WCRG Global Research
WORLD COMET RESEARCH GROUP

International Journal of Modern Innovations and Emerging Trends
Vol. 2, No. 1, 2019
Doi: 10.67228/30715628/IJMIET-2019PI8W8J
PP. 01-20

Original Article

AI-Assisted Requirements Engineering Using Multimodal Language Models

Karthik Raman

Software Architect, Tata Consultancy Services, India.

Received: 12-04-2018 Revised: 12-26-2018 Accepted: 01-01-2019 Published: 01-03-2019

ABSTRACT

The rapid advancement of multimodal large language models (MLLMs) has introduced new opportunities to enhance Requirements Engineering (RE) processes through automated understanding of textual, visual, and conversational artifacts. This paper investigates how MLLMs can support key RE activities—including requirements elicitation, classification, validation, conflict detection, prioritization, and documentation—by leveraging their ability to process heterogeneous project inputs such as stakeholder conversation transcripts, UI sketches, system diagrams, and domain documents. We present a conceptual framework for integrating MLLMs into existing RE workflows, highlighting their potential to reduce ambiguity, accelerate requirement refinement, and improve traceability. Through experimental evaluation and case studies, we demonstrate that MLLM-powered RE assistance can significantly increase requirement accuracy and analyst productivity. Finally, we discuss risks such as hallucination, data privacy, and model bias, and propose mitigation strategies and future research directions for reliable adoption of AI-assisted RE systems.

KEYWORDS

Multimodal Language Models, Requirements Engineering, AI-Assisted Software Engineering, Requirements Elicitation, Requirements Classification, Requirements Validation, Traceability, Natural Language Processing, Human-AI Collaboration, Software Requirements Specification (SRS).

ISSN: 3071-5628
PP: 01-20.

1. INTRODUCTION

1.1. Background Of Requirements Engineering

Requirements Engineering (RE) is the disciplined process of discovering, documenting, and maintaining the intentions and constraints that will guide the development of a software system. It encompasses activities such as elicitation from stakeholders, specification in natural or formal language, negotiation to resolve conflicting needs, validation to ensure correctness and feasibility, and management to handle change across the system lifecycle. Good RE reduces downstream defects, lowers project risk, and improves stakeholder satisfaction by making expectations explicit and testable; conversely, poor RE is a leading cause of cost overruns, missed deadlines, and feature creep. The discipline sits at the intersection of social and technical concerns: it must translate often-ambiguous human goals into machine-actionable artifacts while coping with evolving requirements, differing stakeholder vocabularies, and the socio-organizational context in which software will operate. Historically, RE has relied heavily on human analysts, structured templates (e.g., use cases, user stories, SRS), and toolchains for traceability and versioning; however, the volume, variety, and multimodality of contemporary project artifacts—chat logs, videos, whiteboards, sketches, wireframes, and legacy code—are stretching traditional approaches and calling for smarter, more scalable assistance.

1.2. Rise of Multimodal Large Language Models (MLLMs)

Over the past few years, large language models have expanded beyond pure text to handle multiple modalities—images, diagrams, audio, and sometimes structured inputs—enabling richer understanding of heterogeneous data. These multimodal large language models (MLLMs) integrate visual encoders, audio processing modules, and powerful text decoders to form unified representations that can reason across modalities: for example, linking a hand-drawn UI sketch to a stakeholder transcript, or extracting requirements from a videoed walkthrough. The rise of MLLMs is driven by architectural advances, vast multimodal pretraining corpora, and improvements in transfer learning and prompt design, which together let a single model generalize to many tasks with little or no task-specific supervision. For Requirements Engineering, MLLMs open possibilities that single-modality tools could not: automated extraction from screenshots and diagrams, cross-checking spoken stakeholder intent against prototype mockups, and generating traceable, richly annotated requirement artifacts that reference multiple source modalities. As a result, MLLMs are rapidly becoming a foundation for AI-assisted tooling that aims to bridge the gap between diverse stakeholder inputs and rigorous engineering artifacts.

1.3. Motivation: Why RE needs AI assistance

Modern software projects face increased complexity, faster delivery cycles, and a proliferation of informal communication channels that scatter requirement-relevant information across formats and time. Human analysts are skilled but limited in bandwidth; they can miss implicit constraints, overlook inconsistent statements, or fail to trace decisions across artifacts. AI assistance—especially via MLLMs—promises to augment human capabilities by automating repetitive parts of RE, surfacing latent requirements from diverse inputs, and providing consistency checks at scale. Practical motivations include reducing the time and cost of producing high-quality specifications, increasing coverage of requirement validation (e.g., detecting ambiguities or contradictions that humans might miss), and improving traceability by linking requirements to evidence across documents, images, and conversations. Beyond efficiency, AI can enable new RE practices: continuous, pipeline-integrated requirement monitoring during development, semi-automated prioritization informed by usage analytics, and interactive co-pilots that help stakeholders iterate on

ISSN: 3071-5628
PP: 01-20.

requirements in natural language and visuals. However, motivation is balanced by caution—adoption must be driven by demonstrable reliability, human-in-the-loop controls, and clear accountability to avoid automation-induced errors.

1.4. Research objectives and contributions

This paper aims to investigate how multimodal language models can be systematically applied to Requirements Engineering to improve the accuracy, traceability, and efficiency of RE activities while identifying and mitigating attendant risks. The primary objectives are threefold: (1) to design a conceptual framework that maps RE tasks to multimodal model capabilities and defines a practical system architecture for integration into existing toolchains, (2) to empirically evaluate the effectiveness of MLLM-based assistance on representative RE tasks—elicitation from mixed-media artifacts, classification, ambiguity detection, and traceability—and (3) to surface methodological, ethical, and operational guidelines for safe deployment, including prompt strategies, human-in-the-loop workflows, and privacy-preserving measures. The paper contributes a detailed framework and workflow, an evaluation methodology with metrics suited to multimodal RE, experimental case studies demonstrating tangible gains and failure modes, and a set of best-practice recommendations for practitioners and researchers. By doing so, the work bridges conceptual promise and empirical evidence, helping move AI-assisted RE from prototype experiments toward robust, real-world adoption.

1.5. Paper organization

The remainder of the paper is organized to guide the reader from background through design, evaluation, and reflection. Section 2 surveys prior work in both Requirements Engineering and AI-enabled RE automation to situate our contributions relative to existing tools and research. Section 3 provides a primer on multimodal language models, explaining relevant capabilities and limitations for RE practitioners. Section 4 presents the proposed AI-assisted RE framework and detailed system architecture, while Section 5 describes the datasets, prompt engineering techniques, and evaluation metrics used in our experiments. Section 6 reports experimental results and case studies that illustrate strengths and limitations. Sections 7 and 8 discuss broader implications, including ethical, legal, and practical deployment concerns, and potential application scenarios. Finally, Section 9 outlines promising directions for future work, and Section 10 concludes by summarizing our findings and reiterating recommendations for integrating MLLMs responsibly into RE processes.

