engineering proceedings

MDPI

Proceeding Paper

A Domain-Specific Language for Modeling and Implementing Agentic AI Workflows †

Lenin Erazo-Garzón [1,2,*], John Chimbo [1], Rodrigo Ortiz [1], Lourdes Illescas [2] and Priscila Cedillo [2]

[1] Computer Science Research & Development Laboratory (LIDI), Universidad del Azuay, Cuenca 010204, Ecuador; john.chimbo@es.uazuay.edu.ec (J.C.); rodrigo.ortiz@es.uazuay.edu.ec (R.O.)
[2] Department of Computer Science, Universidad de Cuenca, Cuenca 010203, Ecuador; lourdes.illescasp@ucuenca.edu.ec (L.I.); priscila.cedillo@ucuenca.edu.ec (P.C.)
* Correspondence: lerazo@uazuay.edu.ec; Tel.: +593-991828833
† Presented at the Sixth International Conference on Information Systems and Software Technologies (ICI2ST 2026), Quito, Ecuador, 23–25 September 2026.

Abstract

The increasing complexity of tasks addressed by generative artificial intelligence (AI) has driven the emergence of agentic AI, which incorporates capabilities for autonomy, collaboration, planning, decision-making, and adaptation to solve such tasks. However, existing agentic AI frameworks lack formal languages for developing standardized, verifiable, reproducible, and interoperable solutions. Integrating agentic AI with Model-Driven Engineering emerges as a promising yet underexplored approach. Therefore, this paper proposes AgenticAIFlow, a domain-specific language to simplify and formalize the design and implementation of agentic AI workflows. The solution was developed following the Technology Transfer Model proposed by Gorschek et al. and empirically evaluated in an academic context through a quasi-experiment based on the Method Evaluation Model. The results revealed favorable participant perceptions regarding the solution's ease of use, usefulness, and intention to use it in future projects, together with high levels of effectiveness and efficiency in performing the experimental tasks.

Keywords: agentic AI; domain-specific language; model-driven engineering; large language model; transformation engine; workflow

1. Introduction

In recent years, the use of large language models (LLMs) has proliferated across a wide range of domains (e.g., education, health, commerce, software engineering, scientific research), significantly transforming the way people interact with technology to perform their daily tasks [1,2]. However, when LLMs operate independently (e.g., chatbots), they exhibit important limitations, such as the lack of explicit control over the reasoning flow, limited traceability, low reproducibility, and difficulties in managing systematic processes composed of multiple tasks [3,4]. In this context, the integration of LLMs with agent-based architectures has led to a new paradigm called agentic AI, in which multiple specialized agents interact, coordinate, and use LLMs and external tools to autonomously solve complex workflows that require planning, collaboration, persistent memory, iterative reasoning, self-evaluation, and adaptation [5–7].

Since 2023, several frameworks have emerged for building agentic workflows based on LLMs (e.g., LangGraph, CrewAI, AutoGen). However, several comparative studies [7–9] agree that this field remains in its early stages of development and highlight multiple open

Check for updates

Academic Editor: Fredy Gavilanes

Published: 18 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

challenges. Among the most relevant problems is the absence of abstract and standardized specification mechanisms, as most frameworks rely on imperative code, employ ad hoc configurations, and lack formal modeling languages with well-defined syntax and semantics. This situation significantly limits the verifiability (static analysis), reproducibility, interoperability, and scalability of agentic AI workflow solutions.

Although Model-Driven Engineering (MDE) has proven to be an effective approach for increasing the level of abstraction, formality, and automation in software development [10], recent studies indicate that the integration between MDE and agentic AI is still in its early stages [11]. Currently, there is no domain-specific language (DSL) and conceptual metamodel that comprehensively represents the architecture of the agentic workflow, the roles and capabilities of agents, workflows and control mechanisms, cooperation strategies, memory and context management, interaction mechanisms with LLMs (prompt engineering), external tools and users, as well as evaluation methods and metrics. This constitutes a significant research gap [8,11,12].

Therefore, this paper proposes AgenticAIFlow, a visual DSL that formalizes and simplifies the design and implementation of agent-supported workflows that use LLMs, shared memory, and external tools to solve complex problems autonomously and in a planned, collaborative manner. In particular, the main contributions of this work are:

- A metamodel that provides a sufficiently complete and expressive modeling language to represent a wide variety of agentic AI workflow architectures, regardless of the application domain.
- A graphical designer that incorporates forms based on high-level concepts from the problem domain to abstract the complexity associated with the design, specification, and implementation of the main components of an agentic AI workflow, including tasks, agents, LLMs, memory, external tools, and prompt engineering.
- A graph-based orchestration engine that coordinates the execution of multiple agents to complete tasks in the order defined by the workflow autonomously. The agents operate according to the specifications established in the model, including agent properties, task input/output parameters, LLM hyperparameters, memory and context configurations, and prompting strategies. Additionally, the orchestration engine supports invoking external tools through APIs, managing their parameters, and facilitating interaction with external services.
- An empirical evaluation of AgenticAIFlow, conducted through a quasi-experiment, providing evidence of the effectiveness, efficiency, ease of use, usefulness, and intention to use the proposed solution.

This paper is organized as follows. Section 2 presents the background and related work. Section 3 briefly describes the methodology employed to develop the proposed solution. Section 4 describes the AgenticAIFlow DSL, detailing its metamodel, architecture, components, and main functionalities. Section 5 reports the empirical evaluation of the proposed solution through a quasi-experiment. Finally, Section 6 presents the conclusions and outlines future work.

2. Background and Related Work

This section provides an overview of the foundations and technological evolution from LLMs to agentic AI, along with an analysis of the state of the art in integrating MDE with generative and agentic AI.

2.1. From LLMs to Agentic AI

AI is at a turning point, evolving from generative AI, based on static and reactive language models capable of generating multimodal content (text, images, code, among

others) from user instructions, toward agentic AI, characterized by dynamic, proactive, and autonomous systems capable of reasoning, coordinating, making decisions, and executing actions [5,9].

This progression begins with LLMs, a class of generative AI models (e.g., GPT, PaLM), trained on vast corpora of data using advanced deep learning architectures, particularly Transformer-based neural networks, multi-head self-attention mechanisms, and Mixture of Experts (MoE) [13–15]. Although LLMs exhibit remarkable capabilities in natural language understanding and generation, they still face several limitations, including hallucinations, limited memory retention, restricted contextual reasoning, low reproducibility, and insufficient alignment with user intent. To overcome these limitations, several techniques have been proposed, among which the following stand out: (i) *instruction fine-tuning*, trains foundation LLMs using supervised examples to adapt them to specific tasks [16]; (ii) *Reinforcement Learning from Human Feedback (RLHF)*, employs human evaluations to train reward models aimed at improving the alignment of LLM-generated responses with user preferences [17,18]; (iii) *prompt engineering*, uses advanced strategies to guide and adapt model behavior without modifying its weights [19] (e.g., Chain of Thought—CoT [20], Tree of Thought—ToT [21], and Reasoning and Acting—ReAct [22]); and (iv) *Retrieval-Augmented Generation (RAG)*, integrates information retrieval mechanisms into the generative workflow to incorporate relevant and up-to-date context from external sources in real time [23]. These techniques, together with the integration of LLMs with *external tools and services* via APIs (e.g., computational, modeling, and code execution tools) [24,25], have improved model alignment, the quality of generated responses, and their ability to perform actions and solve increasingly complex tasks.

These advances have paved the way for AI agents and, more broadly, so-called agentic AI. Although the concept of an AI agent was first introduced in 1998 [26], its potential has significantly increased with the rise of generative AI. In this context, an AI agent is formally defined as an autonomous software entity that perceives its environment, reasons, plans, and acts to achieve specific goals with minimal human intervention [6]. An AI agent typically consists of four fundamental elements: perception mechanisms that capture information from the environment; a cognitive model based on LLMs; memory that preserves context and coherence over time; and tools that interact with the environment, thus closing the perception–reasoning–action cycle [27,28].

In contrast, agentic AI emerged in late 2023 and represents an architectural approach characterized by collaboration among multiple agents, the dynamic decomposition of complex high-level objectives into tasks and their strategic assignment to specialized agents, as well as the use of shared memory and coordinated autonomy [7,9]. This approach seeks to develop scalable agentic infrastructures aimed at supporting workflows that exceed the capabilities and resources of a single agent.

