels, and insufficient environmental specifications, which significantly hampers test reproducibility.

The right side of Figure 1 presents two contrasting examples of defect reports. While many fields are common to both test cases and defect reports, defect reports include two distinctive fields: *Case ID* for traceability to the corresponding test case, and *Type* for defect classification. **DE1** demonstrates an effective defect report with clear descriptions, logical structure, and reproducible steps. In contrast, **DE2** exemplifies a problematic report characterized by ambiguous descriptions that impede developers' understanding and resolution of the issue.

B. Motivation

Effective assessment of crowdsourced testing reports is crucial for identifying high-quality submissions and optimizing testing resources [14]. However, manual assessment faces significant challenges: large report volumes with varying quality [6], complex multi-modal content requiring careful examination [3], [27], and inconsistent reporting styles across diverse crowdworkers, leading to time-consuming and subjective assessment processes [7].

The LLM-as-a-Judge (LLM-J) paradigm offers promising solutions by leveraging sophisticated natural language understanding for scalable and consistent assessment [15]. Recent studies demonstrate LLM-J's effectiveness across software engineering tasks, achieving high consistency with human evaluators while significantly reducing assessment time and cost [19]–[21].

However, existing LLM-based approaches like LLMPrior [28] underutilize LLM capabilities by focusing primarily on ranking rather than comprehensive quality assessment. They lack multi-dimensional assessment perspectives, detailed reasoning mechanisms, and actionable feedback generation—missing opportunities to fully exploit LLM-J's potential for both assessment and guidance.

This motivates our comprehensive LLM-based framework that addresses these limitations through: (1) multi-dimensional assessment covering textual quality, functional adequacy, and competitive value, (2) detailed reasoning and transparency in assessment processes, and (3) actionable feedback to guide continuous improvement in crowdsourced testing report quality.

III. ASSESSMENT DIMENSIONS

To assess multiple aspects of report quality, we propose three distinct assessment dimensions based on current research practices and the characteristics of LLM-as-a-Judge: textuality dimension, adequacy dimension, and competitiveness dimension. These dimensions assess report quality from different perspectives. The **textuality dimension** focuses on the normativity, readability, and logic of individual test cases

TABLE I: The Relationship Between Desirable Properties and Quantitative Indicators

Category	Indicator	Desirable Property*				
		A[a]	**C**[b]	**Cp**[c]	**U**[d]	**R**[e]
Morphological	Text Length	●	×		●	
	Readability				×	
	Punctuation		×		●	
Relational	Itemization		●	●	×	×
	Environment			×		×
	Preconditions			×		×
	Expected Results			×		×
	Extra Information			×		
	Screenshot			×	×	×
Analytical	Interface Elements		●		×	×
	User Behavior			●	×	×
	Defect Response			●		×

* Desirable Properties are abbreviated in the header. [a] **A**: Atomicity; [b] **C**: Conciseness; [c] **Cp**: Completeness; [d] **U**: Understandability; [e] **R**: Reproducibility. '†' '×' represents a direct influence and '●' denotes an indirect influence between the indicator and the desirable property.

or defects. The **adequacy dimension** assesses how well the report covers the required functionalities specified in the test requirement documents. The **competitiveness dimension** primarily focuses on defect reports of crowdsourced testing reports, which considers each crowdsourced testing report's ability to uncover system defects and the novelty of the defects found. By combining these dimensions, we can provide a more comprehensive and accurate assessment of the quality of crowdsourced testing reports.

A. Textuality Dimension

The textuality dimension focuses on assessing the quality of individual test cases and defect reports at a fine-grained level. Drawing inspiration from Chen et al.'s work on TERQAF [11], we identified five key properties that characterize high-quality test reports: atomicity, completeness, conciseness, understandability, and reproducibility. However, in the definition of TERQAF, these five properties are considered as desired properties, which cannot be directly assessed and need to be further refined into quantitative indicators. Based on existing research in indicator taxonomy and according to direct and indirect relationships, TERQAF defines four categories of quantitative assessment indicators. As shown in Table I, to better adapt to the characteristics of LLM-as-a-Judge, we streamlined these four categories of assessment indicators and finally derived our textuality dimension assessment indicators.

Table I illustrates the relationship between different quantitative indicators and their corresponding desirable properties, which are abbreviated as **A**tomicity, **C**onciseness, **C**om**p**leteness, **U**nderstandability, and **R**eproducibility. The indicators in the table are categorized into three main types: Morphological indicators (e.g., text length, readability, punctuation) primarily focus on the surface features of the text; Relational indicators (e.g., itemization, environment, preconditions) emphasize the structured features and content format standardization of the report; and Analytical indicators (e.g., interface elements, user behavior, defect response) are used for in-depth analysis of test steps. The table uses the symbol '●' to denote an indirect influence of an indicator on a desirable property, while '× ' signifies a direct influence, thereby clearly demonstrating how each indicator specifically impacts the various desired qualities.

B. Adequacy Dimension

While the textuality dimension assesses individual report quality, it fails to address how well submitted reports cover required functionalities. This motivates our adequacy dimension, which assesses the extent to which test reports align with and cover specified testing requirements.

Test requirement documents serve as the authoritative specification in crowdsourced testing scenarios [6], yet existing approaches often overlook this crucial guidance. Test coverage measures how thoroughly test cases examine software functionality [29], with high coverage increasing the likelihood of detecting defects.

The adequacy dimension assesses how well each report contributes to overall testing coverage by considering functional point coverage, requirement traceability, and alignment between test cases and testing objectives. By incorporating requirement documents as reference standards, this dimension ensures high-quality reports are strategically valuable for comprehensive testing coverage.

Formally, we define the adequacy score for a crowdsourced testing report R_i as:

$$\text{Adequacy}(R_i) = \frac{|\text{CoveredRequirements}(R_i)|}{|\text{TotalRequirements}|} \times 100\% \quad (1)$$

C. Competitiveness Dimension

While textuality and adequacy dimensions assess individual report quality and requirement coverage, they do not capture the competitive value of defect discovery in crowdsourced testing. Multiple testers often uncover overlapping defects, necessitating our competitiveness dimension, which assesses each report's unique contribution to defect discovery.

