BPMN-Based Design of Multi-Agent Systems: Personalized Language Learning Workflow Automation with RAG-Enhanced Knowledge Access

Hedi Tebourbi , Sana Nouzri * , Yazan Mualla * , Meryem EL Fatimi , Amro Najjar , Abdeljalil Abbas-Turki , Mahjoub Dridi

Posted Date: 16 July 2025

doi: 10.20944/preprints202507.1291.v1

Keywords: MAS; language learning; BPMN; RAG; low-resource languages; Luxembourgish; speech recognition; educational pedagogy

 doi:10.20944/preprints202507.1291.v1

Article

BPMN-Based Design of Multi-Agent Systems: Personalized Language Learning Workflow Automation with RAG-Enhanced Knowledge Access

Hedi Tebourbi [1], Sana Nouzri [1], Yazan Mualla [2,*], Meryem EL Fatimi [3], Amro Najjar [4], Abdeljalil Abbas-Turki [2] and Mahjoub Dridi [2]

[1] University of Luxembourg, 4365 Esch-sur-Alzette, Luxembourg
[2] Université de Technologie de Belfort-Montbéliard, UTBM, CIAD UR 7533, F-90010 Belfort, France
[3] Cadi Ayyad university, Bd Abdelkrim Al Khattabi, Marrakech 40000, Maroco
[4] Luxembourg Institute of Science and Technology, 4362, Esch-sur-Alzette, Luxembourg
* Correspondence: yazan.mualla@utbm.fr

Abstract

The intersection of Artificial Intelligence (AI) and education is revolutionizing learning and teaching in this digital era, with Generative AI and large language models (LLMs) heralding even greater possibilities for the future. This paper presents a novel approach to design a personalized language learning by combining Business Process Model and Notation (BPMN) with Multi-Agent System (MAS) architectures and Retrieval-Augmented Generation (RAG) knowledge bases. Addressing the specific challenges of teaching Luxembourgish, a low-resource language, we design a modular system where BPMN diagrams play a central role in designing and orchestrating the workflows of intelligent agents. Each agent is responsible for a specific learning activity, such as reading, listening, grammar, or conversation, and is equipped with access to a dynamically retrieved, context-rich knowledge base powered by RAG. To ensure realism in learner interaction, we integrate speech-to-text and text-to-speech technologies, creating an immersive, human-like learning environment. The system simulates intelligent tutoring through agents' collaboration and dynamic adaptation to learner progress. We demonstrate our method through a Luxembourgish learning platform that integrates GPT-based agents and educational content from textbooks of the National Institute of Languages of Luxembourg. Our Results demonstrate that BPMN functions not merely as a modeling tool but as a pivotal framework for designing intelligent and adaptive agent workflows in language learning systems. By structuring the sequence, logic, and interdependencies of diverse learning activities, BPMN enables the development of coordinated, goal-oriented agent behaviors. This systematic design approach ensures that each agent, whether dedicated to reading, listening, grammar, or conversation, operates within a coherent and pedagogically aligned flow, thereby supporting greater personalization, learner engagement, and instructional effectiveness.

Keywords: MAS; language learning; BPMN; RAG; low-resource languages; Luxembourgish; speech recognition; educational pedagogy

1. Introduction

Recent advances in conversational Artificial Intelligence (AI), particularly through systems like ChatGPT, have demonstrated the potential for AI-driven language learning [2]. However, current AI-driven language learning applications face significant limitations: they often rely on single-agent architectures that struggle to manage the complexity of comprehensive language education, and they frequently produce content errors or "hallucination" that undermine pedagogical effectiveness [4,5].

These challenges are particularly acute for low-resource languages like Luxembourgish, where training data scarcity exacerbates model limitations. With Luxembourgish being the main language

 doi:10.20944/preprints202507.1291.v1

of only 48.9% of Luxembourg's population as of 2021, down from 55.8% in 2011, and spoken by approximately 275,000 people as their primary language [14], traditional large language models (LLMs) often fail to generate accurate, contextually appropriate educational material [17,22,25].

Furthermore, current language learning applications utilizing LLMs predominantly focus on vocabulary acquisition through role-playing conversations and providing immediate feedback derived from model-generated content [1]. This method, however, can result in exposure to inaccuracies or model "hallucinations", and fails to address the comprehensive nature of the language learning process, which extends beyond conversational practice to include reading comprehension, listening skills, and grammatical competency [3]. Effective language acquisition necessitates robust pedagogy, efficient teaching methodologies, reliable content, and a supportive teacher-student dynamic [6]. While MAS have shown promise in educational contexts through distributed task management [10], their application to language learning for low-resource languages remains underexplored.

Based on our previous works [24,48][1], this article addresses these challenges through a novel integration of three key technologies: Business Process Model and Notation (BPMN) for workflow transparency and specification [9], MAS for distributed task execution, and Retrieval-Augmented Generation (RAG) for auditable knowledge access and enhanced content accuracy [21]. By modeling language learning as a structured business process, where BPMN diagrams explicitly define pedagogical workflows that are then executed by specialized agents, each focusing on specific learning aspects (conversation, reading, listening, grammar), we ensure a comprehensive learning experience that addresses the holistic nature of language acquisition. By grounding agent responses in a RAG-enhanced knowledge base built from validated educational materials from the National Institute of Languages of Luxembourg (INL), we significantly reduce content errors while maintaining pedagogical coherence.

Explainable Artificial Intelligence (XAI) is a field that aims to provide techniques, models, and methods for developing XAI-based systems [32–35]. These systems enable users and other human actors to better understand AI's decision-making, which, in turn, can improve factors such as trust and transparency [36–38], particularly in data-driven AI [39–41].

Our approach extends beyond traditional Intelligent Tutoring Systems (ITS) [7] by implementing dynamic agent collaboration rather than rule-based responses, enabling more adaptive and personalized learning experiences while prioritizing explainability through BPMN's transparent workflow modeling. By formalizing agent interactions and decision logic in BPMN diagrams, we provide educators and learners with auditable, human-interpretable insights into AI-driven pedagogical choices (e.g., activity sequencing, error correction pathways). This explainability addresses the "black-box" limitations of conventional LLMs, aligning with XAI principles of trust and accountability in educational AI. The integration of Speech To Text (STT) and Text To Speech (TTS) technologies further enhances the immersive learning environment, addressing the conversational practice needs critical for language proficiency development. Furthermore, the opaque decision-making of LLMs hinders pedagogical trust. Our framework inherently addresses this by: (i) decomposing complex workflows into auditable agent tasks, (ii) grounding responses in traceable knowledge sources, and (iii) enabling human-in-the-loop validation of XAI for educational systems. The main contributions of this paper are:

- A novel BPMN-to-MAS transformation methodology that converts pedagogical workflows into executable MAS, bridging formal process modeling with AI-driven education.
- Integration of RAG technology to ensure accurate, contextually grounded language instruction while mitigating LLM hallucinations.
- Implementation of a complete Luxembourgish learning platform (A1—B2) with React/ FastAPI frontend, LangGraph core, ChromaDB vector store, and STT/TTS pipelines.

[1] This article is a revised and expanded version of these papers.

 doi:10.20944/preprints202507.1291.v1

- Empirical evaluation showing strong response accuracy (RAGAs: Context Relevancy = 0.87, Faithfulness = 0.82, Answer Relevancy = 0.85) and high learner satisfaction in a pilot (85.8% ease-of-use, 71.4% engagement).
- A generalizable framework for low-resource language education that combines formal process modeling, distributed AI agents, and knowledge-grounded generation.

2. Related Work

2.1. BPMN for Educational Process Modeling

BPMN has been widely adopted to formalize and communicate complex workflows in education. García-López et al. [16] demonstrated how BPMN diagrams capture pedagogical sequences—such as lesson plans and assessment flows—improving collaboration between instructors and system designers. Similarly, Costa and Silva [11] applied BPMN to model adaptive e-learning paths, enabling conditional branching based on learner performance while improving course reusability and personalization. Nevertheless, these efforts typically stop at static documentation and do not produce executable, adaptive process definitions that integrate with AI agents at run time.

2.2. Multi-Agent Systems in Education

Intelligent tutoring systems have long leveraged MAS to distribute educational tasks among specialized agents. Wooldridge and Jennings's foundational survey [28] laid out the theory of agent collaboration, and Ivanova et al. [12] later proposed an agent-oriented architecture for strategy-based e-learning with distinct agents handling content delivery, assessment, and learner tracking. More recent LLM-based MAS frameworks—such as AutoGen [23] and LangGraph [19]—provide generic scaffolding for multi-agent orchestration but have not been tailored to formal pedagogical workflows or low-resource language settings.

2.3. Retrieval-Augmented Generation

RAG combines neural generation with external retrieval to ground outputs in factual documents. Lewis et al. [21] first showed that RAG significantly reduces hallucinations in knowledge-intensive NLP tasks. In education, Wang et al. [27] applied RAG to AI tutoring systems and reported a 40% reduction in factual errors and a 25% increase in answer relevance. However, existing RAG-based solutions do not incorporate formal process models or MAS, leaving a gap in end-to-end, workflow-driven educational AI.

2.4. LLM-Powered Language Learning Chatbots

Generative AI chatbots like ChatGPT have become popular for language practice. Belda-Medina and Calvo-Ferrer [2] showed that LLM conversational partners can boost vocabulary retention. Cavojský [1] integrated GPT-3 in classroom settings, yielding fluency gains, but also observed persistent hallucinations. Huang et al. [5] provide a taxonomy of these hallucinations, and Rzepka et al. [26] analyzed their impact on learner trust. Commercial platforms (e.g. Duolingo Max [15]) now harness GPT-4, yet still emphasize gamified vocabulary gains over a comprehensive, skills-based curriculum.

2.5. Technologies for Low-Resource Languages

Low-resource languages like Luxembourgish face acute data scarcity. Lavergne et al. [20] surveyed existing Luxembourgish NLP tools and found major gaps in corpora and models. Multilingual transformers (XLM-R [29], mBERT [30]) afford zero-shot capabilities but often underperform in specialized domains. For speech, Whisper-based STT fine-tuned on RTL.lu achieves WERs of 18—28% [31], yet has seen little application in structured educational systems.

2.6. Research Gap

While prior work has independently explored: BPMN for high-level educational design [16], MAS for distributed tutoring [12,28], RAG for grounding LLM outputs [21,27], LLM-based chatbots

 doi:10.20944/preprints202507.1291.v1

for language practice [1,2], and low-resource language models [20,31], no existing system brings all of these together into a cohesive, executable framework that:

1. Uses BPMN to specify pedagogical workflows,
2. Orchestrates specialized LLM-powered agents via MAS,
3. Grounds all content in vetted external knowledge with RAG,
4. Incorporates real-time voice interaction (STT/TTS) for a low-resource language.

Our work bridges this gap for Luxembourgish by compiling BPMN into an LLM-agent graph, integrating RAG-enhanced vector retrieval, and embedding STT/TTS pipelines into the MAS—delivering a structured, adaptive, and pedagogically coherent learning experience.

3. System Design and Architecture

Our system design is built around the integration of three core components that collectively enable a modular, intelligent, and process-driven learning environment:

- **BPMN** is used to define and structure the high-level learning workflow. Each BPMN task represents a discrete educational activity (e.g., "Grammar Practice"), providing a visual and organized representation of the learning journey.
- **MAS** executes the modeled processes. Each agent within the MAS is domain-specific, with specialized roles such as Conversational Agent, Grammar Agent, Reading Agent, and Listening Agent. These agents work autonomously yet collaboratively, each handling a distinct aspect of the learning experience.
- **RAG** equips each agent with access to relevant educational content. This is achieved through vector stores constructed from INL textbooks and OCR-processed materials, allowing agents to retrieve and generate accurate, knowledge grounding responses.

The architecture, as shown in Figure 1 (MAS Architecture part), draws inspiration from the structural design of language learning books proposed by INL, which categorize activities into well-defined types: ***SchwÄtzt!: Speak up!, Lauschtert!: Listen!, Liest!: Read!, Kombinéiert!: Combined!, Notéiert!/Schreift!: Note!/Write!, KrÄizt un!: Cross on!, and ënnerstrÄichen!: Underline!*** Each type represents a focused mode of interaction, and this pedagogical segmentation has influenced our system's modularity.