2.2. MDE and Agentic AI

According to the systematic review by Rocco et al. [11] and a complementary literature review carried out in this work on the integration of LLMs and MDE, research efforts have predominantly focused on LLMs as the primary mechanism for supporting intrinsic MDE tasks. Among the most frequently reported applications are model creation, refinement, and completion; text-to-model, model-to-model, and model-to-text transformations; syntactic and semantic validation; and reverse engineering.

In contrast, there is limited adoption of MDE principles for modeling and formalizing key aspects of LLM-based and agentic AI systems, such as prompt engineering, the governance and control of reasoning workflows, knowledge and context management (e.g., RAG), and post-generation activities, including validation and metric-based evaluation, among

others [11]. Among the few works identified along this line is Impromptu [29,30], a DSL designed for generating multimodal prompts for generative AI. This DSL incorporates additional functionalities, such as prompt chaining, version control, prompt adaptation to specific generative AI systems, and validation of the quality of generated responses. Likewise, Morales et al. [31] present Lang-BiTe, a DSL for specifying ethical requirements and using them to customize and automate ethical bias test cases in LLM-based systems. This solution facilitates the evaluation and selection of the most suitable LLM according to the specified ethical requirements. Göbel et al. [32] propose a DSL, KELM, that combines sequence diagrams and enhanced argumentation graphs to represent human–LLM interactions, as well as the relationships among information fragments generated during the conversation. Furthermore, this proposal includes an executable semantics based on model transformations to compute metrics and assess user confidence regarding knowledge extracted from conversations.

In the field of agentic AI, Barriga et al. [12] propose a metamodel for specifying the structural and behavioral aspects of LLM-based multi-agent systems. However, this work is still at a preliminary stage and provides a language with limited expressiveness for defining agent profiles, coordination among agents, task sequencing in workflows, LLM hyperparameters, memory and context mechanisms, prompting strategies, and integration interfaces with external tools. In addition, the proposal has not been subjected to empirical evaluation, which limits the validity of claims regarding its applicability.

Unlike existing approaches, AgenticAIFlow DSL provides a comprehensive solution for designing and implementing agentic AI workflows, broadly addressing key concerns such as workflow architecture, agent roles and functions, inter-agent coordination mechanisms, LLM configuration, shared memory and context management, integration of external tools, and prompt engineering strategies. Furthermore, unlike solutions that rely on imperative code and ad hoc configurations, AgenticAIFlow provides a modeling language based on high-level concepts and an orchestration engine that interprets these specifications to operationalize agentic AI workflows. In this way, the solution abstracts away the complexity of implementation details and facilitates the development of agentic AI workflows with desirable properties, such as verifiability, reproducibility, maintainability, traceability, and reusability.

3. Research Methodology

The methodology employed to develop AgenticAIFlow DSL is based on the Technology Transfer Model proposed by Gorschek et al. [33], which promotes collaboration between researchers and industry practitioners, fostering research aligned with real-world needs and problems in the sector. In general terms, this model begins with problem formulation and a review of the state of the art. Then, candidate solutions are defined and iteratively evaluated empirically and continuously improved through an evolutionary cycle to achieve an appropriate and realistic solution. In this work, the process reached the initial validation of AgenticAIFlow in an academic context through a quasi-experiment designed according to the experimental process proposed by Wohlin et al. [34] and the Method Evaluation Model (MEM) introduced by Moody [35]. Validation of the solution in an industrial context and its subsequent transfer to professional practice will be addressed in future work.

4. AgenticAIFlow DSL

This section describes the metamodel (abstract syntax) underlying the AgenticAIFlow DSL, including its metaclasses, attributes, and relationships. It also explains the DSL's architecture, defining the functionality of its main components and relationships.

4.1. Metamodel

This metamodel was designed in the standard ECORE format using the Eclipse Modeling Framework (EMF) tool [36]. The metamodeling approach was based on the reuse of the most relevant metaclasses identified in the related work. Subsequently, new metaclasses were incorporated to represent fundamental concepts of the agentic AI domain that had not been considered in prior work. The main metaclasses and their associated attributes and relationships are described below (see Figures 1 and 2).

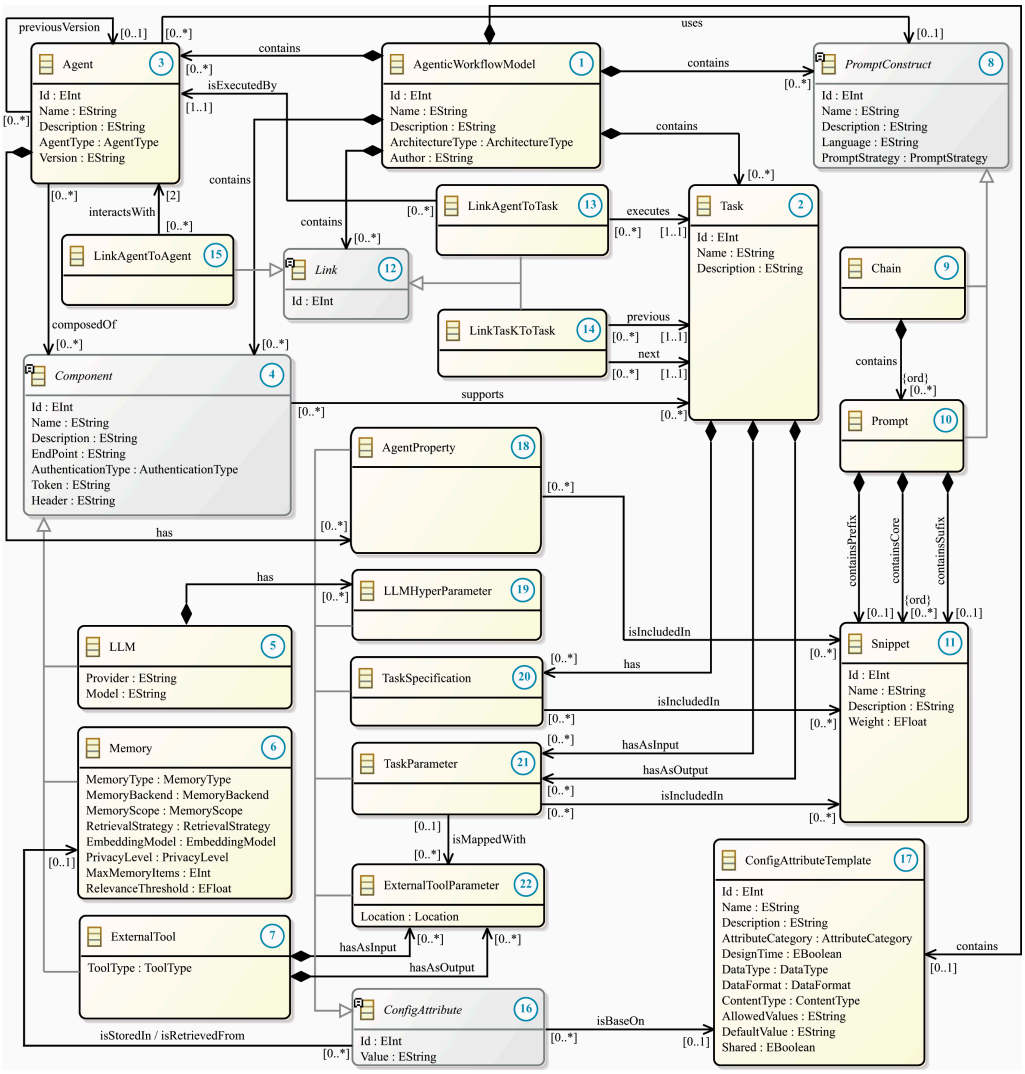


Figure 1. Metaclasses, relationships, and attributes of the AgenticAIFlow metamodel.

AgenticAIWorkflowModel. Root metaclass that encapsulates all components of an agentic AI workflow model. Among the main attributes are: (i) the model's name, description, and author; and (ii) *ArchitectureType*, which specifies the structural and functional organization of the workflow (e.g., pipeline, hierarchical, collaborative) and determines the control, coordination, communication, and information-flow mechanisms among agents during task execution (see Figure 1 (1)).