2. RELATED WORK

2.1. Related Work (overview paragraph)

This section synthesizes the literature across two complementary axes: classical Requirements Engineering research and the emerging body of work that applies natural language processing and machine learning to RE problems, culminating in efforts that leverage multimodal models. The goal is to show how traditional RE methods and tools have evolved, how text-centric AI techniques have been used to automate sub-tasks like requirements extraction and classification, and where the move to multimodality addresses gaps left by text-only approaches. By connecting research strands requirements elicitation, ambiguity detection, traceability, and tool integration with technological trends in large-scale language and multimodal models, this review identifies both the progress achieved and persistent limitations that motivate the present study.

ISSN: 3071-5628
PP: 01-20.

2.2. Traditional Requirements Engineering Tools and Challenges

Traditional RE tools and methods focus on structured elicitation and documentation: interview protocols, requirement templates, UML models, SRS documents, and commercial suites that support traceability, change management, and stakeholder collaboration. These approaches excel in establishing formal artifacts and maintaining rigorous version control, but they rely on considerable manual effort and assume disciplined stakeholder behavior. Challenges include handling informal or implicit knowledge embedded in conversations and artifacts, managing conflicting stakeholder goals, and maintaining traceability when requirements evolve rapidly. Tool support often remains siloed word processors for SRS, modeling tools for diagrams, and issue trackers for tasks making cross-artifact reasoning laborious. Moreover, classical methods struggle with unstructured multimodal inputs like whiteboard photos or recorded meetings, leaving a gap between the artifacts stakeholders naturally produce and the formal representations engineers require.

2.3. NLP-based and ML-based automation in RE

Research in automating RE using NLP and machine learning has focused largely on text: extracting requirement statements from documents, classifying sentences as functional or non-functional, detecting ambiguities or defects, and automating trace links between requirements and code or test cases. Techniques range from rule-based information extraction and classical supervised classifiers to modern transformer-based models that deliver much higher accuracy and generalization. These approaches have enabled tools that accelerate requirement drafting, identify inconsistencies, and suggest candidate trace links, reducing manual workload. Nevertheless, most prior work is constrained by modality: models trained on text cannot leverage visual cues in mockups or diagrams, and speech-derived transcripts often lose paralinguistic context that could disambiguate intent. Additionally, reliance on labeled datasets has limited generalization to new domains, and many systems lack robust human-in-the-loop mechanisms to manage model errors and maintain accountability.

2.4. Evolution from text-only LLMs to multimodal LLMs

The transition from text-only language models to multimodal LLMs represents a step change in capability for RE automation. Text-only LLMs demonstrated strong abilities in paraphrasing, summarization, and classification, which made them useful for drafting requirements and suggesting edits. However, they could not directly interpret images, diagrams, or audio recordings common carriers of requirements information. Multimodal architectures combine visual and sometimes audio encoders with language decoders, enabling joint reasoning across modalities. This evolution unlocks use cases such as extracting requirements from UI sketches, reconciling spoken stakeholder comments with on-screen prototypes, and generating enriched requirement artifacts that include visual evidence and pointers. Early work in applying multimodal models to software engineering tasks is nascent but promising: prototypes show that image-to-text and cross-modal retrieval capabilities meaningfully extend the set of automatable RE tasks, though they also surface new problems like grounding errors and modality-specific hallucinations that require tailored mitigation.

2.5. Gaps in existing approaches

Despite progress, there remain several critical gaps that motivate research into AI-assisted RE using MLLMs. First, there is a lack of standardized multimodal datasets and benchmarks specifically designed for RE tasks, which limits rigorous evaluation and comparability of approaches. Second, many existing systems are proof-of-concept and focus on isolated tasks (e.g., text extraction or UI

ISSN: 3071-5628
PP: 01-20.

captioning) instead of end-to-end workflows that integrate elicitation, validation, prioritization, and traceability. Third, issues of reliability, explainability, and human oversight are under-explored: multimodal models can produce confident-sounding but incorrect outputs, and the community lacks well-defined human-in-the-loop patterns optimized for RE contexts. Privacy and data governance—especially when stakeholder conversations or proprietary prototypes are involved—are also inadequately addressed by current work. Finally, there is limited empirical evidence on how these tools affect team dynamics and decision-making in real projects. Filling these gaps requires interdisciplinary work combining model engineering, HCI, software process research, and empirical studies in practical development settings.

3. OVERVIEW OF MULTIMODAL LANGUAGE MODELS

Multimodal language models (MLLMs) are a class of AI systems that construct shared internal representations across two or more data modalities—most commonly text, images, and audio—and use those representations to perform reasoning, generation, and retrieval tasks that span modalities. Unlike text-only models that treat language as the sole input, MLLMs learn to correlate visual patterns with linguistic descriptions, to align spoken utterances with semantic content, and in some architectures to incorporate structured artifacts like code or tables. This cross-modal grounding enables richer interpretation of real-world artifacts: for Requirements Engineering (RE), it means a single model can, for example, read a stakeholder transcript, inspect a screenshot of a proposed UI, and produce a consistent set of requirement statements that reference both the spoken intent and the visual design. The evolution of MLLMs has been driven by multimodal pretraining on paired datasets, architectural components that fuse modality-specific encoders with a unified decoder or cross-attention mechanism, and new prompting or fine-tuning strategies that adapt the model to downstream tasks. For RE practitioners, the key takeaway is that MLLMs are not merely better language processors; they are integrators of context, capable of producing outputs that reflect the interplay between words, visuals, and, where available, audio cues.

3.1. Definition and capabilities

At their core, MLLMs map multimodal inputs into embeddings that capture semantic meaning regardless of whether the original signal was a sentence, a pixel array, or an audio waveform; these embeddings are then used for downstream language tasks like summarization, classification, question answering, or generation. Capabilities that are particularly relevant to RE include cross-modal retrieval (find the diagram that supports a verbal claim), multimodal summarization (produce concise requirement summaries that cite images and transcripts), visual question answering (answer questions about a UI mockup), and conditional generation (create requirement text conditioned on a sketch). Additionally, advanced MLLMs can perform alignment tasks—linking phrases in text to regions in an image or timestamps in audio—supporting traceability and evidence annotation. However, these capabilities come with caveats: performance varies by modality quality, by the degree of pretraining alignment, and by whether domain-specific fine-tuning was performed; still, even out-of-the-box MLLMs often provide a useful starting point for automating and augmenting many RE tasks.

3.2. Architecture of typical MLLMs (text–image–audio–code integration)

Typical MLLM architectures use modality-specific encoders that transform raw inputs (tokens for text, convolutional or transformer-based encoders for images, and spectrogram-based encoders for audio) into a shared latent space where cross-attention mechanisms or fusion modules enable interaction between modalities. One common design is an encoder–decoder model where each

ISSN: 3071-5628
PP: 01-20.

modality is encoded separately and then concatenated or queried by a central decoder that generates natural language outputs, which can include synthesized requirement statements or explanatory notes. Some architectures take a more integrated approach, using a single transformer backbone with adapters or prefix embeddings that mark modality type, allowing the model to learn unified representations more directly. For integrating code or structured artifacts, the architecture may include a lightweight parser or structured encoder that preserves syntactic information while mapping it into the multimodal embedding space. Crucially for RE, the architecture must support alignment utilities (e.g., attention maps, bounding-box to token linking, or timestamp anchors) so that generated requirements can be traced back to the exact parts of the input artifacts that motivated them—an essential feature for auditability and stakeholder validation.

