From Meeting Audio to Requirements: A Generative-AI Multi-Agent Pipeline

Gabriel Ferreira Silva
Instituto de Informática, UFG
Goiânia, Brazil
ferreira.silva@discente.ufg.br

Heitor S. Rodrigues
Instituto de Informática, UFG
Goiânia, Brazil
heitor.rodrigues@discente.ufg.br

Paulo M. S. Rodrigues
Laboratório de Inovação Goiás - LIGO
Goiânia, Brazil
paulo.rodrigues@goias.gov.br

Jacson Rodrigues Barbosa
Instituto de Informática, UFG
Goiânia, Brazil
jacson_rodrigues@ufg.br

Abstract

Requirements elicitation from meetings is an expensive, manual, and error-prone activity, in which practitioners must simultaneously participate in the discussion and record it as formal artifacts. This work proposes a multi-agent system, based on generative artificial intelligence, that automates this process end to end, transforming meeting audio into structured requirements documentation. The architecture comprises four specialized agents responsible for automatic audio transcription, persona identification and user story generation following the INVEST criteria, requirements document generation in two selectable formats (the IEEE 830 standard and a client-specific format), and domain diagram synthesis; the agents communicate through a file-based sequential pipeline that provides loose coupling, fault isolation, and entry-point flexibility. The system was validated on a real government software project, using a manually produced official requirements document as ground truth. The quantitative evaluation covers transcription accuracy (WER of 6.3%), system-actor identification (F1 of 0.96) and the INVEST conformance of the generated stories (52.5% on the AMI corpus and 66.7% on the real domain), coverage against the official document (72.7%), and error propagation and cost across the pipeline, processing a nearly 100-minute meeting in about 11 minutes at a cost of US$ 0.72.

Keywords

Requirements Engineering, Multi-Agent Systems, Generative AI, Speech-to-Text, User Stories, Agentic Systems

1 Introduction

Requirements engineering (RE) is a fundamental phase of software development, bridging stakeholder needs and technical implementation. In most organizations, requirements are gathered through meetings facilitated by specialized professionals—requirements analysts, product owners, or scrum masters—whose role is to accurately capture complex, evolving discussions and transform them into formal artifacts [3]. This manual workflow introduces well-known risks: information loss caused by memory decay between the meeting and the documentation step, cognitive overload on the analyst who must simultaneously participate and record, and inconsistency in granularity and format across the resulting artifacts. Because requirements documents serve as the foundation for every subsequent development decision, poor or incomplete specifications translate directly into misaligned implementations, rework, and cost overruns.

Commercial meeting intelligence tools—such as Fellow.ai [6], Jamie [10], and Otter.ai—automate transcription and action-item extraction but produce general-purpose notes rather than formal software requirements artifacts. The translation from meeting discussion to a structured Software Requirements Specification (SRS) remains a manual, expertise-intensive step. In the academic literature, studies applying Large Language Models (LLMs) to RE address predominantly isolated subtasks: user story generation from existing textual requirements [5], conceptual model extraction [20], or requirements classification; few works explore complete end-to-end pipelines from raw meeting audio to stakeholder-specific, formally structured documentation [15].

This work addresses the gap through a *multi-agent architecture* in which four specialized agents collaborate to transform meeting audio into requirements documentation. Decomposing the pipeline into agents—rather than a single monolithic LLM prompt—provides concrete engineering advantages: *separation of concerns*, so each agent evolves independently; *entry-point flexibility*, letting users join the pipeline at any stage; *per-agent model selection*, matching LLM capability and cost to task complexity; and *observable error propagation*, making it possible to trace how transcription errors cascade downstream—a first-class engineering concern in agentic systems.

Initial experiments with Agents 1 and 2—conducted as part of this research—demonstrated the viability of persona identification and INVEST-compliant story generation. This paper extends that implementation to its *complete form*, adding Agents 3 and 4 for document generation and domain diagram synthesis, introducing two selectable output formats (IEEE 830 SRS and a client-specific format), and conducting real-world validation on actual software projects. It also replaces the preliminary qualitative evaluation with a quantitative harness covering transcription accuracy, extraction metrics, coverage against an official document, and error propagation analysis.

