Towards Modeling Human-Agentic Collaborative Workflows: A BPMN Extension

Adem Ait
University of Luxembourg
Esch-sur-Alzette, Luxembourg
adem.ait@uni.lu

Javier Luis Cánovas Izquierdo
IN3 - UOC
Barcelona, Spain
jcanovasi@uoc.edu

Jordi Cabot
University of Luxembourg
Luxembourg Institute of Science and Technology
Esch-sur-Alzette, Luxembourg
jordi.cabot@list.lu

***Abstract*—Large Language Models (LLMs) have facilitated the definition of autonomous intelligent agents. Such agents have already demonstrated their potential in solving complex tasks in different domains. And they can further increase their performance when collaborating with other agents in a multi-agent system. However, the orchestration and coordination of these agents is still challenging, especially when they need to interact with humans as part of human-agentic collaborative workflows. These kinds of workflows need to be precisely specified so that it is clear whose responsible for each task, what strategies agents can follow to complete individual tasks or how decisions will be taken when different alternatives are proposed, among others. Current business process modeling languages fall short when it comes to specifying these new mixed collaborative scenarios. In this exploratory paper, we extend a well-known process modeling language (i.e., BPMN) to enable the definition of this new type of workflow. Our extension covers both the formalization of the new modeling concepts required and the proposal of a BPMN-like graphical notation to facilitate the definition of these workflows. Our extension has been implemented and is available as an open-source human-agentic workflow modeling editor on GitHub.**

I. Introduction

In our current information-rich society, the integration of agents, especially agents powered by Large Language Models (LLMs), is becoming more and more important to quickly perform many tasks [17]. Agents can interact with the environment, make their own decisions, and learn from the received feedback. Moreover, often, agents do not work in isolation but as part of Multi-Agent Systems (MAS) where agents cooperate (or compete) to achieve a common goal [6]. This collaborative process is known as an agentic system. These systems have already demonstrated their superior performance against single-agent solutions [6].

While agentic systems are performant in many tasks, complex scenarios require the participation of humans [16]. Therefore, there is a need to precisely define this collaboration and how each participant interacts with each other. Unfortunately, we argue that current process modeling languages, such as BPMN, lack the modeling constructs to specify this collaboration between humans and agentic systems as new primitives to define the confidence of the agents, the strategies they can use to perform a task, or the process to reach a decision. Furthermore, frameworks targeting the implementation of agentic workflows, such as LangGraph[1], minimize the participation of humans in the process and therefore are not expressive enough to model the human-agent interaction beyond very simple cases.

The goal of this paper is to enable the precise definition of human-agentic workflows. To this aim, we study how the Business Process Model and Notation (BPMN), one of the most well-known modeling languages for workflows, could be used to represent this new type of workflow, and then, based on the identified limitations, we propose a BPMN extension to enable their definition in BPMN. Note that our approach can serve as a blueprint for extending other workflow languages, as the conceptual elements we identify are largely notation-independent. This extension has been implemented in an open source modeling tool available on GitHub.

The rest of the paper is structured as follows. Section II provides the background and a running example. Section III shows the mapping of agentic concepts to BPMN, while Section IV presents the extension to the BPMN to overcome the limitations found. Section V and VI provide the extension definition and the proof of concept, respectively. Section VII presents the related work. Finally, Section VIII concludes the paper and presents the roadmap.

II. Background & Running Example

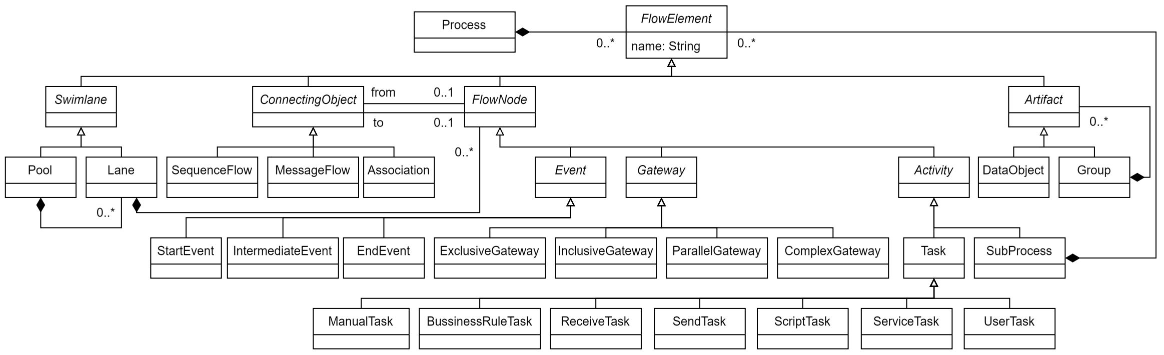
In this section, we briefly describe BPMN, the language we aim to extend; and present the main concepts of agentic systems. We end the section with a running example.

A. BPMN

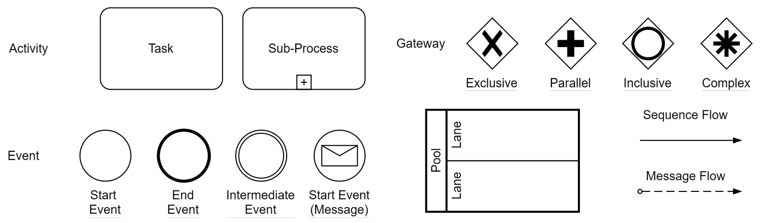
Business Process Management (BPM) studies how work is performed in an organization to ensure consistent outcomes and to take advantage of improvement opportunities [3]. BPM is about managing entire chains of events, activities, and decisions (called business processes) that ultimately add value to the organization and its customers. Thus, the formal representation of such process facilitates the orchestration, monitoring, and improvement of an organization's workflow.