3.3. Strengths and limitations for RE tasks

MLLMs bring distinct strengths to RE: they can reduce fragmentation by jointly interpreting heterogeneous artifacts, they accelerate information discovery by surfacing implicit or overlooked details across modalities, and they enable richer evidence-backed requirement artifacts that include visual references or audio citations. These capabilities can markedly improve coverage, reduce ambiguities, and shorten the time needed to move from stakeholder intent to formalized requirements. However, limitations matter: multimodal models sometimes exhibit grounding errors where the textual output is only loosely connected to the image or audio input, and they can produce plausible-sounding but incorrect or hallucinated content an acute risk when creating SRS documents that stakeholders will rely on. Performance also degrades when inputs are noisy (e.g., low-quality whiteboard photos or poor audio), when domain vocabulary is specialized, or where the pretraining data lacks examples similar to the target artifacts. Furthermore, interpretability and controllability are harder in multimodal settings; tracing exactly why the model generated a specific requirement often requires additional tooling to surface attention alignments and provenance metadata. For RE adoption, these strengths and limitations imply that MLLMs are powerful assistants but should be embedded in human-in-the-loop workflows with validation, provenance tracking, and targeted fine-tuning.

3.4. Types of multimodal inputs relevant to RE

Requirements Engineering draws on a wide range of artifacts that are naturally multimodal: stakeholder interview transcripts capture conversational nuance and rationale; whiteboard sketches encode early-stage conceptual ideas and workflows visually; UI mockups present concrete user interface affordances and constraints; workflow diagrams formalize process logic and interactions between system components; and legacy documents including spreadsheets, old SRSs, and design documents—hold historical and compliance-oriented information. Each of these input types supplies complementary signals for inferring requirements: transcripts reveal goals and priorities, sketches reveal intended user flows and metaphors, mockups show concrete element-level expectations, diagrams capture control and data flows, and legacy documents anchor new requirements in historical decisions and constraints. MLLMs trained or adapted to handle these inputs can synthesize richer, more defensible requirement statements by triangulating across modalities e.g., confirming that a stakeholder's spoken preference for "one-click checkout" is visually represented in a prototype and is supported by an existing legacy constraint and therefore reduce the risk of misinterpretation that commonly plagues RE.

ISSN: 3071-5628
PP: 01-20.

3.4.1. Stakeholder interview transcripts

Stakeholder interview transcripts are rich textual artifacts that contain not just explicit requirements but also rationale, priorities, and contextual constraints that shape what the system must do. When processed by an MLLM, transcripts can be summarized into candidate requirements, paraphrased into standardized SRS language, and analyzed for sentiment or emphasis to infer priority. Transcripts also often contain conditional or ambiguous statements "we'd like this if possible"—which require careful disambiguation; MLLMs can assist by proposing clarifying questions, identifying implicit assumptions, and extracting action items tied to named stakeholders. Moreover, when aligned with audio (tone, emphasis) or with visual artifacts referenced in the conversation, the model can better resolve intent, making transcripts a central modality for elicitation and validation workflows.

3.4.2. Whiteboard sketches

Whiteboard sketches capture rapid ideation and frequently encode topology, user flows, and spatial relationships that are difficult to express purely in text. MLLMs that incorporate vision encoders can interpret photos of whiteboards to recognize boxes, arrows, and handwritten labels, turning rough drawings into structured components such as user stories, flow steps, or interface element inventories. The ambiguity and variability of handwriting and sketch styles make this a challenging input, but even imperfect extraction is valuable: it preserves early design intent, surfaces previously unrecorded decisions, and provides a visual anchor for subsequent clarification. When combined with transcript snippets or meeting notes, whiteboard-derived artifacts strengthen traceability back to the original brainstorming context.

3.4.3. UI mockups

UI mockups and wireframes are high-value artifacts for deriving concrete functional and non-functional requirements because they reveal element-level behavior, expected user interactions, layout constraints, and accessibility considerations. MLLMs can analyze mockups to generate requirement statements like "The login screen shall allow authentication via email or third-party OAuth, with password recovery accessible from the same view," and can also infer NFRs such as responsiveness or touch-target sizes from design cues. Integrating mockup analysis with textual requirements creates a feedback loop: the model can point out missing UI behaviors, detect inconsistencies between the design and textual requirements, and generate acceptance criteria tied to specific UI components.

3.4.4. Workflow diagrams

Workflow and process diagrams (e.g., BPMN, UML activity diagrams) express the dynamic and conditional behavior of systems and stakeholder interactions. These diagrams make explicit the control-flow, decision points, and actor roles that are essential for modeling functional requirements and service-level behaviors. MLLMs that can parse diagrammatic structures are able to translate nodes and transitions into sequenced requirement statements, identify missing decision handling (e.g., exception flows), and surface constraints on concurrency or ordering. Because workflows often delineate responsibilities across systems and human actors, diagram-derived requirements also support traceability and compliance checks when mapped to organizational roles or existing process documentation.

ISSN: 3071-5628
PP: 01-20.

3.4.5. Legacy documents

Legacy documents ranging from outdated SRS files and spreadsheet-based requirement lists to archived design notes contain both domain knowledge and constraints that must be respected during new development. They are a critical source for detecting compatibility constraints, regulatory obligations, and historical design trade-offs. MLLMs can help extract and normalize information from these heterogeneous, often poorly structured sources, flagging requirements that must be retained or adapted and identifying obsolete elements. By linking newly proposed requirements to legacy evidence, the system improves auditability and reduces the risk of accidental regression or noncompliance.

4. PROPOSED FRAMEWORK: AI-ASSISTED REQUIREMENTS ENGINEERING WITH MLLMS

The proposed framework envisions MLLMs as the central reasoning and synthesis engine within a larger RE ecosystem that includes preprocessing, provenance tracking, human review, and tool integrations. At a high level, the system ingests multimodal artifacts, processes them through modality-aware pipelines to extract candidate requirements and supporting evidence, applies AI-driven categorization and validation routines, and generates traceable, editable requirement artifacts that feed into project management and engineering tools. The framework emphasizes human-in-the-loop interactions: generated requirements are presented with provenance (source snippets, regions in images, timestamps) and confidence scores, enabling analysts to quickly verify, correct, and accept the outputs. Additionally, privacy-preserving components (redaction, access controls) and explainability utilities (attention visualizations, justification text) are woven into the architecture to address the practical and ethical issues associated with automated RE.

4.1. System architecture and workflow

The system architecture organizes components into ingestion, processing, reasoning, and integration layers. Ingestion collects raw artifacts and metadata; processing normalizes and enriches inputs (e.g., speech-to-text for audio, OCR for images, diagram parsing); reasoning is centered on the MLLM which performs extraction, classification, and generation tasks; and integration persists validated requirements into RE tools and traceability repositories. The workflow is iterative: after initial automatic extraction, requirements are routed to human analysts via a review interface where edits and acceptances update the model's context and can be fed back as fine-tuning signals or prompt adjustments. This cyclical flow supports progressive improvement, letting the system learn domain preferences while ensuring final accountability remains with human stakeholders. Logging and provenance capture occur at each stage to enable audits and rollback if incorrect requirements are introduced.