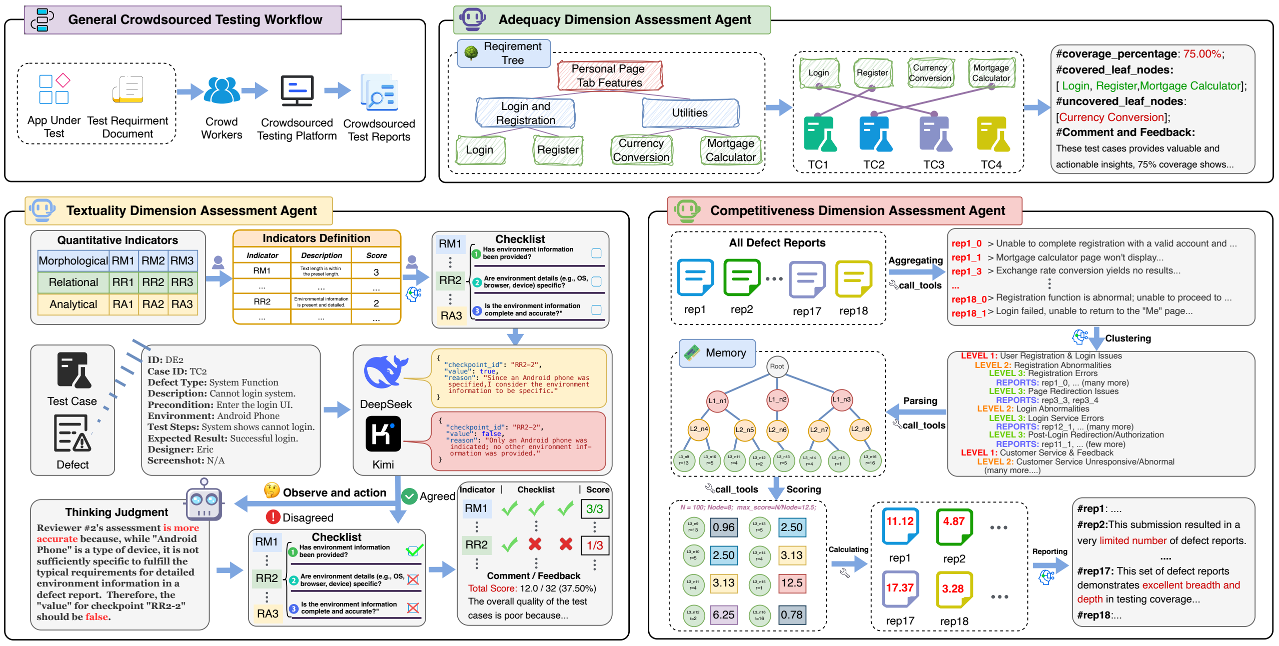
The competitiveness dimension focuses on defect reports, recognizing that testing's ultimate goal is identifying previously unknown software defects [1]. It addresses: (1) How effectively does a report contribute to uncovering hidden system defects? and (2) How novel are the discovered defects compared to those found by other testers?

Unlike traditional approaches assessing reports in isolation, this dimension requires comparative analysis across all reports within a testing task. It considers defect severity, uniqueness, discovery difficulty, and potential system impact. Reports uncovering critical, previously unknown defects receive higher scores, while duplicate or trivial issues receive lower scores.

Formally, we define the competitiveness score for a crowdsourced testing report R_i as:

$$\text{Competitiveness}(R_i) = \sum_{d \in D_i} \text{Rarity}(d, \mathcal{D}) \quad (2)$$

where D_i represents defect reports in testing report R_i, d denotes an individual defect report, and $\mathcal{D}$ represents all defect reports in the testing task. The rarity function $\text{Rarity}(d, \mathcal{D}) \in [0, +\infty)$ measures defect d's scarcity relative to all other defects, with higher values indicating rarer discoveries.



General Crowdsourced Testing Workflow
App Under Test
Test Requirment Document
Crowd Workers
Crowdsourced Testing Platform
Crowdsourced Test Reports
Adequacy Dimension Assessment Agent
Reqirement Tree
Personal Page Tab Features
Login and Registration
Utilities
Login
Register
Currency Conversion
Mortgage Calculator
TC1
TC2
TC3
TC4
#coverage_percentage: 75.00%;
#covered_leaf_nodes:
[Login, Register,Mortgage Calculator];
#uncovered_leaf_nodes:
[Currency Conversion];
#Comment and Feedback:
These test cases provides valuable and actionable insights, 75% coverage shows...
Textuality Dimension Assessment Agent
Quantitative Indicators
Indicators Definition
Checklist
DeepSeek
Kimi
Test Case
Defect
Thinking Judgment
Observe and action
Agreed
Disagreed
Comment / Feedback
Total Score: 12.0 / 32 (37.50%)
The overall quality of the test cases is poor because...
Competitiveness Dimension Assessment Agent
All Defect Reports
rep1
rep2
rep17
rep18
Aggregating
call_tools
Clustering
Memory
Parsing
Scoring
Calculating
Reporting
#rep1: ....
#rep2:This submission resulted in a very limited number of defect reports.
#rep17: This set of defect reports demonstrates excellent breadth and depth in testing coverage...
#rep18:...

Fig. 2: Framework of the proposed method, which consists of three assessment agents.

IV. FRAMEWORK

A. Overview

Figure 2 presents our comprehensive framework for multi-dimensional assessment of crowdsourced testing reports using LLM-based agents. The framework follows a systematic workflow that transforms the traditional manual review process into an automated, multi-agent assessment system.

The assessment process begins with the **General Crowdsourced Testing Workflow**, where the app under test and test requirement documents are provided to crowdworkers through a crowdsourced testing platform. The crowdworkers conduct testing activities and generate crowdsourced test reports, which serve as the primary input for our assessment framework.

Our framework consists of three specialized LLM-based agents working in coordination:

Textuality Dimension Assessment Agent assesses the individual quality of test cases and defect reports using quantitative indicators across three categories: morphological (e.g., text length, readability), relational (e.g., itemization, environment specifications), and analytical (e.g., interface elements, user behavior). This agent generates detailed checklists based on the textual quality assessment criteria defined in Section III-A. To ensure assessment consistency and stability, the agent employs two LLMs to independently examine the checklist. When disagreements arise between the two LLMs, the agent performs a "thinking and judgment" process to reconcile the differences and produce a unified checklist result. Finally, the agent calculates a score based on the unified checklist.

Adequacy Dimension Assessment Agent operates in three sequential steps. First, it extracts a hierarchical requirement tree from the test requirement document, where leaf nodes represent specific functional points. Second, it maps test cases from each crowdsourced testing report to these functional points, establishing connections between test cases and the requirements they address. Finally, based on this mapping, the agent identifies covered and uncovered functional points, calculates coverage percentages, and generates actionable feedback highlighting areas lacking sufficient test coverage.

Competitiveness Dimension Assessment Agent first aggregates all defect reports submitted by testers within the same testing task, then constructs a hierarchical defect clustering tree to organize similar defects. The agent calculates the competitiveness score for each defect based on the number of reports contained in the leaf nodes of the clustering tree, where defects appearing in fewer reports receive higher competitiveness scores due to their rarity. Finally, the agent computes the overall competitiveness score for each crowdsourced testing report by aggregating the competitiveness scores of all defects contained within that report.

The assessment workflow operates as follows: First, the defect clustering agent processes all submitted reports and requirement documents to establish the competitiveness context. Then, the three assessment agents work in parallel, each assessing reports from their respective dimensions. The textuality agent focuses on individual report quality, the adequacy agent assesses requirement coverage using the requirement tree, and the competitiveness agent assesses the uniqueness and value of discovered defects. Finally, the framework aggregates scores from all dimensions and generates detailed feedback reports.

This architecture provides several advantages: (1) *Comprehensive assessment* - each dimension addresses different

TABLE II: Quantitative Indicators Defination

Cat.	Indicator	ID	Description	Score
Morph.	Length	RM1	Text length is within the preset range.	3
	Readability	RM2	Concise, fluent, and easy to understand.	2
	Punctuation	RM3	Punctuation is used correctly.	3
Relat.	Itemization	RR1	Numbered operational steps.	5
	Environment	RR2	Detailed environment information.	3
	Precondition	RR3	Preconditions are described and complete.	2
	Expected Results	RR4	Expected results are filled in standardly.	2
	Extra Information	RR5	All other important fields are filled.	2
	Screenshot*	RR6	Screenshot provided for defects.	3
Analyt.	User Interface	RA1	Clear and enough UI-element description.	5
	User Behavior	RA2	Interactive behavior explained.	5
	Defect Feedback*	RA3	Defect feedback included.	3

*Applicable only to defect reports.

aspects of report quality; (2) *Scalability* - agents can process multiple dimensions simultaneously; (3) *Consistency* - LLM-based assessment reduces subjective biases inherent in manual reviews; and (4) *Actionable feedback* - detailed assessments help testers improve future submissions.

