A Chatbot that Uses a Multi-Agent Organization to Support Collaborative Learning

Mateus da Silveira Colissi[1], Renata Vieira[2][0000−0003−2449−5477], Viviana Mascardi[3][0000−0002−2261−9926], and Rafael H. Bordini[1][0000−0001−8688−9901]

[1] School of Technology, Pontifical Catholic University of Rio Grande do Sul, Av. Ipiranga, 6681 – Porto Alegre, RS, Brazil
Mateus.Colissi@edu.pucrs.br, rafael.bordini@pucrs.br

[2] CIDEHUS, University of Évora, Palácio do Vimioso Largo do Marquês de Marialva, n.º 8 - Évora, Portugal
renatav@uevora.pt

[3] Department of Informatics, Engineering, Robotics and Systems Engineering, University of Genova, Via Dodecaneso, 35, 16146 Genova GE, Italy
Viviana.mascardi@unige.it

Abstract. This work investigates and apply the use of a multi-agent system to assist in the coordination of group tasks, specifically in educational environments, in which the interaction occurs indirectly, that is, asynchronously. The system has a web interface integrated with a chatbot for more natural interaction. The chatbot communicates with the multi-agent system that is responsible for the organization of the group, that is, it contains information about the tasks and members of the groups, in addition to restrictions that can be imposed according to the organization of the group, and it is also able to return the requested information in natural language through the chatbot. This approach was validated in a practical undergraduate course of software engineering. The students assessed the functionalities and usability of the system while working in groups in order to develop software collaboratively. Our system was used to assist students in a real project. With this assessment, it was found that the system was able to support the development of the group tasks, ensuring quick and consistent responses to the student's request.

Keywords: Multi-Agent System · JaCaMo · Chatbot · Dialogflow · Group Coordination

1 Introduction

Institutions use different types of learning methods, such as classroom learning and virtual learning. However with different learning methods, we need different learning approaches. To assist these learning methods, several techniques and tools are being used in virtual environments, such as: chats, conversational agents and others. With the use of virtual learning environments, concerns arise for example regarding collaboration between people in learning techniques such as groupwork.

According to King [11], collaborative learning may motivate studies more than individual study, so in educational environments it is important to promote collaborative learning to enable group participation and interaction in various tasks, where knowledge is built through dialogues that enable the sharing of ideas and information within the group [12]. Also, it can provide important feedback for the teacher to know how were the interactions and discussions made by the group, as well as the individual contribution of students in problem solving [1].

However, there are various causes of inefficiencies in groupwork such as poor capacity balance, incorrect team dynamics, poor communication or difficult social situations [2]. Solving these inefficiencies for virtual teams would help improve the relationship of team members in their online learning environment, allowing them to complete tasks daily, improving collaboration, productivity, and task tracking.

One way to support the management of group learning is to create environments that facilitate knowledge sharing and other valuable learning attitudes, helping to promote student discussion and interaction skills [7]. With the advancement of Artificial Intelligence (AI), there is growing use of conversational agents such as chatbots to aid learning. However, most of the work in this area are for individual learning.

In a functional group, each member is responsible for one or more tasks. One of the main requirements for group work to perform well is that its members operate in harmony. Contributing toward solutions for the problem of groupwork coordination, this work aims mainly to: improve information sharing among group members; increase group productivity; improve overall performance in collaborative tasks; and to allow the person responsible for the groups to be aware of noteworthy events during the performance of the tasks.

This work explores the use of virtual assistants in a collaborative environment, where agents represent their users as part of a groupwork project, and they also assist in the organization and communication between users. The communication is done through the chatbot and not directly.This multi-agent system has the main objective of facilitating the dissemination of information about the current status of the project and allowing the person in charge of the group to monitor performance during the execution of tasks without disturbing the the individual members. In particular, the focus is on academic environments, where it is very important that the group members as well as the lecturers are aware of how the project is evolving.

2 Related Work

Islam et al. [10] proposed a browser-based client interface approach with Express and SocketIO framework, Node.js, Jade and AngularJS, that can be accessed using only a web browser available on PCs, tablets, laptops, or mobile devices. The teacher uploads the lecture to the site for students to study/solve by collaborating in pairs. Collaborative peer activities were proposed where students

collaborated synchronously or asynchronously in conjunction with the teacher's lecture.

Tegos and Demetriadis [14] proposed a prototype dialogue support system, acting as an instant messaging application, in order to accomplish one or more learning tasks in an online activity, with a conversation agent that uses text-to-speech to read its interventions, offering academically productive talk (APT). To encourage collaboration, the agent introduces concepts into the group discussion displaying outside the main chat frame using the Levenshtein-based string similarity algorithm and a WordNet lexicon for synonyms to calculate a proximity score for each identified concept.