Instead of relying on a single, monolithic language model to handle all educational tasks, our architecture embraces a multi-agent LLM approach. This decentralized design mirrors the textbook structure, assigning specific learning tasks to dedicated agents, thus allowing each LLM-based agent to focus on a single modality or skill. This results in more targeted, efficient, and context-aware instruction. By distributing responsibilities among specialized agents, the system enables cooperative problem-solving and interactive learning, fostering a more dynamic and adaptable educational experience.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2025 doi:10.20944/preprints202507.1291.v1

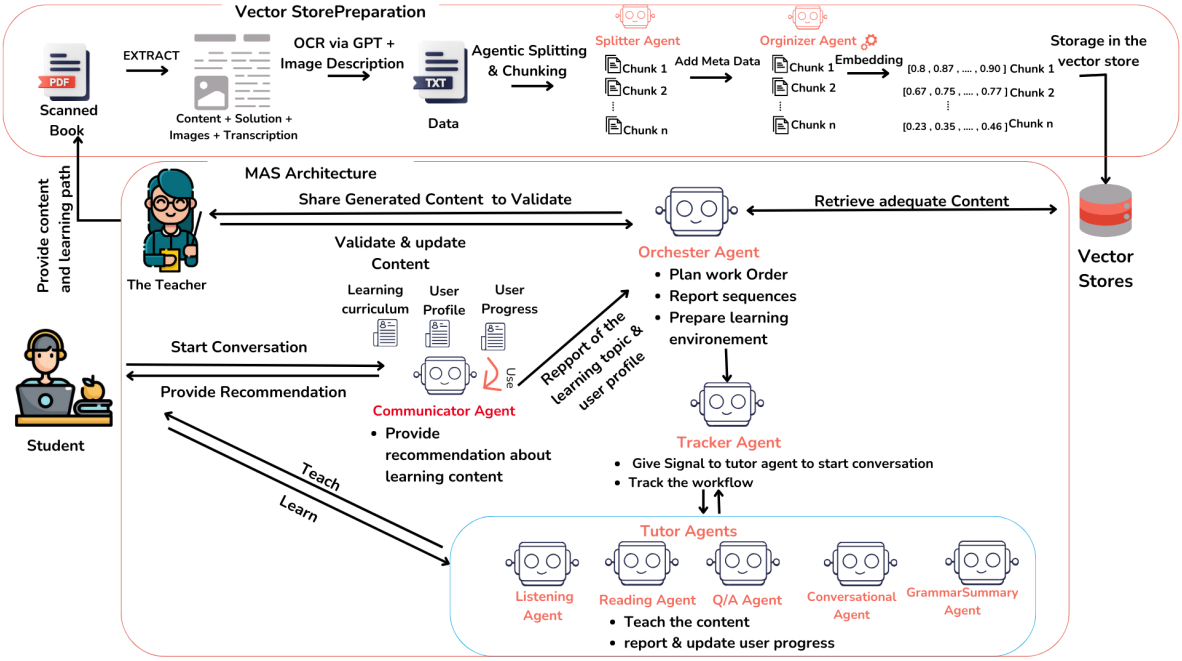

Figure 1. Collaborative Learning via Agent Communication and RAG-Based MAS for Language Learning

3.1. Core Agent Roles and Responsibilities

1. **Communicator Agent:** Serves as the primary interface between the system and the user. Its responsibilities include:
   - Interacting with users in natural language.
   - Recommending learning activities based on the user's preferences, progress, and performance.
   - Personalizing the learning path according to user goals and pacing.
2. **Orchestrator Agent:** Functions as the system's manager, responsible for:
   - Managing the overall workflow of the learning session.
   - Retrieving and sequencing appropriate content for each session.
   - Coordinating the actions of the specialized tutor agents to ensure cohesive learning progression.
3. **Tracker Agent:** Monitors and ensures the correct execution of the learning workflow. Its tasks include:
   - Signaling the activation of Tutor Agents at the appropriate stages.
   - Waiting for each agent to complete its task before initiating the next.
   - Maintaining the flow and timing of the session until completion.
4. **Specialized Tutor Agents:** At the heart of the platform are the Tutors Agents, each designed to focus on a specific aspect of language learning. These agents reflect the structure found in Luxembourgish textbooks, where different learning activities are categorized to target distinct language skills:
   - **Conversational Agent:** Focuses on spoken language skills through interactive conversation. Activities include Role-playing scenarios, Vocabulary-focused dialogues, and Real-time feedback for pronunciation and fluency.
   - **Reading Agent:** Enhances comprehension and vocabulary acquisition through reading tasks, assists with reading accuracy and understanding, and encourages summarization and contextual vocabulary usage.

 doi:10.20944/preprints202507.1291.v1

- **Listening Agent:** Builds listening comprehension through repetition and follow-up exercises, listening-based actions, and feedback to improve recognition and application of new vocabulary.
- **QA (Question & Answer) Agent:** Provides interactive tasks and exercises based on book-inspired formats such as Kombinéiert!, Notéiert!/Schreift!. It presents quiz-style questions, evaluates responses and delivers detailed explanations, and offers corrective and motivational feedback.
- **GrammarSummary Agent:**
  Specializes in grammatical instruction by delivering concise and targeted grammar rules, supporting correct grammatical usage in user outputs.

5. **Human Tutor** A teacher will be involved to ensure the linguistic accuracy and coherence of the learning materials. This includes verifying the alignment between text content, audio transcriptions, and visual illustrations. The teacher also validates the relevance and pedagogical sequencing of the content retrieved by the Orchestrator, ensuring it is appropriate for the user's proficiency level and learning objectives.

3.1.1. Knowledge Provenance

Every RAG-augmented response in our system is transparently linked back to its original textbook source via the metadata produced by the Organizer agent (see Figure 2). Prior to storage in the vector database, each chunk is annotated with the following fields, extracted verbatim from the INL A1—B2 materials and cross-checked against the scanned PDF pages, to ensure exact provenance: Thema, Kategorie, Agent, and Inhalt (content).

When an agent generates a response, it appends these metadata attributes (e.g. "Thema: Wéi heescht Dir?", "Kategorie: Dialogs and name spelling exercises") alongside the answer. This mechanism guarantees that learners—and auditors—can always trace each generated utterance back to the precise location within the original INL textbook.

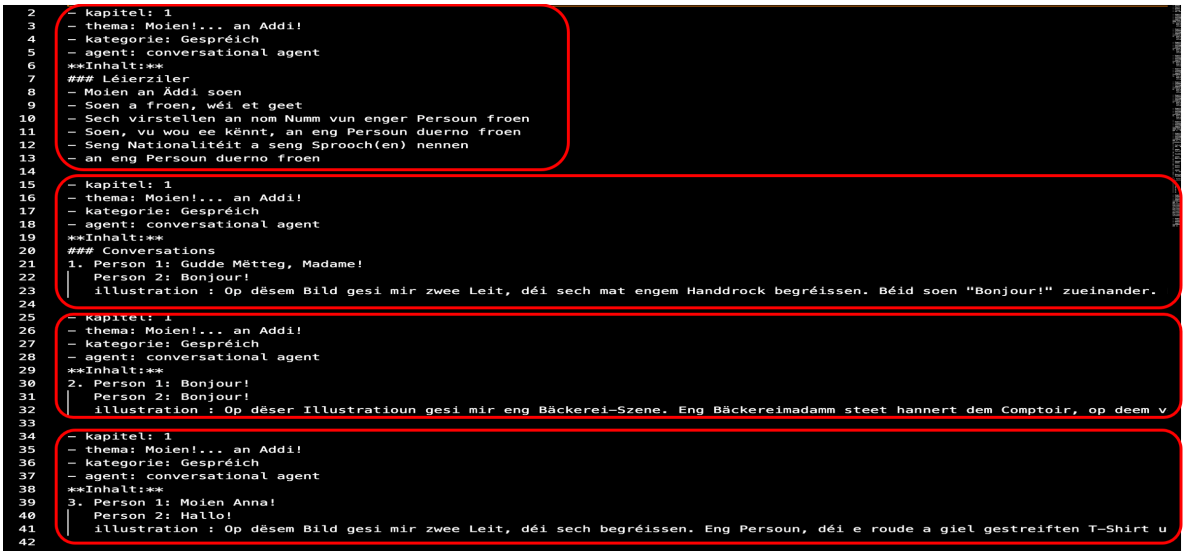

Figure 2. Organizer Agent output: textbook chunks annotated with pedagogical metadata and source references.

3.1.2. Human Auditing

Validators (teachers) review:

- The raw BPMN-defined sequence of activities,
- LangSmith logs of each agent's decision context,
- The alignment between chunk metadata and delivered exercises.

This ensures pedagogical sequencing remains faithful to expert intent.

 doi:10.20944/preprints202507.1291.v1

3.1.3. Workflow Transparency

BPMN diagrams (Figure 4 and Figs. A1-A4 in Appendix A) explicitly document every decision point, for example:

$$\text{IF pronunciation\_error\_count} > 2 \longrightarrow \text{Re-prompt}$$

This makes the branching logic visible to both developers and instructors.

3.2. *Explainability Mechanisms*

We embed an interactive explainability layer into our multi-agent workflow, ensuring transparency in what would otherwise be a "black-box" system. By integrating LangSmith [42], educators gain detailed insights into the rationale behind each adaptive decision made by the LangGraph engine, creating clear audit trails from learner data to agent actions.

3.2.1. LangSmith Role

LangSmith is an interactive observability and debugging platform provided by LangChain Inc. [42]. It captures, stores, and visualizes every prompt, tool call, router decision, and state change across a LangGraph-driven workflow. Through its web-based dashboard, educators and developers can:

- Explore complete prompt histories for each agent node,
- Inspect intermediate state variables and message payloads,
- Visualize conditional routing paths and loop iterations,
- Search and filter on specific student interactions or decision predicates.

By integrating LangSmith, our system turns opaque agent behavior into an auditable, human-readable trace, facilitating both technical debugging and pedagogical review.

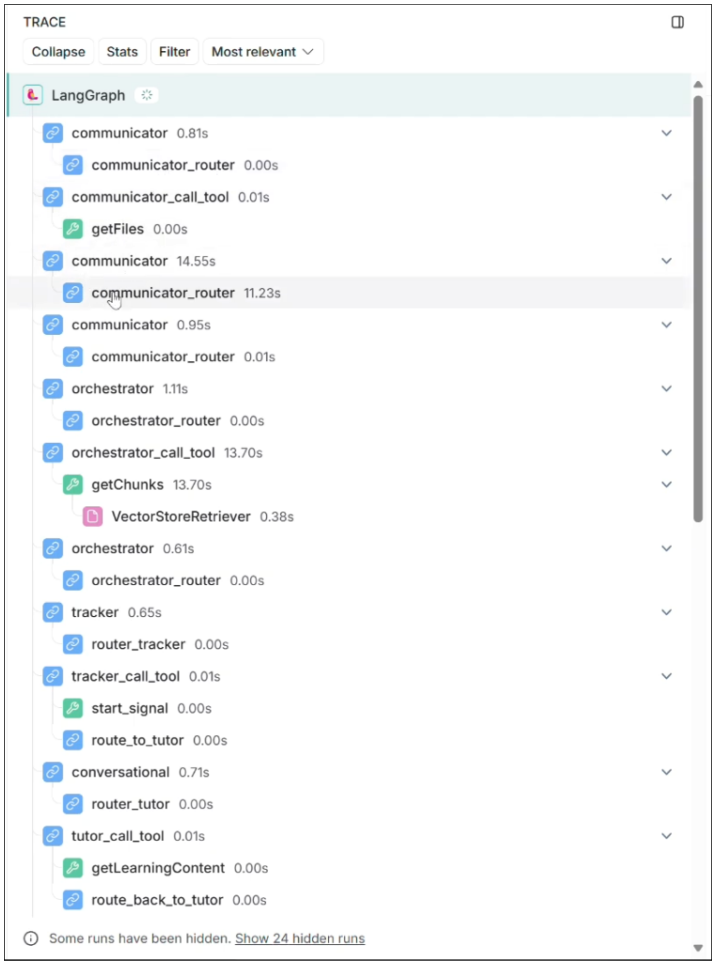

Figure 3. LangSmith trace of decision steps for a sample learner session.

 doi:10.20944/preprints202507.1291.v1

4. BPMN to MAS Transformation

4.1. BPMN Modeling of Learning Workflows

To capture how our agents communicate, cooperate, and supervise one another, we represent their interaction flow with a BPMN diagram (see Figure 4). By modeling our MAS in BPMN, we make the system's logic explicit, easily understandable to both technical and non-technical stakeholders, and readily analyzable for correctness and optimization.

