colMOOC

# D2.2

# Conversational Agent Module

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the
Erasmus+ Programme
of the European Union

# The colMOOC:

# Integrating Conversational Agents and Learning Analytics in MOOCs

# D2.2 - Conversational Agent Module

| | |
|---|---|
| **Project number:** | 588438-EPP-1-2017-1-EL-EPPKA2-KA |
| **Grant Agreement No:** | 2017-2841/001-001 |
| **Project acronym:** | colMOOC |
| **Project title:** | Integrating Conversational Agents and Learning Analytics in MOOCs |
| **Programme, Key action, Action type, Topic:** | E+ KA2: Cooperation for innovation and the exchange of good practices, Knowledge Alliances |
| **Start date of the project:** | 01/01/2018 |
| **Duration:** | 36 months |
| **Project web site:** | https://colmooc.eu/ |

| | |
|---|---|
| **Deliverable type:** | R |
| **Deliverable reference number:** | D2.2 |
| **Deliverable title:** | Conversational Agent Module |
| **WP contributing to the deliverable:** | WP2 |
| **Delivery date:** | 30/12/2018 |

| | |
|---|---|
| **WP Leader:** | Stavros Demetriadis |
| **Responsible organization:** | AUTH |
| **Abstract:** | This report will document the deployment, configuration, modification of the CA module. The report includes information regarding the life cycle development methodology and the modifications that took place through the development process, the architecture of editor and player components, and the demonstration of agents intervention strategies. |
| **Keywords:** | Agent, conversational agent, MOOC, model design, intervention strategy, deployment |

| | |
|---|---|
| **Dissemination level:** | Public |

## Authors and Reviewers

Stavros Demetriadis, Aristotle University of Thessaloniki (AUTH), Greece, sdemetri@csd.auth.gr

Stergios Tegos, Aristotle University of Thessaloniki (AUTH), Greece, stegos@csd.auth.gr

Georgios Psathas, Aristotle University of Thessaloniki (AUTH), Greece, gpsathas@csd.auth.gr

**REVIEWERS**

Yannis Dimitriadis, University of Valladolid (UVA), Spain, yannis@tel.uva.es

Santi Caballé, Open University of Catalunia (UOC), Spain, scaballe@uoc.edu

## Document Change Log

| Version | Date (mm/dd/yyyy) | Author (s) | Sections Changed |
|---|---|---|---|
| 0.1 | 10/15/2018 | See author list | ALL |
| 0.2 | 11/01/2018 | See author list | ALL |
| 1.0 | 12/02/2018 | See author list | Minor revisions |

## Executive Summary

This deliverable presents the configuration of the Conversational Agent (CA) module of the colMOOC platform. More analytically, it describes the system design, development and deployment processes and provides additional information on the system architecture as well as low-level components. Finally, it displays the modifications that took place while working towards the implementation and release of the newer versions of the CA module.

The deliverable is split into three major sections:

1. *Configuration of Conversational Agent Module*, which analytically demonstrates the agent intervention strategies (static and dynamic).
2. *Conversational Agent Module Deployment*, which presents: the agile life cycle development methodology, the major components of the software architecture (agent editor and agent player), and a series of use cases for the editor and player system components.
3. *Modifications to the Conversational Agent Module*, which presents the modifications that took place throughout the development process emphasizing the evolution of the graphical user interface, providing also comprehensive screen examples.

## Table of Contents

## List of Tables

## List of Figures

## List of Acronyms

| Acronym | Description |
|---|---|
| AB | Advisory Board |
| APT | Accademically Productive Talk |
| CA | Conversational Agent |
| CSS | Cascade Style Sheets |
| DLP | Deliverable Lead Partner |
| EB | Ethics Board |
| IPG | Intellectual Property Management Group |
| MOOC | Massive Open Online Course |
| PC | Project Coordinator |
| QA | Quality Assurance |
| QC | Quality Control |
| QEG | Quality Evaluation Group |
| SB | Project Supervisory Board |
| SM | Scientific Manager |
| TM | Technical Manager |
| ToC | Table of Contents |
| UG | User Group |
| WPL | Work Package Leader |

# 1 Introduction

## 1.1 Purpose of this document

The objective of this deliverable is to:

A. Present the configuration of the Conversational Agent (CA) module of the colMOOC platform
B. Describe the system design, development and deployment processes and provide additional information on the system architecture as well as low-level components
C. Discuss the modifications that took place while working towards the implementation and release of the newer versions of the CA module

## 1.2 Document structure

To accomplish the above objective, the present deliverable is split into three major sections:

### 1.2.1 The Configuration of Conversational Agent Module

In the first chapter, we present the intervention strategies, which can be either an event or a combination of events triggering a specific conversational agent intervention (D2.1, 2018).