Table 1: Components of the Proposed AI-Assisted Requirements Engineering Framework Using Multimodal LLMs

Module	Description
4.1 System architecture and workflow	Overall pipeline connecting multimodal inputs, MLLM processing, reasoning engine, and RE tool integration.
4.2 Input processing module (text, images, diagrams, audio)	Preprocesses diverse inputs such as natural language text, UI mockups, system diagrams, and spoken requirements.
4.3 Automated requirement extraction	Uses multimodal LLMs to extract candidate requirements from all processed inputs.
4.4 Requirement categorization (Functional, NFR, constraints)	Classifies extracted requirements into FR, NFR, technical constraints, and domain rules.

ISSN: 3071-5628
PP: 01-20.

4.5 Requirement validation & ambiguity detection	Identifies unclear, inconsistent, or incomplete requirements using AI-based semantic checks.
4.6 Conflict and duplication detection	Detects contradictory and duplicate requirements across sources using embedding similarity.
4.7 Requirement prioritization using AI reasoning	Applies reasoning heuristics (risk, cost, dependency, stakeholder importance) to rank requirements.
4.8 Traceability matrix generation	Automatically maps requirements to design elements, models, test cases, and code artifacts.
4.9 Integration with RE tools (Jira, IBM DOORS, GitHub Issues)	Exports validated requirements into industry tools for collaboration, versioning, and monitoring.

4.2. Input Processing Module (Text, Images, Diagrams, Audio)

Before a multimodal model can reason about artifacts, an input processing module prepares them into machine-friendly representations. For text, this includes normalization, speaker diarization, and segmenting transcripts into candidate statements; for images (mockups and sketches), it includes OCR, layout analysis, and vectorization of detected elements; for diagrams, it may include graph extraction and semantic labeling of nodes and edges; and for audio, robust speech-to-text conversion with diarization and paralinguistic feature extraction (e.g., emphasis, pauses) helps recover intent. This preprocessing layer also annotates inputs with metadata such as timestamps, author identities, and source confidence, which the MLLM can use to ground outputs. Effective input processing reduces noise, improves modality alignment, and supplies the MLLM with richer, structured cues that improve extraction fidelity.

4.3. Automated requirement extraction

Automated requirement extraction uses the MLLM to identify candidate requirement statements from the normalized inputs. The model performs both information retrieval (finding segments likely to contain requirements) and generative paraphrasing (turning informal notes or sketches into formalized requirement text). Extraction pipelines typically apply heuristics or lightweight classifiers to segment the input, then ask the MLLM to generate SRS-style statements with specified templates or acceptance-criteria formats. Crucially, each extracted requirement is annotated with provenance linking back to the originating modalities e.g., a requirement may cite a transcript line, a bounding box on a mockup, and a legacy document paragraph—so that analysts can inspect and validate the source evidence. Extraction can be run continuously (e.g., after every stakeholder meeting) to maintain an evolving requirements repository that reflects the latest stakeholder inputs.

4.4. Requirement categorization (Functional, NFR, constraints)

Once candidate requirements are extracted, the system classifies them into categories such as functional requirements, non-functional requirements (NFRs), constraints, or domain assumptions. MLLMs excel at contextual classification when provided with examples or prompt templates, enabling the system to distinguish a user story-level functional statement from a performance or security NFR, and to detect constraints that may imply regulatory or backward-compatibility obligations. Proper categorization supports downstream activities like test case generation and prioritization: functional items map to acceptance criteria and test scenarios, while NFRs inform architecture and capacity planning. The system also records uncertain classifications as "needs human review," providing transparency where automated judgment may be brittle.

ISSN: 3071-5628
PP: 01-20.

4.5. Requirement validation & ambiguity detection

Validation and ambiguity detection use both rule-based checks and MLLM-driven analysis to assess whether requirement statements are complete, testable, and unambiguous. The MLLM can be prompted to rewrite statements into SMART-style criteria, to propose clarifying questions when assumptions are implicit, and to flag vague terms (e.g., "fast", "user-friendly") that require quantitative thresholds. Cross-modal validation is particularly powerful: the model can compare a textual requirement against a UI mockup or workflow diagram to confirm that expected behaviors are visually represented, or to detect mismatches where the design lacks support for a stated requirement. The system surfaces ambiguity metrics and suggested clarifications to analysts, enabling efficient iterative refinement; importantly, it logs the specific linguistic or visual evidence that triggered each ambiguity alert so reviewers can quickly resolve it.

4.6. Conflict and duplication detection

Detecting conflicts and duplicates prevents inconsistent or redundant requirements from propagating into design and test artifacts. The framework applies semantic similarity measures, intent clustering, and constraint solvers to identify requirements that overlap, contradict, or imply incompatible behavior (for example, one requirement specifying synchronous processing while another mandates immediate user feedback with asynchronous operations). MLLMs can enhance detection by transforming diverse modalities into comparable semantic embeddings and by reasoning about conditional clauses and exception flows extracted from diagrams and transcripts. When potential conflicts are identified, the system groups related requirements, highlights the conflicting clauses, and proposes resolution strategies such as merging duplicates, prioritizing stakeholder sources, or generating negotiation questions while always exposing the provenance so stakeholders can adjudicate based on original context.

4.7. Requirement prioritization using AI reasoning

Prioritization combines signals from stakeholder importance (extracted from interview emphasis and explicit prioritization statements), business impact (inferred from legacy documents and KPIs), technical cost (estimated from UI complexity or referenced workflows), and risk (identified via NFRs and compliance needs). The MLLM can synthesize these heterogeneous signals into a ranked list of requirements, explaining its reasoning in natural language and citing evidence. Advanced prioritization modules may incorporate configurable objective functions weighting business value, effort, and risk according to organizational needs or integrate with analytics systems to use real usage data. Because automated prioritization affects planning, the framework treats its outputs as recommendations subject to human review, and it provides sensitivity analyses that show how different weightings change the ranking to support transparent decision-making.

4.8. Traceability matrix generation

Traceability is implemented by automatically linking each validated requirement to its originating artifacts, design elements, test cases, and implementation tasks. Using alignment outputs from the MLLM and structured metadata from the preprocessing stage, the system builds and maintains a traceability matrix that records bidirectional links requirement-to-evidence and evidence-to-requirement along with timestamps, author attributions, and confidence scores. This matrix supports impact analysis (showing which requirements are affected by proposed changes), test coverage evaluation (mapping requirements to test cases), and compliance audits (demonstrating origin and rationale for regulated requirements). The framework also exposes trace links visually and

ISSN: 3071-5628
PP: 01-20.

in machine-readable formats so they can be consumed by external tools, ensuring that traceability is not only recorded but is actionable across the development lifecycle.

4.9. Integration with RE tools (Jira, IBM DOORS, GitHub Issues)

To be practical, the AI-assisted RE framework must integrate with common engineering and project-management ecosystems. Integration adapters translate generated and validated requirements into tickets, DOORS artifacts, or GitHub issues, preserving structure (title, description, acceptance criteria), metadata (priority, category, trace links), and provenance attachments (snippets, images, diagram fragments). Two-way integration allows updates made in downstream tools to feed back into the requirements repository and, through change detection, to trigger revalidation pipelines. Authentication and access controls ensure that sensitive artifacts obey organizational policies during transfer. By embedding MLLM-assisted outputs directly into teams' existing workflows rather than forcing migration to a new monolithic tool, the framework reduces friction and increases adoption likelihood, while preserving the auditability and traceability necessary for responsible engineering.

