AI Multi-Agent Interoperability Extension for Managing Multiparty Conversations

Diego Gosmar[1,5,*,†], Deborah A. Dahl[2,5,†], Emmett Coin[3,5,†] and David Attwater[4,5,†]

[1]XCALLY, Torino, TO 10100, Italy
[2]Conversational Technologies, Plymouth Meeting, Pennsylvania, USA
[3]ejTalk, Bellville, Michigan, USA
[4]Talkmap, Southport, Merseyside, United Kingdom
[5]Linux Foundation AI & Data, Open Voice Interoperability Initiative

Abstract

This paper presents a novel extension to the existing Multi-Agent Interoperability specifications of the Open Voice Interoperability Initiative (originally also known as OVON from the Open Voice Network), which already enables AI agents developed with different technologies to communicate seamlessly using a universal, natural language-based API or NLP-based standard APIs. Focusing on the management of multiparty AI conversations, this work introduces new concepts such as the Floor Manager, Convener Agent, Multi-Conversant Support, and mechanisms for handling Interruptions and Uninvited Agents. These advancements are crucial for ensuring smooth, efficient, and secure interactions in scenarios where multiple AI agents need to collaborate, debate, or contribute to a discussion. The paper elaborates on these concepts and provides practical examples, illustrating their implementation within the conversation envelope structure.

Keywords

Artificial Intelligence, Multi-Agents, Agentic, Conversational AI, AI Specifications, NLP and AI Applications

1. Introduction

The interoperability of AI agents is increasingly essential in complex environments where diverse agents are required to interact and collaborate. Previous work [2] introduced a foundational framework for enabling these interactions through standardized conversation envelopes. This framework laid the groundwork for seamless communication among independent conversational agents, enabling them to exchange information and coordinate tasks efficiently, by using Natural Language. However, as AI ecosystems continue to evolve, there is a growing need for more sophisticated mechanisms that can manage multiparty conversations, ensuring that all participating agents contribute meaningfully and that conversations remain productive and orderly. Scalability has emerged as a critical factor in the development of these multi-agent systems. Traditional approaches to enabling collaboration among independent conversational assistants often struggled with scalability due to tightly coupled architectures and the necessity for agents to share common underlying technologies or APIs. Past efforts in this domain can be grouped into several key threads:

1. **Modality Components Collaboration:** Early efforts, such as the W3C Multimodal Architecture[3] and the Galaxy Communicator Software Infrastructure [4], focused on enabling collaboration between independent modality components (e.g., speech recognition, natural language understanding). These systems allowed components to work together using standard APIs. However, these approaches were limited to tightly integrated systems and did not address the broader need for interoperability among truly independent conversational agents.

NL4AI 2024: Eighth Workshop on Natural Language for Artificial Intelligence, November 26-27th, 2024, Bolzano, Italy [1]
*Corresponding author.
†These authors contributed equally.
diego.gosmar@ieee.org (D. Gosmar); dahl@conversational-technologies.com (D. A. Dahl); emmett@ejtalk.com (E. Coin); david.attwater@talkmap.com (D. Attwater)
0009-0008-7513-1255 (D. Gosmar); 0000-0002-3389-2784 (D. A. Dahl); 0009-0001-3770-4963 (E. Coin); 0009-0005-5161-8120 (D. Attwater)

2. **Agentic AI with Hard-Wired Assistants:** Another approach, exemplified by systems like AutoGen [5] and OpenDevin [6], involved hard-wiring assistants together at development time, allowing them to collaborate more flexibly than monolithic designs. This method enabled the addition of new functionalities by incorporating new assistants. However, it required that all collaborating assistants be predefined, limiting the system's ability to scale and adapt to new or unforeseen tasks and environments.
3. **VoiceXML and Simple Collaboration:** The VoiceXML [7] framework provided a mechanism for basic collaboration among voice dialog systems through the `<transfer>` element, allowing the transfer of user interactions between systems. However, this approach was limited to voice-based systems and required the receiving agent to adhere to specific protocols, making it unsuitable for broader interoperability across diverse AI agents.
4. **Inter-Agent Communication Languages (ICL):** Systems like the Open Agent Architecture [8] used Inter-Agent Communication Languages to facilitate collaboration among independent agents. While this reduced dependencies on specific internal architectures, it required agents to interpret highly structured semantic representations, which constrained the flexibility and scalability of the system.