In the scope of the colMOOC project, patterns (D2.1, 2018) fall into one of the following two categories:

- Static intervention strategies
- Dynamic interventions strategies

In this section, we analyze all static and dynamic interventions strategies implemented by the colMOOC agent, we refer to the condition(s) that may trigger each intervention strategy, and we introduce examples of intervention messages.

### 1.2.2 Conversational Agent Module Deployment

In the second section, we analyze the agile life cycle methodology adopted during the development of the colMOOC conversational agent module.

Moreover, we discuss colMOOC architecture and the components that constitute a conversational agent. These are:

- The colMOOC agent editor, which can be used by the teachers in order to set up a conversational agent activity
- The colMOOC agent player, which is responsible for presenting to the learners a chat-like user interface for online collaborative activities

Additionally, we present a series of use cases for the editor and player system components. These use cases serve as a guide for (a) the instructors, who are interested in setting up a conversational agent activity via the colMOOC editor, and (b) students, who participate in a colMOOC activity.

### 1.2.3 Modifications to the colMOOC Conversational Agent Module

In the third section, we discuss the modifications that took place throughout the development of the colMOOC conversational agent module. We demonstrate the evolution of the graphical user interface, and we provide screen examples of each development stage. In this scope, we describe

some recommendations made by the design team along the way and discuss how those affected the development of the agent user interface and functionality.

### 1.3 Audience

This document is open and publicly available.

## 2 The Configuration of Conversational Agent Module

### 2.1 Conversational Agent Intervention Design

As we have already described in deliverable D.2.1, teachers can configure and shape the domain model using the colMOOC editor. More specifically, teachers can create conceptual links, consisting of one concept or two concepts connected with a verb or a modifier. Those conceptual links feed the intervention component of the colMOOC architecture. While building conversational intervention mechanisms, attention is given on identifying efficient and effective techniques of modeling and triggering constructive peer interactions through fine-tuned agent interventions.

Conversational agent interventions emerge following the identification of an associated 'intervention strategy'. An intervention strategy refers to an event or a combination of events, which take place in an online chat activity and may be of interest for the agent to detect and analyze as an opportunity for triggering an intervention. Usually, an intervention strategy is regarded as a combination of something uttered by the human discussants along with some contextual information of what is going on in the chat environment. In the scope of the colMOOC project, patterns fall into one of the following categories:

(a) static patterns, referring to one or more events that are independent of the dynamic evolvement of the peer interaction, and
(b) dynamic patterns, referring to contextual events arising from the analysis of peer utterances and the identification of certain key concepts (keywords/phrases and their synonyms), set by the teacher when building the conceptual links, i.e. the agent domain ontology.

In this section, we present a series of static and dynamic patterns, which could be detected during a colMOOC chat activity.

### 2.2 Static Intervention Strategies

Static intervention strategies arise from one or more predefined events that relate to the context of the chat activity but do not directly emerge from the agent domain model, i.e. the available conceptual links. Such intervention strategies audit interactions with the player component user interface, such as button presses, or environmental changes, such as the loss of the Internet connection. When these interventions strategies are detected, the correspondent intervention message appears in the chat dialogue activity.