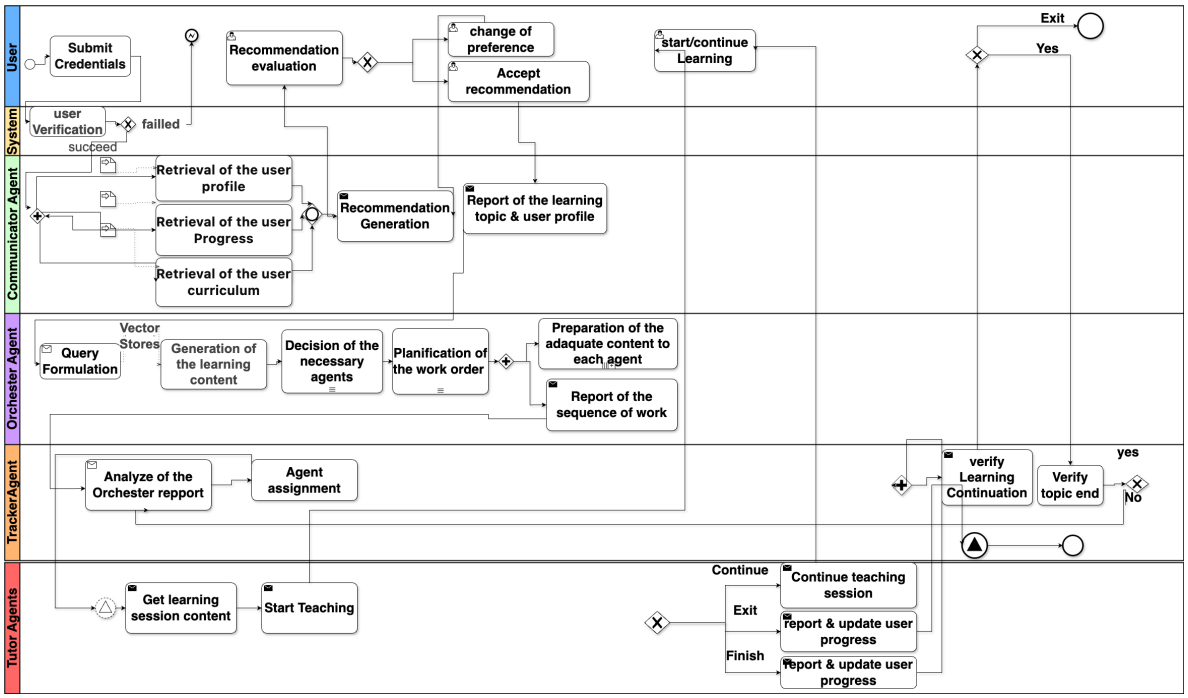

Figure 4. BPMN diagram for the agent interaction

4.1.1. Top-Level Orchestration Diagram (Figure 4)

Table 1. Pools and Lanes in the Top-Level BPMN Orchestration

Pool / Lane	Role / Responsibility
User	Human learner interacting via the UI.
Communicator Agent	Retrieves profile data; proposes personalized learning paths; emits RAG queries.
Orchestrator Agent	Fetches and validates content; plans which tutor agents to invoke and in what order.
Tracker Agent	Drives step-by-step activation of tutor agents; logs completion or early-exit signals.
Tutor Agents	Swimlane for specialized tutors (Conversation, Reading, Listening, QA, GrammarSummary).

Pools and lanes correspond to the BPMN swimlanes shown in Figure 4.

1. **Start Event (Message)** The User logs in, triggering a message start event in the Communicator lane.
2. **User Data Retrieval (Service Tasks)** In the Communicator lane, three service tasks are retrieved:
   - `UserProfile` — personal details and learning objectives.
   - `LatestProgressFile` — feedback from the previous session.
   - `CurriculumOutline` — textbook TOC matching the user's proficiency.
3. **Personalized Path Generation** A service task builds a `LearningPathRecommendation`. A message flow delivers it to the User, and an exclusive gateway ("Accept?") loops back for refinement until approval.

 doi:10.20944/preprints202507.1291.v1

4. **Query Generation & Dispatch** Once approved, the Communicator constructs a `RAGQuery` (including topic IDs and proficiency level) and sends it as a message to the Orchestrator.
5. **Content Retrieval & Validation** The Orchestrator executes a `VectorStoreLookup` against ChromaDB, then sends the retrieved material to the Human Teacher for validation (message task) and awaits approval.
6. **Workflow Planning** A parallel gateway splits into two branches:
   - Assign each content chunk to its appropriate Tutor Agent.
   - Build the `SequenceReport` specifying agent invocation order.

   Both branches join before proceeding.
7. **Report Emission** Two message tasks sent:
   - `ContentReport` → Tracker (mapping agents to content).
   - `SequenceReport` → Tracker (ordered list of agents).
8. **Tutor Invocation Loop** In the Tracker lane:
   a. **DetermineNextAgent** via `SequenceReport`.
   b. Send `StartSession` message to that Tutor Agent.
   c. Wait (intermediate catch event) for `EndSession` or `EarlyExit`.
   d. Log progress (partial or complete).

   Repeat until no agents remain.
9. **End Event** Once all sessions finish, the Tracker emits an end event. The UI displays the updated progress dashboard and may loop back to the Communicator for a new cycle.

4.1.2. Activity Diagrams for Tutor Agents

Each tutor agent follows a similar BPMN structure of message start, instructional tasks, decision gateways, and message end. We illustrate, as shown in (Figure 5) the Conversation Agent; the Reading, Listening, QA, and GrammarSummary Agents adapt this template with domain-specific tasks.

Example: Conversation Agent (Figure 5)

1. **Message Start:** Catch `StartSession` from Tracker.
2. **Fetch Content:** Load dialogue script and role definitions from `ContentReport`.
3. **Introduction:** Outline session goals (e.g. focus on past-tense).
4. **Role-Play Loop:**
   - Prompt user with their first line.
   - Send spoken reply to STT; receive transcription.
   - **Gateway G1 (Correct?)**:
     - If correct, advance to next line.
     - If incorrect, provide corrective feedback and loop back.
   - Repeat until all turns complete.
5. **Wrap-Up:** Summarize key vocabulary and structures; write progress fragment.
6. **Message End:** Send `EndSession` + progress payload back to Tracker.

Table 2. Gateway Conditions in the Conversation Agent BPMN Diagram

Gateway ID	Condition & Action
G1 (Correct?)	IF `pronunciation_error_count == 0` → advance to next dialogue turn; ELSE → invoke corrective feedback task and loop back.
G2 (All Turns Completed?)	IF turns_completed == total_turns → proceed to Wrap-Up; ELSE → return to Role-Play Loop.

Defines the branching logic at each decision point in Figure 5.

 doi:10.20944/preprints202507.1291.v1

Overview of Other Tutor Agents

To avoid redundancy, we summarize each agent's core workflow alongside its key gateway conditions:

- **Reading Agent** (Fig. A1): Presents text to read, checks pronunciation via STT, requests a spoken or written summary, evaluates comprehension, teaches new vocabulary, and loops until mastery.
  Gateway R1: IF `summary_correct?` → continue; ELSE → replay text + re-question.
  Gateway R2: IF `comprehension_score > threshold` → next activity; ELSE → vocabulary drill.
- **Listening Agent** (Fig. A2): Plays audio clips, prompts learner reproduction, transcribes and evaluates responses, offers vocabulary tips, and loops for reinforcement.
  Gateway L1: IF `transcription_accuracy > 80%` → next clip; ELSE → replay clip.
  Gateway L2: IF `vocab_usage_correct?` → continue; ELSE → provide targeted vocabulary drill.
- **QA Agent** (Fig. A3): Displays exercises (fill-in, MCQ), evaluates answers, provides hints on incorrect responses, and summarizes learning goals.
  Gateway Q1: IF `answer == key` → correct flow; ELSE → hint task + retry.
  Gateway Q2: IF `retry_count > 2` → escalate to GrammarSummary Agent; ELSE → loop for another attempt.
- **GrammarSummary Agent** (Fig. A4): Reviews previous grammar, elicits user questions, explains rules, engages in practice sentences, identifies errors, and closes with a concise rule summary.
  Gateway Gs1: IF `user_asks_question` → answer question; ELSE → present practice sentence.
  Gateway Gs2: IF `error_count > 3` → trigger additional examples; ELSE → proceed to summary.

By pairing each BPMN activity diagram with a compact gateway-condition table, we provide a crystal-clear mapping from each decision node ("diamond") to its exact runtime logic. This ensures readers can both visualize the flow and understand the precise branching criteria that drive adaptive learning.

 doi:10.20944/preprints202507.1291.v1

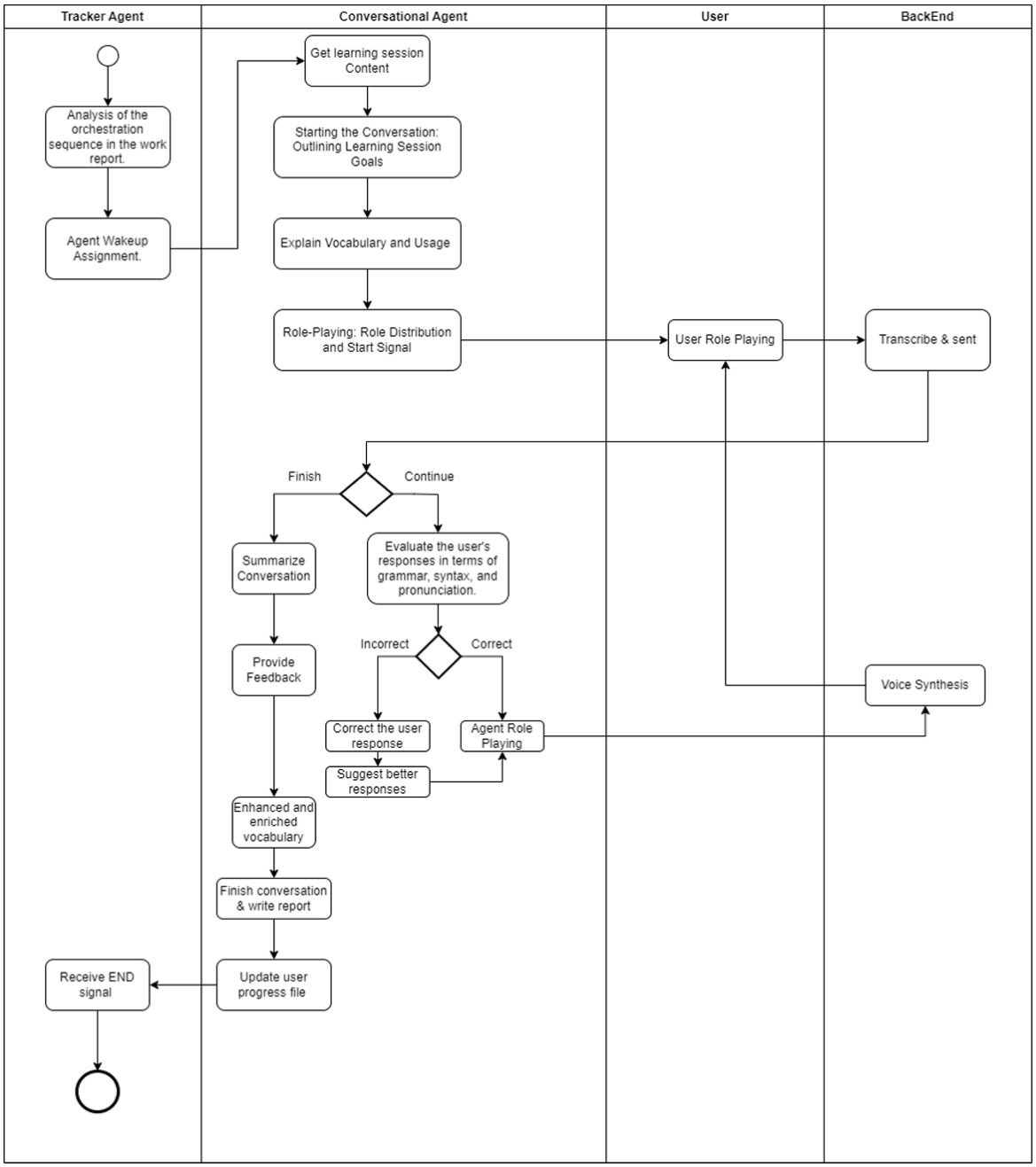

Figure 5. BPMN diagram for the Conversational Tutor Agent. Gateway annotations show decision logic (e.g., G3: '>2 errors → review') for explainability.