1.1 Research Questions

This paper addresses the following research questions:

RQ1. What is the automatic transcription accuracy (WER/CER) of Agent 1 for Portuguese meeting audio?

RQ2. How accurately does Agent 2 identify stakeholder personas and generate user stories (Precision/Recall/F1, INVEST compliance)?

RQ3. How completely does the generated document cover a manually produced official requirements specification (coverage/Recall)?

RQ4. How do errors propagate across pipeline stages, and how does per-agent model selection affect the cost–quality trade-off?

1.2 Main Contributions

This work contributes: a complete, open-source 4-agent system for end-to-end requirements elicitation from meeting audio, supporting two document formats (IEEE 830 SRS and a client-specific format); real-world validation on a government software project against a manually produced official requirements document as ground truth; a quantitative evaluation covering WER/CER (Agent 1), Precision/Recall/F1 with inter-rater agreement via Cohen's kappa (Agent 2), and document coverage (Agent 3); and a structured empirical analysis of cross-agent error propagation, with per-stage latency, API cost, and the residual human correction the pipeline still requires.

The remainder of this paper is organized as follows. Section 2 reviews related work on LLMs in RE, automated user story generation, and multi-agent systems. Section 3 presents the system architecture and the real-world case study. Section 4 describes the evaluation design. Section 5 reports results and discussion. Section 6 addresses threats to validity. Section 7 concludes with future work.

2 Related Work

2.1 Large Language Models in Requirements Engineering

The application of Large Language Models to requirements engineering has attracted growing interest [9]. Motger et al. [15] conducted a systematic mapping study identifying 62 publicly available datasets for LLM-based RE research, finding that only 2 address elicitation tasks and that 58 of 62 are English-only—a scarcity that directly motivates Portuguese-language validation such as that presented here.

Regarding artifact generation, Delgado et al. [5] compared N-gram heuristics with GPT-3 for user story generation from existing textual specifications (ROUGE-N = 0.46, BERTScore = 0.69 for the LLM). Ren et al. [20] showed that Chain-of-Thought prompting enables LLMs to extract conceptual class diagrams from user stories, and Naimi et al. [16] generated descriptive use case text from UML diagrams encoded as XML. Collectively, these works demonstrate LLM viability for generating RE artifacts, but share a common prerequisite: structured textual input. None addresses the extraction of requirements from unstructured meeting audio.

2.2 Speech Recognition for Meeting Transcription

Automated speech recognition (ASR) is the enabling technology for audio-based requirements elicitation. Radford et al. [19] introduced Whisper, a large-scale weakly supervised model demonstrating robust multilingual transcription across diverse acoustic conditions, including non-native speakers and background noise—characteristics common in real stakeholder meetings. Kasakowskij and Haake [12] developed T^3 Talk2Text, a near-real-time ASR system that captures full-group transcripts of virtual meetings for downstream analysis.

Despite these advances, ASR in requirements elicitation meetings faces domain-specific challenges: technical vocabulary, overlapping speech, and project-specific acronyms that general-purpose models may transcribe incorrectly. Such errors do not remain isolated—they propagate to downstream agents, a failure mode we analyze empirically as RQ4.

2.3 User Story Quality and the INVEST Framework

User stories are the predominant requirements artifact in agile development, following the format *"As a role, I want goal so that benefit"* [13]. Wake [21] proposed the INVEST heuristics—Independent, Negotiable, Valuable, Estimable, Small, Testable—as six criteria for assessing whether a user story is actionable and well-scoped.

Lucassen et al. [13] formalized the Quality User Story (QUS) framework comprising 13 criteria covering syntax, semantics, and pragmatics, and introduced AQUSA, an NLP-based tool that detects quality defects. Evaluated on 1,023 stories from 18 software organizations, AQUSA demonstrates the feasibility of automated quality assessment but requires manually written stories as input. Xu et al. [22] combined INVEST with a seven-criteria model to automate quality evaluation in agile pipelines. Our work extends this thread by generating user stories from meeting audio and evaluating them against INVEST using an LLM-judge protocol validated against human raters.