BPMN [11] has become the de-facto standard for business processes diagrams. It is defined by the Object Management Group (OMG) and specified as ISO standard (ISO/IEC 19510:2013). BPMN provides a metamodel and a notation to define and visualize business process models. It provides an extension mechanism to allow modeling domain-specific

[1]https://www.langchain.com/langgraph



(a)



(b)

Fig. 1: Simplified (a) metamodel of BPMN and the corresponding (b) notation.

elements not included in the specification. In particular, BPMN 2.0 extension mechanism enables the approach of extension by addition, which consists of attaching new domain-specific elements to the predefined elements of the language.

Figure 1a shows a simplified metamodel of BPMN. The different *FlowElements* compose the *Process*. The *Swimlane* is used to organize the *Process*, which can be either *Pool*, or *Lane*. *Pools* can be formed by several *Lanes*. *FlowObject* can be: (1) *Events*, which are used as triggers (e.g., timers); (2) *Activities*, the work unit, which can be atomic (i.e., *Tasks*) and non-atomic (i.e., *Sub-Process*); and (3) *Gateways* for controlling the flow. *ConnectingObject* is used to specify the order of *FlowElements* which is divided into: (1) *SequenceFlow*, to establish the flow within pools, which can also be conditional (i.e., the flow will continue if a condition is fulfilled); (2) *MessageFlow*, to communicate between pools; and (3) *Association*, used to connect user-defined text (an *Annotation*) with *Flow Nodes*. *Artifact* describes the *DataObject* shared in the process and *Group*, a visual mechanism to group elements of a diagram informally. Figure 1b shows the notation of the main elements described.

B. Agentic systems

LLMs excel at multiple tasks [15], but creating specialized LLMs instances to target specific tasks has shown promising results. These specialized LLMs are typically known as agents [17]. LLM-based agents use the potential of LLMs, facilitating sophisticated interactions, decision-making, tool-use capabilities, and in-context learning through memory [6].

However, LLMs exhibit non-deterministic behavior. To alleviate this, reflection strategies have been proposed to refine their answers [13], namely: (1) **self-reflection**, where agents generate feedback on the plan and reasoning process to refine themselves; (2) **cross-reflection**, where the feedback is provided by other agents; and (3) **human-reflection**, where humans provide the feedback.

To tackle more complex problems, a common approach is to increase the number of agents, forming what is known as a LLM-based MAS, or, more recently, rebranded as agentic systems, to emphasize the cooperation of the agents in the MAS which has already been proven to outperform single-agent solutions [6]. A key aspect in agentic systems is how agents work collaboratively to solve tasks, leveraging their interactions with the environment or other agents. The works by Guo et al. [6] and Liu et al. [9] introduce the first attempts at characterizing how agentic systems work.

Agentic systems can adhere to different types of cooperation patterns. The core ones are: (1) **voting-based**, where agents independently propose alternative solutions and reach consensus by voting; (2) **role-based**, where each agent, or group of agents, has assigned a role, thus making the decision according to such roles; and (3) **debate-based**, where agents submit and receive feedback to adjust the thoughts until a consensus is reached. Furthermore, an additional scenario is **competition-based** collaboration, where agents, instead of cooperating, compete and the fastest (or the most reliable output) is selected.

All these agentic aspects will need to be part of the process modeling language if we want to model in detail human-agentic collaborations as, for instance, in the scenario we use as running example.

TABLE I: Partial mapping of human-agentic workflow concepts to BPMN elements.

HUMAN-AGENTIC WORKFLOW CONCEPT	VARIETY	BPMN ELEMENT
Agent	Single-agent	Pool or Lane
	Multi-agent	Multi-instance pool
Reflection	Self-reflection	Extra loop activity
	Cross-reflection	Loop with gateways and activities
	Human-reflection	Loop with gateways and activities
Agent Collaboration	All	Group or message flow
Merging Collaboration Efforts	All	Complex gateway or message flow

C. Running Example

To illustrate our proposal, we will use a running example based on a simple resolution process for bug reports in a software project. The example process comprises five participants, two humans and three agents.

The humans are a user, who reports the bug; and a maintainer, who reviews the final change proposal and resolves the bug. The agents are responsible for solving the bug by implementing change proposals and deciding together the best option. There are three agents, one is used as a reviewer and the other two are specialized coding agents. Each agent comes with a level of uncertainty regarding the quality of all its actions. This value could be derived from the underlying LLM and/or the agent setup.

Once the reviewer agent validates the bug definition, using a reflective strategy to double-check on the first assessment, the two coding agents are in charge of proposing a solution to the bug. They both work independently in parallel and, following a role-based cooperation strategy, is the agent with the reviewer role who has the final decision, also considering the uncertainty of each coding agent in case of discrepancies.

III. USING BPMN TO MODEL HUMAN-AGENTIC WORKFLOWS

In this section, we study the current support of BPMN to model human-agentic workflows. The characteristics of agentic systems we aim at covering are the ones that concern the reflection and cooperation, while also addressing their non-deterministic behavior. Table I shows a possible mapping of BPMN elements to model human-agentic workflows. Note that this mapping is partial, as the current support of BPMN for this type of workflow is very limited, as we show in this section.