#### 2.2.1 Welcome

**Table 1:** Welcome static intervention strategy description

| | |
|---|---|
| Intervention Name | Welcome |
| Intervention Strategy Version | 1.0 |

| | |
|---|---|
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | When a user connects to the chat activity, the agent provides a welcome message. |
| Purpose | Enhances group awareness regarding the users' online status. |
| Triggers When (Transaction Pattern) | A user connects to the chat activity. |
| Requires Modelling Of | User online status. |
| Transaction Pattern Example | A user enters the chat room of the activity. |
| Intervention Message Example | *"Hello [Student Name], we are waiting for another classmate to join. Please be patient!"* |

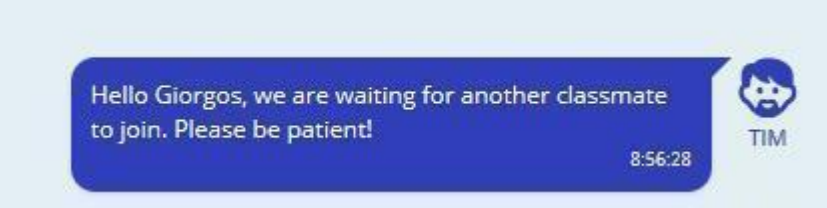


**Figure 1:** Welcome static intervention strategy example

### 2.2.2 Onboarding

**Table 2:** Onboarding static intervention strategy description

| | |
|---|---|
| Intervention Name | Onboarding |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | As soon as all group members are connected (online), the agent informs peers about the ongoing task and attempts to initiate the discussion. |
| Purpose | Contributes to the user onboarding process and can be used as an "ice-breaking" tactic. |
| Triggers When (Transaction Pattern) | Both users are connected to the chat activity. |
| Requires Modelling Of | User online status. |
| Transaction Pattern Example | The two users, who were matched with each other and belong to |

| | |
|---|---|
| | the same team, enter the chat room of the activity. |
| Intervention Message Example | *"Now that you are both connected we can start! In this collaborative assignment, you are expected to provide a joined response to the task displayed at the top. Let's begin our discussion by doing the introductions. Let me start. My name is [Agent Name] and I am passionate about collaborative learning"* |

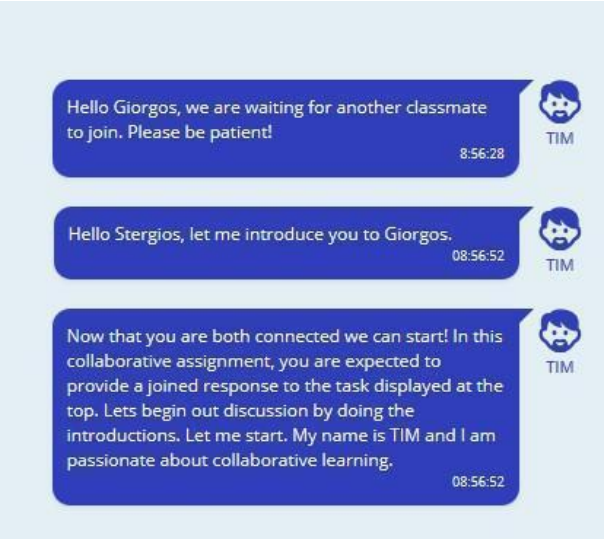


**Figure 2:** Onboarding static intervention strategy example

### 2.2.3 Disconnection

**Table 3:** Disconnection static intervention strategy description

| | |
|---|---|
| Intervention Name | Disconnection |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | In case a user gets disconnected from the chat activity (appears offline), the agent informs their partner about the disconnection event by displaying a related message. |
| Purpose | Enhances group awareness. |
| Triggers When (Transaction Pattern) | A user is disconnected from the chat activity. |
| Requires Modelling Of | User online status. |

| | |
|---|---|
| Transaction Pattern Example | A user loses connection to the Internet. |
| Intervention Message Example | *"It appears that [Student Name] was disconnected from the activity. I will let you know as soon as he/she joins back."* |