B. Textuality Dimension Assessment

The Textuality Dimension Assessment Agent focuses on assessing the individual quality of test cases and defect reports at a fine-grained level, which implements a comprehensive assessment framework based on five key desirable properties. However, since these properties cannot be directly assessed, the agent operationalizes them through quantitative indicators organized into three distinct categories as shown in Table I. To facilitate LLM-based assessment, we developed an indicator definition table that assigns identifiers and definitions to each indicator. Following the practice of Zhang et al.'s work [13], we defined score upper limits for each indicator based on the direct and indirect relationships between desirable properties and quantitative indicators, as detailed in Table II.

Checklist Generation addresses a fundamental limitation of LLM-as-a-Judge approaches. It is well understood that LLM-as-a-Judge cannot provide accurate scoring in continuous ranges, usually only output integer scores [16]. Therefore, to ensure scoring stability, we follow the design principles of RocketEval [30] and use checklists to guide LLM judgments. Instead of asking LLMs to directly assign numerical scores, we decompose the assessment task into a series of binary checkpoint assessments, where each quality indicator corresponds to multiple checkpoints.

The agent generates assessment checklists based on Table II. For each quantitative indicator, the agent creates specific checkpoints according to the indicator's description, where the number of checkpoints corresponds to the assigned score value (e.g., a 3-point indicator generates 3 binary checkpoints, a 5-point indicator generates 5 checkpoints). To ensure the quality of textuality dimension assessment, the generated checklists may require manual review and refinement by domain experts to validate that each checkpoint accurately captures the intended quality criterion and maintains objectivity in assessment.

Dual LLM Assessment and Disagreement Resolution mechanisms are also adopted to further ensure scoring stability. It implements a sophisticated consistency assurance mechanism where two independent LLMs simultaneously assess the same report against the generated checklist. Each LLM receives an identical assessment prompt containing the formatted report content and the applicable checklist with detailed checkpoint descriptions. The models perform independent assessment, determining for each checkpoint whether the report meets the specified quality criteria and providing reasoning for their judgments. When the two LLM assessments produce conflicting judgments on specific checkpoints, the agent itself performs a "thinking and judgment" process where it analyzes the disagreement context, reviews the evidence and reasoning presented by both initial assessments, and makes a final determination based on the quality criteria.

This mechanism tackles the inherent variability in LLM responses and potential biases that might arise from single-model assessment. By using two independent assessments and integrating a conflict resolution mechanism, the agent can systematically pinpoint and address areas of uncertainty or inconsistency in applying assessment criteria. This boosts the overall reliability of the assessment process. Furthermore, the agent provides detailed justifications for its final decisions, creating transparency in the conflict resolution process and upholding the integrity of the assessment framework.

Score Calculation synthesizes the unified checklist results into quantitative quality scores. The scoring mechanism directly sums up all the checkpoints that pass (i.e., those marked as True) in the checklist to obtain the final textuality dimension score. For each test case or defect report, the final textuality dimension score is calculated using the formula:

$$TextualityScore = \sum_{i=1}^{N} \sum_{j=1}^{|C_i|} \mathbb{I}(c_{i,j} = \text{True}) \tag{3}$$

where N represents the total number of indicators, C_i denotes the set of checkpoints for indicator i, and $\mathbb{I}(c_{i,j} = \text{True})$ is an indicator function that equals 1 when checkpoint j of indicator i passes and 0 otherwise. This approach provides a cumulative score reflecting the total number of quality criteria satisfied by the report.

The assessment results include not only numerical scores but also qualitative feedback derived from the LLM reasoning processes, enabling testers to understand specific strengths and weaknesses in their submissions and facilitating continuous improvement in crowdsourced testing report quality.

C. Adequacy Dimension Assessment

The Adequacy Dimension Assessment Agent addresses a fundamental challenge in crowdsourced testing: assessing how comprehensively test reports cover the specified functional requirements. Unlike approaches that assess reports in isolation, this agent ensures that high-quality reports are not only well-written but also strategically valuable for achieving comprehensive testing coverage as defined in requirement documents. The core principle is that test adequacy should be measured against explicit functional requirements, ensuring that crowdsourced testing efforts collectively address all critical system functionalities rather than focusing on redundant or less important areas.

Requirement Tree Construction systematically transforms unstructured requirement documents into hierarchical trees where leaf nodes represent atomic functional points. This phase operates through three sub-steps: First, the agent performs structural analysis using LLM-based document understanding to identify main functional modules, hierarchical relationships, and logical groupings within the requirement text. The LLM analyzes document structure, utilizing explicit numbering systems when available, or inferring logical groupings based on semantic relationships when the structure is implicit. Second, the agent implements requirement decomposition to break down complex, compound requirements into atomic functional points. The LLM identifies requirements that describe multiple independent functionalities connected by conjunctions such as "and," "or," or "as well as." For example, a requirement stating "users can login and register accounts" would be decomposed into two atomic functional points: "users can login to the system" and "users can register new accounts." This decomposition ensures that each leaf node represents a single, independently testable functionality. Third, the agent constructs the hierarchical requirement tree by establishing parent-child relationships between decomposed nodes. Each node in the tree contains a unique identifier, functional description, parent reference, children list, and a path from root to the current node. Leaf nodes specifically represent atomic functional points that can be directly mapped to test cases, while internal nodes serve as organizational categories.

Test Case Mapping establishes connections between individual test cases and the atomic functional points in the requirement tree. For each test case, the agent extracts comprehensive information including test case ID, name, description, test steps, and expected results. This information is then formatted into a structured prompt that presents both the test case details and the complete list of available leaf nodes with their paths and descriptions. The agent employs sophisticated LLM-based semantic analysis to determine which functional points each test case addresses. The LLM analyzes the test case content, considering not only explicit functional references but also implicit coverage through test steps and validation criteria. The mapping process produces a list of requirement paths that the test case covers, ensuring that each test case contributes to specific functional coverage rather than generic testing efforts. This semantic mapping approach is particularly effective because it can identify coverage relationships that might be missed by keyword-based or rule-based approaches.

Coverage Assessment synthesizes the mapping results to produce comprehensive adequacy assessments for each crowdsourced testing report. The core goal is to assess how well the collection of test cases within a single crowdsourced testing report covers the specified functional requirements. For each crowdsourced testing report R, the agent calculates the functional coverage percentage using the formula:

$$\text{Coverage}(R) = \frac{|\bigcup_{T_i \in R} \text{CoveredLeafNodes}(T_i)|}{|\text{LeafNodes}|} \times 100\% \tag{4}$$

where R represents a crowdsourced testing report containing multiple test cases, T_i denotes an individual test case within report R, $\text{CoveredLeafNodes}(T_i)$ represents the set of atomic functional points (leaf nodes) covered by test case T_i, and $|\text{LeafNodes}|$ denotes the total number of atomic functional points in the requirement tree. The union operation ensures that if multiple test cases within the same report cover the same functional point, it is counted only once.

The assessment phase generates detailed coverage reports for each crowdsourced testing report that include: (1) Overall coverage percentage indicating the proportion of functional requirements addressed by all test cases within the report, (2) Lists of covered functional points with their corresponding test cases, providing traceability between requirements and validation efforts, and (3) Identification of uncovered functional points, highlighting gaps in testing coverage that require additional test cases.