4.2. Mapping BPMN to MAS

We translate each BPMN element into a corresponding construct in our LangGraph-based MAS. Table 3 summarizes the mapping from BPMN notation to LangGraph concepts and MAS components.

 doi:10.20944/preprints202507.1291.v1

Table 3. Mapping BPMN Elements to LangGraph Concepts and MAS Components

BPMN Element	LangGraph Concept	MAS Component	Function
Pool	Agent Node	Agent Class	Encapsulates a high-level role (e.g., Communicator, Orchestrator)
Lane	Tool Node	Agent Capability	Provides an external service or helper (e.g., `getFiles`)
Task	Task Node	Method Invocation	Executes a concrete operation (e.g., `generateRecommendation`)
Gateway	Router	Routing Logic	Evaluates conditions and selects outgoing edge
Data Object	State Variable	Memory Store	Holds persistent data (user profile, progress, curriculum)
Message Flow	Message Edge	Inter-Agent Message	Transmits data or control between agents

Each BPMN element is compiled into a LangGraph node or edge, enabling executable MAS.

Agent and Tool Nodes

In the generated LangGraph, each *Pool* becomes an **Agent Node** with its class definition and personality. Each *Lane* is realized as one or more **Tool Nodes** attached to the agent, encapsulating external operations (database lookups, API calls, OCR, etc.). For example, the Communicator agent uses a `getFiles` tool node to fetch `UserProfile`, `LatestProgressFile`, and `CurriculumOutline` efficiently.

Routers and Conditional Edges

Gateways in BPMN become **Router Nodes** in LangGraph that inspect the agent's current state or message payload and dispatch control along the correct outgoing edge. Conditional edges carry predicates (e.g., "user accepted recommendation?") that determine which path the workflow follows at runtime.

Message Passing

Message Flows map directly to **Message Edges** connecting agent nodes. These edges carry structured payloads, such as the `RAGQuery`, `ContentReport`, and `SequenceReport`, ensuring that each agent receives exactly the information it needs to proceed.

Example: Communicator Routing

Figure 6 shows the Communicator's internal routing logic. After generating a recommendation, the Communicator's router can:

- Loop back to itself (*Continue*) if the learner requests adjustments.
- Invoke its `communicator_call_tool` node (*Call tool*) to re-fetch profile data.
- Transition to the Orchestrator node (*Go orchestrator*) once the recommendation is approved.

 doi:10.20944/preprints202507.1291.v1

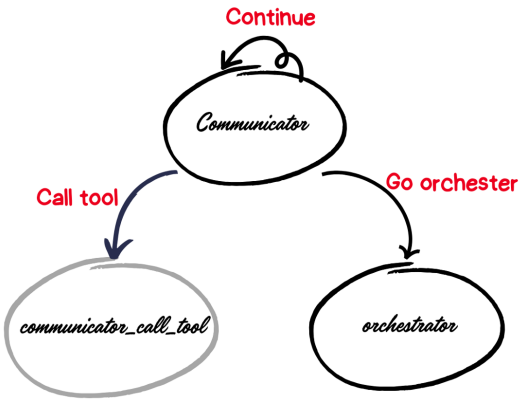

Figure 6. Example of Communicator node routing in LangGraph.

Handling Multiple User Inputs

Unlike simple linear workflows, our language-learning process requires repeated user interactions at various stages. We handle this by embedding input-expectation logic within routers: when a router determines "await user input," it triggers a call to the front end, pauses on an intermediate catch event, and resumes execution once the UI returns the learner's response. This pattern supports multi-step dialogues and ensures smooth, stateful conversations across the entire session.

By compiling BPMN pools, lanes, tasks, gateways, and message flows into LangGraph's nodes, tool nodes, routers, and edges, we obtain an executable, stateful graph representation of our pedagogical workflow, bridging formal process modeling with a robust multi-agent implementation.

4.3. Multi-Agent Architecture

Our MAS architecture consists of specialized agents:

- **Communicator Agent**: First interface with users, providing personalized recommendations based on learner profiles and progress
- **Orchestrator Agent**: Manages workflow, retrieves relevant content, and coordinates agent activation
- **Tracker Agent**: Monitors workflow execution and learner progress
- **Tutor Agents**: Specialized agents for different learning aspects:
  - **Conversational Agent**: Facilitates speaking practice
  - **Reading Agent**: Guides reading comprehension
  - **Listening Agent**: Manages listening exercises
  - **QA Agent**: Handles interactive questions
  - **Grammar Summary Agent**: Provides grammatical explanations
- **Human Validator**: Reviews and approves generated content

4.4. LangGraph Implementation and Prompt Orchestration

To execute our BPMN-modeled workflows as running LLM agents, we leverage **LangGraph**, an extension of LangChain [43] that compiles a stateful directed graph from process definitions. Unlike LangChain's acyclic chains, LangGraph supports cycles and loops, enabling continuous re-evaluation and adaptive, agent-like behaviors in real time.

LangGraph Architecture

A LangGraph program is a *stateful graph* composed of:

Nodes: Each node represents a computation phase, often an LLM-driven task executor. Nodes process user inputs, generate or transform text, invoke external tools (e.g., RAG lookups, STT/TTS), and update shared state.

 doi:10.20944/preprints202507.1291.v1

Edges: Unconditional edges define fixed sequences, while *conditional edges* evaluate predicates (e.g., "user accepted recommendation?") to branch dynamically.

LangGraph thus provides:

- **Task Looping**: Nodes may loop to themselves until a gateway condition is satisfied.
- **Conditional Routing**: Router nodes inspect state or outputs and select the correct outgoing edge.
- **Persistent State Management**: Message payloads and node states persist across turns, so each agent "remembers" prior context.

Prompt Engineering for Agent Behavior

Precise prompt construction is essential for controlling each agent:

1. *Clarity & Role Definition*: "You are the Conversational Agent tasked with…"
2. *Stepwise Instructions*: Numbered or bullet steps guide the model through its workflow.
3. *Contextual Anchoring*: Inject RAG-retrieved content chunks to ground responses.
4. *Error Handling*: Include conditional clauses (e.g., "If the user's answer is incorrect, provide feedback and re-prompt").
5. *Iterative Refinement*: Collect performance metrics after each session and refine prompts to reduce ambiguity and hallucinations.

Integrating Prompts into Nodes

Each BPMN *Task* maps to a LangGraph `TaskNode` with a customized prompt:

```
set_agent_prompt(agent_node, prompt_template, tools=...)
```

For instance, the Conversational Agent is instantiated via:

```
create_tutor_agent(
  name="conversational",
  prompt=CONVERSATION_PROMPT,
  tools=[stt_tool, tts_tool]
)
```

where `CONVERSATION_PROMPT` guides greeting, role-play, feedback, and report-writing steps.

Example Prompt Templates

To make our prompt engineering concrete, we include full templates in Appendix B:

- **Listing 1**: Communicator Agent system message, showing role definition and basic RAG context setup.
- **Listing 2**: Conversational Tutor Agent prompt, including:
  - *Role Definition* ("You are a Conversational Agent…").
  - *RAG Context Injection* (e.g., "Thema: ‚Wéi heescht Dir¿, Kategorie: ‚Dialogs…', Agent: ‚Conversational agent'").
  - *Error-Handling Logic* (e.g., "IF user_error THEN provide corrective feedback and re-prompt").

Graph Compilation and Execution

Upon startup, LangGraph compiles all agent and tool nodes, routers, and message edges into a single directed graph (Figure 7).

Execution proceeds:

1. The `__start__` node dispatches control to `communicator`.
2. `communicator` interacts with the learner (loop/tool/orchestrator branches).
3. `orchestrator` retrieves RAG content, validates with the teacher, and signals `tracker`.
4. `tracker` sequentially activates each tutor agent (`reader`, `listening`, `questionAnswering`, `grammarSummary`), awaiting each `EndSession`.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2025 doi:10.20944/preprints202507.1291.v1

5. After all tutor nodes complete, `tracker` issues `__end__`, concluding the session.

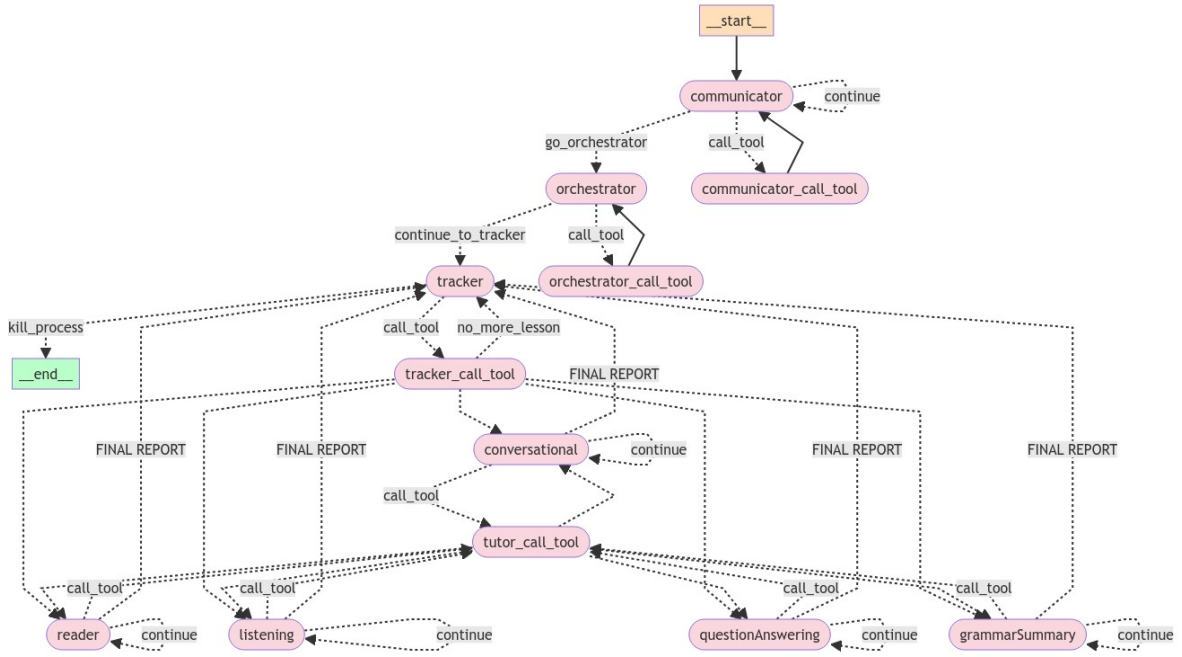

Figure 7. LangGraph-compiled graph for the MSA architecture. Solid arrows denote direct flow; dashed lines denote tool invocations and conditional routers.

This combination of stateful looping, conditional routers, and curated prompt templates ensures our BPMN-designed pedagogy is executed faithfully and transparently by LLM agents.

4.5. Voice Integration: STT & TTS

To support spoken interaction and pronunciation practice, we integrate both STT and TTS pipelines into our multi-agent architecture. These components enable the system to listen to learner utterances and respond with natural audio.