**Figure 3:** Disconnection static intervention strategy example (Giorgos' participant view)

### 2.2.4 Exit

**Table 4:** Exit static intervention strategy description

| | |
|---|---|
| Intervention Name | Exit |
| Intervention Strategy Version | 1.0 |
| Development State | To be implemented. |
| Intervention Description | If a user chooses to exit the chat activity, the agent informs their partner about the disconnection event by displaying a related message. |
| Purpose | Supports group awareness and suggests a solution to the user who is left alone in the chat room. |
| Triggers When (Transaction Pattern) | A user confirms that they would like to exit the chat activity. |
| Requires Modelling Of | User exit confirmation |

| | |
|---|---|
| Transaction Pattern Example | A user clicks on the 'X' (close) button of their activity tab in their browser and verifies, using the popup confirmation modal box, that they would like to terminate and exit their current chat activity. |
| Intervention Message Example | *"Unfortunately, [Student Name] has just decided to exit the chat activity. You can continue working in the activity and submit the task answer on your own or restart the activity with another user."* |

### 2.2.5 Re-connection

**Table 5:** Re-connection static intervention strategy description

| | |
|---|---|
| Intervention Name | Re-connection |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | The agent informs the user waiting for their partner, that the disconnected user has just returned. |
| Purpose | Enhance group awareness. |
| Triggers When (Transaction Pattern) | A disconnected team member re-connects to the chat activity. |
| Requires Modelling Of | User online status. |
| Transaction Pattern Example | A user gets disconnected but logins again into the chat activity. |
| Intervention Message Example | *"Welcome back [Student Name]. Lets continue our conversation"* |

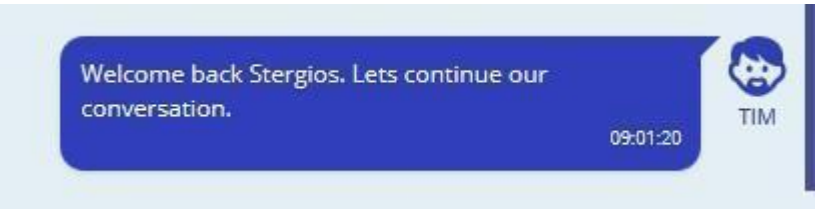


**Figure 4:** Reconnection static intervention strategy example

### 2.2.6 Task completion

**Table 6:** Task completion static intervention strategy description

| | |
|---|---|
| Intervention Name | Task completion |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | The agent informs participants about their successful task answer submission. |
| Purpose | Enhance group awareness. |
| Triggers When (Transaction Pattern) | The group clicks on the Submit Answer button and confirms the submission of the task answer. |
| Requires Modelling Of | Task answer submission confirmation. |
| Transaction Pattern Example | A user clicks on the Submit Answer button and, following the confirmation modal box that appears, the user confirms their action and submits the answer of the group. |
| Intervention Message Example | *"Nice! Your team has successfully submitted an answer to the task!"* |

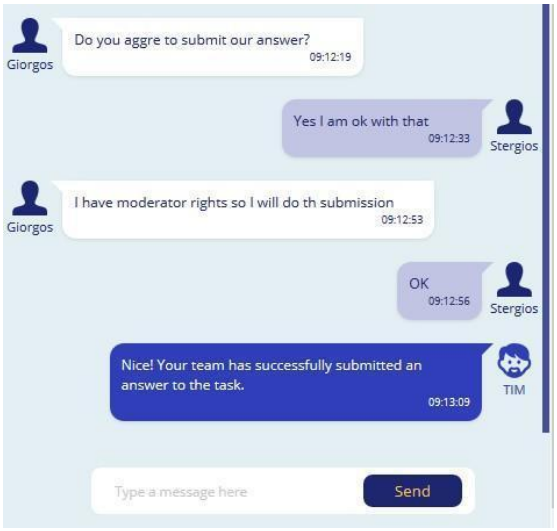


**Figure 5:** Task Completion static intervention strategy example

### 2.2.7 Help

**Table 7:** Help static intervention strategy description

| | |
|---|---|
| Intervention Name | Help |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | If a user presses the help button, the agent provides text assistance to the user. |
| Purpose | Supports group awareness regarding the task and how to complete the activity. |
| Triggers When (Transaction Pattern) | A user presses the “Help” button. |
| Requires Modelling Of | User’s Clickstream data. |
| Transaction Pattern Example | A user presses the “Help” button to ask for activity explanation |
| Intervention Example | *“Think about the agents` questions later when you try to answer the task. This will help you to provide a better task answer and deepen your understanding…”* |