2.4 Multi-Agent Systems for Software Engineering

Multi-agent architectures have emerged as a productive paradigm for automating complex software engineering tasks. ChatDev [18] showed that specialized LLM agents can collaborate through structured dialogue across design, coding, and testing phases, and MetaGPT [8] encoded Standardized Operating Procedures into prompt sequences, assigning SE roles—product manager, architect, engineer, QA—to distinct agents. Luo et al. [14] applied an agentic pipeline to repository-level code documentation, surpassing human-authored baselines. A recent survey by Cai et al. [4] of 94 papers on LLM-based multi-agent systems for SE tasks found *Role-Based Cooperation* to be the most commonly adopted design pattern—the same pattern underlying our pipeline.

The closest work to ours is MARE [11], a multi-agent collaboration framework for the entire RE process. MARE decomposes RE into four tasks (elicitation, modeling, verification, specification) across five specialized agents, outperforming state-of-the-art RE baselines by up to 15.4% in F1. However, MARE operates from

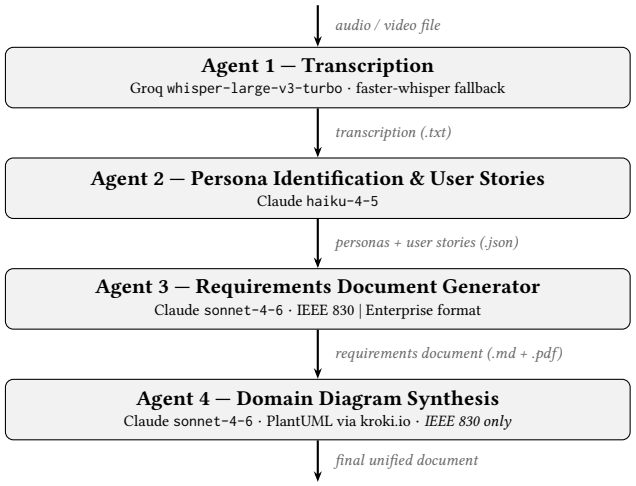

Figure 1: Four-agent pipeline. Agents communicate via a shared file system monitored by watchdog. Agent 4 is skipped in enterprise-format runs.

a textual *rough idea* provided by a human and simulates elicitation through agent-to-agent dialogue; it does not process meeting audio and therefore cannot be deployed in a post-meeting, fully automated scenario. Our system addresses precisely this gap.

2.5 Identified Gaps

Four gaps emerge from the reviewed literature. First, **no audio-to-SRS pipeline exists**: commercial tools [6, 10] produce general-purpose meeting notes rather than formal RE artifacts, and LLM-based RE systems such as MARE [11] consume structured text, not raw conversational audio. Second, prior LLM work on RE targets **isolated subtasks**—user story generation [5], use case description [16], conceptual modeling [20]—without integrating transcription, persona identification, and document generation into a single pipeline. Third, **real-world validation is scarce**: most studies use synthetic or controlled corpora, and few evaluate against a manually produced official requirements document from an actual project. Fourth, **error propagation** across agents is acknowledged but never empirically measured. This paper addresses all four through a 4-agent pipeline validated on real software projects with quantitative measurement of cross-agent error propagation.

3 Method

3.1 System Overview and Architecture

The proposed system comprises four specialized agents connected through a file-based sequential pipeline, as illustrated in Figure 1. Each agent monitors the output directory of its predecessor using the watchdog library and triggers processing automatically on each new file, giving loose coupling between stages, fault isolation, and auditable intermediate artifacts. Users may join the pipeline at any stage—supplying raw audio, an existing transcription, or a prior analysis JSON—reusing results across runs.

The pipeline supports two selectable output formats, chosen once per run: **IEEE 830**, the standard Software Requirements Specification structure used throughout this research; and **Enterprise**, a client-specific document layout requested by the industrial partner (Section 3.4). In enterprise runs, Agent 4 is skipped, as the client format does not include domain diagrams. The format affects only the document template and some agent prompts; all upstream stages (transcription, persona identification, user story generation) are format-agnostic.

3.2 Agent Design

3.2.1 Agent 1 — Meeting Transcription. Agent 1 accepts any audio or video file and produces a plain-text transcription. Processing begins with **ffmpeg** normalization to a 16 kHz mono FLAC file, reducing size while preserving transcription-relevant signal; if the result exceeds the Groq API size limit, it is split at silence boundaries into chunks, each transcribed independently and merged into a single output.