Agents could be represented as pools or lanes, depending on the process context. When agents are part of a project process, lanes could be used, but if they represent external contributors, pools could be used instead. Sets of agents could be represented as multi-instance pools. Information about the non-deterministic behavior of the agents could only be represented via text annotations.

Reflection strategies cannot be represented in standard BPMN. We could simulate them using additional activities and gateways. Self-reflection could be represented as a loop activity that follows the activity to be refined. For cross- and human-reflection, the feedback loop could be represented as several gateways and activities to keep refining the answer until the desired output is obtained.

To model the collaboration between agents, we consider two scenarios, depending on whether agents are located in lanes of the same or different pool/s (i.e., collaboration diagram). When located in the same pool, gateways could be used to route the flow towards multiple lanes, and groups could be utilized to set the collaboration strategy. If located in different pools, message flows could be used, but text annotations should be added to describe the strategy. For merging collaborations, in the first scenario we could use a complex gateway and specify the desired merging strategy as an annotation, while in the second scenario we could specify the strategy as an annotation in the incoming message flow. Note the need to use text annotations to describe the behavior. BPMN also offers additional collaboration mechanisms such as choreographies (formalized coordination of interactions between participants), but these still require supplementary descriptions for agent-specific behaviors since their focus is on the exchange of information rather than the orchestration of their work.

Figure 2 shows the BPMN model for the running example. As can be seen, we used five lanes to represent the different actors. The *User* and *Maintainer* are the humans reporting the bug and validating the final proposed solution, respectively. The *AgentReviewer* is the agent designated to solve the bug, while *AgentCoder* and *AgentCoder2* represent specialized coding agents that will help with coding tasks. To apply self-reflection on the *Check bug validity* activity, we define an extra loop activity (see *Provide feedback on answer*) with a loop condition defined in natural language to not stop until a refined answer is provided. Once we obtain a reliable answer, we proceed to fix the bug if the report is valid. To illustrate the cooperation, we use a group (see *cooperation*). Then, as a merging gateway, we use a complex gateway to define our own condition through an annotation, which is decided by *AgentReviewer*. Thus, only the selected solution will be the one delivered to the next activity.

Representing human-agentic workflows in BPMN is challenging, as the standard BPMN does not provide specific elements to model agents or their interactions. In the running example, we cannot set uncertainty to determine the agents' reliability or identify lanes or pools as agents. We can partially represent the collaboration and reflection strategies. We rely on natural language to describe the collaboration and reflection strategies, which can lead to ambiguity and misinterpretation. Furthermore, the intricate process of representing reflection can hinder the readability and understandability of the model.

BPMN was chosen for several key reasons: (1) widespread adoption, BPMN is the *de facto* standard for business process modeling with broad industrial and academic support; (2) expressivity, BPMN already provides comprehensive constructs for modeling workflows involving human participants; (3) extensibility, BPMN offers an extension mechanism that enables domain-specific additions while maintaining compliance with the standard; (4) execution capabilities, BPMN models can be directly executed by business process engines, making the

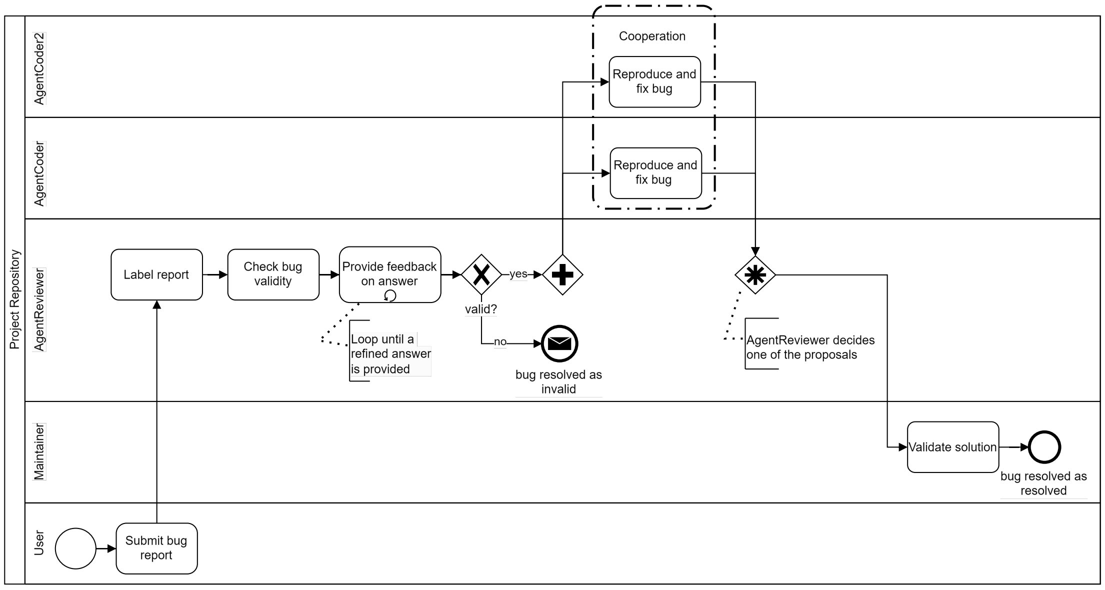


Fig. 2: Running example with standard BPMN.

transition from conceptual models to implementation easier; and (5) integration with existing processes, many organizations already use BPMN, making it easier to incorporate agent-based elements into existing business processes.