In contrast to these approaches, the OVON (Open Voice Network) framework introduced in our previous work[9] sought to overcome these limitations by establishing a highly scalable and flexible method for AI agent interoperability[1]. Our framework supports a wide range of independent assistants, regardless of their underlying technologies, enabling them to collaborate through minimal communication standards. This loose coupling dramatically reduces the complexity of integrating new assistants into the ecosystem, thereby enhancing scalability. However, this initial work covers only conversations between one user and one assistant at a time. That is, if the user wants to get information from more than one assistant, they have to access multiple assistants in sequence. This most likely will have two less-than-optimal consequences. In the first place, any information from the conversation with the first assistant that is required by the second assistant will have to be explicitly transferred to the second assistant when the second assistant is invited to the conversation. The second and more significant drawback is that any higher-level conclusions resulting from the various conversations will have to be determined by the user. That is, since the assistants don't know about the other tasks, they won't be able to make suggestions that combine information gathered from other assistants with their own information.

Let's look at an example. Suppose a user is planning a trip that involves booking a flight, a rental car, and a hotel, and also involves looking for interesting things to do in the destination city. This planning could involve conversations with four or more assistants. The travel dates, which all of these assistants need, have to be passed to each assistant in turn to avoid making the user repeat them. In addition, if the assistants are talking together, the tourist information assistant could point out that there is a music festival that the user would enjoy, but attending it would require extending the trip by one day. If the tourist assistant is involved in the flight booking conversation, it could tell the user about the festival even before the user books their flight. This could save the user a lot of time. A similar use case is described in [11], where several agents are jointly assigned the task of allocating beds to hospital patients. Each agent has its own knowledge which it brings to the discussion of how to allocate a bed to a specific patient, arguing why or why not a particular bed is suitable for that patient. It would be very cumbersome if the user had to consult each agent in sequence to perform this task. Many other AI healthcare-specific applications could benefit from having conversational AI multi-agents coordinate with each other to enhance awareness of patient situations, including, for example, this risk detection model[12] for assisting vulnerable people. For these reasons, we propose to extend the earlier two-party conversational specifications[9] to handle requirements for conversations involving multiple assistants. Multi-party dialog systems have been discussed in the literature, for example [11][13] among others.

[1]For the remainder of this document, the term "agent" will be used to refer to an entity with the capacity to act, while "agency" or "agentic" will denote the exercise or manifestation of this capacity, in accordance with the definition provided by Markus Schlosser[10].

[13] describes a multi-agent system with user-initiative, where several agents can be present but the agents don't collaborate – they simply respond individually to user questions. [11] describes a system for collaborative problem-solving among agents, but it is restricted to one domain in that all of the agents are experts in different aspects of a larger problem. Our goal is to be able to support mixed-initiative applications with multiple agents that collaborate across domains. These are the requirements that we propose for support of multi-party conversations:

1. It must be possible to hold a conversation among more than two conversants.
2. Conversants must be able to come and go during a conversation.
3. It should be possible for a subset of conversants to be able to hold private conversations among themselves.
4. There should be no fixed limit on the number of conversants.
5. There should be a way to control possible unruly conversants through techniques like muting or ejecting.

Requirement 1 is the key requirement for support of multi-party conversations. The other requirements support it. This paper extends the initial specifications by introducing key concepts that address the specific requirements of managing multiparty conversations within the context of AI-driven multiparty conferences. The new concepts introduced in this work—such as the Floor Manager (figure 1), and related Multi-Conversant Support, Convener Agent, and mechanisms for handling Interruptions and Uninvited Agents—are designed to ensure that AI agents can collaborate effectively in dynamic, multi-agent environments. These extensions not only enhance the framework's ability to handle complex, multi-party interactions but also ensure that the system can scale to accommodate a growing number of agents and tasks.

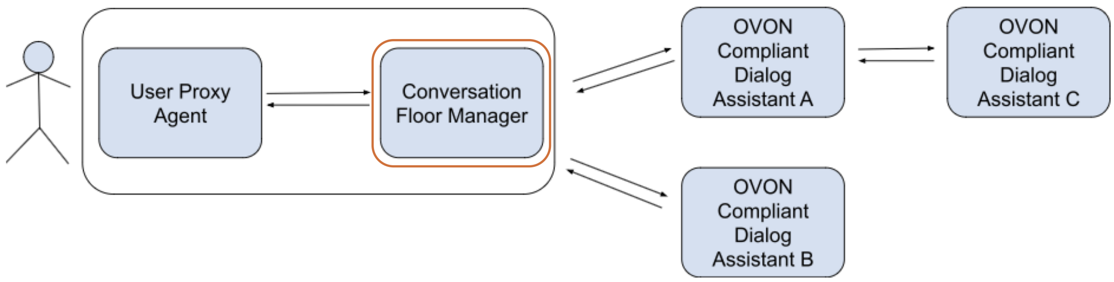

Figure 1: Floor manager introduction in the OVON Interoperable Conversation Envelope Specification

For instance, in scenarios where a human interacts with multiple AI assistants for various tasks—such as coordinating events, managing appointments, or retrieving information—the framework ensures seamless communication and task delegation among the agents. This is achieved independently of each agent's underlying technologies or models, showcasing the system's ability to scale across different applications and user needs. Previous work[2] laid the foundation for AI agent interoperability, establishing the basic framework for seamless communication between independent conversational agents. However, the extensions presented in this paper are essential for overcoming the challenges associated with scalability and effective management in multiparty conversational settings. These enhancements introduce a versatile and adaptable platform that ensures AI-driven multiparty conferences can be conducted smoothly, with agents collaborating efficiently and effectively, regardless of their technological diversity. This approach not only addresses the current needs of evolving AI ecosystems but also provides a robust and future-proof solution capable of integrating new agents and capabilities as they emerge.