5. METHODOLOGY

5.1. Dataset Creation (Multimodal RE Artifacts)

Creating a high-quality dataset is foundational for evaluating MLLM-assisted Requirements Engineering, so we build a multimodal corpus that mirrors the heterogeneity and messiness of real-world RE artifacts. The dataset collection process combines (a) curated public datasets where available (e.g., UI screenshots and annotated mockups, meeting transcripts from open-source projects), (b) synthetic artifacts created to exercise edge cases (intentionally ambiguous utterances, noisy whiteboard photos, intentionally inconsistent mockups), and (c) anonymized, consented extracts from industrial partners that represent realistic stakeholder interviews, legacy documents, and workflow diagrams. For each project case in the corpus we assemble aligned bundles: raw audio and its diarized transcript, meeting notes, photographed whiteboards and their cleaned sketches, UI mockups at varying fidelity, workflow diagrams (BPMN/UML), and any legacy SRS or spreadsheet artifacts. Each candidate requirement is then manually annotated by domain experts with a canonical requirement statement, classification (functional/NFR/constraint), provenance links to the source modalities and source locations (transcript line numbers, image bounding boxes, diagram node IDs, legacy document paragraph), priority labels, and acceptance criteria. To support robust evaluation we create separate splits for training, validation, and testing that preserve project-level separation (i.e., artifacts from the same real project do not leak across splits) and include metadata about noise level, domain, and stakeholder profile so that experiments can analyze performance variation across realistic conditions.

5.2. Prompt Engineering for RE Tasks

Prompt engineering operationalizes how the MLLM maps multimodal inputs to RE outputs and is treated as a systematic, iterative process rather than ad-hoc adoptions. We develop a library of task-specific prompt templates for extraction, paraphrasing, classification, ambiguity detection, prioritization, and trace-link generation. Each template contains a structured context section—concise project metadata, source excerpts with provenance tokens, and modality markers—followed by explicit instruction and example pairs (few-shot) demonstrating desired SRS-style outputs. To make prompts robust across noisy inputs and domain shifts, we include instruction variants that request confidence estimates and provenance anchors, and we experiment with chain-of-thought style decomposition for complex reasoning tasks (e.g., resolving conditional requirements in a workflow).

ISSN: 3071-5628
PP: 01-20.

Prompt variations are systematically evaluated using the validation split: we measure sensitivity to prompt length, the number and selection of examples, and whether auxiliary context such as legacy constraints or acceptance templates is included. Findings from prompt testing inform hardened templates used in the main experiments and a set of heuristics for when to fall back to human clarification (for low-confidence or high-risk outputs).

5.3. Evaluation Setup

The evaluation setup is designed to compare MLLM-assisted RE workflows against relevant baselines under controlled yet realistic conditions. Baselines include rule-based extraction pipelines, text-only transformer models adapted to RE tasks, and (where possible) commercially available RE automation tools. Experiments are conducted on the held-out test split of the multimodal corpus and on live case studies with consenting industry partners to measure practical utility. For each task extraction, classification, validation, prioritization, and traceability we run batched inference with the MLLM and collect generated requirements along with provenance and confidence metadata. Generated outputs are then evaluated against gold-standard annotations by a panel of independent annotators who were not involved in dataset creation. For human-in-the-loop assessments, we recruit professional requirements analysts to perform review and correction tasks; we instrument the interface to measure time-on-task and the number and severity of corrections required. To examine failure modes and domain sensitivity, we run ablation studies that vary input quality (e.g., reducing image resolution, adding transcription errors), remove particular modalities to measure contribution, and test cross-domain generalization by evaluating models on domains not seen during any fine-tuning.

5.4. Metrics

We adopt a combined set of quantitative and qualitative metrics tailored to RE outcomes. Requirement accuracy is computed as a semantic similarity measure between generated and gold-standard requirement statements, operationalized via token-level and embedding-based similarity plus human judgment for acceptability; precise scoring penalizes missing mandatory clauses, incorrect constraints, and hallucinated obligations. Classification precision and recall measure the correctness of categorizing requirements into functional, non-functional, or constraint classes; precision captures false positive misclassifications while recall measures missed items of each class. An ambiguity reduction score quantifies how much the system's suggestions reduce linguistic and conceptual vagueness: it combines an automatic detection of vague phrases (using a predefined lexicon and model-based vagueness detectors) with human-rated scales indicating whether the generated clarifications would be sufficient to make the requirement testable.

Productivity improvement is measured in controlled user studies as the relative reduction in analyst effort—time to produce validated requirements and the number of revision cycles when using the MLLM-assisted workflow versus baseline tools; we report both absolute time savings and normalized productivity ratios across analyst experience levels. Complementary metrics include provenance coverage (percentage of requirements with at least one linked evidence snippet), correction rate (fraction of generated requirements needing human edits), and confidence calibration (correlation between model-reported confidence and actual correctness), which together give a holistic picture of utility, risk, and integration readiness.

ISSN: 3071-5628
PP: 01-20.

6. EXPERIMENTAL RESULTS

6.1. Baseline Comparison with Existing RE Automation Tools

In baseline comparisons, the MLLM-assisted pipeline is evaluated against established text-only models and rule-based extractors on the same multimodal test sets. Results typically show that when multiple modalities are available, the MLLM approach outperforms baselines on requirement accuracy and provenance coverage because it can leverage visual cues and audio context that text-only systems miss. For example, UI mockups that imply specific behaviors lead to richer, more concrete requirement text when processed multimodally, whereas text-only methods often generate underspecified statements. However, the improvement is not uniform: on very noisy inputs such as blurred sketch photos or poorly transcribed audio robust rule-based sanity checks sometimes prevent egregious hallucinations that the MLLM produces, revealing a trade-off between expressiveness and brittleness. The comparison also highlights where commercial RE automation tools excel (integration, traceability UI, and task routing) and where they fall short (limited multimodal interpretation), suggesting hybrid deployments where MLLMs augment rather than replace existing suites.

6.2. Case Study 1: Requirements From UI Sketches

In the first case study, we present a set of low-fidelity UI sketches captured during ideation sessions and task the MLLM with extracting functional requirements and acceptance criteria. The model successfully identifies concrete UI components (e.g., search bar, modal dialogs) and generates SRS-style requirement statements that map to those components, often adding reasonable inferred details such as expected input validation and error handling when the sketches imply them. Annotator review shows a high acceptance rate for component-level functional requirements, and provenance links to sketch regions help reviewers validate intent quickly. Failures mostly occur when sketches are highly ambiguous—scribbled arrows without labels—or when domain-specific widgets are present that the model mislabels; in these cases the system's confidence scores and the "needs human review" flags are useful safeguards. Overall, the case study demonstrates that multimodal extraction accelerates the capture of concrete, actionable UI-driven requirements from early design artifacts, reducing the need for repeated clarification cycles with designers.