Therefore, despite the identified limitations, we believe that BPMN still provides a good starting point compared to alternative modeling languages. While languages like UML Activity Diagrams or YAWL could potentially be extended in similar ways, BPMN's widespread usage in industries means that practitioners are already familiar with its core concepts, reducing the learning curve for our extended notation.

In the next section, we propose an extension to BPMN that introduces formalized syntax for defining collaboration and reflection strategies, rules for merging collaborative outcomes, and agents' profiling (i.e., role and trustworthiness).

IV. Extending BPMN to Model Human-Agentic Workflows

To address the limitations identified for modeling human-agentic workflow concepts in BPMN, we propose an extension to the BPMN specification. In particular, we propose to extend the following elements: lane, task, message flow, and parallel and inclusive gateways. In the following, we describe each extension point, showing the domain models of the extended BPMN elements. When illustrating the extension, classes of the BPMN 2.0 metamodel will be highlighted in gray. Note that, without loss of generality, we describe our extension as a BPMN extension, but a similar approach could have been used to extend other process modeling languages, as they offer a largely overlapping set of concepts.

A. Agent Profiling

Agent profiling requires modeling (1) the role of the agents in collaboration scenarios, as it is how an LLM is initialized as an agent; and (2) the reliability of the agent's output,

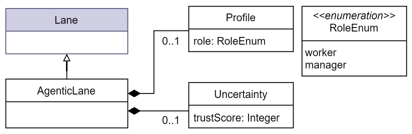


Fig. 3: Domain model of the *AgenticLane*.

or trust score, to address the non-deterministic nature of agents behavior. As shown in Section III, in standard BPMN, representing agents as lanes or pools overlooks these aspects: (1) when having multiple lanes, each representing specialized agents, only lane names can be specified; and (2) there is no way to specify uncertainty for agent's output.

Figure 3 shows the domain model of the extension to address agent profiling. We differentiate between regular participants and *agentic* participants (see *Lane* and *AgenticLane*). The *Pool* element is the graphical representation of a participant. However, we decided to extend the *Lane* class rather than the *Pool* class to allow setting a profile for each agent within a pool. This way, a group of agents can be represented as a pool, where each agent can have different trust scores since they would be represented as lanes of the pool.

To represent the role and the trust score, we define the attributes in the *Profile* and *Uncertainty* classes, respectively. The role value distinguishes between manager and workers, but the corresponding enumeration could be extended to fulfill further roles (e.g., coder). When set to manager, the agent represented by the lane is the one in charge of selecting the valid output in role-based or debate-based cooperation. The trust score parameter is a percentage value (i.e., 0-100) of the trustworthiness of a particular agent.

B. Agent Reflection

Reflection is key in agentic systems, enabling agents to evaluate their actions and adapt their behavior accordingly.

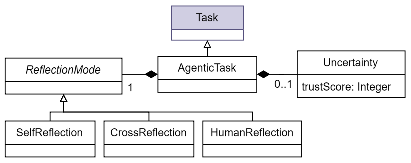


Fig. 4: Domain model of the *AgenticTask*.

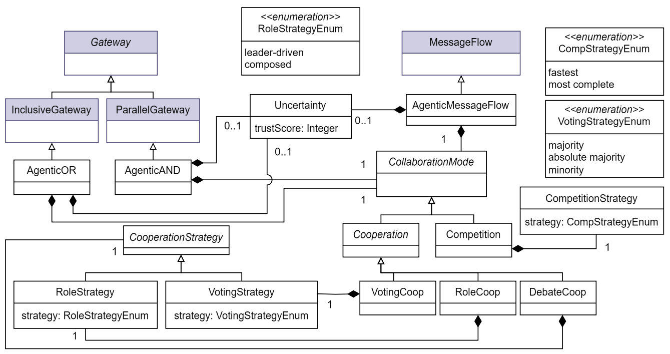


Fig. 5: Domain model of the *AgenticOR*, *AgenticAND* and *AgenticMessageFlow*.

Although standard BPMN can model loops and decision points, it lacks the constructs to formally define and enforce self-reflection, cross-reflection, or human-reflection processes.

Figure 4 shows the domain model of the extension to address agent reflection. We extend the *Task* BPMN element, defining the *AgenticTask*, which can be associated to one of the three reflection strategies. We model the identified reflection strategies from the literature into three classes (see *SelfReflection*, *CrossReflection*, and *HumanReflection* classes), which are subclasses of the *ReflectionMode* class. Depending on the degree of reliability and the resources available, one could define the reflection as human-reflection, to avoid undesired outputs. However, if one wants a completely automated task, but also ensure some degree of reliability, one could use self-reflection or cross-reflection strategies, where the latter would be more expensive, since it requires instances of other agents, but it might provide better results.

Furthermore, tasks can have attached a trustworthiness score (see *Uncertainty*), used to represent the reliability of the task output. This trust score can be further used in the workflow to decide the next steps.

C. Agent Collaboration

In standard BPMN, collaboration between agents could be represented as groups, which allows for a basic depiction of collaborative efforts, but falls short in specifying the cooperation (and merging) strategies employed by the agents.