Task. Represents a workflow task that must be performed by an agent. The metamodel allows the order of task execution to be established using the *LinkTaskToTask* metaclass, enabling tasks to be executed sequentially or in parallel. Moreover, the metamodel provides a general description of the task (*description* attribute) and defines detailed specifications and input/output parameters through the *TaskSpecification* and *TaskParameter* metaclasses,

respectively. This task information can be incorporated into prompt snippets (*isIncludedIn* relationships), so that the LLMs associated with the agents have the necessary context to accomplish the assigned tasks (see Figure 1 (2, 14, 20, 21)).

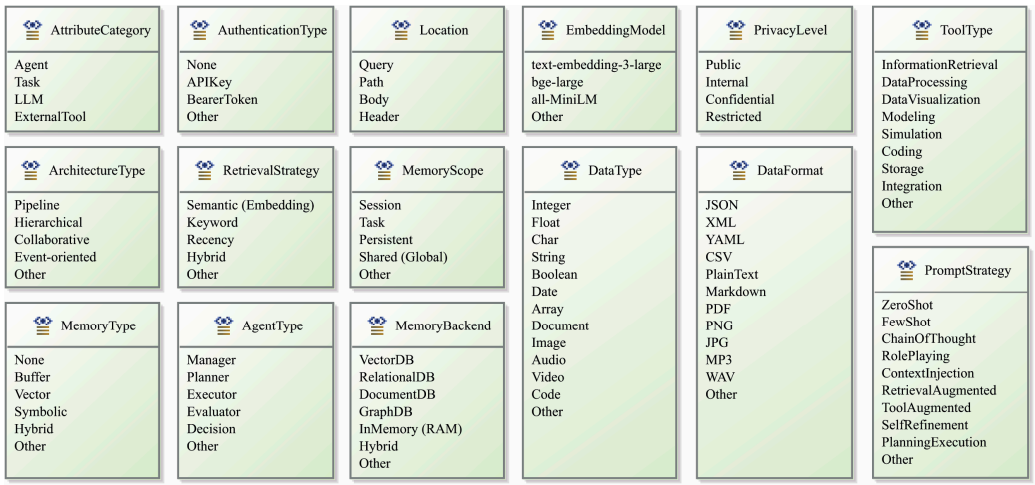


Figure 2. Enumerations of the AgenticAIFlow metamodel.

Agent. Defines an autonomous or semi-autonomous computational entity that perceives the environment, reasons about available information, and executes actions with the support of various components (LLMs, memory, external tools). Through these capabilities, agents can collaborate to solve one or more tasks within the workflow (*LinkAgentToTask* metaclass). In turn, the metamodel enables defining how agents interact with each other (*LinkAgentToAgent* metaclass), based on the workflow architecture and the functional role (*AgentType* property) they play, such as manager, scheduler, executor, or evaluator. Likewise, the metamodel enables a general description of the agent (*description* attribute) and the definition of specific properties (*AgentProperty* metaclass), which can be injected into prompt snippets (*isIncludedIn* relationship). Finally, agents can be versioned to track their evolution over time (see Figure 1 (3, 13, 18)).

Component. An abstract metaclass representing an agent component that can be specialized into an LLM, memory, or external tool. This metaclass includes the following attributes: (i) component name and description; (ii) *EndPoint*, an address used to access the services exposed by the component; (iii) *AuthenticationType*, an authentication mechanism used by the component (e.g., APIKey, BearerToken); (iv) *Token*, a credential used to authenticate and authorize service requests; and (v) *Header*, which contains metadata associated with the service request or response (see Figure 1 (4)).

LLM. A large language model that acts as the agent's cognitive component. Based on the information provided via prompts, the LLM interprets the available context, makes inferences, and plans actions to fulfill the workflow's specific tasks and objectives. This metaclass includes attributes to define the provider name (e.g., OpenAI, Anthropic, Google) and the model identifier associated with the LLM (e.g., GPT-4o, claude-sonnet-4-20250514, gemini-1.5-flash). Additionally, the metamodel allows for the configuration of hyperparameters that control the behavior and response generation of the LLM (e.g., temperature, top_p, max_tokens, frequency_penalty) (see Figure 1 (5, 19)).

Memory. A component that stores, indexes, and retrieves information about past interactions, intermediate states, and domain-specific knowledge, facilitating the preservation of context over time, coherence across prolonged interactions, and the adaptation of behavior to changing environments. Among the main attributes are (i) *MemoryType*, the representation strategy for stored information; (ii) *MemoryBackend*, the underlying storage

infrastructure; (iii) *MemoryScope*, the temporal or contextual scope of memory (session, task, persistent across sessions, shared among agents); (iv) *RetrievalStrategy*, the mechanism used to retrieve stored information; (v) *PrivacyLevel*, the level of memory access control; (vi) *MaxMemoryItems*, the maximum number of items that can be stored before replacement or compression policies are triggered; and (vii) *RelevanceThreshold*, the minimum relevance or similarity score required for a memory item to be considered during retrieval (see Figure 1 (6)).

ExternalTool. A computational component that exposes an API that can be invoked by the agent to extend its capabilities beyond those provided by LLMs. Among the main functionalities that external tools provide to an agent are: access to external information sources, data processing, code execution, modeling, storage, and, in general, integration with the environment and third-party services. In turn, the metamodel allows specifying the input parameters required for the tool's execution, as well as the output information generated by the tool execution (*ExternalToolParameter* metaclass). These input/output parameters of the tools must be mapped (*isMappedWith* relationship) to the input/output parameters of the tasks in order to associate pre-generation or post-generation processes with the execution of the LLMs (see Figure 1 (7, 22)).

PromptConstruct. An abstract metaclass representing the key elements for constructing a prompting strategy that guides the behavior, reasoning, and response generation of an LLM during the execution of workflow tasks. This metaclass can specialize as a chain or a prompt. Moreover, it includes attributes to define the language and techniques (e.g., ZeroShot, FewShot, Chain of Thought) used to formulate prompts (see Figure 1 (8–10)).

Chain. Represents an ordered sequence of interconnected prompts, where the output of one prompt can be used as the input for the next. The purpose of a chain is to divide a workflow task into multiple steps, enabling the LLM to reason incrementally and in a controlled manner. In addition, the chain supports the storage of partial results between steps through working memory, as well as the accumulation and management of context throughout the entire sequence of prompts (see Figure 1 (9)).

Prompt. Represents the basic unit of instruction sent to an LLM to induce the generation of a specific output. A prompt is composed of an ordered sequence of snippets. Each prompt may contain at most one *prefix* snippet and one *suffix* snippet, while multiple *core* snippets may be included (see Figure 1 (10)).

Snippet. Defines a reusable content fragment designed to be incorporated into a prompt. Specifically, snippets can encapsulate agent properties, task specifications, and descriptions of the task's input/output parameters using the *isIncludedIn* relationship. The content encapsulated in the snippets may include text, documents, code, images, and other types of data. Each snippet is assigned a weight (0–1) indicating its relevance to the prompt. For example, a prompt for eliciting software requirements may include a prefix snippet defining the agent's role: *"You are a software analyst with expertise in requirements engineering …"*, a core snippet specifying the main task: *"Your task is to identify and structure the functional and non-functional requirements for a data science portal …"*, and a suffix snippet specifying the output constraints: *"Deliver the response in a document structured into two sections: functional and non-functional requirements, using technical language …"* (see Figure 1 (11)).

ConfigAttribute. An abstract metaclass representing a configuration attribute used to dynamically configure and characterize a component of the agentic AI workflow model. In particular, this metaclass can be specialized into agent properties, LLM hyperparameters, task specifications, and input/output parameters. Configuration attributes have a *value* field, whose content can be set both at design time and during the execution of the agentic AI workflow. Furthermore, these attributes and their values are dynamically embedded

into prompts to influence LLM behavior and the operation of external tools. For attributes whose data type corresponds to resources, such as documents, images, audio, or video, the *value* field stores a Uniform Resource Identifier (URI) that enables the associated resource to be identified, located, and accessed (see Figure 1 (16)).

ConfigAttributeTemplate. Defines a template for a configuration attribute, allowing common specifications to be reused across multiple components of the same type, either within the current model or in other models. For example, a template can define a temperature hyperparameter for reuse across multiple LLMs or establish a specification based on a standard that can be associated with various tasks. When creating an attribute template, the following must be specified: (i) the attribute's name and description; (ii) *AttributeCategory*, defining the type of component with which the attribute is associated (e.g., agent, task, LLM, or external tool); (iii) *DesignTime*, determining whether the attribute's value should be assigned during the design phase or at workflow runtime; (iv) *DataType* and *DataFormat*, specifying the type and format of information that the attribute can contain; (v) *AllowedValues* and *DefaultValue*, defining the allowed values and the default value used if no explicit value is specified for the attribute, respectively; and (vi) *Shared*, determining whether the attribute can be shared and reused across different models (see Figure 1 (17)).