Transcription is performed by **Groq** running `whisper-large-v3-turbo` [7, 19], chosen for its combination of high multilingual accuracy and low latency via hardware-accelerated inference. If the Groq API is unavailable, the agent falls back automatically to **faster-whisper** running the `small` model locally [17], trading accuracy for offline availability.

3.2.2 Agent 2 — Persona Identification and User Story Extraction. Agent 2 reads the transcription and executes two sequential LLM calls using **Claude `haiku-4-5`** [2]. The first call identifies distinct stakeholder personas present in the conversation, extracting for each: name or identifier, role, defining characteristics, and representative dialogue excerpts. The second call generates persona-specific user stories following the INVEST criteria [21] in the format *"As a [persona], I want [goal] so that [benefit]."* Decomposing identification and generation into two calls reduces context interference and improves output structure. Results are saved as a typed JSON document that serves as the contract between Agents 2 and 3.

3.2.3 Agent 3 — Requirements Document Generation. Agent 3 reads the JSON produced by Agent 2 and generates a requirements document using **Claude `sonnet-4-6`** with **Jinja2** templates. In IEEE 830 mode, the output follows the standard SRS structure (introduction, general description, stakeholders, functional requirements, user stories table). In enterprise mode, it follows the client layout (functional features, detailed functional requirements with comments, business rules, use cases, non-functional requirements). The document is rendered to Markdown and then to PDF via **WeasyPrint**, using CSS Paged Media for table of contents with automatic page numbers.

3.2.4 Agent 4 — Domain Diagram Synthesis. Agent 4, active only in IEEE 830 runs, reads the Agent 3 document and the original Agent 2 JSON and uses **Claude `sonnet-4-6`** to extract domain entities and their relationships. The resulting PlantUML class diagram is rendered to PNG via the **kroki.io** public API and embedded in a final unified document consolidating all previous sections.

3.3 Model Selection Strategy

Model selection per agent reflects an explicit cost–quality trade-off. Agent 2 performs structured extraction with a well-defined output schema (typed JSON); the task is repetitive and runs on every meeting, making cost-per-call a primary concern—hence the use of the lighter `haiku-4-5` model. Agents 3 and 4 perform open-ended synthesis and document generation requiring stronger reasoning and language fluency; `sonnet-4-6` is used for both. Agent 1 uses a dedicated ASR model (Whisper) rather than a general-purpose

LLM, as speech recognition is a specialized task for which Whisper achieves substantially better accuracy and speed.

3.4 Real-World Case Study

To validate the system in an authentic setting, we established a partnership with the technology team of a Brazilian public sector agency. The team provided a recorded stakeholder meeting and a manually produced official requirements document for the same system, which serves as ground truth for evaluation (RQ3–RQ4).

The meeting involved **10 participants** and ran for **1 hour, 37 minutes, and 47 seconds**, conducted entirely in Brazilian Portuguese. The subject was the redesign of a budget balance management module in a government financial planning system. The raw video file (609 MB) was processed end-to-end by the pipeline: ffmpeg normalization, silence-based chunking, Groq transcription, persona identification, and enterprise-format document generation.

The enterprise format was adopted at the partner's request, as the organization does not use IEEE 830 internally—the requirement that motivated the dual-format architecture. The generated document and the official gold standard are compared in Section 4.

4 Evaluation Setup

The evaluation is organized around the four research questions, each associated with an agent or the pipeline as a whole. Four data sources support it: the **AMI corpus** [1] (seven annotated meetings, for RQ2); a **Portuguese audio corpus** (Mozilla Common Voice, for RQ1); the **real-world case study** (the recorded meeting and official document of Section 3.4, for RQ3–RQ4); a **second real case** (a meeting transcript from a software company, an additional RQ2 case); and a **third real case**, an internal meeting of a software company on a centralized authentication and access-control microservice, whose generated document was reviewed by a practitioner and adopted as the company's internal requirements specification (RQ4).