Figure 5 shows the domain model of the collaboration types along with their merging strategies, and how they are related to the extended *Gateway* and *MessageFlow* classes. The collaboration will be enclosed between diverging and merging gateways. The former specifies the collaboration strategy, while the latter indicates the merging strategy.

We model the *CollaborationMode* as the root of the hierarchy of collaboration modes, which can be either cooperation or competition, the former being the root of three types of cooperation (i.e., voting, role and debate). We extend both the inclusive and parallel gateways (see *AgenticOR* and *AgenticAND*, respectively), which also set the collaboration strategy (see *CollaborationMode* association). The inclusive gateway allows specifying a condition so that only specific agents are activated, while the parallel gateway diverges the flow to all its outputs, which is desired when all the participants from the diverged flow are required to collaborate. Since agentic gateways are intended to represent collaboration, we only extend these two gateways. We do not consider the exclusive gateway, since the flow is allowed only to one output.

The merging strategies set how to decide the solution to select in collaboration scenarios. When competing, the *CompetitionStrategy* class indicates the merging strategy. Regarding the cooperation strategies, the merging strategies are represented in the *CooperationStrategy* class. *VotingStrategy* is used when the decision is made through voting, while *RoleStrategy* is used when the merging is taken by a manager role or by the inherent roles of the agents (e.g., specialized agents proposing each part of the output). While voting-based and role-based cooperation have explicit merging strategies because of their description, the debate-based cooperation might leverage from one of the two strategies.

The gateways allow the collaboration within pools, but if the set of agents is represented as another pool, message flows should be used (see *MessageFlow*). We extend the message flow, rather than choreography tasks, since it allows for an elementary depiction of the collaboration. Like done for gateways, the outgoing message flow sets the collaboration strategy, while the incoming message flow sets the desired merging strategy.

Finally, the agentic gateways and message flow are also associated with a trust score. In gateways, it can be used as a decision point, while in message flows can be used to understand the reliability of the received token.

D. BPMN Notation Extension

According to the BPMN specification, extensions notation must not alter the notation of its elements, and must be as close as possible, in terms of look and feel, to it [11]. Our extension introduces four new elements (see Table II) with a graphical representation close the BPMN element being extended.

When choosing these elements, we also aimed to stick to the principles for effective visual notations by Moody [10]. For instance, to be compliant with the semiotic clarity principle, we maintain a 1:1 correspondence between semantic constructs and graphical symbols, with each agentic concept (lane, task, gateway, message flow) having its own distinct visual representation (see Table II).

To identify the extended elements, we use a marker representing an agent. Our agent marker follows the perceptual discriminability principle, since it clearly distinguishes agentic elements from standard BPMN elements, while maintaining the basic shape of the original BPMN symbols to preserve familiarity. Furthermore, the agent icon intuitively suggests intelligence and automation being semantically immediate, following the semantic transparency principle.

On the other hand, we minimize the introduction of new symbols by using a consistent agent marker with letter modifiers rather than creating entirely new shapes, following the graph economy and dual coding principle (i.e., use both graphical elements and textual annotations to enhance cognition).

The agentic lane is identified with the agent marker centered below the name for vertical lanes, or centered at the right of the name for horizontal lanes. The trust score is set between the name of the lane and the agent marker, following the position of the text. The role is set as a letter below the marker, where "w" stands for worker and "m" for manager.

The agentic task is represented with the agent marker in the top-left corner, as it is done with specific tasks (e.g., *ManualTask*). The reflection and collaboration strategies are indicated with a marker at the bottom of the shape. The reflection strategy is set as a letter inside the marker, where "s" stands for self-reflection, "c" for cross-reflection, and "h" for human-reflection. Note that the notation shown in Table II the "x" is used as a placeholder.

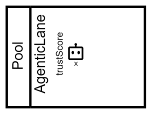
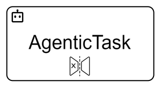
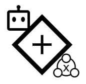
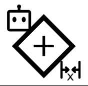
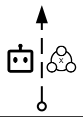
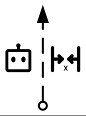
The agentic gateways are represented with the agent marker at the top-left side, without disturbing the shape of the gateway. The diverging gateway contains the collaboration marker in the bottom-right corner. This way the visual distance between symbols is greater [10]. The same principle has been applied for the remaining notations. The collaboration strategy is set as a letter below the marker, where "c" stands for competition, "d" for debate cooperation, "r" for role cooperation, and "v" for voting cooperation. The merging gateway must contain the merging marker in the bottom-right corner. The merging strategy is represented with two sets of letters below the marker, the first as the strategy class and the second as the merging strategy type. Thus, for voting strategies are "v-ma" for majority, "v-a" for absolute majority, and "v-mi" for minority; for role-based strategies are "r-l" for leader-driven and "r-c" for composed; while for competition strategies are "c-f" for fastest, and "c-mc" for most complete.

The agentic message flow is represented with the agent marker centered on the left side, if the message flow is vertical, or centered above the message flow, if it is horizontal. The collaboration or merging strategy is set with a marker on the right side, if the message flow is vertical, or centered below the message flow, if it is horizontal. As with the agentic gateway, the outgoing message flow must contain the collaboration strategy, while the incoming message flow must contain the merging strategy.

E. Using our Extension to Model the Running Example

Figure 6 shows the running example using our extension. The different agents are denoted with *agentic lanes*, and have a trust score attached (see three top lanes). The second task (see *Check bug validity*) is an *agentic task* that applies self-reflection to the output.