Paikari et al. [13] proposed a chatbot called Sayme to address the detection and resolution of possible code conflicts that may arise in the development of parallel software. Sayme is implemented using Python, MySQL, and Google Cloud Platform. Slack is used as the chat channel to communicate with developers. In addition, the chatbot operates autonomously, initiating conversations with developers based on information collected from Git. Sayme also monitors when developers save files using Git commands to automatically extract files that are being changed and on which lines they are being changed. To detect possible indirect conflicts between files, all files are analyzed using the Abstract Syntax Trees (AST) library.

Sayme provides information about potential direct and indirect conflicts, helping developers to resolve those conflicts. Its main function is to proactively detect possible conflicts between developers working in parallel, notifying them before the conflicts become too complex. Its secondary function is to respond to a variety of requests that help developers understand the state of work of other developers.

Unlike other approaches in the literature, our architecture allows a formal representation of the group organization through a MAS, managing multiple groups simultaneously, in which each group has different roles and tasks for members to carry out. This improves the management of knowledge related to the proposed organization in collaborative work. On the other hand, there are challenges in the use of these systems in collaborative educational environments, for example, due to the efforts required in the construction of agent organizations, in which roles and tasks must be well defined and explicitly represented within the system.

3 Our Approach

The overall architecture of the proposed system involves software development in different environments to allow better user interaction and coordination. Our system is able to receive requests in the form of questions (about the state of the organization) or actions to be performed (register new tasks, select new tasks, among others) and return a coherent answer with the user's group through plans in the JaCaMo multi-agent systems development platform [3].

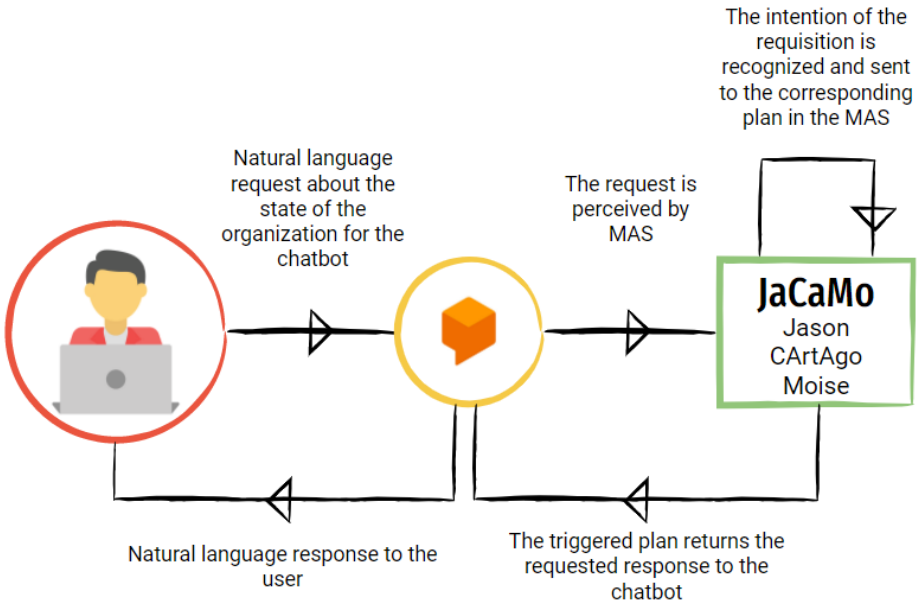

Fig. 1. Our Approach

Figure 1 shows our approach. The student makes a request in natural language about the tasks they need to complete and the chatbot returns information from the MAS. The tasks that students had to achieve were related to software development, that is, front-end and back-end development tasks, among others. The following are examples of tasks to be performed: modeling the persistence layer, generating the initial physical and logical database schema, creating the POST method to receive the registration data, creating the GET endpoint in the users API, and creating the profile screen and user route management.

A website was developed to define permission levels and assist in the identification of agents in JaCaMo, in addition to allowing the leader to register/delete users and make requests to the chatbot. The website was developed using the Angular frontend framework, for the backend Firebase was used with the Realtime Database. To coordinate the groups, the JaCaMo system is able to collect information from the database to initialize the groups according to commands from the leaders. The information generated at runtime by the organization is not stored in the database, but in the JaCaMo artifact for each group. For the development of the chatbot, the Dialogflow platform was used. The next sections present the details of the architecture components mentioned above.