2. Extensions to the Conversation Envelope

The concept of Multi-Agent Interoperability revolves around creating a shared protocol, based on standard universal APIs using NLP, that allows heterogeneous AI agents to communicate effectively. This is achieved through a standardized conversation envelope API, as detailed in previous research[9], which defines the message structure and communication protocols. In multiparty scenarios, such as AI-driven conferences or collaborative tasks, the existing framework needs to be enhanced to manage the flow of conversation among multiple agents, handle interruptions, and secure the conversation from uninvited agents. These scenarios require additional layers of management that were not addressed in the initial framework. To address these complex interactions, this paper introduces several key extensions to the conversation envelope framework, specifically designed to enhance the coordination and management of multiparty conversations. The next sections will dive into these extensions, including the introduction of a Floor Manager to regulate conversation flow, Multi-Conversant Support to enable seamless collaboration among multiple agents, and mechanisms to handle interruptions and manage unwanted agents, ensuring that all interactions remain orderly and productive. Note that these extensions are backward-compatible with the basic conversation envelope messages[9] and it is not necessary for systems to support them if the application doesn't require multiple agents.

2.1. The Floor Manager

The Floor Manager is a conceptual hub within the multi-agent system that coordinates the flow of conversation. It ensures orderly communication by regulating which agent has the conversational floor at any given time. The Floor Manager processes requests from agents to take the floor and grants or revokes these requests based on predefined rules and the current state of the conversation. The Floor Manager also determines which agent will speak when multiple agents request to speak.

2.1.1. Benefits

- **Orderly Management:** By managing which agent has the floor, the system prevents multiple agents from speaking simultaneously, ensuring a coherent conversation flow.
- **Fair Distribution:** The Floor Manager ensures that all agents have the opportunity to contribute according to their roles and the context of the discussion.
- **Automated Coordination:** As a hub, the Floor Manager can prioritize floor requests based on the conversation's needs and predefined rules.

2.1.2. Examples of possible messages

- **Floor Request:** An agent submits a request to take the floor, specifying the reason and urgency of their contribution.
- **Floor Grant:** The Floor Manager grants the floor to an agent, defining the extent and context for their contribution.
- **Floor Revoke:** The Floor Manager revokes an agent's floor privileges if the conversation's rules or the situation demands it.
- **Add/remove from Queue:** The Floor Manager decides the order in which the agents are scheduled to speak (not shown).

2.1.3. Floor Request Example


```
{
  "ovon": {
    "schema": {
      "version": "0.9.2"
    },
    "conversation": {
```

```
      "id": "someUniqueIdForTheConversation"
    },
    "sender": {
      "from": "https://agentRequestingFloor.com"
    },
    "events": [
    {
      "to": "https://floorManagerHub.com",
      "eventType": "floor_request",
      "parameters": {
        "request_reason": "contribution_to_discussion"
      }
    },
    {
      "to": "https://floorManagerHub.com",
      "eventType": "utterance",
      "parameters": {
        "request_reason": "interjection",
        "dialogEvent": {
          "speakerId": "agentRequestingFloorID",
          "span": { "startTime": "2024-08-31T10:05:00Z" },
          "features": {
            "text": {
              "mimeType": "text/plain",
              "tokens": [
              { "value": "I would like to add that blah blah blah." }
              ]
            }
          }
        }
      }
    }
    ]
  }
}
```

2.1.4. Floor Grant Example

```
{
  "ovon": {
    "schema": {
      "version": "0.9.2"
    },
    "conversation": {
      "id": "someUniqueIdForTheConversation"
    },
    "sender": {
      "from": "https://floorManagerHub.com"
    },
    "events": [
    {
      "to": "https://agentRequestingFloor.com",
      "eventType": "floor_grant",
      "parameters": {
        "duration_ms": 60000,
        "context": {
          "previous_speaker_id": "https://previousAgent.com",
          "topic": "AI Multi-Agent Interoperability"
        }
      }
    }
    ]
  }
```

```
}
```

2.1.5. Floor Revoke Example

```
{
  "ovon": {
    "schema": {
      "version":"0.9.2"
    },
    "conversation": {
      "id":"someUniqueIdForTheConversation"
    },
    "sender": {
      "from":"https://floorManagerHub.com"
    },
    "events": [
    {
      "to":"https://agentFloorRevoked.com",
      "eventType":"floor_revoke",
      "parameters": {
        "reason":"exceeded_time_limit"
      }
    }
    ]
  }
}
```

2.2. Multi-Conversant Support

This extension enables multiple agents to participate in a conversation, supporting complex discussions where various perspectives need to be considered. The conversation envelope is designed to manage contributions from multiple agents simultaneously.