TABLE II: Graphical notation of the extended elements.

EXTENSION ELEMENT	NOTATION
Agentic lane	
Agentic task	
Diverging agentic gateway	
Merging agentic gateway	
Outgoing agentic message flow	
Incoming agentic message flow	

To represent the collaboration between agents, we use the *agentic gateway*, since agents are represented as lanes of the same pool. The collaboration strategy applied is *role cooperation*, as denoted by the notation. After stating the collaboration strategy, the flow is divided towards the agents that collaborate. All flows from this collaboration are merged into an *agentic gateway* following the *leader-driven*, strategy as denoted by the notation. The remaining elements are compliant to standard BPMN.

V. Our Extension as a Standard BPMN Extension Definition

BPMN provides an extension mechanism to allow modeling domain-specific elements not included in the specification.

Nevertheless, there is a lack of methodological guides to develop and publish specific extensions. Stroppi et al. [14] define a method (called BPMN+X) to transform a domain model into a BPMN-compliant extension by using UML profiles, which is the typical approach used in the UML world to define lightweight extensions[2] to the language. Given a BPMN extension defined as a profile, the BPMN+X method uses mapping rules and automated model transformations to generate an XML Schema Extension Definition Document conforming to the official BPMN extension mechanism.

We follow this approach to redefine our extended BPMN metamodel as a BPMN extension. To this purpose, Figure 7 shows the metamodel from Figures 3, 4 and 5 defined as a profile.

[2]The term lightweight extension is used to denote language extensions that are compatible with language semantics and that can be expressed using the own language extension mechanisms, enabling the direct use of the extension in any tool that supports the language. This is in contrast to heavyweight extensions that offer more complex extensions that enable richer semantics for the extension but require dedicated tooling support

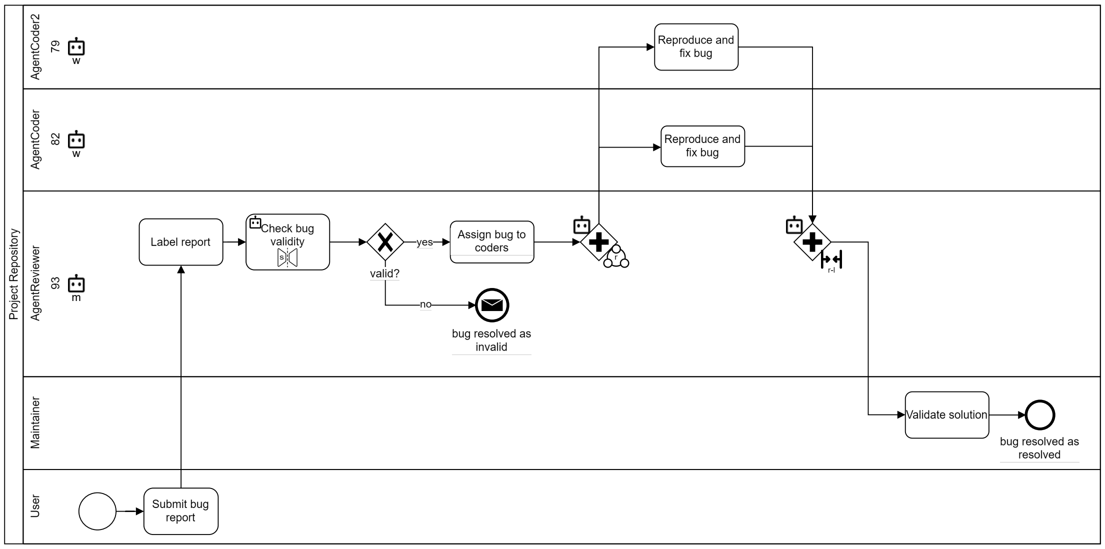


Fig. 6: Running example with our extension notation.