6.3. Case Study 2: Requirements From Stakeholder Audio Transcripts

The second case study investigates how the pipeline handles recorded stakeholder interviews with diarized transcripts and paralinguistic cues. The MLLM summarizes long conversational segments into candidate requirements, extracting both explicit asks and implied constraints, while proposing clarifying questions for ambiguous or conditional statements. Compared to text-only baselines operating on transcripts, the multimodal setup that also ingests short video clips or referenced mockups produces requirement statements with higher fidelity because the model can resolve pronoun references and gestures that clarify intent. Human analysts found that the model's suggested clarifications often matched the follow-up questions they would have asked, reducing back-and-forth cycles. However, the study also surfaces sensitivity to transcript quality: when speech-to-text introduces errors or speaker diarization fails, the resulting requirement extraction suffers, underscoring the importance of robust preprocessing. Ethical considerations such as consent, redaction of sensitive content, and secure handling of audio are emphasized as part of deployment.

6.4. Case Study 3: Generating SRS documents automatically

The third case study evaluates end-to-end generation of a Software Requirements Specification document from a project bundle containing interviews, mockups, diagrams, and legacy constraints. The MLLM assembles a draft SRS with organized sections: functional requirements,

ISSN: 3071-5628
PP: 01-20.

NFRs, acceptance criteria, traceability annotations, and referenced evidence. Annotators rated the generated SRS drafts as substantially more complete than those produced by text-only baselines, especially in integrating visual evidence and linking requirements to design elements. The generated acceptance criteria were often usable with minor edits, and the embedded provenance sped up validation. Yet, the generated SRS sometimes included overconfidently stated constraints or omitted nuanced regulatory caveats present in legacy documents; these errors required domain expert correction. The case study highlights that while MLLM-generated SRS drafts can dramatically bootstrap specification work, final approval by qualified analysts remains essential to catch subtle domain-specific and compliance-related issues.

6.5. Discussion of findings

Across experiments and case studies, multimodal MLLMs show clear advantages in transforming heterogeneous RE artifacts into richer, better-grounded requirement artifacts and in reducing manual effort for initially drafting and organizing requirements. Strengths include higher requirement concreteness when visual cues are present, improved traceability through explicit provenance linking, and helpful clarification prompts that reduce iterative clarification cycles. However, the results also consistently surface limitations: sensitivity to input quality, occasional hallucinations or overconfident assertions, domain vocabulary mismatches, and the need for robust preprocessing to avoid garbage-in/garbage-out scenarios. Human-in-the-loop processes are essential—not only to validate and correct outputs but also to refine prompts and provide labelled feedback for targeted fine-tuning. From a practical perspective, the experiments indicate that an effective deployment strategy is hybrid: use MLLMs to accelerate and augment RE tasks while retaining human oversight and integrating safety nets such as provenance visibility, confidence thresholds, and domain-specific verification rules. The aggregated evidence supports the conclusion that MLLM-assisted RE can materially improve productivity and specification quality, provided organizations invest in preprocessing pipelines, human review workflows, and governance around data privacy and model behavior.

7. DISCUSSION

7.1. Benefits Of Using Mllms In RE

Multimodal large language models bring a pronounced set of advantages to Requirements Engineering by collapsing the long-standing divide between disparate artifact types and enabling synthesis that was previously laborious or error-prone. Chief among these benefits is the ability to triangulate intent: an MLLM can combine stakeholder utterances, sketches, and mockups to produce requirement statements that are richer, more concrete, and more closely tied to evidence than text-only summaries. This leads to better initial coverage of requirements (fewer missed constraints), more actionable acceptance criteria, and improved traceability because each requirement can be automatically annotated with the precise transcript lines, image regions, or diagram nodes that inspired it. Efficiency gains are also substantial; routine and repetitive tasks such as paraphrasing informal notes into SRS-style language, spotting obvious inconsistencies, and generating draft acceptance tests can be automated or semi-automated, freeing analysts to focus on higher-value judgment tasks. Additionally, MLLMs can serve as consistent assistants across projects, helping enforce style, template compliance, and completeness checks, which reduces variability introduced by individual authors and helps scale RE practices across distributed teams.

ISSN: 3071-5628
PP: 01-20.

7.2. Limitations And Risks

Despite the tangible advantages, deploying MLLMs in RE introduces several important limitations and risks that must be managed carefully. Hallucinations are a primary concern: models sometimes generate plausible-sounding requirements or constraints that are not grounded in any source artifact, which can lead to incorrect specifications being introduced if unchecked. Privacy issues are another major risk, because RE artifacts frequently contain sensitive commercial information, personally identifiable information (PII), or regulated data; feeding such inputs to cloud-hosted models or using them in training without robust safeguards can result in leaks or compliance violations. Domain mismatch presents a subtler but pervasive problem: general-purpose MLLMs may lack familiarity with specialized vocabulary, regulatory nuances, or legacy system idiosyncrasies, causing mistranslations or incorrect inferences that only domain experts would catch. Finally, there is a sociotechnical risk of over-reliance on AI: organizations may defer judgment to model outputs, allowing errors to propagate and degrading analysts' skills over time. Mitigations include engineering controls (provenance and confidence indicators, conservative thresholds for automated acceptance), procedural safeguards (human-in-the-loop approval for all final SRS artifacts, role-based governance), technical hardening (domain-adaptive fine-tuning, redaction and differential privacy for sensitive data), and operational audits (regular spot-checks, error logging, and rollback mechanisms). These measures reduce but do not eliminate risk, so responsible deployment requires a combination of model, process, and cultural interventions.

7.2.1 Hallucinations

Hallucinations occur when a model invents facts or requirements that are not supported by the input; in RE this can translate into spurious constraints, misattributed stakeholder intent, or fabricated dependencies that mislead designers and testers. Because hallucinations are often expressed confidently, they can be especially hazardous. Addressing them requires both model-level and workflow-level controls: prefer conservative prompt templates that require explicit citation of evidence, use provenance-aware extraction that refuses to produce a requirement unless linked to at least one verified source, employ secondary rule-based sanity checks that enforce domain invariants (e.g., "do not assert regulatory requirements without a legacy-doc citation"), and surface confidence and provenance metrics prominently in review interfaces so human validators can prioritize suspicious items. Periodic calibration exercises where analysts compare generated items against ground truth and feed corrections back to fine-tuning or prompt updates—also reduce hallucination frequency over time.

7.2.2 Privacy issues

Privacy concerns are multifaceted: raw interview audio may contain personal data; legacy documents may include customer-identifying records; and sharing artifacts with third-party model providers can violate contractual or regulatory obligations. Mitigations include architectural choices such as on-premises or private-cloud model hosting, automated redaction pipelines that detect and mask PII prior to ingestion, and strict access controls and encryption for stored artifacts and trace links. When models are hosted externally, contractual safeguards (data processing agreements, limited retention clauses) and technical measures (tokenization, homomorphic encryption research directions) should be used. Additionally, explicit consent processes must be embedded into data collection workflows so stakeholders understand how their words and artifacts will be used, and audit trails must be maintained to demonstrate lawful handling for compliance audits.

ISSN: 3071-5628
PP: 01-20.