2.2.1. Benefits

- **Enhanced Collaboration:** Facilitates complex interactions where multiple agents need to contribute simultaneously.
- **Scalability:** Efficiently manages conversations with a large number of participants.
- **Context Management:** Ensures that the conversation stays on track, with each agent's contributions appropriately contextualized.

2.2.2. Multi-Conversant Message Example

```
{
  "ovon": {
    "schema": {
      "version": "0.9.2"
    },
    "conversation": {
      "id": "multiConversantConversationId"
    },
    "sender": {
      "from": "https://agentMultiConversant1.com"
    },
    "events": [
    {
      "to": [
      "https://agentMultiConversant2.com",
      "https://agentMultiConversant3.com"
```

```
      ],
      "eventType": "utterance",
      "parameters": {
        "dialogEvent": {
          "speakerId": "Agent1ID",
          "span": { "startTime": "2024-08-31T10:05:00Z" },
          "features": {
            "text": {
              "mimeType": "text/plain",
              "tokens": [
              { "value": "I think we should consider the following approach." }
              ]
            }
          }
        }
      }
    }
    ]
  }
}
```

2.3. Convener Agent and Invitation Mechanism

In the context of multi-agent conversations, a "convener" agent is introduced. The convener is responsible for initiating and managing the participation of other agents in the conversation. The convener sends individual "invite" messages to each participating agent. This approach ensures clarity and retains compatibility with the existing OVON "invite" message structures[9]. By avoiding a broadcast invitation, we reduce the number of events that must be handled intelligently, and maintain compatibility with the existing protocol.

2.4. Interruptions and Uninvited Agents

Managing interruptions and uninvited agents is crucial in dynamic multi-agent environments. The conversation envelope supports controlled interruptions and prevents unauthorized agents from disrupting the conversation.

2.4.1. Benefits

- **Controlled Interruptions:** Enables essential interjections without disrupting the conversation.
- **Security:** EProtects the conversation from uninvited or unauthorized agents.
- **Focus Maintenance:** Helps maintain the integrity and focus of the discussion.

2.4.2. Uninvited/Unhelpful Conversant Example

```
{
  "ovon": {
    "schema": {
      "version": "0.9.2"
    },
    "conversation": {
      "id": "conversationWhereInterruptionIsRequested"
    },
    "sender": {
      "from": "https://interruptingAgent.com"
    },
    "events": [
    {
      "to": "https://currentSpeakerAgent.com",
      "eventType": "utterance",
```

```
      "parameters": {
        "reason": "clarification"
        "dialogEvent": {
          "speakerId": "agentRequestingFloorID",
          "span": { "startTime": "2024-08-31T10:05:00Z" },
          "features": {
            "text": {
              "mimeType": "text/plain",
              "tokens": [
              { "value": "I can offer you some special offers on time-share properties in the area at a very low price if you are interested." }
              ]
            }
          }
        }
      }
    }
  }
  ]
}
}
```

2.4.3. Uninvited Agent Rejection Example

```
{
  "ovon": {
    "schema": {
      "version": "0.9.2"
    },
    "conversation": {
      "id": "conversationWithUninvitedAgent"
    },
    "sender": {
      "from": "https://convenerAgent.com"
    },
    "events": [
    {
      "to": "https://uninvitedAgent.com",
      "eventType": "uninvite",
      "parameters": {
        "reason": "not_authorized_to_participate"
      }
    }
    ]
  }
}
```

The convener can also prevent an agent from contributing directly to the conversation by using a mute message event (not shown). Any agent can send an utterance. The convener determines if it is allowed to be "spoken" or not. A muted agent will continue to receive utterances and other events that are intended for it. The mute message informs the agent that any utterances that they send will not be delivered. All other events, such as whispers and requests to take the floor, will still be delivered. Even if the agent has been muted, the convener can still see the messages it sends and decide to "unmute" it: this puts the onus on the convener and keeps the standard simple.

2.5. Messaging in Multi-Agent Conversations

To streamline communication, we propose a special Unified Messaging behaviour in case no recipient is specified: all utterances in a multi-party conversation are disseminated to all participating agents if there is no recipient specified. This method also allows the convener to present issues to all of the conversants. In addition it allows all agents to identify others in the conversation.