For individual report assessment, the agent calculates adequacy scores based on the coverage percentage achieved by each report. Reports with higher functional coverage receive higher adequacy scores, as they demonstrate more comprehensive testing of the specified requirements. This scoring mechanism encourages testers to design diverse test cases that collectively address a broader range of functional requirements rather than focusing on redundant testing scenarios. The agent also generates actionable feedback highlighting specific uncovered functional areas, suggesting potential test scenarios for gaps, and providing requirement traceability information to help testers understand the broader functional context of their testing efforts. This feedback mechanism not only supports immediate assessment but also guides future testing activities toward more comprehensive coverage of the software's functional requirements.

D. Competitiveness Dimension Assessment

The Competitiveness Dimension Assessment Agent addresses the unique challenge of assessing defect discovery value in crowdsourced testing environments where multiple testers independently work on the same software. Unlike traditional assessment approaches that assess reports in isolation, this agent performs comparative analysis to identify the rarity and uniqueness of discovered defects, thereby promoting innovation and thoroughness in testing strategies. The core principle underlying this assessment is that defects discovered by fewer testers are more valuable and should receive higher competitiveness scores, as they represent unique insights or innovative testing approaches that uncover hidden software issues.

Defect Aggregation collects all defect reports submitted by different testers within the same testing task. Each defect report contains essential information including defect ID, title, reproduction steps, and actual results. This comprehensive aggregation ensures that the subsequent clustering process has access to the complete defect landscape for comparative analysis.

LLM-based Hierarchical Clustering leverages the natural language understanding capabilities of large language models to automatically organize similar defects into a hierarchical clustering tree. The agent constructs a detailed prompt that instructs the LLM to analyze defect reports based on their root causes, observable symptoms, and functional impact. The LLM generates a multi-level tree structure where similar defects are grouped together, with the tree depth typically ranging from 3 to 4 levels to ensure fine-grained differentiation. This approach is particularly effective because it can identify semantic similarities that traditional clustering algorithms might miss, such as defects with different surface manifestations but identical underlying causes.

The clustering prompt specifically instructs the LLM to pursue fine-grained categorization, considering factors such as specific error messages, user operation sequences, affected modules, preconditions, and symptom variations. Each unique defect variant ideally becomes a separate leaf node or small cluster, ensuring that genuinely different issues are distinguished while grouping truly similar reports.

Defect Scoring implements a rarity-based scoring mechanism where defects appearing in fewer reports receive higher competitiveness scores. For each leaf node in the clustering tree, the agent calculates the competitiveness score using the formula:

$$Score(d) = \frac{S_{max}}{|N_d|} \tag{5}$$

where S_{max} represents the maximum possible score, N_d denotes the leaf node containing defect d, and $|N_d|$ represents the number of defect reports in that node. This inverse relationship ensures that unique defects discovered by only one or few testers receive significantly higher scores than commonly reported issues.

Report Score Aggregation computes the final competitiveness score for each tester's report by summing the scores of all unique defect categories they contributed to. Importantly, if a tester submits multiple defects that belong to the same leaf node in the clustering tree, they receive the score for that node only once, preventing score inflation from reporting multiple instances of the same underlying issue.

This assessment methodology encourages testers to explore diverse testing scenarios, discover edge cases, and employ innovative testing strategies. Reports that uncover rare, previously unknown defects receive higher competitiveness scores, while those reporting common or duplicate issues receive lower scores accordingly. The approach maintains fairness by recognizing genuine contributions while discouraging redundant submissions, ultimately improving the overall effectiveness and quality of crowdsourced testing efforts.

V. EXPERIMENTS

To comprehensively evaluate the effectiveness of our proposed method, we conduct extensive experiments addressing three key research questions.

RQ1: Human Evaluator Subjectivity and Consistency: How significant are the subjectivity and inconsistency issues in manual assessment of crowdsourced testing report quality by human evaluators?

RQ2: Consistency with Human Evaluators: How consistent is our LLM-based multi-dimensional assessment method compared to human expert assessments across different assessment dimensions?

RQ3: Assessment Efficiency: How significantly does our automated LLM-based assessment method improve the efficiency of crowdsourced testing report review compared to manual assessment processes?

A. Experimental Setup

1) Dataset: We collected crowdsourced testing reports from three different mobile applications through a leading crowdsourced testing platform in China. The applications represent diverse domains: a financial management app, an online community platform, and a utility application. For each application, we gathered the corresponding test requirement documents and collected all submitted crowdsourced testing reports from participating crowdworkers. The dataset comprises crowdsourced testing reports from 64 different crowdworkers, resulting in 1,588 individual test cases and 235 defect reports.

2) LLM Configuration: Considering API calling cost factors, we made strategic choices in our LLM configuration. Since the textuality dimension assessment involves high token consumption due to dual LLM assessment mechanisms and the amonut of test cases and defect reports, we selected cost-effective models *DeepSeek-V3-0324* and *kimi-latest-8k* as the assessment models for this component. To ensure precision in clustering and other critical operations, all three assessment agents utilize *gpt-4o-2024-05-13* as their base LLM, including the think and judge component in textuality assessment for resolving disagreements between the dual assessment models. This configuration balances cost efficiency with assessment accuracy across different operational requirements. For all models, we set the temperature to 0.1 and top-p to 0.9 to minimize output randomness and ensure consistent, reproducible assessment results while maintaining sufficient diversity in model responses.

3) Manual Assessment: To establish reliable baselines for evaluating our automated assessment method, we conducted comprehensive manual assessments across all three dimensions using six experienced software testing experts. Each expert possessed at least 5 years of industrial testing experience and demonstrated proficiency in crowdsourced testing assessment. The manual assessment process was designed to ensure consistency and objectivity while providing ground truth labels for our evaluation.

For the textuality dimension, human evaluators independently scored individual test cases and defect reports following the same scoring criteria outlined in Table II. Each evaluator assessed reports based on the quantitative indicators across morphological, relational, and analytical categories, assigning scores according to the predefined scoring rubric. To ensure consistency, evaluators underwent training sessions where assessment criteria were explained and sample assessments were conducted collectively until consensus was reached on scoring standards.

To ensure consistency in functional point identification and coverage assessment, we provided evaluators with the requirement tree generated by our adequacy assessment agent, containing all extracted functional points. Human evaluators analyzed the test cases within each crowdsourced testing report to determine the number of functional points covered by that report. This approach eliminated potential discrepancies arising from different interpretations of requirement documents while focusing the evaluation on coverage assessment accuracy.

For the competitiveness dimension, we supplied evaluators with the unique defect set derived from our hierarchical clustering tree. Human evaluators reviewed each crowdsourced testing report against this standardized defect set, recording the number of unique defect categories mentioned in each report. This methodology ensured that all evaluators worked with the same defect taxonomy, enabling consistent assessment of each report's contribution to defect discovery while maintaining the competitiveness scoring principle.

4) Evaluation Metrics: To comprehensively assess our method's performance, we employ multiple evaluation metrics:

- **Single-measure Intra-class Correlation Coefficient (ICC(2,1))** [31]: Estimates the reliability of an *individual* judge's score by quantifying the proportion of variance attributable to true differences between reports versus rater-specific error.
- **Average-measure Intra-class Correlation Coefficient (ICC(2,k))** [31]: Estimates the reliability of the *mean* score obtained by aggregating k judges (here, $k = 6$); higher values reflect the error-reducing effect of averaging across raters.
- **Quadratic Weighted Kappa (κ)** [32]: Primary metric for measuring agreement between LLM assessments and human expert assessments.
- **Spearman's Rank Correlation (ρ)** [33]: Measures monotonic relationship between ranking orders produced by automated and manual assessments.
- **Kendall-Tau (τ)** [34]: Assesses the ordinal association between automated and human assessment rankings.