RQ1 (transcription). We measure the Word Error Rate (WER) and Character Error Rate (CER) of the produced transcription against a reference, over the Portuguese audio corpus, and report the Real-Time Factor (RTF); we compare Groq transcription with a local faster-whisper run. Reference and hypothesis are normalized (lowercased, punctuation removed, spelled-out numbers converted to digits) before scoring. **RQ2 (extraction).** Persona identification is evaluated via Precision, Recall, and F1 against a reference set of system actors. User story quality is measured by the conformance rate to the six INVEST criteria using an LLM-as-judge protocol. As automatic assessment is sensitive to the judge model, we compare two judges (a light model and a reasoning model) and adopt the latter as reference, validating it by blind manual annotation; agreement between the reference judge and the human annotator is quantified with Cohen's kappa. **RQ3 (coverage).** We compare the generated document with the manually produced official document through an item-by-item matching of requirements; the central metric is Recall (coverage) per requirement category. **RQ4 (error propagation and cost).** Error propagation is analyzed as a qualitative case study tracing how a transcription error manifests downstream; we additionally measure per-stage processing time and estimated API cost, and characterize, in the third real case, the corrections a practitioner still had to apply to the generated document before adopting it.

Table 1: Transcription accuracy (corpus-level WER/CER) and mean RTF of Agent 1, over 200 Portuguese Common Voice clips.

Engine	Samples	WER	CER	RTF
Groq (`whisper-large-v3-turbo`)	200	6.3%	2.3%	0.91
Local (*faster-whisper*, `small`)	200	13.5%	4.9%	0.56

5 Results and Discussion

5.1 RQ1 — Transcription Accuracy

Table 1 reports corpus-level WER/CER and mean RTF over 200 Portuguese Common Voice clips. Groq (`whisper-large-v3-turbo`) achieved a WER of 6.3% and CER of 2.3%, on par with the Whisper state of the art for Portuguese; 76% of clips were transcribed with no error, and the remaining errors are mostly phonetic. The local engine's WER was more than double (13.5%), evidencing the **accuracy–availability** trade-off. Both engines ran with RTF below 1.0. One caveat applies: Common Voice is read speech in short clips, so this WER **underestimates** the difficulty of real meetings (overlapping speech, noise, and the domain vocabulary whose failure mode we trace in Section 5.4).

5.2 RQ2 — Persona Identification and User Stories

Persona identification. Comparing the identified roles to reference roles, micro-averaged over the seven AMI meetings, Agent 2 achieved **Precision of 1.00, Recall of 0.92, and F1 of 0.96**: it introduces no spurious actors and recovers most discussed actors, missing only lightly emphasized profiles.

INVEST conformance. The quality of the 122 generated stories was assessed by conformance to the six INVEST criteria via an LLM-as-judge protocol; a story is conformant if it satisfies all six simultaneously. Conformance proved **highly sensitive to the judge model**: evaluating the same set with a light judge (`haiku-4-5`) and a reasoning judge (`sonnet-4-6`, with more detailed instructions), global conformance rose from **5.9%** to **52.5%** (Table 2). Inspecting the disagreements revealed that the light judge systematically underrated *Estimable*, *Small*, and *Testable*; we therefore adopt `sonnet-4-6` as the reference judge. Under it, AMI stories are mostly **valuable** (98.4%) and **independent** (100%), with non-conformances concentrated on granularity—confirming *Small* as the predominant failure mode. A blind manual annotation of 25 stories yielded close global rates (60% human vs. 48% judge) but story-by-story agreement of Cohen's $\kappa = 0.13$; the divergence concentrates on the testability of aesthetic requirements—"appealing design" is valuable but admits no objective acceptance criterion—a conceptual and defensible divergence rather than annotation noise.

Conformance in the real domain. AMI consists of design meetings for a consumer product, which favor aspirational stories. Applying the same protocol to the 24 government-partner stories, conformance rises to **66.7%**, with 100% of stories *testable* and *valuable* (Table 2). Requirements of a real corporate system are concrete and verifiable by construction—a story about querying the available balance by action, expense group, and funding source has an objective acceptance criterion, unlike the aesthetic stories of the AMI domain. Blind human validation of these 24 stories reinforces this:

Table 2: INVEST conformance by domain, under the reference judge (`sonnet-4-6`): AMI (122 stories), the government partner (24), and a software company (20). The light-judge column (`haiku-4-5`) is shown only for AMI, evidencing judge sensitivity.