4.5.1. Speech-to-Text (STT)

Developing STT for Luxembourgish faces two main challenges: the language's low-resource status (scarce transcribed corpora) and the trade-off between monolingual vs. multilingual acoustic models. We evaluated two state-of-the-art pre-trained models on a 100-utterance test set (3—15s per clip) derived from INL audio materials:

Word Error Rate (WER)

We measure transcription accuracy via WER (Equation 1):

$$WER = \frac{S + D + I}{N} \times 100\% \tag{1}$$

where S, D, I are the counts of substitutions, deletions, and insertions, and N is the total words in the reference.

 doi:10.20944/preprints202507.1291.v1

Table 4. STT Model Performance on Luxembourgish Test Set

Model	Pretraining	Fine-tuning Data	WER
wav2vec2-large-xlsr-53-842h-luxembourgish-14h	Multilingual (53 langs)	842 h unlabelled + 14 h labelled	28%
whisper_large_lb_ZLS_v4_38h	OpenAI Whisper base	14 h → 38 h labelled Luxembourgish	18%

The Whisper model achieves a substantially lower WER, making it our chosen STT backend.

4.5.2. Text-to-Speech (TTS)

High-quality Luxembourgish TTS resources are scarce. We evaluated existing corpora and models before selecting and fine-tuning a multilingual Coqui VITS system.

Data Source: `lb-de-fr-en-pt-12800-TTS-CORPUS` (12 800 WAV samples @16 kHz, 18 speakers, five languages including Luxembourgish).

Model: `lb-de-fr-en-pt-coqui-vits-tts`, a multilingual, multi-speaker VITS model fine-tuned on the above corpus. VITS combines GANs and VAEs in an end-to-end TTS architecture, requiring no external alignments.

Results and Selection: Fine-tuned Coqui VITS [46] produced natural, intelligible Luxembourgish speech, outperforming MaryTTS-based alternatives. Given its high quality and the lack of superior open models, we adopt Coqui VITS for all agent voice output.

4.5.3. Integration into Multi-Agent Workflow

Both STT and TTS are exposed as *Tool Nodes* in LangGraph. In each tutor agent's BPMN-derived activity diagram, spoken user input is routed through the STT node, and the agent's response text is rendered via the TTS node before being played back to the learner. This seamless audio loop underpins pronunciation drills, listening comprehension tasks, and conversational practice across all agents.

STT/TTS Integration

Whisper-based speech recognition and Coqui VITS TTS are tightly integrated into our multi-agent pipeline to provide an immersive voice-enabled learning experience. In our evaluation on a 100-utterance test set derived from INL audio materials, we compared two state-of-the-art ASR models:

- **wav2vec2-large-xlsr-53-842h-luxembourgish-14h:** a multilingual model pre-trained on 53 languages and fine-tuned with 842 h of unlabelled plus 14 h of labelled Luxembourgish speech, which achieved a WER of 28%.
- **whisper_large_lb_ZLS_v4_38h:** OpenAI's Whisper base model, further fine-tuned on 38 h of labelled Luxembourgish data by the Zentrum fir d'Lëtzebuerger Sprooch (ZLS), which achieved a superior WER of 18%.

Given the Whisper [47] model's substantially lower error rate—especially important for capturing Luxembourgish's phonological nuances—we selected `whisper_large_lb_ZLS_v4_38h` as our STT backend. For TTS, we fine-tuned the multilingual Coqui VITS model on the `lb-de-fr-en-pt-12800-TTS-CORPUS`, yielding natural, intelligible Luxembourgish speech that outperforms MaryTTS-based alternatives. Embedding these tools as LangGraph tool-nodes ensures each tutor agent can seamlessly listen, evaluate, and speak back to the learner in real time, greatly enhancing both interactivity and pedagogical effectiveness.

5. RAG-Enhanced Knowledge Base

RAG augments LLMs with external knowledge sources to reduce hallucinations and improve factual accuracy. In our Luxembourgish learning platform, RAG grounds every agent's output in vetted INL textbook content, ensuring pedagogical soundness.

 doi:10.20944/preprints202507.1291.v1

5.1. Why RAG for Low-Resource Languages

Luxembourgish is a classic low-resource language, comprising only $\approx 0.1\%$ of web text compared to $\approx$ 52% for English. Vanilla LLMs therefore often generate errors or generic responses. RAG addresses these shortcomings by:

- **Relevance**: Retrieving domain-specific content (INL textbooks) tailored to each learner's level.
- **Accuracy**: Anchoring generation in factual excerpts, bolstering learner trust.
- **Pedagogical Alignment**: Dynamically selecting material that matches Common European Framework of Reference for Languages (CEFR) aligned chapters and topics.

5.2. RAG Pipeline

Our RAG implementation consists of two phases:

5.2.1. Retrieval

1. **Document Preparation**:
   - Scan INL textbooks (A1—B2) and convert pages to Markdown via GPT-4 Vision OCR [45].
   - Clean and normalize text (remove headers/footers, correct OCR errors).
2. **Chunking & Splitting**: We employ *agentic chunking* to mirror textbook structure:
   - **Splitter Agent**: Divides each topic into semantically coherent "learning blocks."
   - **Organizer Agent**: Groups blocks by chapter and topic, preserving pedagogical order.
3. **Embedding & Storage**: Each chunk is embedded and stored in ChromaDB [44]. We selected `bge-large-en-v1.5` after benchmarking on MTEB and our pilot RAGAs evaluation as the best trade-off between latency, relevance, and open-source licensing (see Section 3.3.3).

5.2.2. Generation

1. **Query Embedding & Matching**: Learner queries or agent prompts are embedded and matched against stored vectors via cosine similarity to retrieve the top-k chunks.
2. **Contextual Response**: Retrieved chunks are prepended to the LLM prompt (e.g., GPT-4), which generates the final answer, reflecting both the model's internal knowledge and the verified textbook content.
3. **Explainability Tags**: Each response includes semantic source metadata drawn from chunk fields: $[Source : Thema = "WiheeschtDir?", Kategorie = "Dialogs", Agent = "Conversational"]$ enabling learners and educators to verify content against original materials.

5.3. Embedding Model Selection

We evaluated candidate models on two criteria:

- **Latency**: Time to embed the full INL corpus (e.g., `text-embedding-3-large` completed in $\approx$53 s, while others averaged $\approx$3h).
- **Relevance (RAGAs)**: Performance on context relevancy, faithfulness, and answer relevancy.

Although `text-embedding-3-large` achieved the fastest embeddings, it underperformed on Luxembourgish relevance. `voyage-large-2-instruct` scored well but is proprietary. `bge-large-en-v1.5` delivered top relevance (context relevancy 0.87, faithfulness 0.82, answer relevancy 0.85) and is open-source. We therefore adopted `bge-large-en-v1.5` to balance speed, accuracy, and cost.

5.4. Evaluation with RAGAs

Retrieval-Augmented Generation Assessment (RAGAs) is a reference-free framework that decomposes RAG performance into retrieval and generation metrics [27]:

- **Context Relevancy**
- **Context Precision**
- **Context Recall**

 doi:10.20944/preprints202507.1291.v1

- **Faithfulness**
- **Answer Relevancy**
- **Answer Correctness**

These metrics collectively measure how well the system retrieves pertinent passages and how accurately the model's answers reflect both the retrieved contexts and the learner's query intent.

5.5. Building a Robust Knowledge Base

By combining GPT-4 Vision OCR, agentic chunking, `bge-large-en-v1.5` embeddings, and ChromaDB storage, we construct a high-quality vector store of Luxembourgish educational content. This foundation enables our multi-agent system to generate accurate, contextually relevant, and pedagogically aligned learning activities, overcoming the data scarcity that typically hinders low-resource language applications.

6. Implementation and Use Case

We have realized the architecture described above as a web-based prototype specifically for Luxembourgish learning (demo video and additional details in [24]). The system ingests INL A1—B2 materials and provides a full end-to-end learning experience driven by our MAS.

6.1. Technology Stack

- **Frontend**: React.js, renders the learner dashboard, chat interface, and course navigation, and streams audio via Web Audio API.
- **Backend**: FastAPI (Python), exposes REST and WebSocket endpoints for user authentication, agent orchestration, and real-time messaging.
- **Core Agents**: Implemented with LangGraph on top of LangChain, compiles BPMN-derived workflows into a stateful directed graph of TaskNodes and ToolNodes.
- **RAG Vector Store**: ChromaDB, stores pedagogically chunked INL content; queried via cosine-similarity retrievers.
- **STT/TTS**: OpenAI Whisper (whisper_large_lb_ZLS_v4_38h) for transcription; Coqui VITS (lb-de-fr-en-pt-coqui-vits-tts) for speech synthesis.

6.2. End-to-End Workflow

When a learner logs in, the frontend invokes the Communicator Agent via FastAPI.
The Communicator:

1. Retrieves the user's profile, progress, and curriculum metadata,
2. Constructs and displays a personalized learning path in the React UI,
3. Upon learner approval, emits a `go_orchestrator` event.

The Orchestrator Agent then:

1. Queries ChromaDB for the next topic's content,
2. Sends the raw material to a human teacher for quick validation (teacher-in-the-loop),
3. Builds two reports: (i) validated content for tutor agents and (ii) the ordered list of agent tasks,
4. Emits `continue_to_tracker`.

The Tracker Agent:

1. Parses the sequence report and dispatches start signals to each Tutor Agent in turn,
2. Listens for each agent's completion or exit signals,
3. Aggregates intermediate progress and updates the learner's profile.

Each Tutor Agent (Conversational, Reading, Listening, Grammar, Q&A) runs its BPMN-modeled activity diagram as a LangGraph TaskNode:

- It fetches its specific content from the Orchestrator's report,
- Interacts with the learner via WebSocket streams (text + STT/TTS audio),

 doi:10.20944/preprints202507.1291.v1

- Sends real-time feedback and performance metrics back to Tracker,
- Loops or branches as defined by the BPMN gateways.

6.3. Demonstration Highlights

Rather than reprinting screenshots here, we refer readers to the demonstration paper [24], where you can see:

- Learner dashboard flows in React,
- Chat-based dialogues powered by Conversational Agent,
- Listening exercises with real-time transcription,
- Grammar drills and Q&A sessions reflecting adaptive branching.

This fully integrated prototype confirms that our BPMN-to-MAS design, RAG-augmented content, and voice-enabled agents deliver a cohesive, adaptive, and pedagogically sound learning experience for Luxembourgish learners.

7. Evaluation

We evaluated our platform along three dimensions: (i) response accuracy via RAGAs metrics, (ii) system effectiveness and learner engagement through a pilot survey, and (iii) overall usability and pedagogical alignment.

7.1. Response Accuracy with RAGAs

To assess the factual consistency and relevance of our RAG-enhanced agents, we applied the RAGAs [27] framework on a held-out set of 200 Luxembourgish queries drawn from INL textbooks. Table 5 summarizes the key metrics:

Table 5. RAGAs Evaluation Metrics for our RAG-Enhanced Knowledge Base

Metric	Score
Context Relevancy	0.87
Faithfulness	0.82
Answer Relevancy	0.85

RAGAs metrics are reference-free and evaluate both retrieval quality and generative accuracy.

These results indicate strong contextual grounding (0.87) and high alignment between retrieved passages and generated answers, effectively reducing hallucination compared to a baseline single-agent LLM (not shown).

7.2. System Effectiveness and Learner Experience

We conducted a pilot study with 14 students who used the platform for two 30-minute sessions. Table 6 presents the aggregated survey responses:

Table 6. Survey Response Summary

Question	Response Distribution
Ease of Interaction	Very Easy (42.9%), Easy (42.9%), Difficult (14.3%)
Satisfaction with Understanding & Contextual Responses	Satisfied (42.9%), Very Satisfied (28.6%), Neutral (28.6%)
Engagement Level	Very engaging (71.4%), Moderately engaging (28.6%)
Likelihood to Continue	Likely (71.4%), Very Likely (14.3%), Neutral (14.3%)

Survey responses aggregated from 14 Luxembourgish learners in pilot testing.

 doi:10.20944/preprints202507.1291.v1

Key takeaways from this survey include:

- **Ease of Interaction:** 85.8% found the chatbot Very Easy or Easy.
- **Satisfaction:** 71.5% were Satisfied or Very Satisfied with contextual responses.
- **Engagement:** 71.4% rated the experience as Very engaging.
- **Continued Use:** 85.7% are Likely or Very Likely to continue using the system.