B. Experimental Results

1) RQ1: Human Evaluator Subjectivity and Consistency: As shown in Table III, the inter-rater reliability analysis reveals significant variability in individual human evaluator consistency across the three datasets. For App 1 (18 reports), App 2 (23 reports), and App 3 (23 reports), the single-measure intra-class correlation coefficient ICC(2,1) ranges from 0.49 to 0.63, which falls within the "moderate" agreement range according to the conventional Landis–Koch scale[1]. This indicates that approximately 37%–51% of the observed score variance in individual crowdsourced testing report assessments stems from rater-specific bias and random error, rather than genuine differences in textual quality.

TABLE III: Inter-rater reliability and standard error of measurement analysis of human raters across three datasets.

Dataset	n	k	ICC(2,1)	ICC(2,k)	$\text{SEM}_{\text{single}}$	SEM_{mean}
App 1	18	6	0.49	0.85	±1.62	±0.66
App 2	23	6	0.52	0.87	±2.42	±0.99
App 3	23	6	0.63	0.91	±2.00	±0.82

This substantial individual variability persisted despite comprehensive evaluator training with predefined scoring rubrics, suggesting that the observed rater-specific bias and random error have deeper underlying causes. The persistent subjectivity stems from the fundamental nature of human assessment as a complex cognitive process influenced by multiple factors that are difficult to standardize through training alone. The primary contributing factors to this variance include: (1) *Individual experience and cognitive biases*, where evaluators' diverse professional backgrounds and knowledge structures lead to different interpretations of the same content; (2) *Psychological and environmental factors*, including order effects when reviewing multiple reports sequentially, fatigue-induced attention decline; and (3) *Reference frame variations*, where evaluators inevitably compare current reports against their previous assessment experiences, causing deviations from the standardized reference framework established during training.

Fortunately, this individual-level subjectivity can be substantially mitigated through aggregation. The reliability improves dramatically when averaging scores across multiple evaluators, as demonstrated in Table III, where the average-measure ICC(2,6) achieves values between 0.85–0.91, representing "good" to "excellent" reliability. Correspondingly, the standard error of measurement is reduced by more than half when using averaged scores (e.g., decreasing from ±2.42 to ±0.99 for App 2). This consistent pattern across all three applications indicates that while individual judges share a common understanding of quality criteria, their individual assessments exhibit substantial divergence. The aggregation of multiple evaluators' assessments effectively mitigates the aforementioned sources of subjectivity, resulting in more reliable and consistent quality assessments.

Answer to RQ1: Manual assessment by human evaluators exhibits significant subjectivity and inconsistency issues, with individual rater reliability remaining only at moderate levels (ICC(2,1) = 0.49–0.63). This substantial variability in human judgment highlights the inherent limitations of manual assessment approaches and underscores the need for more consistent automated assessment methods.

[1]According to the conventional Landis–Koch scale, values between 0.41 and 0.60 indicate moderate agreement, whereas 0.61–0.80 is substantial.

2) RQ2: Consistency with Human Raters: Table IV presents the agreement metrics between our LLM-based assessment method and human expert assessments across both textuality and adequacy dimensions. The results demonstrate consistently high agreement levels across all three applications and both assessment dimensions.

For the **textuality dimension**, the agreement metrics between our method and individual human raters exhibit substantial variability, further confirming the subjectivity and inconsistency issues identified in RQ1. As shown in Table IV, the Quadratic Weighted Kappa (κ) scores with individual raters vary dramatically across applications: ranging from 0.49–0.86 for App 1, 0.42–0.76 for App 2, and 0.42–0.94 for App 3. Similarly, Spearman correlation coefficients (ρ) show significant fluctuations, ranging from 0.49–0.86 for App 1, 0.54–0.77 for App 2, and 0.49–0.88 for App 3. This substantial variation in agreement metrics with individual evaluators directly reflects the inherent subjectivity and inconsistency in manual assessment processes.

To more intuitively illustrate the aforementioned subjectivity and inconsistency, we plotted boxplots of the textuality scores for each report across the three applications, visualizing the uncertainty and fluctuation in manual assessments (see Fig. 3 (a)–(c)). Each box represents the interquartile range (IQR) of the six evaluators' scores, the whiskers indicate the overall range, orange dots denote outliers, purple triangles represent the mean, purple dashed lines show the mean trend, green dashed lines indicate the median trend, and red pentagrams and red lines correspond to the scores and trend of our method.

Across Fig. 3 (a)–(c), the boxplots make human subjectivity explicit. Many reports show wide IQRs, long whiskers, and outliers, indicating substantial dispersion and skew among the six raters. The variability is particularly pronounced in APP2; APP1 also exhibits visible spread, whereas APP3 is overall more compact yet still shows occasional skew or long whiskers. In contrast, our method's red stars mostly fall inside the boxes and close to the medians, and the red trend line closely tracks the purple triangles and purple dashed line (the six raters' average score and its trend), with only small deviations at a few reports. These patterns indicate that, despite notable human variability, our method remains stable and consistently aligns with the group consensus.

Consistent with these observations, our method exhibits strong agreement with the averaged scores from the six human evaluators. This is corroborated by the data in the consistency quantification table (Table IV). The QWK with averaged ratings achieves stable and high values of 0.72, 0.70, and 0.82 across the three applications, indicating substantial to near-perfect agreement. The Spearman correlations with averaged scores are consistently high (0.79–0.81), and Kendall-Tau values remain in a narrow range (0.60–0.65). This sustained high agreement with collective human judgment validates the effectiveness of our LLM-based assessment approach and underscores the reliability gains from multi-rater aggregation.

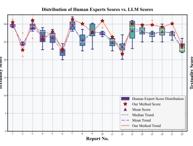
To evaluate robustness of the textuality results with respect to gold-label aggregation, we conduct a concise sensitivity analysis by comparing four aggregation schemes on the six expert scores: (1) *Simple Average*, (2) *Expert Weighted* (weights derived from inter-rater consistency), (3) *High Consistency Subset* (low-disagreement items use simple average; others use weighted average), and (4) *20% Trimmed Mean*. Then we compute Spearman's ρ between our method and each scheme. As shown in Table V, correlations are uniformly high (0.802–0.805, all $p < 0.001$) with negligible differences (maximum absolute difference $\approx$ 0.4%, standard deviation 0.010–0.020), indicating insensitivity to aggregation choice and supporting simple averaging as a reasonable, reproducible option.

For the **adequacy dimension**, a similar pattern emerges. Individual rater agreements show considerable variation, with Quadratic Weighted Kappa scores ranging from 0.77–0.85 for App 1, 0.80–0.89 for App 2, and 0.91–0.97 for App 3. Spearman correlations with individual raters also fluctuate significantly, particularly for App 2 (0.57–0.75) and App 3 (0.63–0.88). However, our method achieves remarkably consistent and high agreement with the averaged human assessments, with Quadratic Weighted Kappa scores of 0.89, 0.90, and 0.97 respectively. The Spearman correlations with averaged scores are stable and strong (0.64–0.88), demonstrating even higher consistency than the textuality dimension.

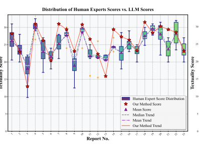
TABLE IV: Agreement between our method and six human raters on three apps across textuality and adequacy dimensions.