3.1 Chatbot

To communicate with the JaCaMo MAS, *intents* were created in Dialogflow for each question to be answered about the group's organization and for each

action to be performed. To organize requests made by users, intents were trained using parameters. The phrases were trained with a parameter name (*@name*) for queries of the type "name_last-name_discipline_role Question". The *@name* parameter was created to identify the agent in MAS and is managed by an entity that recognizes a pattern through regular expressions.

3.2 Multi-Agent System

The system was implemented on the JaCaMo platform using a Dialogflow integration developed by Engelmann et al. (available in https://github.com/DeboraEngelmann/helloworld_from_jason/wiki). The available JaCaMo-Dialogflow integration was adapted to suit the needs of our project.

JaCaMo enables the integration of three multi-agent programming dimensions: agents, organizations, and environment. A JaCaMo system consists of the following platforms: Jason [5] for agent programming, CArtAgO [4] for environment programming, and Moise [8] for programming organizations. JaCaMo integrates these three platforms for a uniform and consistent programming model, with the goal of simplifying the combination of these dimensions when programming MAS [3].

Organizations can be used to ensure coordination in multi-agent systems. Coordination is related to agents' social skills, where agents communicate with each other to share data, beliefs, goals, and plans [5]. With coordination, agents can achieve joint objectives and plans that otherwise would not be possible, and ensure that tasks are performed consistently and efficiently by synchronizing their actions and interactions with other agents [9,6].

In JaCaMo, agents who are members of a group play various roles, each with a set of distinct missions that must be fulfilled for the project to be completed. These roles and missions must be part of a Moise scheme, previously organized, usually before the system started. Note that such schemes must be developed according to the instructions and needs of the responsible teacher for each application.

For the development of the JaCaMo MAS, three types of agents were used:

Request coordinator is the agent who focuses on the artifact related to the requests from Dialogflow. It has plans to deal with each Dialogflow intent, that is, for each chatbot intent there is a corresponding plan. As it receives all requests from Dialogflow, it is responsible for the communication between the real user and the respective agent, that is, it forwards the information to the respective agent. It also has plans responsible for starting a leader agent at runtime, sending the necessary beliefs for that agent to start a group.

Leader is the agent that is responsible for starting a group at runtime, creating the members of the team dynamically with the information provided by the request coordinator. It has plans responsible for: initializing and defining the team's workspace, group, and initial tasks; in addition to initializing team members, it is also responsible for assigning missions and roles to these agents; and finally it is responsible for answering the questions regarding the group,

that is, organizing the group's information and forwarding it to the request coordinator according to the requests.

Student is the agent that has plans to dynamically join a group and workspace as directed by the group leader. It also has plans related to the tasks to be completed for the resolution of the group project, that is, the plans are related to the situation of the tasks in relation to the Moise scheme, such as: selecting, dropping, and completing tasks. Finally, it is also able to directly answer the request coordinator's queries regarding tasks.

Table 1 shows the commands available in the system. The `command` operation has two representations in the system, one for leaders that shows: tasks [register task, remove task, list tasks, tasks not being performed and uncompleted tasks], group [group members, list due date for tasks, remove member and mark project as complete], log and members' productivity and one for students: tasks [list registered tasks, select task and list unfinished tasks], my tasks and group [group members and project status (completed or not completed)].

Table 1. System commands

Command	Description	Responsible agent
`commands`	Show the commands available in the system	Leader and Student
`create group`	Initialize the group in the MAS	Leader
`tasks`	Show commands for tasks	Leader and Student
`group`	Show commands for groups	Leader and Student
`log`	Show the group log	Leader
`productivity`	Show group productivity	Leader
`update`	Show the last update made by the group	Leader and Student
`objective`	Show the project description	Leader and Student
`delivery date`	Show the due date for tasks	Leader and Student
`completed project`	Mark the project as completed	Leader
`how long to deliver the task`	Show how much time is left before the delivery date	Student
`project`	Show the state of completion of the project	Student
`concluded`	Mark task as complete	Student

4 Experiment Results and Analysis

Our system was evaluated during two months of tests with students in an undergraduate course on software engineering. It was tested with two groups (two software development projects), in which the groups consisted of eighteen students and a lecturer responsible for the group. The students performed several sprints during the development of the project; with each sprint, new tasks were added to the MAS, specifically in the Moise organization, as well as corrections and improvements in the system's response.

The students' questions and answers were checked through the Dialogflow platform itself to assess whether the chatbot was able to correctly identify and trigger the desired intent. Of the 577 requests made to the chatbot, only 54 were incorrectly identified.

The MAS was able to deliver quick responses (which is necessary due to the time that Dialogflow waits for a query to be answered) and consistent with the group and its representation in the system. The exchange of messages between the agents of the system to consult information about the group was an interesting approach regarding the organization, since all information about a group was stored in one place, specifically with the group leader.