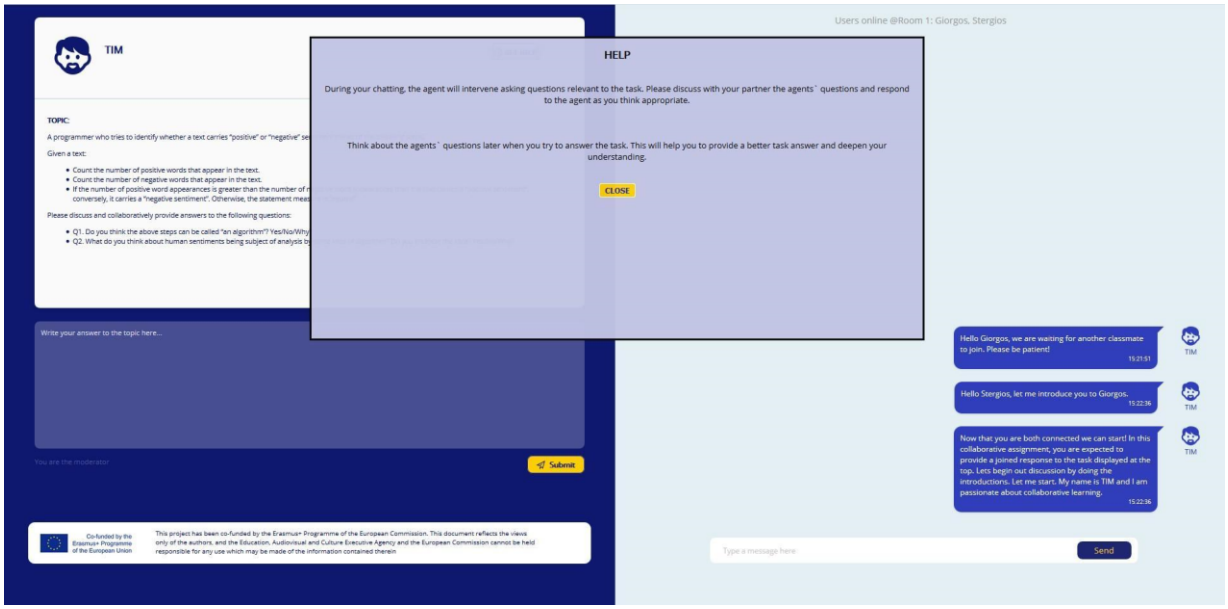


**Figure 6:** Help static intervention strategy example

## 2.3 Dynamic Intervention Strategies

A dynamic intervention strategy is typically based on a conceptual link, created in the colMOOC editor, and possibly on one or more predefined events relating to the context of the chat activity. Dynamic intervention strategies can keep track of various parameters, such as the time between subsequent agent interventions or user messages, the time spent in the activity, the number of words entered in a text answer, and many others. When a dynamic intervention pattern is detected, the correspondent intervention message appears in the chat dialogue activity.

### 2.3.1 Suggestion

**Table 8:** Suggestion dynamic intervention strategy description

| | |
|---|---|
| Intervention Name | Suggestion |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | If not event one domain concept X was discussed after 3 minutes, the agent intervenes encouraging students to discuss the key concept X. |
| Purpose | Provide guidance and encourage students to include a key domain concept in their discussion. |
| Triggers When<br>(Transaction Pattern) | None of the available domain concepts were mentioned 3 minutes after the initiation of the chat activity. |
| Requires Modelling Of | - Domain concepts already discussed (chat history)<br>- Time passed in the activity |
| Requires Conceptual Link(s) | With 1 concept<br>*[Concept A]* |
| Transaction Pattern Example | *"[10:12:55] Agent: Welcome to the chat activity …*<br>*[10:13:04] Giorgos: Hi, Stergios*<br>*[10:13:11] Stergios: Hello Giorgos*<br>*…*<br>*[10:15:51] Giorgos: OK I am ready"* |
| Intervention Message Example | *"[10:15:55] Agent: You didn't discuss a lot about [Concept A]. Is there anything else you wanna mention that could help your team advance in the task?"* |