Link. An abstract metaclass that defines a relationship (visual link) between the main components (agents and tasks) of an agentic AI workflow model. It can be specialized into *LinkAgentToTask*, *LinkTaskToTask*, and *LinkAgentToAgent* (see Figure 1 (12–15)).

4.2. Architecture and Components of the AgenticAIFlow DSL

The AgenticAIFlow DSL includes a graphical designer for modeling workflows supported by multiple AI agents, in accordance with the syntax, semantics, and constraints defined by the metamodel presented in the previous subsection. This graphical designer improves the modeling experience by abstracting the technical complexity inherent in the design and development of agentic AI workflows. Likewise, it provides a high degree of flexibility through user-defined configuration parameters for the main components of the model, including agents, tasks, LLMs, memories, external tools, and prompting strategies, thereby enabling the DSL to adapt to the specific requirements of different application domains. Complementing these capabilities, AgenticAIFlow includes a graph-based orchestration engine that coordinates the execution of tasks by agents in accordance with the workflow model's specifications. This engine integrates an interface that allows users to visualize the traceability and status of executed activities, as well as to consult the intermediate and final results generated during workflow execution. Figure 3 presents the AgenticAIFlow user interface, which includes the modeling area where the DSL's graphical notation (concrete syntax) is displayed. It also shows the graphical designer's tool palette, the maintenance menu for the main components of an AI agent, and the monitoring window with the workflow execution logs and generated results.

The DSL architecture is structured into four decoupled layers that promote separation of concerns, modularity, maintainability, and scalability. Each layer is described below, including its functionality, components, and their interactions (see Figure 4).

Presentation Layer. This layer represents the user interface of the AgenticAIFlow DSL, implemented as a Single Page Application (SPA) using React 19, Vite, React Router DOM, and TailwindCSS. This layer runs entirely in the browser and is responsible for dynamic rendering of the interface, navigation between views without full-page reloads, and consumption of RESTful services exposed by the backend. The interface provides a modeling canvas for graphically designing (through drag-and-drop) and parameterizing (through forms) the components of an agentic AI workflow. In this way, the sequence of tasks and the assignment and coordination relationships between tasks and agents can be

visually defined. Likewise, the interface allows users to debug and execute workflows and visualize the results generated during execution, as shown in Figure 3.

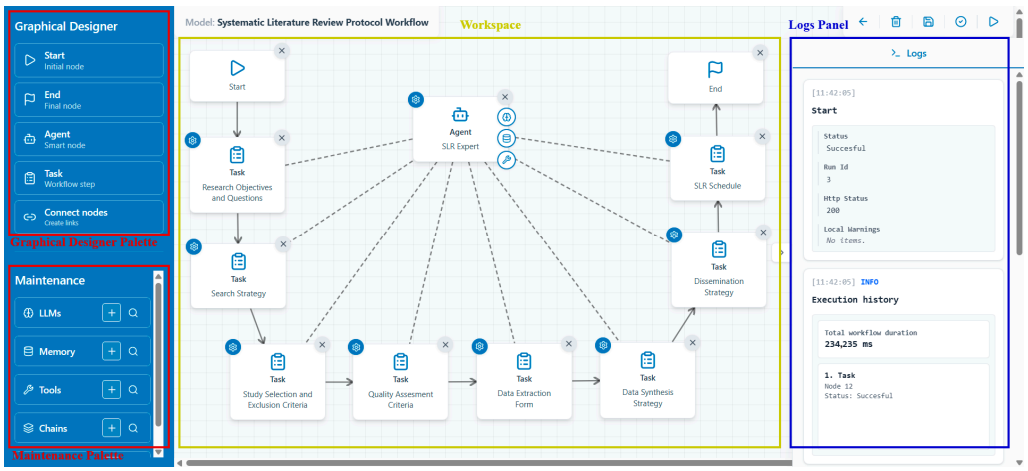


Figure 3. AgenticAIFlow DSL user interface.

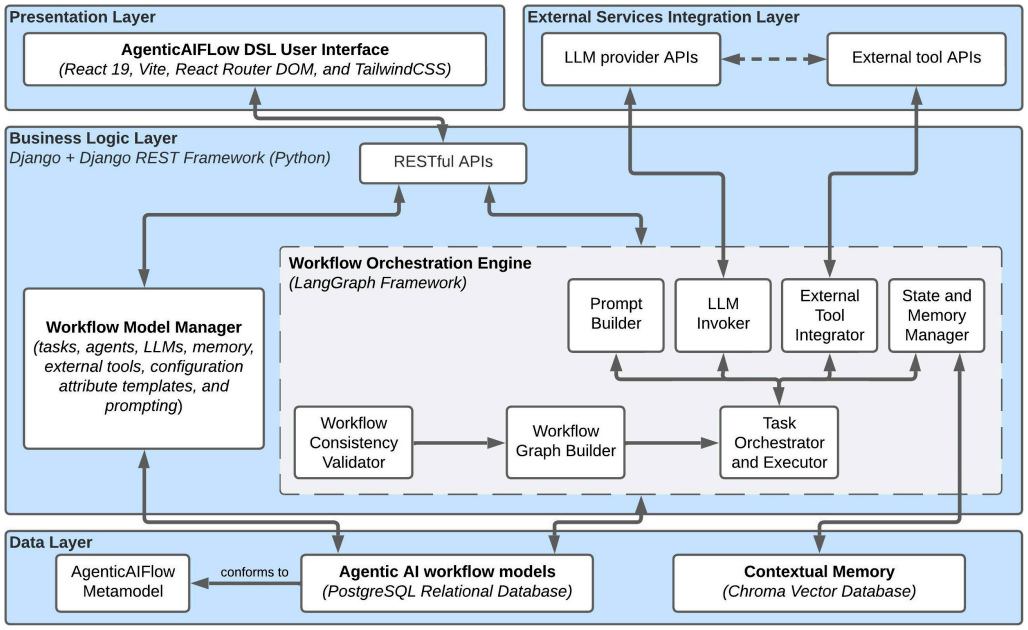


Figure 4. AgenticAIFlow DSL architecture.

Business Logic Layer. Built with Django and Django REST Framework, this layer exposes a set of RESTful services that handle requests from the presentation layer, invoke class methods to execute business rules, and serialize Django models to JSON for information exchange with the SPA. This layer is divided into two modules: *Workflow Model Manager* and *Workflow Orchestration Engine*. The first module provides methods for defining and maintaining agentic AI workflow models, including tasks, agents, LLMs, memory, external tools, attribute templates, and prompting strategies (hierarchically structured into chains, prompts, and snippets), as well as the relationships among these components. The second module is based on the LangGraph framework [37] to perform the following tasks: (i) debugging of the workflow model according to the syntax, semantics, and constraints defined in the metamodel; (ii) construction of a shared-state directed graph (StateGraph), where the nodes represent the workflow tasks and the edges represent the transitions between them; (iii) initialization and dynamic execution of graph nodes (tasks) based on

the system's shared state, which represents contextual information; (iv) dynamic prompt generation for task execution by agents, considering the model specifications, the data inputs provided by the user at runtime, and the shared state; (v) invocation of the LLMs for task resolution; (vi) update and propagation of the shared state throughout the entire execution of the workflow, ensuring its persistent storage (e.g., through vector memory supporting RAG); and (vii) integration with external tools for the execution of preprocessing and postprocessing operations on the inputs and outputs generated by the LLMs, respectively.

Regarding the static analysis of models to verify their consistency, syntax, and semantics, AgenticAIFlow incorporates validation mechanisms at two stages. During the modeling and component configuration stage, the graphical designer restricts the creation of invalid relationships among workflow elements based on the types and cardinalities defined in the metamodel. Subsequently, during the model debugging stage (prior to execution), the orchestration engine validates the model topology against the selected workflow architecture and the constraints and rules defined in the metamodel. This validation includes, among others, the following aspects: (1) consistency among agent types, their relationships, and the selected workflow architecture; (2) definition of a continuous execution flow, with properly established start and end points and valid transitions between tasks, with no orphaned or isolated tasks; (3) assignment of at least one agent to each task and at least one task to each agent; (4) availability of an LLM and a memory component for each agent, while external tools are optional; (5) inclusion of at least one prompt in each chain; (6) configuration of each prompt with at most one prefix and one suffix snippet, and at least one core snippet; and (7) association of the configuration attributes of the components with at least one snippet.