Criterion	AMI (light)	AMI (ref.)	Gov. (ref.)	Company (ref.)
Independent	76.3%	100.0%	87.5%	60.0%
Negotiable	90.7%	82.0%	87.5%	100.0%
Valuable	96.6%	98.4%	100.0%	100.0%
Estimable	17.8%	73.8%	95.8%	15.0%
Small	21.2%	77.9%	87.5%	20.0%
Testable	16.1%	77.9%	100.0%	15.0%
Conformant	**5.9%**	**52.5%**	**66.7%**	**15.0%**

Table 3: Coverage (Recall) of the generated document against the official one (gold standard) in the comparable scope (item 2.1).

Category	Covered	Total	Recall
Functional requirements	2	5	40.0%
Business rules	3	3	100.0%
Use cases	3	3	100.0%
Overall	**8**	**11**	**72.7%**

$\kappa = 0.31$ (fair agreement), more than double AMI's, with observed agreement rising from 56% to 75%. A qualitative observation is also worth noting: some generated stories are semantically close but describe the same capability for **distinct actors** (e.g., the balance query requested by the Budget Manager and by the Balance Consultant); this is not redundancy but the correct granularity of a user story, always written from a single actor's viewpoint—an observation that reverses one level up, where the same interaction reproduced once per actor *is* redundancy between use cases (Section 5.4). The software company case, a platform-planning session closer to epics, scored only **15.0%**, reinforcing that INVEST quality depends heavily on the granularity of the source-meeting discourse.

5.3 RQ3 — Coverage Against the Official Document

We compared the generated document (enterprise format) with the partner's official document, produced manually and adopted as gold standard. A caveat precedes the comparison: the official document details **only one item** of the project (the Balance Maintenance screen, item 2.1) whereas the meeting covers items 2.1 to 2.7, so coverage is restricted to that **comparable scope**. Within it, both documents identified the same core points, including a highly specific detail—the flag for including a balance with a zeroed value or without appropriation—evidencing precise extraction. Recall from the item-by-item matching is reported in Table 3.

Overall coverage was **72.7%**, with 100% on business rules and use cases. The apparently low functional-requirements coverage (40%) deserves scrutiny: the three uncovered requirements are all of the "delete screen X" type, screen-manipulation operations whose capture would require perceiving **what was displayed on screen during the meeting**—a visual input the current audio-only architecture does not process. This is not an extraction failure but a modality limitation of the input: excluding those three, coverage of the functional content discussable by voice approaches 100%. The system faithfully recovers the **spoken** content, and the gaps concentrate on what exists only **visually**—mitigable in future work with multimodal input.

5.4 RQ4 — Error Propagation and Per-Agent Cost

In a sequential pipeline, an early error can propagate to all subsequent artifacts. While processing the partner meeting, Agent 1 transcribed the acronym "IPOF" as "HIPOF" in several passages. The behavior was subtle: in an earlier run, with Agent 3 on a lighter model, the incorrect term survived to the final document; with Agent 3 on the reasoning model (`sonnet-4-6`), the downstream agents **corrected** the acronym from context. The first agent's accuracy (RQ1) thus conditions the potential quality of the whole pipeline, but capable intermediate agents act as a **mitigation** layer.

To assess practical feasibility, Table 4 reports per-stage time and estimated API cost over the real meeting. The pipeline processed the nearly 1 h 40 min meeting in about **11 minutes** (677 s) at a cost of **US$ 0.72**. Transcription reached an RTF of about 0.018 (roughly 55× faster than real time) at a cost of US$ 0.065. Cost and time concentrate on Agent 3 (82% of the cost and 74% of the time), which **justifies the model-selection strategy**: the light model for Agent 2's repetitive extraction and the more expensive reasoning model reserved for document generation.