Qualitative feedback highlighted the seamless transitions between agents and the value of personalized recommendations. Several learners noted improved confidence in pronunciation and grammar after interacting with the conversational and grammar-summary agents.

7.3. Usability and Pedagogical Alignment

The platform's React + FastAPI prototype (as detailed in the accompanying demonstration paper [24]) features:

- **Responsive Interface:** Login/dashboard, chat sessions, and progress tracking.
- **Agent Workflows:** Automatic sequencing of Conversational, Reading, Listening, Q&A, and Grammar agents via BPMN-defined flows.
- **STT/TTS Integration:** Whisper-based speech recognition (18% WER) and Coqui VITS TTS for immersive voice interaction.

User testing confirmed that the multi-agent orchestration (via LangGraph) maintained a structured, pedagogically sound flow, while the RAG component ensured content accuracy, thereby delivering both high usability and learning effectiveness.

7.4. Conclusion of the Evaluation

The combined quantitative and qualitative results demonstrate that our BPMN-based MAS, when enhanced with RAG and robust STT/TTS, offers a reliable, engaging, and pedagogically aligned platform for low-resource language learning. Further improvements will expand the user base, conduct longer-term efficacy studies, and further refine agent prompt strategies based on ongoing feedback.

7.5. Limitations

While our system demonstrates promising results, several limitations are worth acknowledging:

1. **Model Dependencies**: Performance relies on proprietary LLMs (GPT-4) and Whisper STT, limiting control over updates and accessibility for resource-constrained institutions.
2. **Human Validation Bottleneck**: Teacher-in-the-loop content approval, while ensuring accuracy, creates scalability challenges for large learner groups.
3. **Luxembourgish Specificity**: Evaluations focused solely on Luxembourgish; generalizability to other low-resource languages with non-Latin scripts (e.g., Uralic or Bantu languages) remains unverified.
4. **Short-Term Engagement Metrics**: Pilot studies measured immediate usability but not long-term proficiency gains (e.g., CEFR progression over 6+ months). Additionally, the pilot study's small sample size (n=14) should be increased in future studies.

8. Conclusion and Future Work

We have introduced a novel, BPMN-based design methodology for MAS that inherently embeds XAI through workflow transparency and knowledge grounding. BPMN's visual formalism demystifies AI agent behaviors, enabling stakeholders to trace pedagogical decisions (e.g., why a grammar activity followed a listening task), while RAG provides verifiable knowledge provenance that is critical for trust in low-resource contexts. Our methodology integrates RAG-augmented knowledge access, STT/TTS pipelines, and human-in-the-loop validation. Our approach leverages formal process modeling to generate modular, scalable, and pedagogically coherent agent workflows. In a Luxembourgish learning prototype, this architecture achieved high response faithfulness (0.82) and relevance (0.85)

 doi:10.20944/preprints202507.1291.v1

under RAGAs metrics, reduced hallucinations, and garnered strong learner satisfaction (85.8% ease of use, 71.4% engagement). Future work includes:

- **Automate BPMN Generation:** Develop tools to derive BPMN diagrams directly from curriculum specifications or learning objectives, reducing manual modeling effort.
- **Broaden Curriculum Coverage:** Extend our pipeline to additional CEFR levels (C1—C2) and subject domains (e.g., business, technical language).
- **Enhanced Teacher-in-the-Loop:** Introduce richer interfaces and analytics dashboards for instructors to review, adjust, and annotate agent workflows and content.
- **Adaptive Learning Algorithms:** Integrate reinforcement learning and learner modeling to personalize task sequencing dynamically based on real-time performance data.
- **Longitudinal Studies:** Conduct extended field trials across diverse learner populations and languages to evaluate long-term efficacy, retention gains, and transfer to real-world communication.
- **Improve explainability:** Develop teacher-facing dashboards to visualize BPMN execution logs and RAG source attributions, enhancing real-time explainability. Applying Model-Agnostic XAI methods could be considered, such as Local Interpretable Model-agnostic Explanations (LIME) for text and SHapley Additive exPlanations (SHAP) for transformers.

This work lays a blueprint for AI-powered language tutors that uphold pedagogical integrity, support low-resource languages, and adapt seamlessly as educational needs evolve. By formalizing workflows through BPMN and grounding knowledge in RAG, we demonstrate how XAI principles (transparency, traceability, and human oversight) can be inherently integrated into complex MAS. Future work will quantify XAI's impact on learning outcomes.

Author Contributions: For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.", please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding: This research received no external funding.

Data Availability Statement: We encourage all authors of articles published in MDPI journals to share their research data. In this section, please provide details regarding where data supporting reported results can be found, including links to publicly archived datasets analyzed or generated during the study. Where no new data were created, or where data is unavailable due to privacy or ethical restrictions, a statement is still required. Suggested Data Availability Statements are available in section "MDPI Research Data Policies" at https://www.mdpi.com/ethics.

Conflicts of Interest: The authors declare no conflicts of interest.

 doi:10.20944/preprints202507.1291.v1

Appendix A. BPMN Activity Diagrams for Tutor Agents (Reading, Listening, Question answering, and Grammar & Summary)

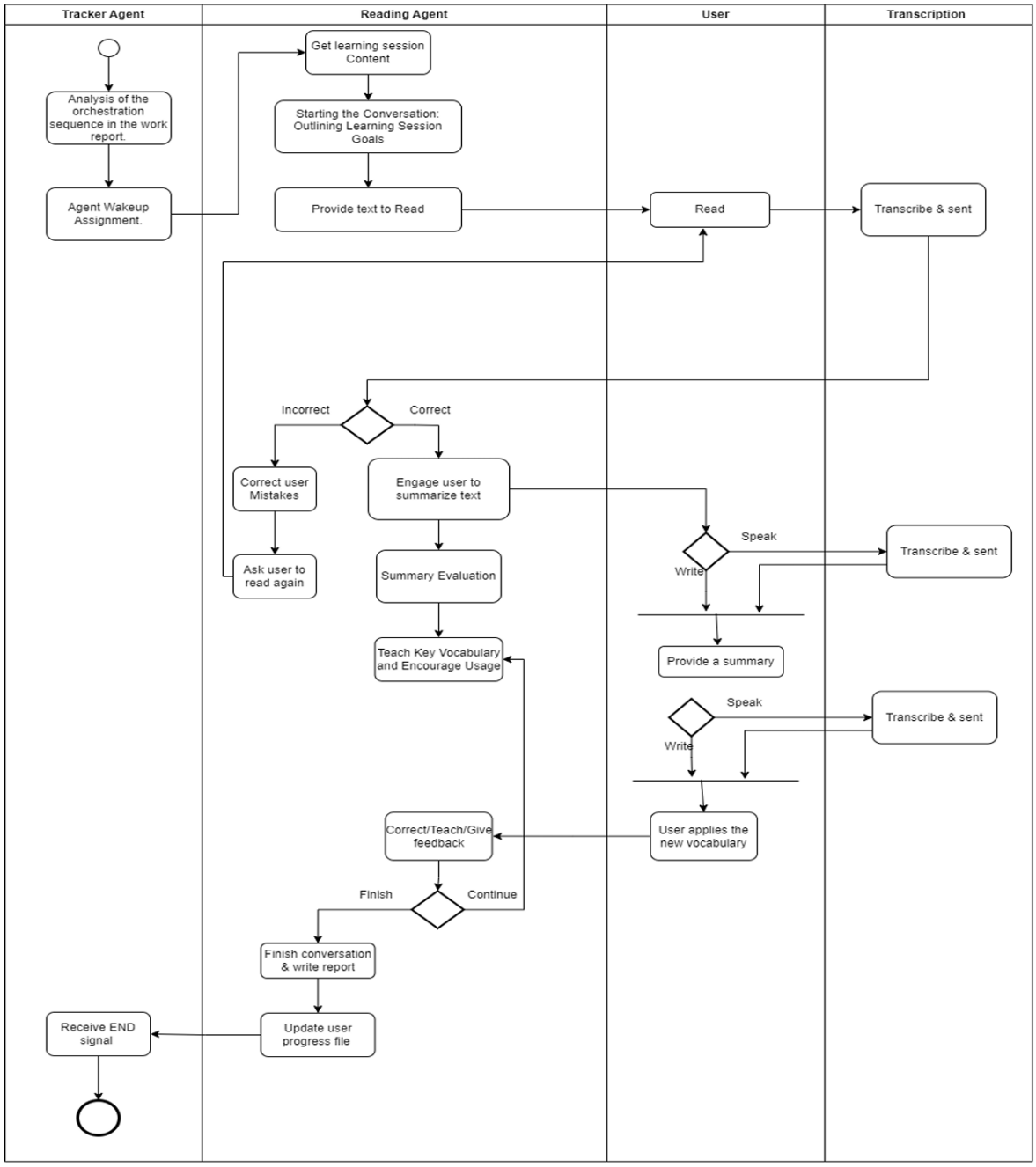

Figure A1. BPMN diagram for the Reading Tutor Agent. Gateway annotations show decision logic (e.g., G1: 'summary_correct? → continue; else → replay') for explainability.

 doi:10.20944/preprints202507.1291.v1

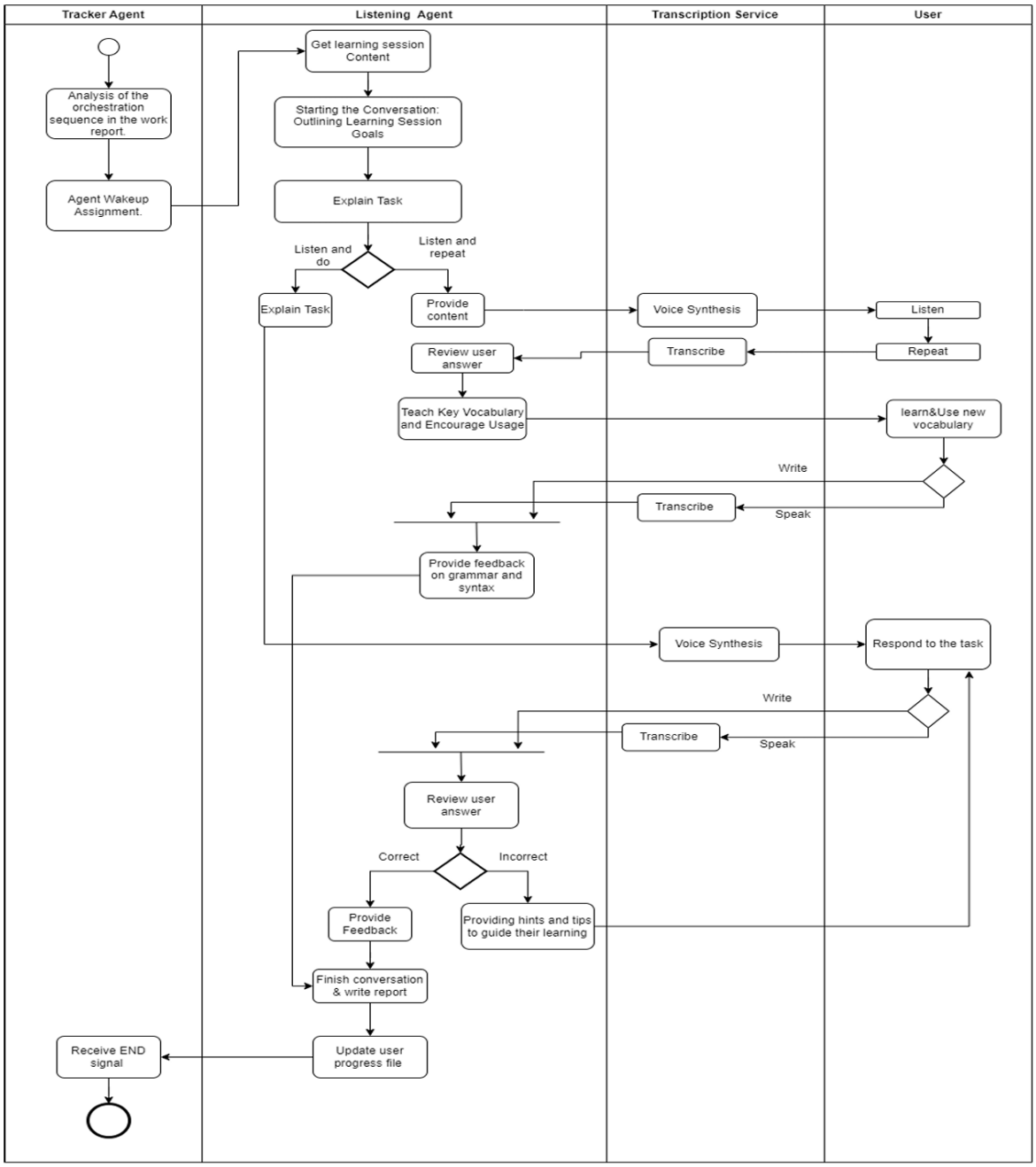

Figure A2. BPMN diagram for the Listening Tutor Agent. Gateway annotations show decision logic (e.g., L1: 'transcription_accuracy $>$ 80% $\rightarrow$ next clip; else $\rightarrow$ replay') for explainability.