External Services Integration Layer. This layer manages interoperability and communication with external services, tools, and systems by consuming APIs specified in the workflow models. This enables the dynamic invocation of external tools during the execution of agentic AI workflows, thereby extending the functional capabilities of agents.

Data Layer. This layer stores the agentic AI workflow models in a PostgreSQL database. Data access and manipulation are abstracted through Django's ORM (Object–Relational Mapping). Likewise, this layer incorporates a vector database implemented in Chroma to support the persistent storage of the contextual state and its semantic retrieval through RAG during workflow execution. The DSL is technology-independent and supports the integration of different vector database systems through the configuration of their respective endpoints and connection parameters.

5. Empirical Evaluation of the Solution

This section presents an empirical evaluation of the AgenticAIFlow DSL through a quasi-experiment to assess user performance (effectiveness and efficiency) when using the proposed solution, as well as their perceived ease of use (PEOU), perceived usefulness (PU), and intention to use (ITU) the DSL. To this end, the experiment was designed based on the MEM [35]. The following subsections briefly describe the evaluation scenario and the design, execution, and analysis of the experimental results.

5.1. Evaluation Scenario

The experimental study was conducted in the context of software requirements engineering and consisted of the design and implementation of a generic agentic AI workflow to support the preparation of software requirements specification (SRS) documents in any domain, based on the methodology proposed by Sommerville [38] and the IEEE-830 standard [39]. The experiment was conducted with the participation of final-year students

from the Computer Science program at the Universidad del Azuay who had successfully completed the Requirements Engineering course.

The following requirements were defined for the quasi-experiment:

- Design of a pipeline-based workflow that includes the following tasks: (i) identification of the system context and stakeholder needs; (ii) elicitation of software requirements; (iii) analysis, prioritization, and classification of software requirements; (iv) modeling and generation of use case diagrams; (v) preparation of the SRS document according to the IEEE-830 standard structure; and (vi) validation of the SRS based on the quality criteria (e.g., correct, complete, unambiguous, verifiable, etc.) of the aforementioned standard. For each task, participants had to define a general description, along with additional specifications regarding the standard to be used, restrictions, quality criteria, and output formats. Furthermore, for the first task, a runtime input parameter provided by the user had to be defined. This parameter corresponds to the problem statement document, which serves as a trigger for the workflow and summarizes the system context and business need.
- Creation of two agents: the first (Requirements Engineer) specialized in the analysis, specification, and validation of software requirements, while the second (UML Modeler) focused on the design of UML diagrams using PlantUML code [40]. Specific properties were also defined for each agent, such as role, specialization, constraints, and writing language and style.
- Configuration of the components associated with the agents: (i) an LLM based on the claude-sonnet-4-20250514 model from Anthropic [41]; (ii) a vector database implemented in Chroma as contextual memory [42]; and (iii) PlantUML as an external tool for generating UML diagrams [40]. The LLM and memory were shared by both agents. It should be noted that all necessary information regarding endpoints and authentication mechanisms was provided to parameterize the components. For the LLM, the temperature hyperparameter was set to 0.2 by default to achieve low creativity and high accuracy in requirements generation.
- Definition of the prompting strategy, composed of a chain and multiple prompts, one for each workflow task. Prefix snippets were associated with agent properties, core snippets with task specifications, and suffix snippets with task output formats.

During the workflow execution, participants used a problem statement related to the need to develop a data science services portal. Figure 5 presents the resulting agentic AI workflow model from the evaluation scenario, designed in the AgenticAIFlow DSL.

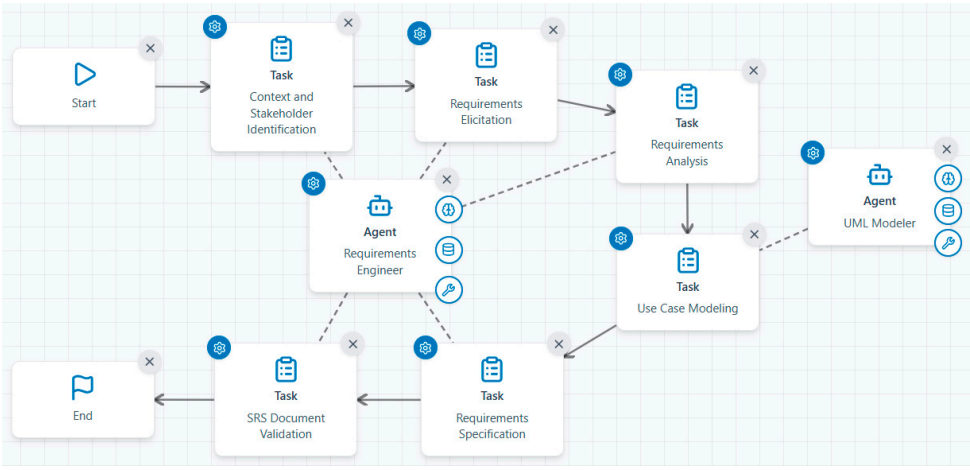


Figure 5. Agentic AI workflow model for software requirements specification development.

5.2. Design of the Quasi-Experiment

The quasi-experiment was designed according to the experimental methodology proposed by Wohlin et al. [34]. Table 1 presents the *objective of the quasi-experiment* following the Goal-Question-Metric (GQM) paradigm [43], while Table 2 details the corresponding *research questions and hypotheses*, formulated in terms of the performance and perception variables defined in the MEM, as well as their causal relationships.

Table 1. Goal-Question-Metric for the quasi-experiment.

Evaluate:	AgenticAIFlow DSL for the design and implementation of agentic AI workflows.
With the purpose of:	• Determining user performance in terms of effectiveness and efficiency when using AgenticAIFlow. • Understanding user perceptions of the ease of use, usefulness, and intention to use AgenticAIFlow in the future.
From the viewpoint of:	Researcher.
In the context of:	Senior-year students of the Computer Science Engineering program at the Universidad del Azuay.

Table 2. Research questions and hypotheses for the quasi-experiment.

Questions	Hypotheses
RQ1. Is the AgenticAIFlow DSL perceived as easy to use and useful? If so, are user perceptions a result of their performance when using the proposed solution?	• $H1_0$: The AgenticAIFlow DSL is perceived as difficult to use, $H1_0 = \neg H1_1$. • $H2_0$: The AgenticAIFlow DSL is not perceived as useful, $H2_0 = \neg H2_1$. • $H4_0$: PEOU is not determined by efficiency, $H4_0 = \neg H4_1$. • $H5_0$: PU is not determined by effectiveness, $H5_0 = \neg H5_1$. • $H6_0$: PU is not determined by PEOU, $H6_0 = \neg H6_1$.
RQ2. Is there an intention to use the AgenticAIFlow DSL in the future? If so, is the intention to use a result of user perceptions?	• $H3_0$: There is no intention to use the AgenticAIFlow DSL in the future, $H3_0 = \neg H3_1$. • $H7_0$: ITU is not determined by PEOU, $H7_0 = \neg H7_1$. • $H8_0$: ITU is not determined by PU, $H8_0 = \neg H8_1$.

The *context of the quasi-experiment* was determined by: (i) the participation of final-year students from the Computer Science Engineering program at the Universidad del Azuay; (ii) the evaluation scenario described in the previous subsection; and (iii) the proposed AgenticAIFlow solution.

The *tasks of the quasi-experiment* were based on the established methodological process for the design and implementation of agentic AI workflows using AgenticAI-Flow, and are listed below: (i) configuration of the prompting strategy (chains, prompts, and snippets); (ii) configuration of LLMs, memory, and external tools for the agents; (iii) design of the AI agents and component assignment; (iv) modeling of workflow tasks and assignment of the responsible agents for their execution; (v) debugging and execution of the agentic AI workflow; and (vi) review of the logs and the generated SRS document.

The *dependent variables of the quasi-experiment* used to evaluate AgenticAIFlow were user performance (effectiveness and efficiency) and user perceptions (PEOU, PU, ITU), as established in the MEM [35].