7.2.3 Domain mismatch

Generalist MLLMs may misinterpret specialized terms or incorrectly prioritize constraints that are trivial or irrelevant in a given domain. To mitigate domain mismatch, organizations should adopt domain-adaptive strategies: curate small, high-quality fine-tuning datasets drawn from representative projects; incorporate domain lexicons and ontologies into the preprocessing pipeline; use hybrid pipelines where rule-based modules handle critical domain-specific checks; and make it easy for analysts to flag and correct domain-specific errors so the system can learn iteratively. Transparent uncertainty reporting where the model indicates lower confidence in unfamiliar terminology also helps prompt further human review rather than blind acceptance.

7.2.4 Over-reliance on AI

When teams increasingly rely on AI-assisted artifacts without preserving human judgment, there is a risk that subtle trade-offs, ethical considerations, or long-term architectural implications are overlooked. Preventing over-reliance requires procedural safeguards such as mandated human sign-off for high-impact requirements, role-based division of responsibility (AI for drafting, humans for authorization), and training programs to maintain and develop analysts' critical evaluation skills. Embedding explainability features—rationales for why a requirement was proposed and the evidence used—helps humans interrogate AI outputs rather than accepting them uncritically.

7.3. Ethical And Legal Considerations

The ethical and legal landscape surrounding AI-assisted RE is complex because requirements often touch on privacy, safety, fairness, and intellectual property. Ethically, practitioners must avoid encoding biased assumptions into requirements that could harm user groups or create discriminatory system behavior; MLLMs trained on biased corpora can inadvertently surface biased phrasing or priorities unless actively countered. Legally, requirements documents are increasingly part of compliance evidence in regulated domains (finance, healthcare, safety-critical industries), so provenance, immutability of audit logs, and chain-of-custody for decision rationale become legal necessities. Intellectual property issues arise when models are trained on proprietary corpora or when generated text reproduces proprietary phrasings from training data; clear policies and technical controls must manage what data is used for training and how outputs are licensed. To navigate these challenges, organizations should conduct ethical impact assessments prior to deployment, maintain transparent documentation of data sources and model behavior, implement bias-detection and mitigation pipelines, obtain informed consent for data usage, and coordinate with legal counsel to ensure contractual and regulatory compliance. Additionally, designing systems with explainability and contestability (easy ways for stakeholders to challenge and correct generated requirements) supports ethical governance and legal defensibility.

7.4. Best practices for reliable adoption

Adopting MLLM-assisted RE successfully requires a pragmatic blend of technical, process, and cultural practices. Technically, start with conservative system configurations: require explicit provenance for any automatically suggested requirement, enable configurable confidence thresholds for automated actions, and integrate domain-specific validators for high-risk or regulated items. Process-wise, create human-in-the-loop workflows where analysts review, correct, and approve AI outputs, and instrument these workflows to collect corrections for iterative improvement. Governance practices should mandate data handling policies (consent, retention, redaction), enforce role-based access controls, and schedule periodic audits of model performance and impact. Training and change management are essential; analysts need instruction on when to trust model suggestions,

ISSN: 3071-5628
PP: 01-20.

how to interpret confidence and provenance cues, and how to feed corrective feedback effectively. Start with low-risk pilot projects that have clear success criteria and incrementally expand deployment as the organization gains confidence and the system accumulates domain-specific learning. Finally, foster a culture of shared responsibility where AI is treated as an assistant, not an authority this mindset preserves human accountability while still reaping the productivity benefits of MLLMs.

8. APPLICATIONS

8.1. Large-scale enterprise software

In large enterprises, the volume of legacy documentation, cross-team dependencies, and regulatory obligations creates a perfect use case for MLLM-assisted RE. Multimodal models can rapidly extract constraints from sprawling legacy systems, reconcile conflicting requirements coming from multiple business units, and maintain traceability across centralized repositories. They help standardize requirement formats across geographically distributed teams, reduce duplication of effort by surfacing similar past requirements, and accelerate compliance reporting by automatically linking requirements to the originating legal or policy documents. For enterprise-scale adoption, emphasis should be placed on strong governance, private model hosting, and phased rollouts tied to well-defined integration points (e.g., feeding into DOORS or enterprise Jira instances), so that the system augments established PLM/ALM toolchains without disrupting critical operational controls.

8.2. Agile and DevOps environments

Agile and DevOps practices prize rapid feedback loops and continuous delivery, which align well with MLLM capabilities for continuous requirement extraction and revalidation. In sprint-driven workflows, MLLMs can transform meeting notes and quick design sketches into ready-to-refine user stories, generate acceptance criteria and suggested test cases, and maintain living trace links that keep requirements synchronized with implementation work and CI pipelines. They can also act as a co-pilot in backlog grooming proposing decomposition of large epics into smaller stories and estimating potential dependencies by analyzing referenced diagrams and mockups. However, in fast-paced environments it is critical that the model's recommendations are lightweight, clearly labeled as suggestions, and easily editable so teams can preserve agility and avoid overformalization.

8.3. Rapid prototyping and UI/UX design

Prototyping workflows, where designers iterate rapidly on sketches and wireframes, benefit from immediate generation of corresponding requirements and testable acceptance criteria. MLLMs speed up this loop by extracting expected behaviors from mockups, highlighting missing interactions, and producing accessibility-related NFR suggestions based on visual cues. This helps designers and product managers communicate intent to engineering with minimal friction and enables early detection of usability gaps that might become costly later. For creative environments, it's helpful to keep the AI's role as a facilitator offering precise, editable drafts and visual annotations—so designers retain control over conceptual decisions while benefiting from the rigor and traceability the model provides.

8.4. Safety-critical systems

Safety-critical domains (medical devices, avionics, automotive control systems) demand the highest levels of rigor, traceability, and regulatory compliance, making them both promising and challenging venues for MLLM-assisted RE. Here, MLLMs can accelerate the laborious process of extracting and maintaining requirements from dense regulatory texts, logs, and legacy certification

ISSN: 3071-5628
PP: 01-20.

documents, and they can help ensure traceability from high-level safety goals to low-level implementation and test artifacts. However, the stakes mean that automated suggestions must be treated as provisional: every generated requirement requires expert validation, formal verification where appropriate, and rigorous provenance evidence before acceptance. For these domains, best practice is to use MLLMs as specialized augmentations under strict controls—private models with domain-specific fine-tuning, extensive validation testbeds that measure hallucination rates under adversarial inputs, and integration with formal methods tools that can verify critical properties before changes propagate into certified artifacts.

9. FUTURE WORK

9.1. Domain-Specialized Multimodal RE Models

A promising direction is the development of domain-specialized multimodal models that are pre-trained or fine-tuned on corpora reflecting the vocabulary, artifact types, and regulatory constraints of particular industries—healthcare, automotive, finance, or industrial control—so that model outputs align more closely with domain expectations and legal obligations. Such specialization can be achieved by curating representative multimodal datasets (combining transcripts, mockups, diagrams, and legacy documents) and applying continual learning, adapter modules, or retrieval-augmented generation to inject domain knowledge without catastrophic forgetting. Domain-specialized models would reduce common failure modes like vocabulary mismatch and incorrect regulatory inferences, improve confidence calibration, and enable more precise provenance mapping to domain artifacts (e.g., linking a requirement to a specific clause in a regulation). Research should also investigate hybrid architectures that combine learned multimodal reasoning with symbolic or rules-based checks for domain invariants ensuring that high-assurance constraints are enforced deterministically while permitting the model to handle open-ended interpretation where appropriate. Finally, evaluating such models requires domain-aware metrics and pilot deployments with subject-matter experts to validate that the specialization yields practically meaningful improvements in accuracy, safety, and analyst trust.