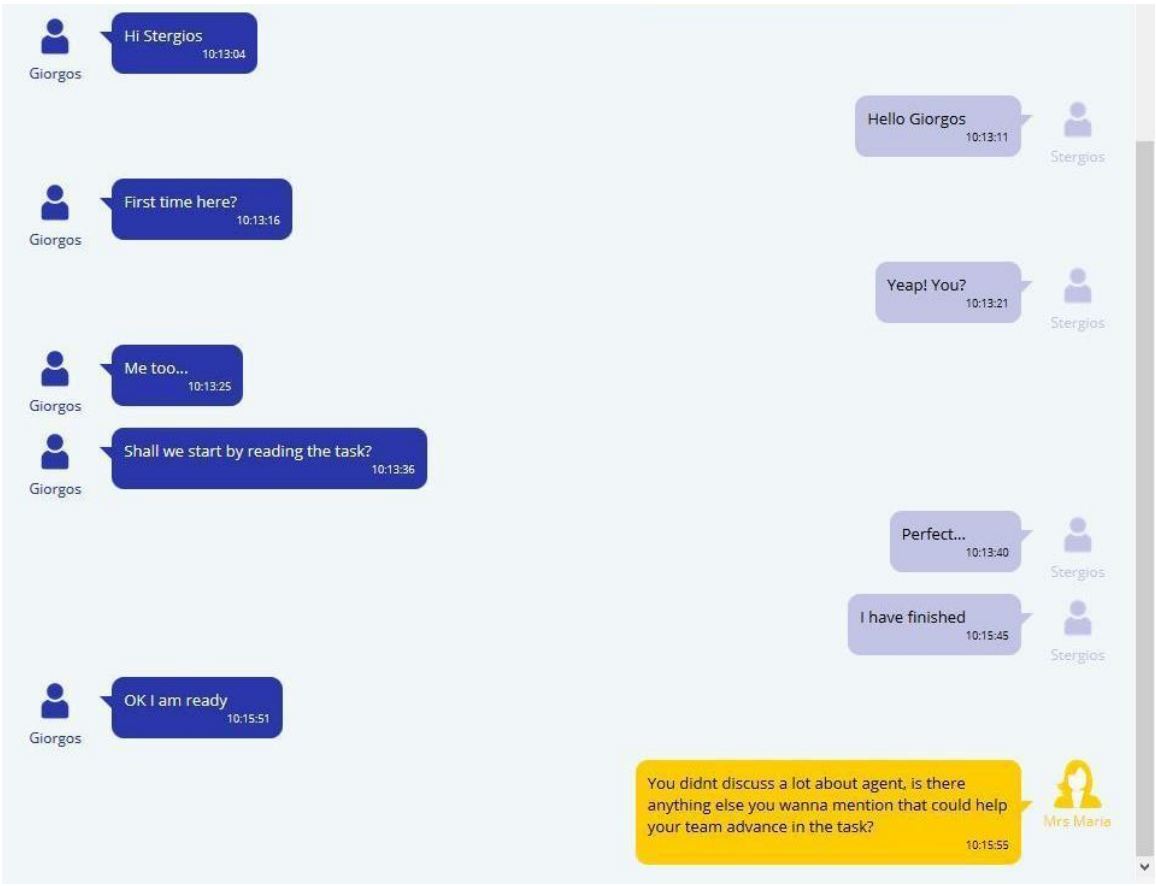


**Figure 7:** Suggestion dynamic intervention strategy example

### 2.3.2 AddOn

**Table 9:** AddOn dynamic intervention strategy description

| | |
|---|---|
| Intervention Name | AddOn |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | After a student introduces a particular concept into the dialogue and their partner have remained silent or provided a very short response within 10 seconds, the agent intervenes asking their partner if he/she has anything to add on that specific concept. |
| Purpose | - Prompt a student for further participation and encourage them to explicate their thoughts on a domain concept introduced by their partner.<br>- Support accountability to the Learning Community. |
| Triggers When<br>(Transaction Pattern) | 10 seconds after a domain concept was introduced by a student, their partner has either remained silent or sent a short reply, consisting of 3 or less words. |
| Requires Modelling Of | - Domain concept detected in last student utterance<br>- Domain concepts already discussed (chat history)<br>- 10-second monitoring of partner's actions (detection of utterance size) |
| Requires Conceptual Link(s) | With 1 concept<br>*[Concept A]* |
| Transaction Pattern Example | *"[10:32:28] Student A: I think [Concept A] is really important since...*<br>*[10:32:32] Student B: ok"* |
| Intervention Message Example | *"[10:32:38] Agent: [Student A], would you like to add something to what your partner [Student B] said about [Concept A]?"* |