The *experimental materials* were distributed via a website and comprised (i) a presentation and a training exercise used by the researchers during solution training; (ii) a bulletin and a worksheet with the description and instructions for the experimental exercise, including fields to collect metrics on the activities performed by the participants and the

results obtained; and (iii) a Google Forms questionnaire designed to assess participants' perceptions of AgenticAIFlow.

5.3. Adaptation of the MEM for the Evaluation of the Solution

Figure 6 presents the adapted MEM model, including the measurement instruments used to evaluate the AgenticAIFlow DSL. The effectiveness and efficiency variables were assessed using formulas that measure the degree of fulfillment and the time spent to complete all experimental tasks, respectively. The perception variables (PEOU, PU, ITU) were evaluated using a five-point Likert scale questionnaire, with a score of three considered neutral and used as the reference value for determining acceptance or rejection of the proposed solution. This questionnaire is available at https://doi.org/10.5281/zenodo.20550780. Finally, the causal relationships among the MEM variables were analyzed using simple linear regression.

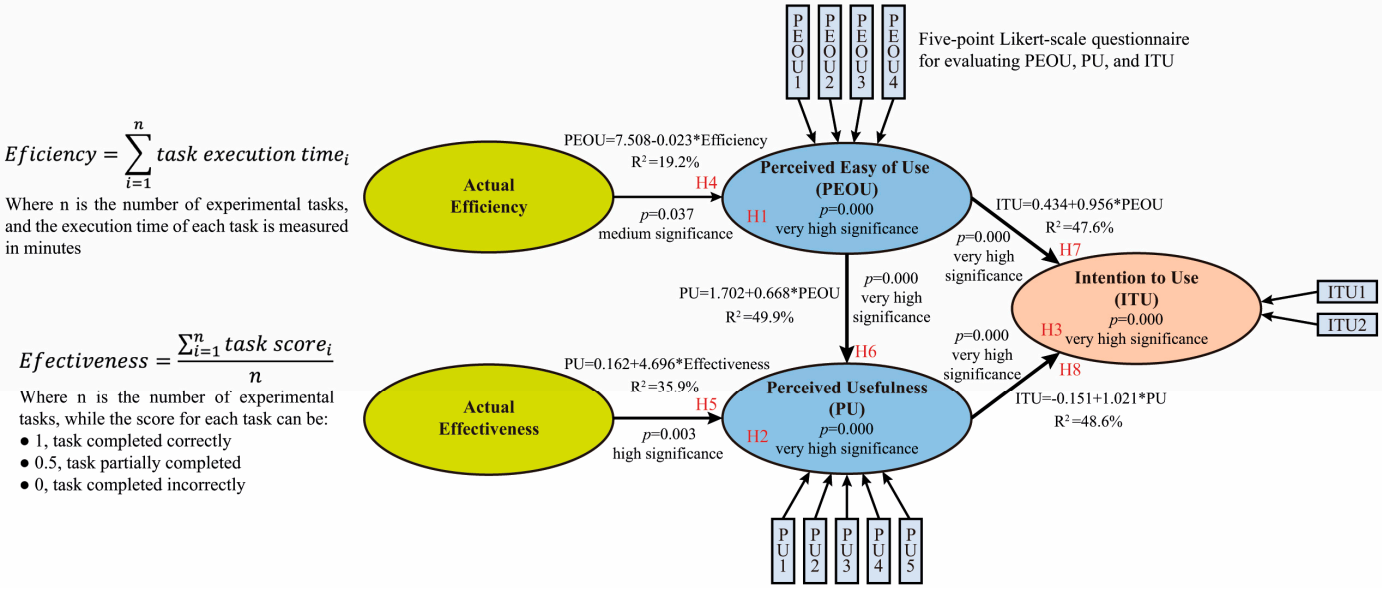


Figure 6. Summary of AgenticAIFlow DSL evaluation using a MEM-adapted model.

5.4. Execution of the Quasi-Experiment

The quasi-experiment involved 35 students and lasted 4.5 h, divided into two sessions. The first session lasted 2 h and was aimed at training students in generative and agentic AI, as well as in the use of AgenticAIFlow, through the design and implementation of a workflow for developing Systematic Literature Review (SLR) protocols. The second session lasted 2.5 h and focused on conducting the experimental exercise. At the end of the experiment, each participant completed the perception questionnaire via Google Forms and submitted the worksheet corresponding to the experimental exercise by email, along with the SRS document generated by the agentic AI workflow.

5.5. Results of the Quasi-Experiment

The data collected during the quasi-experiment were analyzed using descriptive statistics, box plots, and statistical tests in SPSS ($\alpha = 0.05$) to evaluate the proposed hypotheses. The significance levels defined by Moody [44] were used to determine the statistical decision for each null hypothesis. Table 3 presents the results for the performance and perception variables, with outliers excluded from the statistical analyses.

Table 3. Results of user perception and performance variables.

Variable	Min.	Max.	Mean	Std. Dev.	Shapiro-Wilk Test *p*-Value	*t*-test/Wilcoxon Test *p*-Value
PEOU	3.2500	5.0000	4.2065	0.5093	**0.172 > 0.05**	**0.000 < 0.001**
PU	3.4000	5.0000	4.5130	0.4818	**0.009 < 0.05 ***	**0.000 < 0.001 ****
ITU	3.0000	5.0000	4.4565	0.7057	**0.000 < 0.05 ***	**0.000 < 0.001 ****
Effectiveness	0.7500	1.0000	0.9266	0.0615		
Efficiency (minutes)	128.0000	163.0000	140.6522	9.4993		

* The variable does not follow a normal distribution. ** Results of the one-tailed one-sample Wilcoxon test.

5.5.1. Analysis of User Perceptions

Overall, the mean scores for the perception variables (PEOU, PU, ITU) were above the neutral point (3) on the Likert scale (see Table 3). Furthermore, the Shapiro–Wilk test was conducted to assess the normality of the data distributions. The results indicated that PEOU followed a normal distribution ($p > 0.05$), while PU and ITU did not ($p < 0.05$). Therefore, a one-sample *t*-test was applied to PEOU, while one-tailed one-sample Wilcoxon tests were conducted for PU and ITU. In both cases, a test value of 3, corresponding to the questionnaire's Likert scale neutral point, was used. Finally, the results revealed a very high level of statistical significance for all three variables ($p < 0.001$), leading to the rejection of the null hypotheses $H1_0$, $H2_0$, and $H3_0$ (see Table 3). These findings indicate that participants perceive the proposed solution as easy to use and useful, and intend to use it in future projects.

5.5.2. Analysis of User Performance

The participants achieved an average effectiveness of 92.66%, indicating that the vast majority successfully completed the experimental tasks involved in designing and implementing the agentic AI workflow for developing SRS documents (see Table 3).

The most recurrent errors occurred during the configuration of the prompting strategy, mainly due to omissions and inaccuracies in defining the configuration attributes of workflow components, such as agent properties (e.g., role, specialization, and writing style) and task specifications (e.g., standards to be used, constraints, and output formats). In addition, in some cases, participants omitted the mapping between configuration attributes and prompt snippets, as well as the association between LLMs and agents. In contrast, relatively few errors were observed in the graphical modeling of the workflow task sequence and its association with the corresponding agents.

Nevertheless, the formal language (metamodel) and the methodological process supporting the design of agentic AI workflow models, together with the consequent automatic and ordered construction of the orchestration graph and LLM instructions, structured into chains, prompts, and snippets (prefix, core, and suffix), contributed to substantially reducing errors in prompt engineering compared with a chatbot-based solution lacking explicit mechanisms for task-flow and reasoning control. As a result, improvements were observed in the standardization, consistency, homogeneity, reproducibility, and traceability of the participants' outputs.

The participants' efficiency ranged from 128 to 163 min, with an average completion time of 140.65 min and a standard deviation of 9.50 min (see Table 3). This result indicates a moderate level of dispersion, which could be attributed to factors such as participants' analytical skills, the rigor with which they performed the tasks, and their level of knowledge in generative AI (prompt engineering), agentic AI, and the application domain. Consequently, this variability could potentially be reduced through more in-depth training involving a greater number of practical exercises.