Dimension	**Rater**	**APP1**			**APP2**			**APP3**		
		ρ	τ	κ	ρ	τ	κ	ρ	τ	κ
Textuality	Rater 1	0.53	0.38	0.70	0.72	0.54	0.61	0.76	0.60	0.57
	Rater 2	0.86	0.69	0.52	0.54	0.39	0.42	0.61	0.49	0.80
	Rater 3	0.59	0.42	0.68	0.68	0.48	0.62	0.63	0.51	0.79
	Rater 4	0.86	0.72	0.49	0.77	0.58	0.47	0.49	0.38	0.42
	Rater 5	0.49	0.34	0.51	0.69	0.51	0.62	0.88	0.74	0.94
	Rater 6	0.79	0.60	0.77	0.77	0.62	0.76	0.75	0.58	0.76
	Avg.*	0.81	0.65	0.72	0.81	0.60	0.70	0.79	0.64	0.82
Adequacy	Rater 1	0.88	0.75	0.80	0.57	0.50	0.81	0.63	0.63	0.91
	Rater 2	0.87	0.72	0.84	0.62	0.54	0.89	0.88	0.88	0.97
	Rater 3	0.80	0.67	0.77	0.75	0.68	0.89	0.76	0.76	0.94
	Rater 4	0.85	0.73	0.85	0.57	0.51	0.80	0.88	0.88	0.97
	Rater 5	0.79	0.66	0.77	0.75	0.68	0.89	0.76	0.76	0.94
	Rater 6	0.82	0.65	0.83	0.69	0.62	0.87	0.88	0.88	0.97
	Avg.*	0.86	0.72	0.89	0.64	0.54	0.90	0.88	0.87	0.97

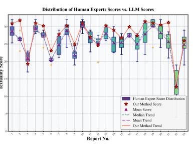
* Avg. represents the consistency measurement between our method and the average scores of the six human raters, not the average of the above six individual rater metrics.



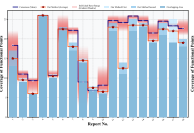
(a) Textuality-APP1



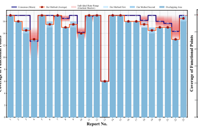
(b) Textuality-APP2



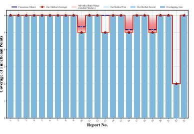
(c) Textuality-APP3



(d) Adequacy-APP1



(e) Adequacy-APP2



(f) Adequacy-APP3

Fig. 3: Visualization chart between our method and human raters: (a-c) Textuality dimension, (d-f) Adequacy dimension

TABLE V: Sensitivity analysis: Spearman correlation (ρ) between our method and four aggregation schemes on textuality.

App	Simple Avg.	Expert Wtd.	High-Consist.	Trimmed Mean
APP1	0.814	0.804	0.814	0.804
APP2	0.812	0.812	0.812	0.830
APP3	0.789	0.789	0.789	0.782
Avg.	**0.805**	**0.802**	**0.805**	**0.805**
SD	0.011	0.010	0.011	0.020

The observed variation in performance across applications can be attributed to the inherent characteristics of different software domains and the complexity of their functional requirements. App 2 shows relatively more variability in individual rater agreements, likely due to its more complex functional structure as an online community platform, while the utility application (App 3) demonstrates the highest overall consistency. Importantly, across both dimensions, our method's agreement with averaged human assessments consistently exceeds commonly accepted thresholds for substantial agreement ($\kappa > 0.60$), with most achieving excellent agreement levels ($\kappa > 0.80$). These results validate the effectiveness of our LLM-based method in replicating reliable human expert judgment patterns while avoiding the subjectivity and inconsistency inherent in individual manual assessments.

Answer to RQ2: While our LLM-based method shows significant variability in agreement with individual human raters (reflecting the subjectivity issues identified in RQ1), it demonstrates consistently high agreement with averaged human assessments across both textuality and adequacy dimensions (average $\kappa = 0.74$ for textuality, $\kappa = 0.92$ for adequacy). Moreover, sensitivity analysis under four defensible aggregation schemes yields nearly identical conclusions for textuality (Table V), confirming that our findings are robust to the choice of aggregation. This validates the effectiveness of our automated assessment approach in replicating reliable collective human judgment while avoiding individual assessment biases.

TABLE VI: Time cost comparison between our automated method and manual human assessment.

Our Method (seconds)

APP	Dim.	Seq.	Con.	Avg.
APP1	Text.	2236.43	145.31	124.25
	Adeq.	582.98	46.67	32.33
	Comp.	182.74	15.23	10.15
APP2	Text.	2923.72	139.47	127.19
	Adeq.	453.14	35.38	19.70
	Comp.	87.62	10.23	3.80
APP3	Text.	2093.76	131.89	91.03
	Adeq.	256.51	22.92	11.15
	Comp.	43.18	8.84	1.87

Human Raters (minutes)

Rater	APP1	APP2	APP3	Avg.
Rater 1	195	221	205	9.70
Rater 2	176	213	195	9.13
Rater 3	135	172	149	7.13
Rater 4	175	231	170	9.00
Rater 5	165	182	162	7.95
Rater 6	245	276	266	12.30
Avg.	**182**	**216**	**191**	**9.20**

3) RQ3: Assessment Efficiency: Table VI presents a comprehensive comparison of time expenditure between our automated LLM-based method and manual human raters for assessing crowdsourced testing reports across three applications. The results demonstrate substantial efficiency improvements achieved through automation.

Our method offers significant time savings, particularly when utilizing concurrent execution. For **textuality dimension assessment**, concurrent execution reduces assessment time to approximately 131-145 seconds per application, compared to sequential execution requiring over 2000 seconds. The **adequacy dimension** shows even greater efficiency gains, with concurrent execution completing assessments in 22-46 seconds versus 256-582 seconds sequentially. The **competitiveness dimension** demonstrates the most dramatic improvements, reducing assessment time to under 16 seconds with concurrent processing.

In contrast, manual human assessment requires substantially more time investment. Human raters spend an average of 7.13 to 12.30 minutes per report, with significant individual variation (ranging from 135 to 276 minutes per application). Rater 6 consistently requires the most time across all applications, while Rater 3 demonstrates the fastest manual assessment speed. The average human assessment time is 9.20 minutes per report, representing a 552-second investment per assessment.

When comparing per-report efficiency, our automated method achieves assessment times ranging from 5.73 to 127.19 seconds per report across concurrent and sequential modes, while human raters require 498.3 to 606.7 seconds per report. This represents approximately a **5 to 90-fold improvement in efficiency** depending on the execution mode, with concurrent processing showing the most dramatic speedup.

The variability in human assessment times also highlights the subjective nature and inconsistency inherent in manual assessment processes. Our automated method not only provides superior speed but also maintains consistent processing times across evaluators, eliminating human subjectivity-related delays.

Answer to RQ3: Our automated LLM-based assessment method significantly improves assessment efficiency, achieving 5 to 90-fold speed improvements compared to manual human assessment (average 5.73-127.19 seconds per report vs. 498.3-606.7 seconds), while maintaining consistent processing times and eliminating subjective variability in assessment duration.

C. Threats to Validity

We acknowledge several potential threats to the validity of our findings and outline the mitigation strategies employed. Although human assessment has certain subjectivity (as shown by individual differences in RQ1), we effectively mitigated this issue through aggregated assessment by six experts and ICC analysis, achieving good reliability levels of 0.85-0.91 for average assessment; LLM clustering in the competitiveness dimension may have certain classification bias, but we adopted multi-level fine-grained clustering strategies to improve accuracy; quantitative indicator weights in the textuality dimension are determined based on existing TERQAF research framework and expert experience, providing a reasonable starting point for this field. Overall, these limitations are acceptable at the current research stage, and we have minimized related risks as much as possible through various technical means and experimental designs.