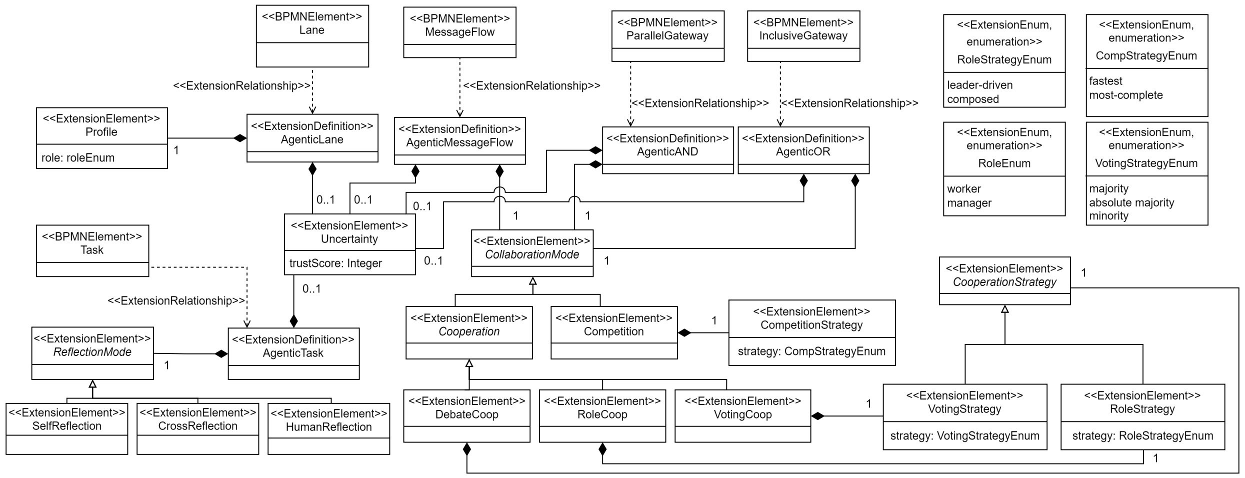

<<BPMNElement>> Lane
<<BPMNElement>> MessageFlow
<<BPMNElement>> ParallelGateway
<<BPMNElement>> InclusiveGateway
<<ExtensionEnum, enumeration>> RoleStrategyEnum
leader-driven
composed
<<ExtensionEnum, enumeration>> CompStrategyEnum
fastest
most-complete
<<ExtensionRelationship>>
<<ExtensionElement>> Profile
role: roleEnum
<<ExtensionDefinition>> AgenticLane
<<ExtensionDefinition>> AgenticMessageFlow
<<ExtensionDefinition>> AgenticAND
<<ExtensionDefinition>> AgenticOR
<<ExtensionEnum, enumeration>> RoleEnum
worker
manager
<<ExtensionEnum, enumeration>> VotingStrategyEnum
majority
absolute majority
minority
<<ExtensionElement>> Uncertainty
trustScore: Integer
<<ExtensionElement>> CollaborationMode
<<BPMNElement>> Task
<<ExtensionElement>> CooperationStrategy
<<ExtensionElement>> Cooperation
<<ExtensionElement>> Competition
<<ExtensionElement>> CompetitionStrategy
strategy: CompStrategyEnum
<<ExtensionElement>> ReflectionMode
<<ExtensionDefinition>> AgenticTask
<<ExtensionElement>> SelfReflection
<<ExtensionElement>> CrossReflection
<<ExtensionElement>> HumanReflection
<<ExtensionElement>> DebateCoop
<<ExtensionElement>> RoleCoop
<<ExtensionElement>> VotingCoop
<<ExtensionElement>> VotingStrategy
strategy: VotingStrategyEnum
<<ExtensionElement>> RoleStrategy
strategy: RoleStrategyEnum
1
0..1

Fig. 7: Extension model.

VI. Proof of Concept

As a proof-of-concept of the proposal, we have implemented a modeling editor that enables any developer to use our extended BPMN language and notation.

The extension has been implemented using Sirius [3], an Eclipse project which allows you to easily create your own graphical modeling workbench by leveraging Eclipse Modeling technologies such as EMF and GMF. Aconite [12], a tool that helps produce Sirius-based graphical notations, has been used to automatically generate the Sirius-based implementation. Figure 8 shows a screenshot.

The tool repository[4] includes the files that use the Aconite annotations and the examples illustrated in this paper. The tool includes a main view of the diagram and a palette to allow the user to drag and drop the elements into the diagram. The palette contains the basic representation of BPMN, plus the extension elements proposed in this paper. Furthermore, the repository also includes several examples from the literature to illustrate the use of the extension.

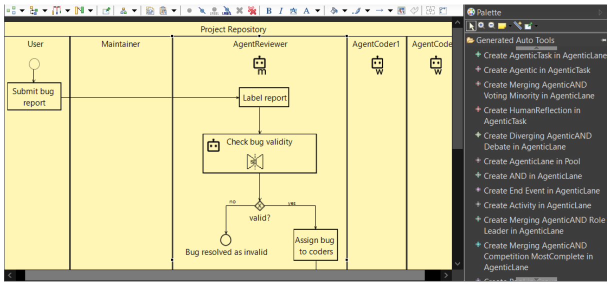

Project Repository
User
Maintainer
AgentReviewer
AgentCoder1
AgentCode
Submit bug report
Label report
Check bug validity
no
yes
valid?
Bug resolved as invalid
Assign bug to coders
Palette
Generated Auto Tools
Create AgenticTask in AgenticLane
Create Agentic in AgenticTask
Create Merging AgenticAND Voting Minority in AgenticLane
Create HumanReflection in AgenticTask
Create Diverging AgenticAND Debate in AgenticLane
Create AgenticLane in Pool
Create AND in AgenticLane
Create End Event in AgenticLane
Create Activity in AgenticLane
Create Merging AgenticAND Role Leader in AgenticLane
Create Merging AgenticAND Competition MostComplete in AgenticLane

Fig. 8: Platform-independent implementation.

[3]https://eclipse.dev/sirius/overview.html
[4]https://github.com/BESSER-PEARL/agentic-bpmn

VII. Related Work

We classify the related work into two groups: proposals that try to model agents in general process modeling languages and agent-specific tools that may include a language to orchestrate agentic workflows.

In the first group we have Küster et al. [8] that combine agent-oriented software engineering with business process design. Nevertheless, they do not extend BPMN and therefore suffer from the limitations stated in Section III. Endert et al. [4] propose a mapping of BDI agents to business processes. However, while this proposal provides ways of mapping agent-specific concepts following the BDI paradigm to BPMN, they do not cover advanced aspects of the current generation of agents such as reflection strategies.