In summary, the vast majority of participants successfully generated an SRS document that met the quality criteria defined by the IEEE-830 standard in considerably less time than would typically be required for manual authoring of such a document using a traditional requirements engineering approach. However, the generated documents should be considered preliminary artifacts intended to support and accelerate the requirements engineering process. Therefore, they must undergo review, validation, and refinement by domain experts and requirements engineers prior to their final use.

Regarding the participants' comments collected through the open-ended question in the questionnaire, most participants noted that the solution is intuitive for graphical workflow modeling. However, they recommended improving the usability of the modal pop-up forms used to maintain the configuration attributes of workflow components, as well as the prompting strategy (chain, prompt, and snippet). They also suggested simplifying and standardizing the processes for associating agents, components, configuration attributes, and snippets. These observations are consistent with the quantitative results obtained in the experiment. Consequently, future versions of the graphical editor will focus on reducing the complexity of these forms through the following actions: (i) improving their organization and presentation by providing clear label descriptions, appropriately selecting input control types, and incorporating tooltips to provide contextual information; (ii) optimizing and standardizing the configuration process for workflow components and their relationships (task, agent, LLM, memory, external tool, chain, prompt, and snippet), thereby reducing the number of iterations required by the user; (iii) incorporating validation mechanisms for form fields and their dependencies (e.g., verifying mandatory associations between workflow components); (iv) leveraging the configuration attribute templates supported by the metamodel to suggest, from a predefined list, commonly used attributes for each component type (e.g., agent role, LLM temperature, or output format for a task); and (v) exploring the incorporation of an AI-based assistant that provides recommendations on potential configuration attributes according to the component type and the application domain of the workflow. These improvements aim to reduce cognitive load and user effort during configuration while promoting greater consistency and efficiency in the modeling process.

5.5.3. Analysis of Causal Relationships

The MEM structure was validated using simple linear regression to identify potential causal relationships among its variables and to evaluate hypotheses $H4_0$, $H5_0$, $H6_0$, $H7_0$, and $H8_0$. Figure 6 shows the results of this analysis, including the linear regression models, their coefficients of determination (R^2), and *p*-values. Hypotheses $H4_0$ and $H5_0$ were rejected, suggesting that participants' current efficiency and effectiveness influence their perceptions of the solution's ease of use and usefulness, respectively. Similarly, hypothesis $H6_0$ was rejected, indicating an association between perceived ease of use and perceived usefulness. Finally, hypotheses $H7_0$ and $H8_0$ were also rejected, confirming that both perceived ease of use and perceived usefulness positively influence the intention to use the solution in the future.

5.6. Threats to the Validity of the Quasi-Experiment

The main threats to the validity of the evaluation results that were mitigated during the quasi-experiment are the following:

- *Internal validity*, associated with participants' experience, potential biases in the experimental material, and the lack of a direct baseline control group, such as comparing performance with code-based implementations using frameworks like CrewAI or LangChain. The first two threats were mitigated through prior training and pilot experiments with experts. The latter threat was partially mitigated by using a test

value of 3, corresponding to the neutral point of the five-point Likert scale used in the questionnaire, to determine whether the perception variables were significantly higher than the neutral level. However, this procedure does not replace a direct comparison against the baseline. Therefore, such an evaluation will be conducted in future work.

- *External validity*, which concerns the representativeness of the participants and the generalizability of the results, was mitigated through the use of a realistic evaluation scenario and the participation of final-year students from the Computer Science program who had previously completed courses related to the experimental context. According to several studies [45–47], the skills of such students can be comparable to those of junior professionals. Nevertheless, to strengthen the generalizability of the findings, future work will include case studies in real-world and industrial contexts across different application domains. In particular, with the participation of professionals and experts from the Department of Quality and the Vice-Rectorate for Research at Universidad del Azuay, agentic AI workflows will be designed and implemented using AgenticAIFlow to collect and systematize information from various internal and external sources and generate scientific production reports for quality assurance and institutional accreditation purposes. Likewise, with the participation of Software Engineering experts from local public institutions, agentic AI workflows will be implemented to support various activities in the software development process, such as requirements elicitation, the modeling of different software perspectives and concerns, and software testing, among others. These case studies will involve more complex scenarios that consider a wide variety of workflow architectures, agent types, LLMs, and external tools. This will allow AgenticAIFlow to be evaluated under different configurations, providing additional evidence of its performance, applicability, flexibility, and scalability for modeling and implementing agentic AI workflows in professional contexts.
- *Construct validity*, associated with the ability of the variables and instruments to adequately measure the constructs under study, was addressed by using the MEM and assessing questionnaire reliability with Cronbach's alpha [48]. The results exceeded the recommended threshold ($\alpha \geq 0.70$) for PEOU (0.77), PU (0.84), and ITU (0.94).
- *Conclusion validity*, related to the interpretation of the results, was mitigated by selecting statistical methods and tests appropriate to the nature of the data.

6. Conclusions and Future Work

The literature review revealed a limited availability of DSLs dedicated to the design and implementation of agentic AI workflows. This limitation hinders the abstract, formal, and standardized representation of the concepts and relationships that characterize such workflows, thereby constraining the ability to model solutions at the problem-domain level rather than at the implementation level. Furthermore, it restricts the development of agentic AI workflow solutions with desirable properties such as verifiability, reproducibility, maintainability, traceability, reusability, and scalability.

In this context, the AgenticAIFlow DSL contributes to addressing this research gap by providing a modeling language sufficiently comprehensive and expressive to represent the composition of an agentic AI workflow, including tasks, agents, LLMs, contextual memory, prompting strategies, and external tools. The proposal is complemented by a graph-based orchestration engine that interprets and automates high-level model specifications to coordinate the execution of agents and their resources to complete workflow tasks. In this way, AgenticAIFlow abstracts the inherent complexity of developing agentic AI solutions, decoupling system specifications from implementation details. The empirical evaluation, conducted as a quasi-experiment, yielded favorable results regarding participants' effective-

ness and efficiency in using AgenticAIFlow for the design and implementation of agentic AI workflows. Likewise, the results indicated that participants perceived the solution as easy to use and useful, and expressed an intention to use it in the future.

As future work, it is proposed to continue improving the solution based on the results of further experiments incorporating direct comparisons with baseline approaches (e.g., CrewAI or LangChain), as well as case studies in real-world and industrial contexts across different application domains. Additionally, the aim is to enrich the modeling language with governance specifications for agentic AI workflows, including security policies, responsible AI principles, quality controls, auditing and traceability, human-in-the-loop supervision, and regulatory compliance. Finally, it is proposed to extend the orchestration engine's capabilities to strengthen coordination among agents and support cyclic workflows oriented toward continuous and iterative improvement.

Author Contributions: Conceptualization, L.E.-G.; methodology, L.E.-G. and P.C.; software, J.C., R.O. and L.E.-G.; validation, L.E.-G., J.C., R.O., L.I. and P.C.; formal analysis, L.E.-G., L.I. and P.C.; investigation, L.E.-G., J.C., R.O., L.I. and P.C.; resources, L.E.-G. and P.C.; data curation, L.E.-G. and L.I.; writing—original draft preparation, L.E.-G., J.C., R.O., L.I. and P.C.; writing—review and editing, L.E.-G., J.C., R.O., L.I. and P.C.; visualization, L.E.-G., J.C. and R.O.; supervision, L.E.-G.; project administration, L.E.-G.; funding acquisition, L.E.-G. and P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is part of the research project "Domain-Specific Language for Designing and Implementing Agentic AI Workflows for Software Engineering Processes and Knowledge Generation. Case Studies: Software Requirements Specification and Systematic Literature Review Protocol Development", Grant 2026-0041.

Data Availability Statement: The raw data supporting the results and conclusions of this study are available upon request from the corresponding author.

Acknowledgments: The authors gratefully acknowledge Universidad del Azuay, Universidad de Cuenca, Universidad Politécnica Salesiana, and Universitat Politècnica de València for their support of this project.