Residual human effort. Coverage and cost say nothing about the correction the pipeline still demands. In the third real case, the generated document (47 pages) was read end to end by a practitioner on that project, who annotated every correction judged necessary; an AI assistant then applied the wording as instructed, so the human effort lies in **reading and deciding**, not in editing. Nine corrections were raised over 71 numbered items and applied in three rounds; the document was adopted as the company's internal specification the next day, shrinking from 47 to 30 pages and from 18 to 11 use cases while its business rules grew from 26 to 34—review made it *more* precise, not merely shorter. The causes matter more than the count. Three corrections were **unreachable from the audio**: a component the team dropped after the meeting, and two requirements presupposing knowledge of the existing system inventory. The agent was faithful to what was said—fidelity to the transcript is not validity at delivery time, and no gain in transcription or extraction would close this gap. Two more resolved a genuine ambiguity in Portuguese (*"login único"* denotes both a unified credential and a single session, and the agent consistently chose the stronger reading), one supplied a missing requirement, one removed redundancy inherent to per-persona generation, one split a meeting covering two subjects, and one was a rendering defect. Only the last two are defects the architecture can fix unaided; the rest indicate that the realistic target is a **reviewed draft**, not a final document.

Table 4: Processing time and estimated API cost per agent, over the partner meeting (enterprise format; Agent 4 not executed).

Agent	Model	Time (s)	Cost (US$)
1 (Transcription)	`whisper-large-v3-turbo`	107	0.065
2 (Identification)	`claude-haiku-4-5`	71	0.063
3 (Document)	`claude-sonnet-4-6`	499	0.593
Total	–	**677**	**0.721**

6 Threats to Validity

Construct. INVEST conformance is a subjective construct: it is sensitive to the judge model (5.9% vs. 52.5%), and human–judge agreement on AMI was only slight ($\kappa = 0.13$), mainly due to divergence on the testability of aesthetic requirements. This is mitigated with a more capable judge, explicit instructions, and blind human validation in both domains. **Internal.** INVEST scoring and requirement matching (RQ3), when done by a single annotator, are subject to bias; this is mitigated, in RQ2, with a second rater and Cohen's kappa. The post-editing analysis of the third case rests on a single, non-blind reviewer who is an author of this paper, so no κ is reportable for it; its nine corrections count what one reader flagged in one pass over 71 items, not an exhaustive defect rate, and review time was not instrumented—only elapsed calendar time is reported. **External.** The validation rests on three real cases and the AMI corpus; results do not automatically generalize to all domains and languages, and only the first case has an official document, limiting RQ3 to a single domain. **Conclusion.** Given the small number of meetings, we avoid conclusions beyond what the data support.

7 Conclusion and Future Work

This paper proposed and evaluated a generative-AI multi-agent system that automates requirements elicitation end to end, transforming meeting audio into structured requirements documentation with two selectable output formats. The quantitative results support its feasibility: a WER of 6.3% for Portuguese (RQ1); an F1 of 0.96 for actor identification with INVEST conformance of 52.5% (AMI) and 66.7% (real domain) (RQ2); 72.7% coverage of the official document in the comparable scope, with gaps restricted to interface details absent from the audio (RQ3); and a nearly 1 h 40 min meeting processed in about 11 minutes for US$ 0.72, with evidence that capable intermediate agents mitigate error propagation and that the realistic target is a reviewed draft rather than a final document (RQ4). The main **limitations** are the three real cases (only one with an official reference document), the subjectivity of INVEST conformance, and the interface details the audio does not carry. As **future work**, we intend to broaden the validation with more meetings and official reference documents; mitigate error propagation with dedicated handling of technical vocabulary and verification steps between agents; segment multi-topic meetings into separate documents; incorporate multimodal input (screens and prototypes); and add a human-in-the-loop review over the intermediate artifacts.

Artifact Availability

The source code, evaluation scripts, and generated artifacts produced in this research are publicly available in an open repository (https://github.com/gabrielsfg/PFC1) under the MIT License.

References

[1] AMI Consortium. 2026. The AMI Meeting Corpus. https://huggingface.co/datasets/knkarthick/AMI. Accessed: June 2026.

[2] Anthropic. 2026. Claude Model Family. https://www.anthropic.com. Accessed: June 2026.