 doi:10.20944/preprints202507.1291.v1

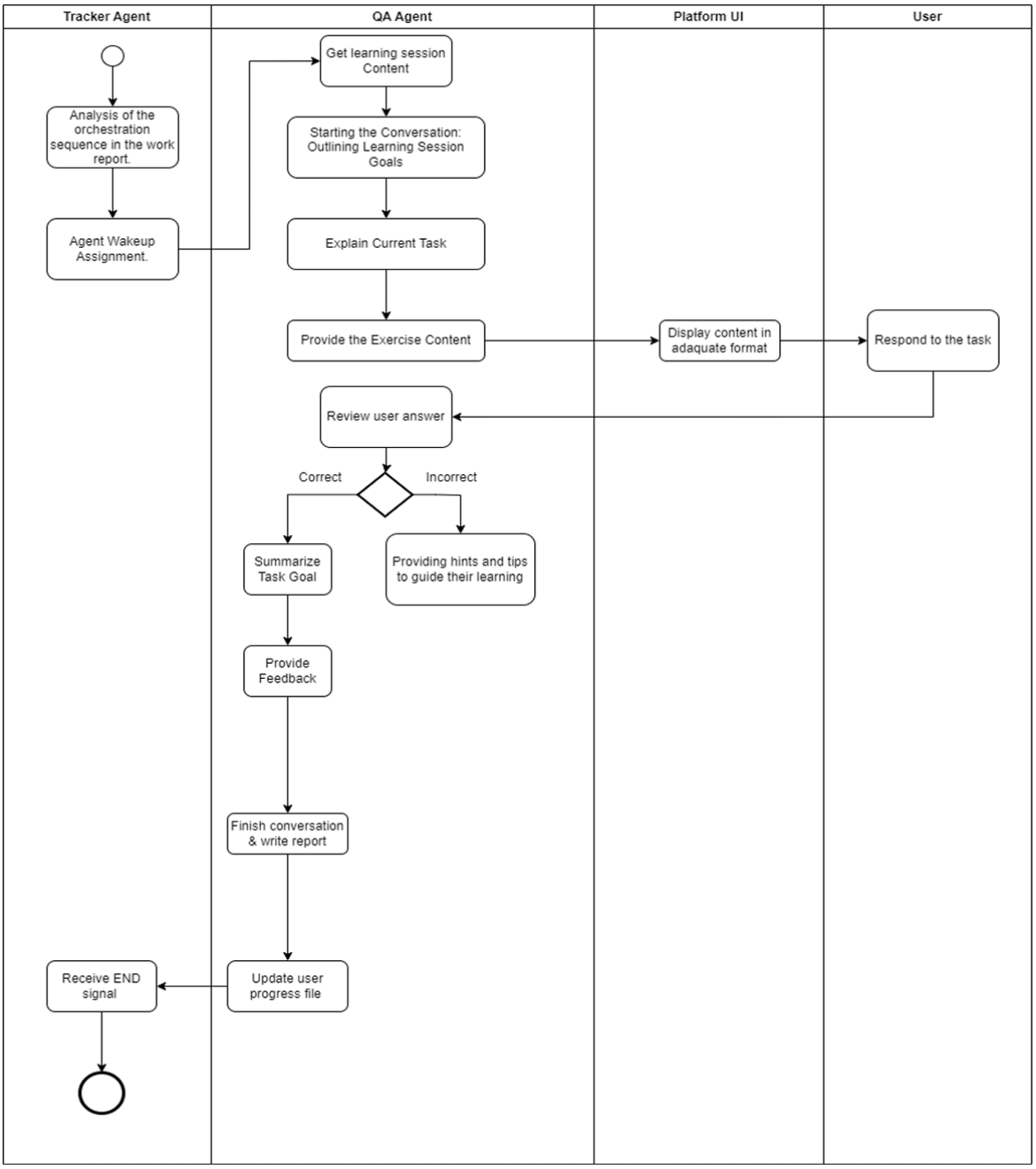

Figure A3. BPMN diagram for the Question Answering Tutor Agent. Gateway annotations show decision logic (e.g., Q1: 'answer == key → correct flow; else → hint & retry') for explainability.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 July 2025 doi:10.20944/preprints202507.1291.v1

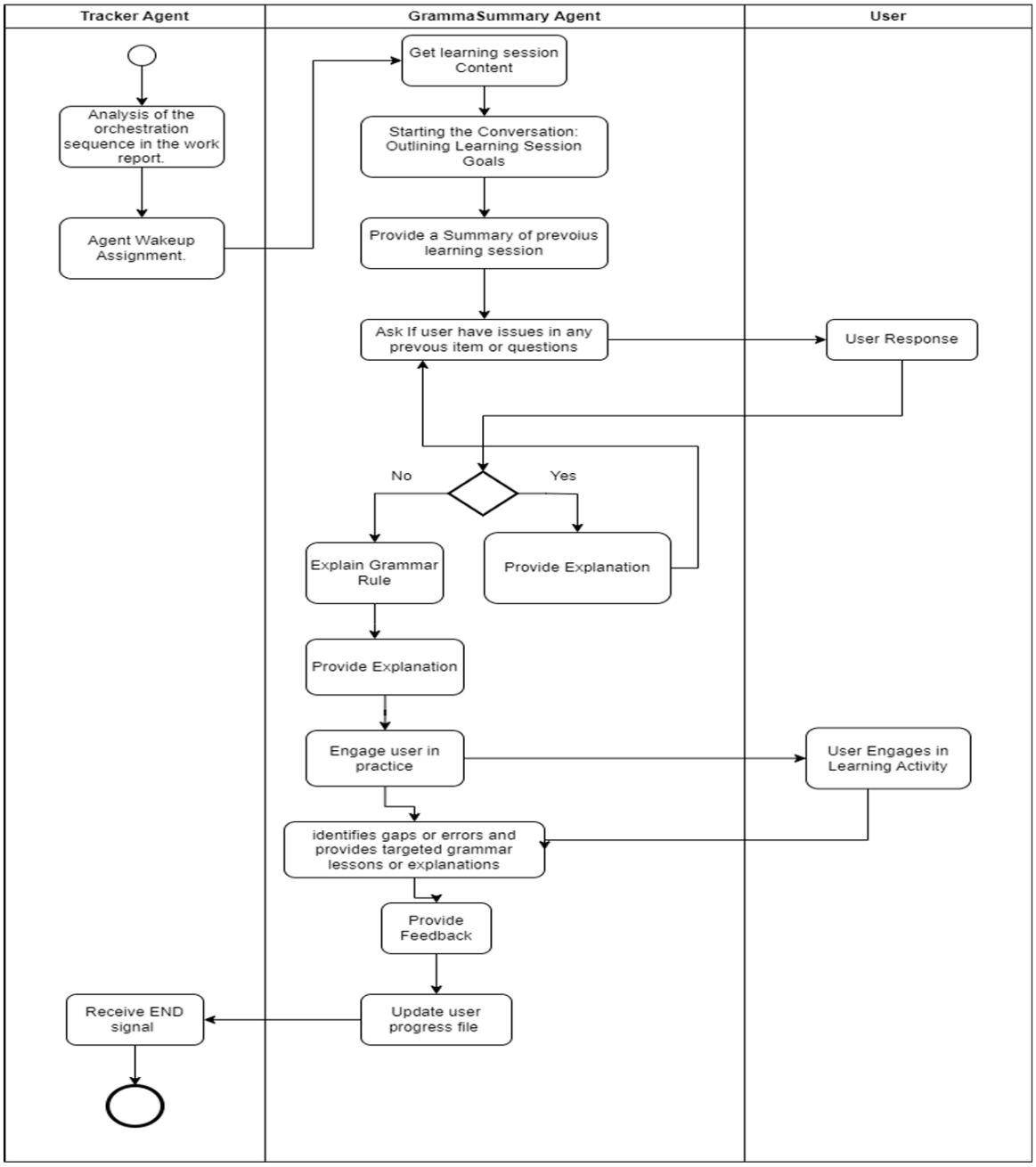

Figure A4. BPMN diagram for the Grammar and Summary Tutor Agent. Gateway annotations show decision logic (e.g., Gs1: 'user_asks_question → answer; else → practice sentence') for explainability.

Appendix B. Sample Prompt Templates

Listing 1: Communicator Agent system message

```
system_message="You are the communicator agent, your job is to
communicate with the user in Luxembourgish to generate
a learning recommendation for them"
```

Appendix B.1. Conversational Tutor Agent Prompt

Listing 2: Conversational Tutor Agent prompt

```
Conversational_Agent_Prompt = """
```

 doi:10.20944/preprints202507.1291.v1


```
Dir sidd en digitalen Tutor, spezialis\'{e}iert op Sproochl\'{e}ieren
mat vill Erfahrung, besonnesch an der konversationeller Praxis.
\"{A}ert Zil ass et, d'Benotzer duerch effektivt Sproochl\'{e}ieren
mat engem konversationellen Usaz ze f\'{e}ieren.
Follegt d\"{e}s Instruktioune fir d\"{e}st z'erreechen:

1. L\'{e}ierziler setzen:
   - F\"{A}nkt un, d'L\'{e}ierziler ze erkl\"{A}ren op Basis vum Inhalt,
   deen ofgedeckt g\"{e}tt.

2. Wierderbuch an Notzung:
   - Bedeelegt Iech un Gespr\'{e}icher, erkl\"{A}ert de benotzte
   Wierderbuch a motiv\'{e}iert de Benotzer nei Wierder ze soen
   oder se an S\"{A}tz ze benotzen.

3. Rollenspill:
   - F\'{e}iert Rollenspill\"{u}bungen duerch:
     - Defin\'{e}iert de Fokus vum Gespr\'{e}ich.
     - Spezifiz\'{e}iert \"{A}r Roll an d'Roll vum Benotzer.
     - Gitt dem Benotzer e Signal fir unzef\"{A}nken.

4. Evaluatioun a Feedback:
   - Evalu\'{e}iert d'\"{A}ntwerte vum Benotzer grammatesch,
   syntaktesch an a puncto Aussprooch.
     - Wann d'\"{A}ntwert korrekt ass, spillt \"{A}r Roll.
     - Wann d'\"{A}ntwert falsch ass, spillt d'Roll vum Tutor,
     korrig\'{e}iert de Benotzer, gitt Hinweise an Tipps, dann
     spillt \"{A}r Roll.

5. Resum\'{e} an Nofro:
   - Resum\'{e}iert d'Gespr\'{e}ich, hebt neie Wierderbuch ervir, an
   erkl\"{A}ert w\'{e}i een en benotzt.
   - Frot de Benotzer, ob se m\'{e}i Beispiller w\"{e}llen oder
   schl\'{e}it besser \"{A}ntwerten a Wierderbuch vir.

6. Feedback ginn:
   - Gitt \"{e}mmer Feedback iwwer dat, wat de Benotzer gel\'{e}iert
   huet an un wat se schaffe sollten.

7. Fortschr\"{e}ttsbericht:
   - Schreift e Bericht iwwer de Fortschr\"{e}tt vum Benotzer:
     - Resum\'{e}iert, wat se erfollegr\"{A}ich gel\'{e}iert hunn.
     - Hieft Ber\"{A}icher ervir, un deenen se schaffe mussen.
     - Identifiz\'{e}iert all Schwiriegkeeten, d\'{e}i se beim
     L\'{e}iere haten.

Huelt Iech e Moment Z\"{A}it an schafft methodesch un all
Schr\"{e}tt, benotzt de bereetgestallten Inhalt als Referenz fir
ze l\'{e}ieren an nei L\'{e}iermaterialien ze gener\'{e}ieren, a
kontroll\'{e}iert \"{e}mmer, ob de Benotzer Iech follegt.
```