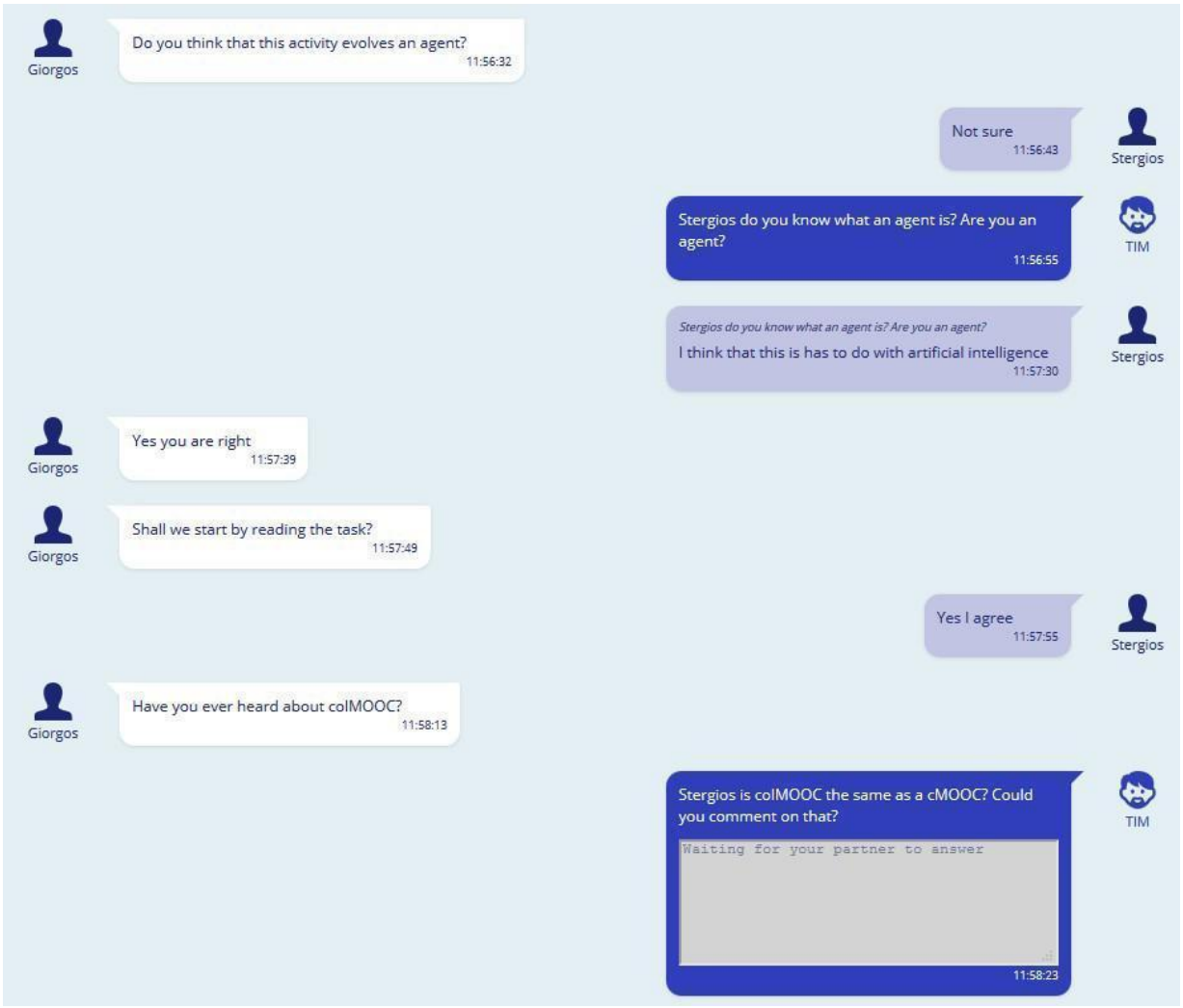


**Figure 8:** AddOn dynamic intervention strategy 1st example (triggered by the concept "colMOOC")


**Figure 9:** AddOn dynamic intervention strategy 2nd example (triggered by the concept "agents")

### 2.3.3 Building on prior knowledge

**Table 10:** Building on prior knowledge dynamic intervention strategy description

| | |
|---|---|
| Intervention Name | Building on prior knowledge |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | The agent detects that a single concept is being discussed and intervenes in the discussion asking to connect this concept to another relevant concept, which is part of the same conceptual link. |
| Purpose | - Encourage students to tie a contribution to another relevant concept of the agent domain model.<br>- Support accountability to knowledge. |
