Pre-print of paper presented at IEEE COMPSAC 2026

Towards a Model-driven Framework for Agentic AI

Antonio Rey-Atienza
Universidad Autónoma de Madrid
antonio.reya@estudiante.uam.es

Juan de Lara
Universidad Autónoma de Madrid
Juan.deLara@uam.es

***Abstract*—Agentic AI applications have become increasingly relevant over the past few years. However, their implementation remains inaccessible to users without significant knowledge of the domain and programming skills. We present a framework for designing agentic workflows using a modeling graph-based language, and provide a low-code platform that incorporates model design, validation, code generation into multiple targets, and execution. The aim is to simplify the design and use of agentic workflows without overly compromising on expressiveness. While we outline areas pending improvement and usability evaluation, our framework represents a first step towards a unified modeling framework for agentic systems.**

***Index Terms*—Graphical Modeling Languages, Agentic AI, Model-driven Engineering, Code generation**

I. Introduction

Agentic systems have gained significant attention due to their potential applications in domains such as finance, customer experience, and intelligent assistants. There are many programming frameworks for building agentic systems, such as LangGraph[1], CrewAI[2], and PydanticAI[3]. However, using them can be challenging, as it requires both familiarity with the agentic AI field and programming experience. Moreover, premature commitment to a specific framework difficults changing into a better one later. Their focus on programming difficults experimenting with several of them for the application in mind, and their low-level nature may obscure the underlying agentic design.

To attack these problems, we present a framework for designing and running agentic workflows without writing code. The aim of our framework is: (i) to lower the entry barrier for non-technical users while allowing for sophisticated applications to be developed, (ii) enable more advanced developers to generate code for different agentic frameworks to ease experimentation, and (iii) be an open platform, facilitating the inclusion of additional compilation targets.

There exist both open source and proprietary lowcode platforms for agent workflow design, such as LangFlow[4], Dify[5], OpenAI Agent Builder[6], or StackAI[7]. These have shown that agentic workflow design can be made more accessible by using visual interfaces. However, they rely on a specific underlying agentic framework (e.g., OpenAI), and do not allow their extension with compilers for other targets. Some research works propose extending BPMN to model human-agent collaboration [1], but their goal is not to model the agents themselves, or to generate code.

Work funded by the Spanish MICINN with project PID2024-155231OBI00

[1] https://www.langchain.com/langgraph [2] https://crewai.com/
[3] https://ai.pydantic.dev/ [4] https://www.langflow.org/ [5] https://dify.ai/
[6] https://openai.com/es-ES/agent-platform/ [7] https://www.stackai.com/

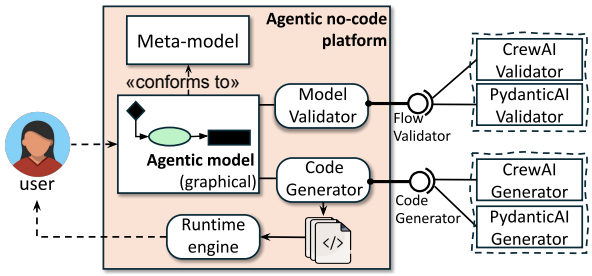

Fig. 1. Working scheme of our MDE agentic framework.

II. A model-driven framework for agentic AI

Our framework (Fig. 1) is based on the principles of model-driven engineering (MDE) [2]. Users design agentic workflows by creating high level representations of the desired behavior in a graphical language. This language captures the core concepts of agentic programming, abstracting away the complexities of lower level programming frameworks and enabling compilation to multiple targets. We review its main components:

The Agentic Graphical Language: The language has been defined by a meta-model (cf. Fig. 2). Agentic workflows are modeled by a Graph. Nodes of different kinds, represented by their shape, perform different types of actions, which are executed in order following the graph's edges. The workflow has a set of variables that can be accessed from any node, called the flow's State.

Agent nodes represent a call to an agent. In addition to the prompt, an agent can have a custom output structure, guardrails, memory, sources of knowledge, and tools. In particular, guardrails rely on state variables and user input to validate the agent output, and can trigger re-execution with custom feedback. Dynamic agent collaboration is supported through handoffs. Router nodes have several outgoing edges and allow for branching based on conditions that use the workflow state, agent outputs and user input. Action nodes allow for modifications to the worflow state using state variables, agent outputs, and user input.

This language aims to be expressive enough to represent a wide range of use cases, yet simple enough to be used by non-technical users through an appropriate modeling environment.