VI. Discussion

A natural question is why a single automated score remains useful when human judgments are subjective. Our goal is not to discover an absolute "true" score, but to provide a more consistent, repeatable, and scalable proxy for collective expert judgment. While strict determinism is not guaranteed, the scorer is configured for high consistency (e.g., checklist mechanism, standardized prompts and low-temperature decoding) and, in practice, exhibits low run-to-run variability; it reduces the intra-/inter-rater variance observed in RQ1 and aligns closely with aggregated expert labels in RQ2, including robustness under alternative aggregation schemes. This consistency enables fair comparison, triage, and prioritization across large report sets where manual assessment is costly and hard to reproduce. Accordingly, the single score should be read as an efficient, low-cost approximation to an expert panel's consensus, not as a replacement for expert deliberation. We also qualify our claims: a single score simplifies an inherently subjective construct; our contribution is a highly consistent, consensus-aligned mechanism that assists (rather than replaces) human decision-making. Future work can expose uncertainty (e.g., confidence intervals or score distributions) to better reflect residual ambiguity.

VII. Related Work

A. Crowdsourced Testing Report Processing

With the increase in software scale and complexity, traditional testing methods are no longer sufficient to meet the demands for comprehensive coverage and efficient issue detection. Crowdsourced testing draws on the concept of crowdsourcing [35], effectively leveraging testers with diverse backgrounds and skill levels by outsourcing testing tasks to individuals or organizations globally, thereby enhancing test coverage and diversity [1], [2]. However, this approach also brings about certain challenges, such as how to select crowdsourced workers [36], [37], how to optimize the crowdsourcing process [38], [39], crowdsourced-assisted automated testing [35], and optimization of test reporting [7], [27], with quality control of crowdsourced testing reports becoming a particularly thorny issue. To address these challenges, many researchers have proposed various methods to assist developers in improving report review efficiency: Yu et al. [40] introduced SemCluster, a semi-supervised clustering method that enhances clustering by extracting features from application screenshots and textual descriptions, while their DeepPrior [7] utilizes image understanding technology for better report prioritization; Feng et al. [3], [6] improved prioritization by extracting features from screenshots and text; Chen et al. [12] modeled test reports based on quality metrics to enable developers to focus on high-quality reports; and report deduplication has emerged as another important direction [41]–[43], with Sun et al. [44], [45] employing information retrieval models to analyze textual and non-textual fields for measuring report similarity, and Nguyen et al. [43] designing DBTM, a tool that uses information-retrieval-based and topic-based features to detect duplicate reports.

B. LLM-as-a-Judge

The LLM-as-a-Judge paradigm [15]–[17] leverages sophisticated natural language understanding capabilities to serve as automated evaluators for both objective and subjective assessments. Compared to reward models that only output scalar scores [46], [47], LLM-as-a-Judge evaluators are more robust and interpretable due to their ability to generate detailed rationales [15], [18], [48]. Past works have employed various evaluation components including fine-grained criteria [15], [49], scoring rubrics [50], [51], and verification questions [52]. Recent studies demonstrate effectiveness across diverse applications, from code generation evaluation [19], [20] to content quality assessment [21], achieving high consistency with human evaluators while significantly reducing time and cost. This paradigm offers unique advantages for assessing crowdsourced testing reports, where the multi-faceted nature of reports aligns well with LLMs' comprehensive analysis capabilities.

VIII. Conclusion

In this paper, we proposed a novel LLM-based multi-dimensional assessment method for crowdsourced testing reports that addresses the critical challenge of efficiently assessing report quality in crowdsourced testing environments. Our approach introduces three complementary assessment dimensions—textuality, adequacy, and competitiveness—that comprehensively assess reports from individual quality, requirement coverage, and defect discovery value perspectives. Through extensive experiments on three mobile applications with 64 crowdworkers, we demonstrated that our method achieves substantial to near-perfect agreement with human expert consensus assessments (average $\kappa = 0.74$ for textuality and $\kappa = 0.92$ for adequacy dimensions) while delivering 5 to 90-fold improvements in assessment efficiency compared to manual assessment. The integration of LLM-as-a-Judge paradigm not only automates the assessment process but also provides detailed feedback to help testers improve their future submissions. Our work contributes to the advancement of automated quality control in crowdsourced testing, offering a scalable and consistent solution that maintains high accuracy while significantly reducing the time and cost associated with manual report review.

Acknowledgment

The authors would like to thank all anonymous reviewers for their insightful comments. This work was partially supported by the National Key R&D Program of China under Grant 2024YFF0908000.

Data Availability

Our source code and prompt templates for the proposed method in this paper are publicly available at this repository.

References

[1] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *Journal of Systems and Software*, vol. 126, pp. 57–84, 2017.

[2] R. Gao, Y. Wang, Y. Feng, Z. Chen, and W. Eric Wong, "Successes, challenges, and rethinking–an industrial investigation on crowdsourced mobile application testing," *Empirical Software Engineering*, vol. 24, pp. 537–561, 2019.

[3] Y. Feng, J. A. Jones, Z. Chen, and C. Fang, "Multi-objective test report prioritization using image understanding," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, 2016, pp. 202–213.

[4] R. Hao, Y. Feng, J. A. Jones, Y. Li, and Z. Chen, "Ctras: Crowdsourced test report aggregation and summarization," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 900–911.

[5] Y. Li, Y. Feng, R. Hao, D. Liu, C. Fang, Z. Chen, and B. Xu, "Classifying crowdsourced mobile test reports with image features: An empirical study," *Journal of Systems and Software*, vol. 184, p. 111121, 2022.

[6] Y. Feng, Z. Chen, J. A. Jones, C. Fang, and B. Xu, "Test report prioritization to assist crowdsourced testing," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 225–236.

[7] S. Yu, C. Fang, Z. Cao, X. Wang, T. Li, and Z. Chen, "Prioritize crowdsourced test reports via deep screenshot understanding," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 946–956.

[8] L. Aversano and E. Tedeschi, "Bug report quality evaluation considering the effect of submitter reputation," in *International Conference on Software Engineering and Applications*, vol. 2. SCITEPRESS, 2016, pp. 194–201.

[9] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann, "What makes a good bug report?" in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, 2008, pp. 308–318.

[10] P. Bhattacharya, L. Ulanova, I. Neamtiu, and S. C. Koduru, "An empirical analysis of bug reports and bug fixing in open source android apps," in *2013 17th European Conference on Software Maintenance and Reengineering*. IEEE, 2013, pp. 133–143.

[11] X. Chen, H. Jiang, X. Li, T. He, and Z. Chen, "Automated quality assessment for crowdsourced test reports of mobile applications," in *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2018, pp. 368–379.

[12] X. Chen, H. Jiang, X. Li, L. Nie, D. Yu, T. He, and Z. Chen, "A systemic framework for crowdsourced test report quality assessment," *Empirical Software Engineering*, vol. 25, pp. 1382–1418, 2020.

[13] H. Zhang, Y. Zhao, S. Yu, and Z. Chen, "Automated quality assessment for crowdsourced test reports based on dependency parsing," in *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2022, pp. 34–41.

[14] Q. Wang, Z. Chen, J. Wang, and Y. Feng, *Intelligent Crowdsourced Testing*. Springer, 2022.

[15] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[16] Y. Liu, D. Iter, Y. Xu, S. Wang, R. Xu, and C. Zhu, "G-eval: Nlg evaluation using gpt-4 with better human alignment," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 2511–2522.

[17] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, "Chateval: Towards better llm-based evaluators through multi-agent debate," *arXiv preprint arXiv:2308.07201*, 2023.