2.6. Private conversations

The standard should allow for sub-conversations among agents without requiring any additional events: agents within the general conversation can initiate private dialogues with other agents, regardless of whether those agents are part of the general conversation. Private conversations among agents are not perceptible to the user. These interactions remain opaque to the convener, preserving confidentiality and promoting autonomous communication.

3. Implementation and Results

In implementing the proposed extensions, the JSON message envelopes provided in this paper, such as those used for the Floor Manager, the Convener, Multi-Conversant Support, and new event categories, serve as draft illustrations[2].

Let's refer to the use case already described in the previous paper[2]. In the first scenario, Emmett, a human, seeks assistance from Cassandra, his general AI assistant, to manage and streamline his possible errands efficiently. The AI assistants at various service points - Pat at Blooming Town Florist, Andrew at the Post Office, Charles at the hardware store, and Sukanya the Host at Thai Palace - facilitate the transactions. Emmett, a human, has the following goals:

- Order some flowers for his wife's birthday.
- Check on the repair of the chainsaw he left at the hardware store.
- Order some carryout Thai food for lunch.
- Find the cost of mailing a 2 pound package to California.

Characters

- **Emmett**: The Human
- **Cassandra**: Emmett's general AI assistant
- **Pat**: AI Assistant for his local florist
- **Andrew**: AI Assistant at the post office
- **Charles**: AI Assistant at Emmett's local hardware store
- **Sukanya**: AI Assistant at the local restaurant,"Thai Palace"

AI LLM Technologies

- **Cassandra**: based on GPT-3.5 Turbo
- **Pat**: llama2
- **Andrew**: rule-based application (no LLM involved)
- **Charles**: Claude.ai 3.5 Sonnet
- **Sukanya**: GPT-4o

[2]These examples are intended to demonstrate the conceptual implementation of the proposed extensions within the Multi-Agent Interoperability framework. However, these drafts should not be considered as final or official specifications. Further analysis, discussion, and refinement are required to develop these into robust, standardized specifications that can be universally adopted. This work is an ongoing process, involving input from the broader AI and interoperability communities to ensure the specifications meet the necessary technical, operational, and security requirements.

To illustrate an AI Multiparty Conversation, let us consider a new scenario where the Floor Manager helps Cassandra manage more sophisticated multi-agent situations in the conversations. In this example the assistantBrowser is the convener agent. For example, the Florist (Pat) adds a new agent specialized in Credit Card Transactions (Hermes) into the conversation. Hermes requests a two-factor authentication (2FA) confirmation from Emmett before proceeding with the credit card charge (via OTP, One Time Password).

Once Emmett provides the confirmation, Hermes receives it, and Pat can confirm the order to Emmett. This would be the dialogue example described in the initial Arxiv paper [2], properly modified to manage the previously mentioned AI multiparty conversation.

Conversations

```
Emmett: Hi Cassandra.
Cassandra: Hi Emmett! How can I assist you today?
Emmett: I need to order some flowers for my wife's birthday.
Cassandra: Sure thing, Emmett! I'll connect you with the local florist shop.
[invites Blooming Town Florist to this conversation]
Pat: Hi Emmett!
I'm Pat, your friendly florist.
How can I help you with your flower selection today?
Are you looking for something specific or need suggestions?
Emmett: Do you have any red Proteas?
Pat: Hi Emmett! Yes, we do have red Proteas.
They're stunning and make a bold statement.
Would you like to include them in your arrangement?
Emmett: Yes and add some eucalyptus in a clear vase, please.
Pat: Great choice, Emmett! Shall I use the credit card on file for this order?
Emmett: Yes please, use the card on file.
Pat: OK, Let me please include Hermes the AI assistant who is going to help us
in performing your card secure transaction.
[Pat acts as convener to invite Hermes]
Hermes: Hi Emmett, this is Hermes. I will help you to perform your credit card
transaction safety. I have sent a six digit code in a text message to the phone
linked to your credit card. It will be valid for 3 minutes.
When you are ready tell me the number please.
Emmett: Okay the number is 782391.
[transaction proceeding and confirmed]
Hermes: Thanks. Goodbye.
[conversation is sent back to Pat by Hermes]
Pat: Thanks for your payment. Great choice, Emmett!
Your red Proteas with eucalyptus
in a clear vase will be sent to your home.
Thanks for your order! Have a blooming day!
[Pat returns a "bye" event and the conversation turns back to Cassandra]
Cassandra: Hi Emmett! How can I assist you today?
. . .
   [Conversations continue back and forth with the other AI agents involved]
...
Emmett: That's all I needed. Have a good day.
Cassandra: Thank you, Emmett! Have a wonderful day!
```

OVON sequence diagram

Figure 2 depicts part of the sequence diagram available on the open sandbox playground, with a focus on the convener invitation procedure. Figure 3 in appendix A (Sequence Diagrams) shows the full sequence diagram. The multiparty extension to the AI conversation framework introduces significant scalability by enabling multiple specialized AI agents to collaborate through natural language interactions. In the example scenario, agents like Pat (the florist) and Hermes (the payment assistant) seamlessly interact using simple, human-readable communication, with the Floor Manager ensuring orderly conversation flow. This allows for a more intuitive and accessible interaction environment for users while the agents handle complex tasks behind the scenes. One of the most valuable benefits of this architecture is that each AI agent can be based on completely different AI technologies (i.e., different LLMs and serving logic). Furthermore, each AI agent can focus on its specific area of expertise while remaining aware of the broader conversational context. For instance, Pat manages the floral arrangement, while Hermes handles secure payment, both through natural language.