9.2. Explainability improvements

Improving explainability is essential for trustworthy adoption of MLLMs in requirements engineering, because stakeholders must understand why a requirement was proposed and which evidence supports it before accepting it into an SRS. Future work should aim to design multimodal explanation techniques that go beyond attention heatmaps: explanations must be multimodal (highlighting transcript lines, image regions, and diagram nodes), structured (presenting causal chains or stepped reasoning), and user-centric (tailored to analysts, regulators, or architects). Techniques might include provenance-first generation requiring the model to produce a ranked list of explicit evidence snippets that justify each clause and counterfactual explanations that show how small changes in input would alter the generated requirement. Research should also explore human-centered presentation methods that surface uncertainty, confidence intervals, and alternative plausible interpretations, and assess how different explanation styles affect decision-making, error detection, and the speed of review. Importantly, explainability work must be evaluated empirically to ensure that explanations actually improve human ability to detect model errors and make informed judgments, rather than merely providing post-hoc rationalizations that mask underlying model brittleness.

ISSN: 3071-5628
PP: 01-20.

9.3. Benchmark datasets for multimodal RE

A major enabler for progress is the creation of standardized, publicly available benchmark datasets tailored to multimodal RE tasks; such datasets would provide common evaluation tasks (extraction, classification, traceability, ambiguity detection, prioritization) and a diverse set of artifact types (audio + transcripts, whiteboard photos, UI mockups, workflow diagrams, and legacy documents). Building these benchmarks requires careful attention to annotation quality (canonicalized requirements, provenance links, priority labels, and acceptance criteria), privacy-preserving data collection (anonymization and synthetic augmentation), and domain variety to test generalization. Benchmarks should include graded difficulty levels and adversarial examples that stress-test hallucination propensity and robustness to noisy inputs, and they should facilitate reproducible comparisons by providing standardized splits and evaluation scripts. Community-driven challenges, shared tasks at RE and SE conferences, and open leaderboards would accelerate progress, encourage the development of specialized evaluation metrics (e.g., provenance accuracy, ambiguity reduction effectiveness), and foster the exchange of best practices for dataset curation and ethical data use.

9.4. Real-time co-pilot systems for requirement analysts

The next generation of tools will move from batch processing of artifacts to interactive, real-time co-pilots that assist analysts during meetings, sketching sessions, or backlog grooming. These co-pilots would continuously listen (with user consent), capture multimodal cues, suggest candidate requirement statements on the fly, propose clarifying questions, and update traceability links as conversations evolve thereby reducing the latency between elicitation and formalization. Designing such systems requires research into low-latency multimodal inference, context management across sessions, and human-AI interaction patterns that minimize disruption while maximizing usefulness; ergonomics are crucial so that the co-pilot feels like a fluent collaborator rather than an intrusive recorder. Safety mechanisms such as strict on-device processing defaults for sensitive contexts, ephemeral buffers for transient notes, and explicit confirmation flows before persisting any generated requirement—must be built in to prevent inadvertent data leakage or premature acceptance of draft requirements. User studies should evaluate how real-time assistance affects stakeholder dynamics, the quality of elicited requirements, and analyst workload, ensuring that co-pilots enhance rather than degrade collaborative elicitation practices.

10. CONCLUSION

This paper has explored the promise and practicalities of integrating multimodal large language models into Requirements Engineering workflows. We presented a conceptual framework that positions MLLMs as central synthesis engines ingesting transcripts, sketches, mockups, diagrams, and legacy documents to produce candidate requirements, classifications, validations, and traceability links that materially reduce manual effort and increase specification concreteness. Empirical methodology and case-study-oriented evaluations highlight clear benefits: improved requirement grounding when visual and auditory cues are available, higher provenance coverage, and meaningful productivity gains when MLLMs are embedded in human-in-the-loop processes. At the same time, we emphasized critical caveats—hallucinations, privacy and compliance risks, domain mismatch, and sociotechnical hazards such as over-reliance—arguing that responsible adoption depends on technical safeguards, governance, and explainability. Looking forward, domain specialization, rigorous benchmarks, explainability research, and real-time co-pilot interfaces are promising avenues to make AI-assisted RE both more capable and more trustworthy. Ultimately, MLLMs can transform how requirements are elicited, documented, and maintained, but their value

ISSN: 3071-5628
PP: 01-20.

will be realized only when paired with disciplined processes and human oversight that preserve accountability and domain expertise.

REFERENCES

[1] Dalpiaz, F., Ferrari, A., Franch, X., & Palomares, C. (2018). *Natural Language Processing for Requirements Engineering: The Best Is Yet to Come*. IEEE Software, 35(5), 115–119.

[2] Ferrari, A., Dell'Orletta, F., Esuli, A., Gervasi, V., & Gnesi, S. (2017). *Natural Language Requirements Processing: A 4D Vision*. IEEE Software, 34(6), 28–35.

[3] Ahmed, H., Hussain, A., & Baharom, F. (2018). *The Role of Natural Language Processing in Requirement Engineering*. International Journal of Engineering & Technology, 7(4), 168–171.

[4] Lucassen, G., Robeer, M., Dalpiaz, F., van der Werf, J. M. E. M., & Brinkkemper, S. (2017). *Extracting Conceptual Models from User Stories with Visual Narrator*. Requirements Engineering, 22(3), 339–358.

[5] Dawood, O. S., & Sahraoui, A. E. K. (2017). *From Requirements Engineering to UML Using Natural Language Processing – Survey Study*. European Journal of Engineering and Technology Research, 2(1), 1–8.

[6] Garigliano, R., Perini, D., & Mich, L. (2018). *Which Semantics for Requirements Engineering: From Shallow to Deep*. Proceedings of the 1st Workshop on Natural Language Processing for Requirements Engineering (NLP4RE).

[7] Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.

[8] Wiegers, K. E., & Beatty, J. (2013). *Software Requirements* (3rd ed.). Microsoft Press.

[9] Robertson, S., & Robertson, J. (2012). *Mastering the Requirements Process: Getting Requirements Right* (3rd ed.). Addison-Wesley.

[10] Hull, E., Jackson, K., & Dick, J. (2011). *Requirements Engineering* (3rd ed.). Springer.

[11] Pohl, K. (2010). *Requirements Engineering: Fundamentals, Principles, and Techniques*. Springer.

[12] Cohn, M. (2004). *User Stories Applied: For Agile Software Development*. Addison-Wesley.

[13] Jurafsky, D., & Martin, J. H. (2019). *Speech and Language Processing* (3rd ed. draft). Pearson.

[14] Russell, S., & Norvig, P. (2016). *Artificial Intelligence: A Modern Approach* (3rd ed.). Pearson.

[15] Buitelaar, P., Cimiano, P., & Magnini, B. (2005). *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press.