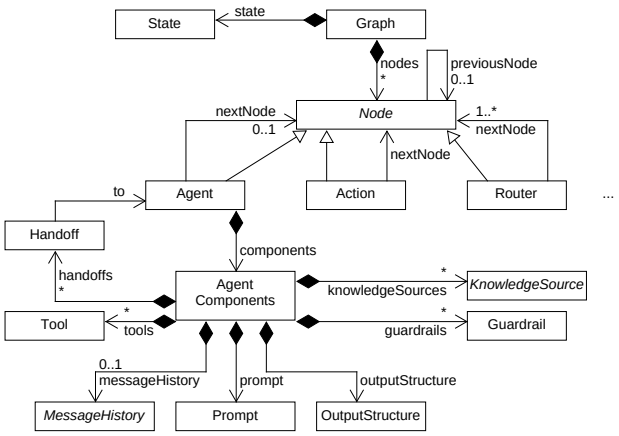

Fig. 2. Meta-model of our agentic language (excerpt).

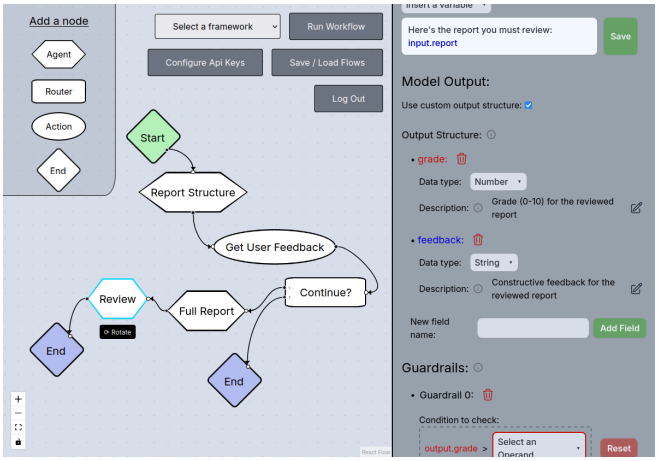

Fig. 3. Screenshot of the modeling environment.

Modeling Environment: A modeling environment allows users to define agentic workflows by adding and connecting nodes, and configuring their properties. The designed workflows can then be run and interacted with through a curated terminal, without the user needing to interact with the underlying code.

Code Generation: The graph models are automatically transformed into code and executed. The user can choose between different target agentic programming frameworks to generate the code in.

Model Validation: A validation module checks the correctness of the models defined by the user before the code generation phase. Validators perform both syntactic and semantic checks.

III. Tool Support

Our modeling environment is a web-based, low-code [3] platform that allows users to design workflows using the described graphical language. As illustrated in Figure 3, the environment provides a user-friendly interface and tooltips to guide users with little experience with AI agents. The platform is completely code free, and even the more complex aspects of the model, such as agent guardrail conditions, can be specified through specialized interfaces without writing code. Agent tools can be selected from a set of pre-made options, and the architecture allows for new tools to be added easily.

Once the model is complete, users can choose to generate code in different agentic programming frameworks (at the time of writing, CrewAI and PydanticAI). Through the built-in execution terminal, users can see the workflow being executed in real time and provide any inputs required by the nodes.

The platform also allows users to store and manage their models using an authentication system. API Keys needed for the execution of the workflows can either be stored in the backend or provided in the code generation phase.

Each of the supported agentic programming frameworks has its own associated model validator and code generator. The validation and code generation modules can easily be extended to support new frameworks. Our graphical language aims to be universal in the sense that any model that is valid for one framework will be valid for all others. Even so, the framework-specific validators serve to inform the user of patterns in their model that would lead to unidiomatic or inefficient code for the selected framework. For instance, including workflow state variables in an agent's system prompts will trigger a warning when generating code for CrewAI, as this framework requires re-initializing the agent for every call to achieve the expected behavior.

All code is available at https://github.com/GAWD-Framework/GAWD.

IV. Conclusions and Future Work

We have presented an MDE framework for agentic AI. Its aim is to: (i) lower the entry barrier to the definition of agentic workflows; and (ii) decouple specification from implementation, so that different agentic libraries can be selected to compile a workflow model.

In the future, we plan to add support for further agentic libraries, and perform usability studies with real users. In our current implementation, user experience could be improved by adding to the graph editor pre-built constructs for patterns such as loops or error handling, and for common agent architectures. Agent capabilities could also be expanded and refined. In particular, the current implementations for agent memory and knowledge sources are limited. The modeling environment could benefit from new features such as the possibility of deploying individual agentic workflows as independent web services, or the analysis of models to detect the most appropriate frameworks to generate code into.

References

[1] A. Ait, J. L. C. Izquierdo, and J. Cabot, "Towards modeling human-agentic collaborative workflows: A BPMN extension," in *SEAA*, ser. LNCS. Springer, 2025, pp. 367–382.

[2] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice, Second Edition*, ser. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2017.

[3] D. D. Ruscio, D. S. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-code development and model-driven engineering: Two sides of the same coin?" *Softw. Syst. Model.*, vol. 21, pp. 437–446, 2022.