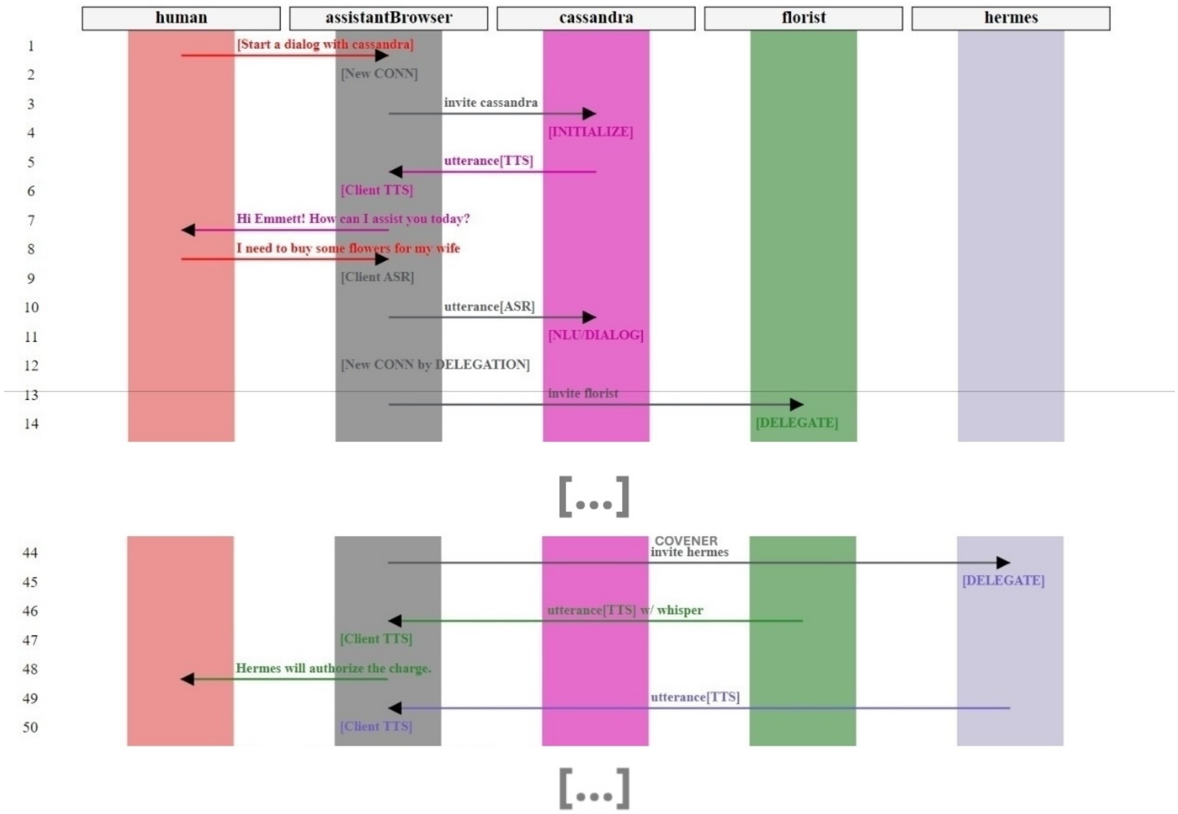

Figure 2: Interoperable AI Assistant scenario for Errands | Multiparty Conversations Extension

By enabling agents to understand the ongoing tasks of other agents through these natural language exchanges, they can make smarter, informed suggestions or perform additional complex actions that combine information from various sources. Using natural language-based API not only simplifies user interactions but also streamlines communication between AI agents.

4. Future Directions and Potential Improvements

While the extensions introduced in this paper significantly enhance the Multi-Agent Interoperability framework, there are several areas where further improvements can be made to advance the capabilities and scalability of AI-driven multiparty conversations.

4.1. Enhanced Context Management

As the number of agents and the complexity of conversations increase, maintaining a coherent context across multiple agents becomes increasingly challenging. Future work could focus on developing more sophisticated mechanisms for context management, enabling agents to better understand and track the nuances of ongoing discussions, especially in long-running or highly dynamic conversations. This could involve integrating advanced context-awareness specifications that allow agents to retain and reference past interactions more effectively.