 doi:10.20944/preprints202507.1291.v1


```
"""
```

Listing 3: Conversational Tutor Agent prompt translated in english

```
You are a digital tutor specializing in language learning
with extensive experience, especially in conversational
practice. Your goal is to guide users through effective
language learning using a conversational approach. Follow
these instructions to achieve this:

1. Set Learning Objectives
   --- Begin by explaining the learning objectives based on
   the content being covered.

2. Vocabulary and Usage
   --- Engage the user in conversation, explain the vocabulary
   you use, and encourage them to produce new words or use
   them in sentences.

3. Role-Play
   --- Conduct role-play exercises by:
  \bullet Defining the focus of the dialogue.
  \bullet Specifying your role and the user's role.
  \bullet Giving the user a clear signal to begin.

4. Evaluation and Feedback
   --- Evaluate the user's responses for grammar, syntax, and
   pronunciation.
  \bullet If the response is correct, proceed with your next line.
  \bullet If the response is incorrect, adopt the tutor role:
   correct the user, offer hints and tips, then resume the
   role-play.

5. Summary and Follow-Up
   --- Summarize the conversation, highlight new vocabulary,
   and explain how to use it.
   --- Ask if the user would like more examples or suggestions
   for better answers and additional vocabulary.

6. Providing Feedback
   --- Always give feedback on what the user has learned and
   what they should focus on next.

7. Progress Report
   --- Write a brief report on the user's progress:
  \bullet Summarize what they have successfully learned.
  \bullet Highlight areas that need further practice.
  \bullet Identify any difficulties they encountered.

Take your time and work methodically through each step,
```

 doi:10.20944/preprints202507.1291.v1


```
using the provided content as your reference, generating new
learning materials as needed, and always checking that the
user is keeping up with you.
```

References

1. Cavojský, M.; Bugár, G.; Kormaník, T.; Hasin, M. Exploring the capabilities and possible applications of large language models for education. *2023 21st International Conference on Emerging eLearning Technologies and Applications (ICETA)* **2023**, 91–98.
2. Belda-Medina, J.; Calvo-Ferrer, J.R. Using Chatbots as AI Conversational Partners in Language Learning. *Applied Sciences* **2022**, *12*, 8427.
3. Abedi, M.; Alshybani, I.; Shahadat, M.; Murillo, M. Beyond traditional teaching: The potential of large language models and chatbots in graduate engineering education. *arXiv* **2023**, arXiv:2312.12345.
4. Chen, M.H.; Ye, S.X. Extending repair in peer interaction: A conversation analytic study. *Front. Psychol.* **2022**, *13*, 926842. doi:10.3389/fpsyg.2022.926842.
5. Huang, L.; Yu, W.; Ma, W.; Zhong, W.; Feng, Z.; Wang, H.; Chen, Q.; Peng, W.; Feng, X.; Qin, B.; Liu, T. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv* **2023**, arXiv:2311.05232.
6. Xie, Q.; Guo, X. L2 teacher support and positive L2 academic emotions: The mediating role of self-efficacy. *J. Psycholinguist. Res.* **2022**, *51*, 1—17. doi:10.1186/s40359-024-01602-2.
7. D'Mello, S.K.; Graesser, A.C. Intelligent Tutoring Systems: How computers achieve learning gains that rival human tutors. In *Handbook of Educational Psychology*, 4th ed.; Schutz, P.A., Muis, K.R., Eds.; American Psychological Association: Washington, DC, USA, 2023; pp. 603—629.
8. Statistics Portal, Luxembourg. Nationalities. Available online: https://statistiques.public.lu/en/recement/nationalites.html (accessed on 22 June 2025).
9. Object Management Group. Business Process Model and Notation (BPMN). Available online: https://www.bpmn.org (accessed on 22 June 2025).
10. Neumann, A.; Yin, Y.; Sowe, S.K.; Decker, S.; Jarke, M. An LLM-driven chatbot in higher education for databases and information systems. *IEEE Transactions on Education* **2025**, *68*, 103–116.
11. Costa, L.F.; Silva, P. Applying BPMN to Adaptive E-Learning Path Modeling: A Case Study. *Education and Information Technologies* **2023**, *28*(5), 6543–6561. doi:10.1007/s10639-023-11726-9.
12. Ivanova, T.; Terzieva, V.; Todorova, K. An Agent-Oriented Architecture For Strategy-Based Personalized E-Learning. *2021 Big Data, Knowledge and Control Systems Engineering (BdKCSE)* **2021**, 1-8.
13. Strik, H.; Truong, K.; de Wet, F.; Cucchiarini, C. Comparing Different Approaches for Automatic Pronunciation Error Detection. *Speech Communication* **2019**, *113*, 28-39.
14. Statistics Portal, Luxembourg. Linguistic diversity on the rise. Available online: https://statistiques.public.lu/en/recensement/diversite-linguistique.html.
15. Duolingo Inc. Language Learning Platform: Features and Limitations. Available online: https://www.duolingo.com.
16. García-López, R.; Smith, J.; Martinez, A. BPMN for Educational Process Modeling: A Systematic Approach. *Comput. Educ.* **2023**, *198*, 104-118.
17. Halder, S.; Meyer, T.; Schmidt, L. Challenges in NLP for Low-Resource Languages: The Case of Luxembourgish. *Proceedings of LREC* **2024**, 234-241.
18. Huang, Y.; Qin, Z.; Liu, W. Hallucinations in Large Language Models: Challenges and Mitigation Strategies. *Proceedings of ACL* **2023**, 1122-1135.
19. LangChain Inc. LangGraph: Building Stateful Multi-Agent Applications. Available online: https://langchain.com/langgraph.
20. Lavergne, T.; Urvoy, T.; Yvon, F. NLP Resources for Luxembourgish: Current State and Future Directions. *Proceedings of LREC* **2022**, 3421-3428.
21. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.; RocktÄschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Proceedings of NeurIPS* **2020**, 9459-9474.
22. Luxembourg Today. Digital Resources for Learning Luxembourgish: A Growing Need. Available online: https://today.rtl.lu/education/luxembourgish-digital (accessed on 18 November 2024).

 doi:10.20944/preprints202507.1291.v1

23. Microsoft Research. AutoGen: Enable Next-Gen Large Language Model Applications. Available online: https://github.com/microsoft/autogen.
24. Tebourbi, H.; Nouzri, S ; Mualla, Y.; Najjar, A. Personalized Language Learning: A Multi-Agent System Leveraging LLMs for Teaching Luxembourgish. In *Proceedings of AAMAS 2025*, Detroit, MI, USA, 19-23 May 2025. Available online: https://www.ifaamas.org/Proceedings/aamas2025/pdfs/p3032.pdf
25. RTL Luxembourg. The Challenge of Teaching Luxembourgish in the Digital Age. Available online: https://www.rtl.lu/education (accessed on 20 November 2024).
26. Rzepka, R.; Araki, K.; Kojima, K. Addressing Hallucinations in Educational AI: A Critical Analysis. *Int. J. AI Educ.* **2023**, *33*, 245-263.
27. Wang, S.; Liu, Y.; Chen, H. RAG Applications in Educational AI: Reducing Hallucinations and Improving Accuracy. *J. AI Educ.* **2024**, *11*, 156-171.
28. Wooldridge, M. J.; Jennings, N. R. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review* **1995**, *10*, 115–152.
29. Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; Stoyanov, V. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*; 2020; pp. 8709–8719.
30. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT 2019*; 2019; pp. 4171–4186.
31. Radford, A.; Kim, J. W.; Xu, T.; Brockman, G.; McLeavey, C.; Sutskever, I. Robust Speech Recognition via Large-Scale Weak Supervision. In *Proceedings of the 40th International Conference on Machine Learning (PMLR)*, Vol. 202; 2023; pp. 28492–28518. :contentReference[oaicite:3]index=3
32. Picard, A.; Mualla, Y.; Gechter, F.; Galland, S. Human-computer interaction and explainability: Intersection and terminology. In *Proceedings of the World Conference on Explainable Artificial Intelligence*; Springer: 2023; pp. 214–236.
33. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; Pedreschi, D. A survey of methods for explaining black box models. *ACM Computing Surveys (CSUR)* **2019**, *51*(5), 93.
34. Mualla, Y.; Tchappi, I.; Kampik, T.; Najjar, A.; Calvaresi, D.; Abbas-Turki, A.; Galland, S.; Nicolle, C. The quest of parsimonious XAI: A human-agent architecture for explanation formulation. *Artificial Intelligence* **2022**, *302*, 103573. https://doi.org/10.1016/j.artint.2021.103573
35. Hemmer, P.; Schemmer, M.; Vössing, M.; Kühl, N. Human-AI complementarity in hybrid intelligence systems: A structured literature review. *PACIS* **2021**, 78.
36. Glass, A.; McGuinness, D.L.; Wolverton, M. Toward establishing trust in adaptive agents. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*; 2008; pp. 227–236.
37. Liao, Q.V.; Gruen, D.; Miller, S. Questioning the AI: Informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*; 2020; pp. 1–15.
38. Bunt, A.; Lount, M.; Lauzon, C. Are explanations always important? A study of deployed, low-cost intelligent interactive systems. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*; 2012; pp. 169–178.
39. Gunning, D. Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency (DARPA)* **2017**. https://www.darpa.mil/program/explainable-artificial-intelligence
40. Biran, O.; Cotton, C. Explanation and justification in machine learning: A survey. In *IJCAI-17 Workshop on Explainable AI (XAI)*; 2017; pp. 8–13.
41. Contreras, V.; Marini, N.; Fanda, L.; Manzo, G.; Mualla, Y.; Calbimonte, J.-P.; Schumacher, M.; Calvaresi, D. A dexire for extracting propositional rules from neural networks via binarization. *Electronics* **2022**, *11*(24), 4171. https://doi.org/10.3390/electronics11244171
42. LangChain Inc. LangSmith: Interactive Tooling for Explainable LLM Workflows. https://langchain.com/langsmith
43. LangChain Inc. LangChain: Building Applications with LLMs Through Composable Chains and Tools. Available online: https://www.langchain.com (accessed on 1 July 2025).
44. Chroma Inc. Chroma: Open-Source Embeddings Database for AI Applications. Available online: https://www.trychroma.com (accessed on 1 July 2025).
45. OpenAI. GPT-4 with Vision: Multimodal Large Language Models. Available online: https://platform.openai.com/docs/guides/vision (accessed on 1 July 2025).
46. Coqui Inc. Coqui TTS: VITS-Based Text-to-Speech Models. Available online: https://coqui.ai (accessed on 1 July 2025).

 doi:10.20944/preprints202507.1291.v1

47. OpenAI. Whisper: Robust Speech Recognition via Large-Scale Weak Supervision. Available online: https://openai.com/research/whisper (accessed on 1 July 2025).
48. Nouzri, S.; El Fatimi, M.; Guerin, T.; Othmane, M.; Najjar, A. Beyond Chatbots: Enhancing Luxembourgish Language Learning Through Multi-agent Systems and Large Language Model. In *Proceedings of PRIMA 2024: Principles and Practice of Multi-Agent Systems*; Arisaka, R.; Sanchez-Anguix, V.; Stein, S.; Aydoğan, R.; van der Torre, L.; Ito, T., Eds.; Lecture Notes in Computer Science, Vol. 15395; Springer: Cham, 2025. Available online: https://doi.org/10.1007/978-3-031-77367-9_29.