[18] Q. Zhang, Y. Wang, T. Yu, Y. Jiang, C. Wu, L. Li, Y. Wang, X. Jiang, L. Shang, R. Tang *et al.*, "Reviseval: Improving llm-as-a-judge via response-adapted references," *arXiv preprint arXiv:2410.05193*, 2024.

[19] T. Ahmed, P. Devanbu, C. Treude, and M. Pradel, "Can llms replace manual annotation of software engineering artifacts?" *arXiv preprint arXiv:2408.05534*, 2024.

[20] R. Wang, J. Guo, C. Gao, G. Fan, C. Y. Chong, and X. Xia, "Can llms replace human evaluators? an empirical study of llm-as-a-judge in software engineering," *arXiv preprint arXiv:2502.06193*, 2025.

[21] Y. Wu, Y. Wan, Z. Chu, W. Zhao, Y. Liu, H. Zhang, X. Shi, and P. S. Yu, "Can large language models serve as evaluators for code summarization?" *arXiv preprint arXiv:2412.01333*, 2024.

[22] L.-H. Chang and F. Ginter, "Automatic short answer grading for finnish with chatgpt," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 21, 2024, pp. 23 173–23 181.

[23] O. Henkel, L. Hills, A. Boxer, B. Roberts, and Z. Levonian, "Can large language models make the grade? an empirical study evaluating llms ability to mark short answer questions in k-12 education," in *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, 2024, pp. 300–304.

[24] C. Senanayake and D. Asanka, "Rubric based automated short answer scoring using large language models (llms)," in *2024 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, vol. 7. IEEE, 2024, pp. 1–6.

[25] O. Fagbohun, N. Iduwe, M. Abdullahi, A. Ifaturoti, and O. Nwanna, "Beyond traditional assessment: Exploring the impact of large language models on grading practices," *Journal of Artifical Intelligence and Machine Learning & Data Science*, vol. 2, no. 1, pp. 1–8, 2024.

[26] B. Jury, A. Lorusso, J. Leinonen, P. Denny, and A. Luxton-Reilly, "Evaluating llm-generated worked examples in an introductory programming course," in *Proceedings of the 26th Australasian Computing Education Conference*, 2024, pp. 77–86.

[27] J. Wang, M. Li, S. Wang, T. Menzies, and Q. Wang, "Images don't lie: Duplicate crowdtesting reports detection with screenshot information," *Information and Software Technology*, vol. 110, pp. 139–155, 2019.

[28] Y. Ling, S. Yu, C. Fang, G. Pan, J. Wang, and J. Liu, "Redefining crowdsourced test report prioritization: An innovative approach with large language model," *Information and Software Technology*, vol. 179, p. 107629, 2025.

[29] H. K. V. Tran, N. b. Ali, M. Unterkalmsteiner, J. Börstler, and P. Chatzipetrou, "Quality attributes of test cases and test suites–importance & challenges from practitioners' perspectives," *Software quality journal*, vol. 33, no. 1, p. 9, 2025.

[30] T. Wei, W. Wen, R. Qiao, X. Sun, and J. Ma, "Rocketeval: Efficient automated LLM evaluation via grading checklist," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=zJjzNj6QUe

[31] R. A. Fisher, "On the" probable error" of a coefficient of correlation deduced from a small sample," *Metron*, vol. 1, pp. 3–32, 1921.

[32] S. Bonthu, S. Rama Sree, and M. Krishna Prasad, "Automated short answer grading using deep learning: A survey," in *Machine Learning and Knowledge Extraction: 5th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2021, Virtual Event, August 17–20, 2021, Proceedings 5*. Springer, 2021, pp. 61–78.

[33] J. Hauke and T. Kossowski, "Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data," *Quaestiones geographicae*, vol. 30, no. 2, pp. 87–93, 2011.

[34] P. K. Sen, "Estimates of the regression coefficient based on kendall's tau," *Journal of the American statistical association*, vol. 63, no. 324, pp. 1379–1389, 1968.

[35] J. Howe *et al.*, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 176–183, 2006.

[36] J. Wang, S. Wang, J. Chen, T. Menzies, Q. Cui, M. Xie, and Q. Wang, "Characterizing crowds to better optimize worker recommendation in crowdsourced testing," *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1259–1276, 2019.

[37] J. Wang, Y. Yang, S. Wang, Y. Hu, D. Wang, and Q. Wang, "Context-aware in-process crowdworker recommendation," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 1535–1546.

[38] J. Wang, Y. Yang, R. Krishna, T. Menzies, and Q. Wang, "isense: Completion-aware crowdtesting management," in *2019 IEEE/ACM 41st international conference on software engineering (ICSE)*. IEEE, 2019, pp. 912–923.

[39] J. Wang, Y. Yang, T. Menzies, and Q. Wang, "isense2. 0: Improving completion-aware crowdtesting management with duplicate tagger and sanity checker," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 4, pp. 1–27, 2020.

[40] S. Yu, C. Fang, Q. Zhang, M. Du, J. Liu, and Z. Chen, "Semi-supervised crowdsourced test report clustering via screenshot-text binding rules," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1540–1563, 2024.

[41] A. Alipour, A. Hindle, and E. Stroulia, “A contextual approach towards more accurate duplicate bug report detection,” in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 183–192.

[42] A. Hindle, A. Alipour, and E. Stroulia, “A contextual approach towards more accurate duplicate bug report detection and ranking,” *Empirical Software Engineering*, vol. 21, pp. 368–410, 2016.

[43] A. T. Nguyen, T. T. Nguyen, T. N. Nguyen, D. Lo, and C. Sun, “Duplicate bug report detection with a combination of information retrieval and topic modeling,” in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, 2012, pp. 70–79.

[44] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, “Towards more accurate retrieval of duplicate bug reports,” in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 2011, pp. 253–262.

[45] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, “A discriminative model approach for accurate duplicate bug report retrieval,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, 2010, pp. 45–54.

[46] T. Wang, I. Kulikov, O. Golovneva, P. Yu, W. Yuan, J. Dwivedi-Yu, R. Y. Pang, M. Fazel-Zarandi, J. Weston, and X. Li, “Self-taught evaluators,” *arXiv preprint arXiv:2408.02666*, 2024.

[47] P. Wang, A. Xu, Y. Zhou, C. Xiong, and S. Joty, “Direct judgement preference optimization,” 2024.

[48] Z. Ankner, M. Paul, B. Cui, J. D. Chang, and P. Ammanabrolu, “Critique-out-loud reward models,” *arXiv preprint arXiv:2408.11791*, 2024.

[49] S. Saha, O. Levy, A. Celikyilmaz, M. Bansal, J. Weston, and X. Li, “Branch-solve-merge improves large language model evaluation and generation,” in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024, pp. 8345–8363.

[50] W. Yuan, R. Y. Pang, K. Cho, X. Li, S. Sukhbaatar, J. Xu, and J. Weston, “Self-rewarding language models, 2024,” *URL https://arxiv.org/abs/2401.10020*.

[51] P. Trivedi, A. Gulati, O. Molenschot, M. A. Rajeev, R. Ramamurthy, K. Stevens, T. S. Chaudhery, J. Jambholkar, J. Zou, and N. Rajani, “Self-rationalization improves llm as a fine-grained judge,” 2024. [Online]. Available: https://arxiv.org/abs/2410.05495

[52] S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz, and J. Weston, “Chain-of-verification reduces hallucination in large language models,” *arXiv preprint arXiv:2309.11495*, 2023.