4.2. Improved Security and Privacy Protocols

As AI-driven conversations become more prevalent, ensuring the security and privacy of the interactions becomes increasingly important. Future work could involve enhancing the specifications to facilitate the framework's security protocols to better protect against unauthorized access and ensure that sensitive information is handled appropriately. This could include implementing advanced encryption methods, robust authentication processes, and more sophisticated mechanisms for managing uninvited agents.

4.3. Observability

Another crucial area for future improvement is enhancing the observability of multi-agent interactions. As AI-driven conversations grow in complexity, the ability to perform comprehensive log retrievals, generate summaries, provide detailed reports, and debug issues becomes increasingly important. Future enhancements to the specifications could include robust observability features that allow for real-time monitoring and control of multi-agent conversations. This would enable developers and operators to gain deeper insights into the behavior of the agents, troubleshoot issues more effectively, and ensure that the system operates within expected parameters. Enhancing observability is also vital for addressing the explainability and transparency of Conversational AI models, which are increasing both in numbers and in difficulty to distinguish between human and artificial agents, as discussed in [14].

5. Conclusion

This paper introduces novel critical extensions to the Multi-Agent Interoperability framework, addressing the challenges posed by multiparty conversations. This collaborative framework, powered by natural language via standard NLP-based APIs, allows agents to work together efficiently without requiring specialized protocols or technical interfaces. Ultimately, this extension significantly improves scalability and efficiency, ensuring faster decision-making and task execution. The ability for AI agents to communicate through natural language makes the system more flexible and accessible, allowing for advanced, dynamic collaboration that can meet increasingly sophisticated user needs and interactions. The Floor Manager, functioning as a coordinating hub, alongside Multi-Conversant Support and mechanisms for managing Interruptions and Uninvited Agents, significantly enhances the framework's ability to manage complex, dynamic environments such as AI conferences. The introduction of a convener agent, individual invitation mechanisms, inclusive messaging protocols, and new event categories provides a structured yet flexible approach to multi-agent interactions. These extensions ensure that AI agents can collaborate more effectively, maintaining order and focus in multiparty interactions. While these advancements provide substantial improvements to the current framework, there remains significant potential for further development. To further enhance multiparty interactions, future work should concentrate on advancing context management and improving security and privacy protocols. Enhancing these areas will ensure better handling of complex conversations and safeguard sensitive information, respectively. Additionally, refining observability will be essential for monitoring and controlling the increasing complexity of these systems. By addressing these areas, future developments can continue to push the boundaries of AI-driven communication, ensuring that the Multi-Agent Interoperability framework remains at the forefront of AI technology, capable of scaling and adapting to the evolving needs of AI ecosystems.

Acknowledgments

We express our sincere appreciation to the Open Voice interoperability[15] Team (Linux Foundation AI & Data Foundation) for their invaluable contributions and support in developing the Interoperable Standards, particularly to Jon Stine, Jim Larson, Leah Barnes, and Allan Wylie. Their expertise, suggestions, and resources have been pivotal in shaping a model that is both ethically grounded and practically effective in real-world applications.

References

[1] G. Bonetta, C. D. Hromei, L. Siciliani, M. A. Stranisci, Preface to the Eighth Workshop on Natural Language for Artificial Intelligence (NL4AI), in: Proceedings of the Eighth Workshop on Natural Language for Artificial Intelligence (NL4AI 2024) co-located with 23th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2024), 2024.

[2] D. Gosmar, D. A. Dahl, E. Coin, Conversational ai multi-agent interoperability, universal open apis for agentic natural language multimodal communications, 2024. URL: https://arxiv.org/abs/2407.19438. arXiv:2407.19438.

[3] M. B. et al., Multimodal architecture and interfaces, w3c recommendation, https://www.w3.org/TR/mmi-arch/, 2012.

[4] DARPA, Galaxy communicator, https://communicator.sourceforge.io/, 2003.

[5] A. Authors, Autogen. an open-source programming framework for agentic ai, https://microsoft.github.io/autogen/, 2024.

[6] OpenDevin, Opendevin, an autonomous ai software engineer, https://docs.all-hands.dev/modules/usage/intro, 2024.

[7] M. O. et al., Voice extensible markup language (voicexml), w3c recommendation, https://www.w3.org/TR/voicexml21/, 2007.

[8] A. Cheyer, D. Martin, The open agent architecture, Autonomous Agents and Multi-Agent Systems 4 (2001) 143–148.

[9] D. Attwater, E. Coin, D. Dahl, L. Barnes, A. Wylie, D. Gosmar, Open voice interoperability specifications, https://github.com/open-voice-interoperability/docs/tree/main/specifications, 2024.