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

References

1. Kaur, P.; Siddharth Kashyap, G.; Kumar, A.; Nafis, M.T.; Kumar, S.; Shokeen, V. From text to transformation: A comprehensive review of large language models' versatility. *arXiv* **2024**, arXiv:2402.16142. [CrossRef]
2. Sajjadi Mohammadabadi, S.M.; Kara, B.C.; Eyupoglu, C.; Uzay, C.; Tosun, M.S.; Karakuş, O. A survey of large language models: Evolution, architectures, adaptation, benchmarking, applications, challenges, and societal implications. *Electronics* **2025**, *14*, 3580. [CrossRef]
3. Corvalán, J.G.; Le Fevre Cervini, E.M.; Sánchez Caparrós, M. *Artificial Intelligence Agents and Agentic Workflows: The New Frontier of Automation*, 1st ed.; UBA IALAB: Buenos Aires, Argentina, 2025.
4. Vinay, V. Failure modes in LLM systems: A system-level taxonomy for reliable AI applications. *arXiv* **2025**, arXiv:2511.19933. [CrossRef]
5. Koubaa, A. From pre-trained language models to agentic AI: Evolution and architectures for autonomous intelligence. *Preprints* **2025**, 2025071809. Available online: https://www.preprints.org/manuscript/202507.1809 (accessed on 16 June 2026).
6. Anthropic. Building Effective Agents. Anthropic Engineering Blog, 19 December 2024. Available online: https://www.anthropic.com/engineering/building-effective-agents (accessed on 16 June 2026).
7. Sapkota, R.; Roumeliotis, K.I.; Karkee, M. AI agents vs. agentic AI: A conceptual taxonomy, applications and challenges. *Inf. Fusion* **2026**, *123*, 103599. [CrossRef]
8. Yu, C.; Cheng, Z.; Cui, H.; Gao, Y.; Luo, Z.; Wang, Y.; Zheng, H.; Zhao, Y. A survey on agent workflow—status and future. In Proceedings of the 2025 8th International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 23–26 May 2025.
9. Abou Ali, M.; Dornaika, F.; Charafeddine, J. Agentic AI: A comprehensive survey of architectures, applications, and future directions. *Artif. Intell. Rev.* **2025**, *59*, 11. [CrossRef]

10. France, R.; Rumpe, B. Model-driven development of complex software: A research roadmap. In Proceedings of the Future of Software Engineering (FOSE'07), Minneapolis, MN, USA, 23–25 May 2007.
11. Di Rocco, J.; Di Ruscio, D.; Di Sipio, C.; Nguyen, P.T.; Rubei, R. On the use of large language models in Model-Driven Engineering. *Softw. Syst. Model.* **2025**, *24*, 923–948. [CrossRef]
12. Barriga, J.A.; Barriga, A.; Clemente, P.J. Model-driven development toward LLM-based multi-agent systems. In *Actas de las XXIX Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2025)*; SISTEDES: Madrid, Spain, 2025.
13. Stryker, C. What Are Large Language Models (LLMs)? IBM. Available online: https://www.ibm.com/think/topics/large-language-models#692473873 (accessed on 7 September 2026).
14. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS 2017), Long Beach, CA, USA, 4–9 December 2017.
15. Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In Proceedings of the 5th International Conference on Learning Representations (ICLR 2017), Toulon, France, 24–26 April 2017.
16. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training language models to follow instructions with human feedback. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, LA, USA, 28 November–9 December 2022.
17. Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv* **2022**, arXiv:2204.05862. [CrossRef]
18. Lee, S.; Lee, G.; Kim, W.; Kim, J.; Park, J.; Cho, K. Human strategy learning-based multi-agent deep reinforcement learning for online team sports game. *IEEE Access* **2025**, *13*, 15437–15452. [CrossRef]
19. Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; Neubig, G. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
20. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi Quoc, E.H.; Le, V.; Zhou, D. Chain-of-Thought prompting elicits reasoning in large language models. In Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, LA, USA, 28 November–9 December 2022.
21. Yao, S.; Yu, D.; Deepmind, G.; Zhao, J.; Griffiths, T.L.; Cao, Y.; Narasimhan, K. Tree of Thoughts: Deliberate problem solving with large language models. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023.
22. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing reasoning and acting in language models. In Proceedings of the 11th International Conference on Learning Representations (ICLR 2023), Kigali, Rwanda, 1–5 May 2023.
23. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-T.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS 2020), Vancouver, BC, Canada, 6–12 December 2020.
24. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. In Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023.
25. OpenAI. Function Calling. OpenAI Developers Documentation. Available online: https://platform.openai.com/docs/guides/function-calling (accessed on 17 June 2026).
26. Castelfranchi, C. Modelling social action for AI agents. *Artif. Intell.* **1998**, *103*, 157–182. [CrossRef]
27. Li, X.; Wang, S.; Zeng, S.; Wu, Y.; Yang, Y. A survey on LLM-based multi-agent systems: Workflow, infrastructure, and challenges. *Vicinagearth* **2024**, *1*, 9. [CrossRef]
28. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A survey on large language model based autonomous agents. *Front. Comput. Sci.* **2024**, *18*, 186345. [CrossRef]
29. Clariso, R.; Cabot, J. Model-driven prompt engineering. In Proceedings of the ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS), Västerås, Sweden, 1–6 October 2023.
30. Morales, S.; Clarisó, R.; Cabot, J. Impromptu: A framework for model-driven prompt engineering. *Softw. Syst. Model.* **2025**, *24*, 1627–1645. [CrossRef]
31. Morales, S.; Clarisó, R.; Cabot, J. A DSL for testing LLMs for fairness and bias. In Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS), Linz, Austria, 22–27 September 2024.
32. Göbel, S.; Lämmel, R. Model-based trust analysis of LLM conversations. In Proceedings of the 2024 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), Linz, Austria, 22–27 September 2024.
33. Gorschek, T.; Garre, P.; Larsson, S.; Wohlin, C. A model for technology transfer in practice. *IEEE Softw.* **2006**, *23*, 88–95. [CrossRef]
34. Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A. *Experimentation in Software Engineering*, 2nd ed.; Springer Nature: Berlin/Heidelberg, Germany, 2024.

35. Moody, D.L. The Method Evaluation Model: A Theoretical Model for Validating Information Systems Design Methods. In Proceedings of the 11th European Conference on Information Systems (ECIS 2003), Naples, Italy, 16–21 June 2003.
36. Eclipse Foundation. Eclipse Modeling Framework (EMF) Documentation. Available online: https://eclipse.dev/emf/docs.html (accessed on 17 June 2026).
37. LangChain Inc. LangGraph Overview—Docs by LangChain. Available online: https://docs.langchain.com/oss/python/langgraph/overview (accessed on 17 June 2026).
38. Sommerville, I. *Software Engineering*, 8th ed.; Pearson Education: Harlow, England, 2006.
39. *IEEE Std 830-1998*; IEEE Recommended Practice for Software Requirements Specifications. IEEE: Piscataway, NJ, USA, 1998.
40. PlantUML Team. *PlantUML Language Reference Guide*, Version 1.2025.0. 2025. Available online: https://plantuml.com/guide (accessed on 17 June 2026).
41. Anthropic. Models Overview—Claude API Docs. Available online: https://platform.claude.com/docs/en/about-claude/models/overview (accessed on 17 June 2026).
42. Chroma. Introduction—Chroma Docs. Available online: https://docs.trychroma.com/docs/overview/introduction (accessed on 17 June 2026).
43. Basili, V.R.; Caldiera, G.; Rombach, H.D. The Goal Question Metric approach. In *Encyclopedia of Software Engineering*; Marciniak, J.J., Ed.; John Wiley & Sons: New York, NY, USA, 1994; Volume 1, pp. 528–532.
44. Moody, D.L. Dealing with Complexity in Information Systems Modeling: Development and Empirical Validation of a Method for Representing Large Data Models. In Proceedings of the 24th International Conference on Information Systems (ICIS 2003), Seattle, WA, USA, 14–17 December 2003.
45. Basili, V.R.; Shull, F.; Lanubile, F. Building Knowledge through Families of Experiments. *IEEE Trans. Softw. Eng.* **1999**, *25*, 456–473. [CrossRef]
46. Höst, M.; Regnell, B.; Wohlin, C. Using Students as Subjects—A Comparative Study of Students and Professionals in Lead-Time Impact Assessment. *Empir. Softw. Eng.* **2000**, *5*, 201–214. [CrossRef]
47. Kitchenham, B.A.; Pfleeger, S.L.; Pickard, L.M.; Jones, P.W.; Hoaglin, D.C.; El Emam, K.; Rosenberg, J. Preliminary Guidelines for Empirical Research in Software Engineering. *IEEE Trans. Softw. Eng.* **2002**, *28*, 721–734. [CrossRef]
48. Cronbach, L.J. Coefficient Alpha and the Internal Structure of Tests. *Psychometrika* **1951**, *16*, 297–334. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.