Other BPMN approaches partially cover uncertainty concerns. Ceballos et al. [2] propose a BPMN Business Process Diagram (BPD) normal form based on Activity Theory [5] that can be used for representing the dynamics of a collective human activity from the perspective of a subject. Herbert and Sharp [7] proposed a BPMN extension introducing the uncertainty in sequence flows. The extension includes probabilistic flows and rewards associated to the execution of tasks. Note that the uncertainty is determined in the sequence flows rather than in the participant or activity BPMN element. Our proposal introduces uncertainty at the participant level, then propagating to all the other elements. Both types of uncertainty could be combined.

Regarding specific tools to create agentic systems, a couple include the graphical modeling of the process. Two representative examples are LangGraph[5] that allows modeling the action flow of an agent (or a set of agents) using cyclic graphs and Flowise[6], an open-source low-code tool for orchestrating LLM agents. In both cases, the workflows are purely focused on agent collaborations. Humans can only trigger the actions but are not supposed to collaborate with the agent to accomplish them. In contrast, our extension considers humans as active participants and enables representing complex human-agent interactions.

VIII. Conclusions and Further Work

We have presented a BPMN extension to model human-agentic workflows. Our extension enables the modeling of complex collaboration patterns between humans and agents, including specifying the agents' reflection strategies.

As further work, we plan to provide a sublanguage to specify in detail the governance and decision-making strategies as part of the merging nodes for agents' collaboration efforts (e.g., should decisions be based on consensus? voting strategies...). We will also propose an uncertainty propagation mechanism that, given the overall uncertainty of the agents and their confidence in the result of a given task, assigns an overall uncertainty of the task, which should then be propagated also to the consecutive tasks.

[5]https://www.langchain.com/langgraph

[6]https://github.com/FlowiseAI/Flowise

Additionally, we plan to work on code generators aimed at producing an executable representation of the model, including the governance aspects and the uncertainty propagation mechanisms to make operational the modeled workflows. The generators could target, potentially, a combination of BPEL engines and agentic platforms, where the BPEL engine would take care of the global orchestration and delegate to the agentic platform the task execution.

Finally, we plan to empirically evaluate our extension and explore its application in industrial use cases. In particular, to ensure the cognitive effectiveness of our notation, we plan to conduct a metric-based assessment based on the alignment with Moody's principles for visual notations [10], following our previous work on collaborative modeling notation evaluation [1].

References

[1] Brambilla, M., Cabot, J., Izquierdo, J.L.C., Mauri, A.: Better call the crowd: using crowdsourcing to shape the notation of domain-specific languages. In: Int. Conf. on Software Language Engineering. pp. 129–138 (2017)

[2] Ceballos, H.G., Flores-Solorio, V., García-Vázquez, J.: A probabilistic BPMN normal form to model and advise human activities. In: Int. Workshop on Engineering Multi-Agent Systems. vol. 9318, pp. 51–69 (2015)

[3] Dumas, M., Rosa, M.L., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer (2013)

[4] Endert, H., Küster, T., Hirsch, B., Albayrak, S.: Mapping BPMN to agents: An analysis. Agents, Web-Services, and Ontologies Integrated Methodologies pp. 43–58 (2007)

[5] Engeström, Y., Miettinen, R., Punamäki-Gitai, R.L.: Perspectives on activity theory. Cambridge University Press (1999)

[6] Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N.V., Wiest, O., Zhang, X.: Large language model based multi-agents: A survey of progress and challenges. In: Int. Joint Conf. on Artificial Intelligence. pp. 8048–8057 (2024)

[7] Herbert, L., Sharp, R.: Precise quantitative analysis of probabilistic business process model and notation workflows. J. Comput. Inf. Sci. Eng. **13**(1) (2013)

[8] Küster, T., Lützenberger, M., Heßler, A., Hirsch, B.: Integrating process modelling into multi-agent system engineering. Multiagent Grid Syst. **8**(1), 105–124 (2012)

[9] Liu, Y., Lo, S.K., Lu, Q., Zhu, L., Zhao, D., Xu, X., Harrer, S., Whittle, J.: Agent design pattern catalogue: A collection of architectural patterns for foundation model based agents. J. Syst. Softw. **220**, 112278 (2025)

[10] Moody, D.: The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on software engineering **35**(6), 756–779 (2009)

[11] OMG: Business process model and notation (bpmn) 2.0.2 specification (Jan 2014), https://www.omg.org/spec/BPMN, accessed on July, 2024

[12] Richardson, N., Kolovos, D., Garcia-Dominguez, A.: Aconite: Towards generating sirius-based graphical editors from annotated metamodels. In: Int. Conf. on Software Language Engineering. p. 16–28 (2024)

[13] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: language agents with verbal reinforcement learning. In: Conf. on Neural Information Processing Systems (2023)

[14] Stroppi, L.J.R., Chiotti, O., Villarreal, P.D.: Extending BPMN 2.0: Method and tool support. In: Int. Workshop of Business Process Model and Notation. vol. 95, pp. 59–73 (2011)

[15] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., Chi, E.H., Hashimoto, T., Vinyals, O., Liang, P., Dean, J., Fedus, W.: Emergent abilities of large language models. Trans. Mach. Learn. Res. **2022** (2022)

[16] Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Li, B., Jiang, L., Zhang, X., Wang, C.: Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. CoRR **abs/2308.08155** (2023)

[17] Xi, Z., Chen, W., Guo, X., He, W., Ding, Y., Hong, B., Zhang, M., Wang, J., Jin, S., Zhou, E., et al.: The rise and potential of large language model based agents: A survey. Sci. China Inf. Sci. **68**(2), 121101 (2025)