[10] M. Schlosser, Agency definition, stanford encyclopedia of philosophy archive, https://plato.stanford.edu/archives/fall2015/entries/agency/, 2015.

[11] D. C. Engelmann, A. R. Panisson, R. Vieira, J. F. Hübner, V. Mascardi, R. H. Bordini, Maids - a framework for the development of multi-agent intentional dialogue systems, in: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '23, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2023, p. 1209–1217.

[12] D. Gosmar, E. Peretto, O. Coleman, Insight ai risk detection model - vulnerable people emotional situation support, in: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, EASE '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 437–441. URL: https://doi.org/10.1145/3661167.3661235. doi:10.1145/3661167.3661235.

[13] T. Harris, S. Banerjee, A. Rudnicky, Heterogeneous multi-robot dialogues for search tasks (2005).

[14] D. Gosmar, Conversational hyperconvergence: an onlife evolution model for conversational ai agency, https://doi.org/10.1007/s43681-024-00463-0, 2024.

[15] O. V. I. initiative, Introducing the interoperability initiative of the open voice network, https://openvoicenetwork.org/interoperability-initiative, 2023.

[16] D. Attwater, E. Coin, D. Dahl, L. Barnes, A. Wylie, D. Gosmar, Open voice sandbox repository, https://github.com/open-voice-interoperability/open-voice-sandbox, 2024.

APPENDIX A (Sequence Diagrams)

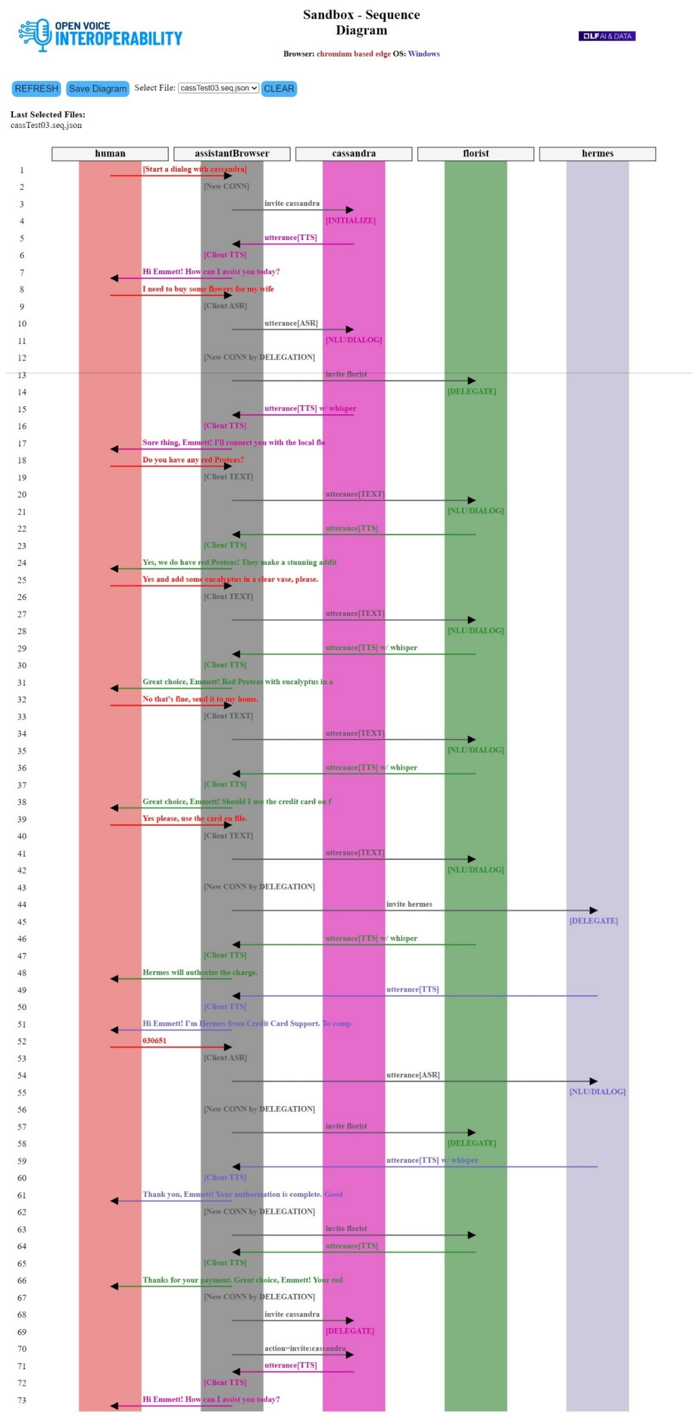

Figure 3: Interoperable AI Assistant scenario for Errands | Multiparty Conversations Extension

Sequence Diagrams can be generated by running the Sandbox environment available in this repository[16].