[3] Mohammad Ubaidullah Bokhari and Shams Tabrez Siddiqui. 2011. Metrics for Requirements Engineering and Automated Requirements Tools. In *Proceedings of the 5th National Conference on Computing for Nation Development (INDIACom)*. New Delhi, India, 10–11.

[4] Yangxiao Cai, Ruiyin Li, Peng Liang, Mojtaba Shahin, and Zengyang Li. 2025. Designing LLM-based Multi-Agent Systems for Software Engineering Tasks: Quality Attributes, Design Patterns and Rationale. *ACM Transactions on Software Engineering and Methodology* (2025). arXiv:2511.08475.

[5] Andrea Delgado et al. 2024. AI-Driven User Story Generation. In *Proceedings of the 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 1–12. doi:10.1109/SANER60148.2024.00000

[6] Fellow.ai. 2025. Fellow: AI Meeting Assistant. https://fellow.ai. Accessed: June 2026.

[7] Groq, Inc. 2025. Groq API: Fast AI Inference. https://groq.com. Accessed: June 2026.

[8] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2024. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*.

[9] Max Hort, Fernando Vallecillos-Ruiz, and Leon Moonen. 2025. Large Language Models: A Survey of Surveys. (2025). Preprint. https://hal.science/hal-05341748.

[10] Jamie. 2025. Jamie: AI Note-Taking Assistant. https://www.meetjamie.ai. Accessed: June 2026.

[11] Dongming Jin, Zhi Jin, Xiaohong Chen, and Chunhui Wang. 2024. MARE: Multi-Agents Collaboration Framework for Requirements Engineering. *arXiv preprint arXiv:2405.03256* (2024).

[12] Thomas Kasakowskij and Joerg M. Haake. 2025. T^3 Talk2Text: A Model for Near Real-Time Voice Transcription in Virtual Group Meetings. *Discover Education* 4 (2025), 146. doi:10.1007/s44217-025-00568-6

[13] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. 2016. Improving Agile Requirements: The Quality User Story Framework and Tool. *Requirements Engineering* 21, 3 (2016), 383–403. doi:10.1007/s00766-016-0250-x

[14] Qinyu Luo, Yining Ye, Shihao Liang, Zhong Zhang, Yujia Qin, Yaxi Lu, Yesai Wu, Xin Cong, Yankai Lin, Yingli Zhang, Xiaoyin Che, Zhiyuan Liu, and Maosong Sun. 2024. RepoAgent: An LLM-Powered Open-Source Framework for Repository-level Code Documentation Generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 436–464.

[15] Quim Motger, Carlota Catot, and Xavier Franch. 2025. ShaRE your Data! Characterizing Datasets for LLM-based Requirements Engineering. In *arXiv preprint arXiv:2510.18787*.

[16] Lahbib Naimi, El Mahi Bouziane, Abdeslam Jakimi, Rachid Saadane, and Abdellah Chehri. 2024. Automating Software Documentation: Employing LLMs for Precise Use Case Description. In *Procedia Computer Science*, Vol. 246. Elsevier, 1346–1354. doi:10.1016/j.procs.2024.09.568

[17] OpenNMT. 2024. CTranslate2: Fast Inference Engine for Transformer Models. https://github.com/OpenNMT/CTranslate2.

[18] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative Agents for Software Development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*. Association for Computational Linguistics.

[19] Alec Radford, Jong Wook Kim, Tao Xu, et al. 2022. Robust Speech Recognition via Large-Scale Weak Supervision. *arXiv preprint arXiv:2212.04356* (2022).

[20] Shuaicai Ren, Hiroyuki Nakagawa, and Tatsuhiro Tsuchiya. 2024. LLM-Based Class Diagram Derivation from User Stories with Chain-of-Thought Promptings. In *Proceedings of the 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 1376–1381. doi:10.1109/COMPSAC61105.2024.00181

[21] Bill Wake. 2003. INVEST in Good Stories, and SMART Tasks. https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Accessed: June 2026.

[22] Xiangqian Xu, Yajie Dou, Liwei Qian, Zhiwei Zhang, Yufeng Ma, and Yuejin Tan. 2023. A Requirement Quality Assessment Method Based on User Stories. *Electronics* 12, 9 (2023), 2155. doi:10.3390/electronics12092155