The most important reason for using MAS is the possibility to create multiple domains. In particular, if there are different people or organizations with different objectives and proprietary information, the MAS is capable of handling those interactions effectively, and possibly efficiently, e.g., when decentralised coordination is required. That is, MAS are able to model an organization's internal affairs in unified approach, avoiding the need to develop an organization that encompasses all representations of roles and tasks, but rather to develop different organizations that are accessible in a single system with its own capabilities and priorities.

5 Conclusion and Future Work

Increasingly, remote (non face-to-face) learning has been used in the education or training of people through digital resources to acquire new knowledge, to develop professionally by acquiring new skills and abilities of the most diverse types, among others. With reasonable technological support, management systems play a large role in collaborative groups and can mainly assist in online education, which is being used as an alternative to face-to-face activities to continue education amid the restrictions imposed by the pandemics of COVID-19.

In this context, online education allows people who do not have access to information in physical environments (for social reasons or for a specific situation, such as the pandemic) to easily, quickly, and dynamically use personalized and efficient knowledge from a digital platform. Although this form of learning has so many good points, it is still far from being the ideal method. There are concerns about the collaborative distance learning method, in which the main problems to be solved in this approach are: balance of skills within a group, incorrect group dynamics, lack of communication in the group and difficulty with social situations.

Our work provides initial evidence that an approach based on multi-agent systems are adequate, in particular with the JaCaMo platform, because it allows us to create and control an organization in terms of members and the management of task assignment. Communication with the user through a chatbot aims to allow a more effective and natural communication with the system. As future work, we intend to work on the chatbot's pro-activity in interaction with the group, thus making it an active member in group decision making.

Acknowledgments

This work was partially funded by the Portuguese Foundation for Science and Technology, project UIDB/00057/2020. The authors also gratefully acknowledge partial support from CAPES and CNPq.

References

1. Allaymoun, M.H., Trausan-Matu, S.: Analysis of collaboration in computer supported collaborative learning chat using rhetorical schemas. In: Proceedings of the International Conference on Information and Communication Systems. pp. 39–44 (2016)
2. Andrejczuk, E., Rodríguez-Aguilar, J.A., Roig, C., Sierra, C.: Synergistic team composition. In: Proceedings of the Conference on Autonomous Agents and MultiAgent Systems. p. 1463–1465 (2017)
3. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with jacamo. Science of Computer Programming **78**, 747–761 (2013)
4. Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E. (eds.): Multi-Agent Programming, Languages, Tools and Applications. Springer (2009)
5. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming multi-agent systems in AgentSpeak using Jason. John Wiley & Sons (2007)
6. Brazier, F.M.T., Mobach, D.G.A., Overeinder, B.J., Wijngaards, N.: Supporting life cycle coordination in open agent systems. In: Proceedings of the MAS Problem Spaces Workshop at AAMAS. pp. 1–4 (2002)
7. Ferschke, O., Tomar, G., Rosé, C.P.: Adapting collaborative chat for massive open online courses: Lessons learned. In: Proceedings of the International Conference on Artificial Intelligence in Education. pp. 13–18 (2015)
8. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the moise$^+$ model: programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering **1**, 370–395 (2007)
9. Huhns, M.N., Stephens, L.M.: Multiagent Systems and Societies of Agents, chap. 2, p. 79–120. Massachusetts Institute of Technology Press, Cambridge, USA (1999)
10. Islam, A., Flint, J., Jaecks, P., Cap, C.: A proficient and versatile online student-teacher collaboration platform for large classroom lectures. International Journal of Educational Technology in Higher Education **14–1**, 29 (Nov, 2017)
11. King, A.: Structuring peer interaction to promote high-level cognitive processing. Theory Into Practice **41**, 33–39 (Mar, 2002)
12. Neto, A.J.M., Fernandes, M.A.: Chatbot and conversational analysis to promote collaborative learning in distance education. In: Proceedings of the International Conference on Advanced Learning Technologies. pp. 324–326 (2019)
13. Paikari, E., Choi, J., Kim, S., Baek, S., Kim, M., Lee, S., Han, C., Kim, Y., Ahn, K., Cheong, C., van der Hoek, A.: A chatbot for conflict detection and resolution. In: Proceedings of the 1st International Workshop on Bots in Software Engineering. pp. 29–33 (2019)
14. Tegos, S., Demetriadis, S.N.: Conversational agents improve peer learning through building on prior knowledge. Educational Technology & Society **20–1**, 99–111 (Jan, 2017)