| Triggers When<br>(Transaction Pattern) | A domain concept, which is linked with another concept in the agent domain model, is introduced into the discussion. |
| Requires Modelling Of | - Domain concept detected in last student utterance<br>- Domain concepts already discussed (chat history) |
| Requires Conceptual Link(s) | With 2 concepts<br>*[Concept A + Relationship + Concept B]* |
| Transaction Pattern Example | *"[10:35:10] Student A: At this point, I guess we could also discuss [Concept A] …"* |
| Intervention Message Example | *"[10:35:11] Agent: "Do you think [Concept A] is somehow related to [Concept B]? How?"* |

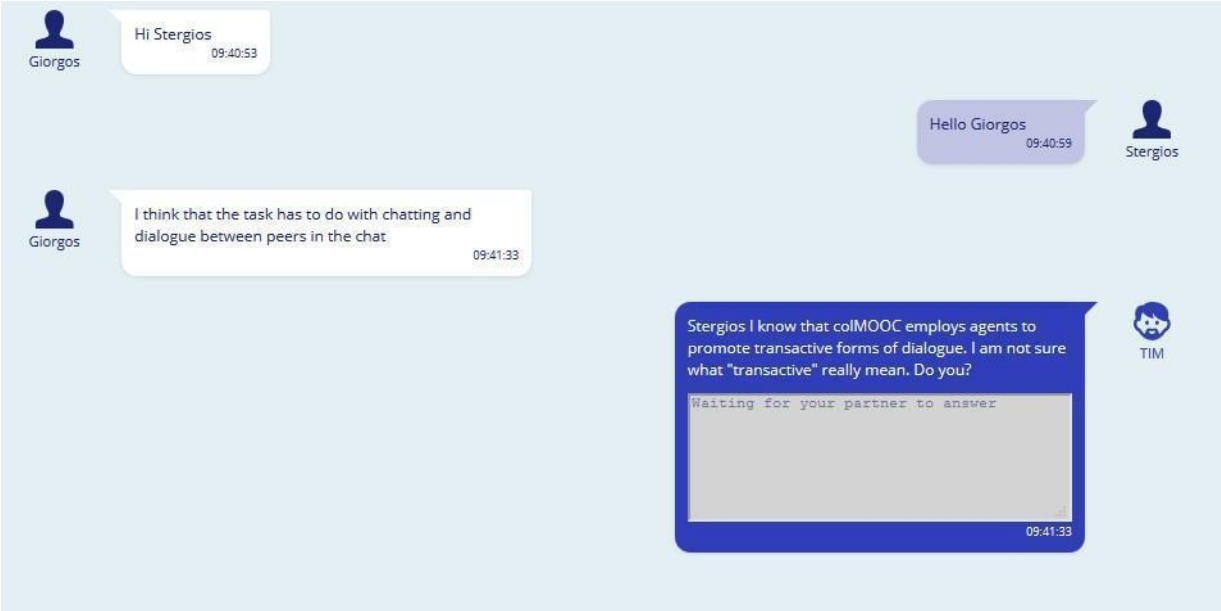


**Figure 10:** Building on prior knowledge dynamic intervention strategy 1st example (triggered by the concept "dialogue" – Giorgos' participant view)

The second agent intervention, depicted in the example below, is triggered by Giorgos' utterance of "agent".

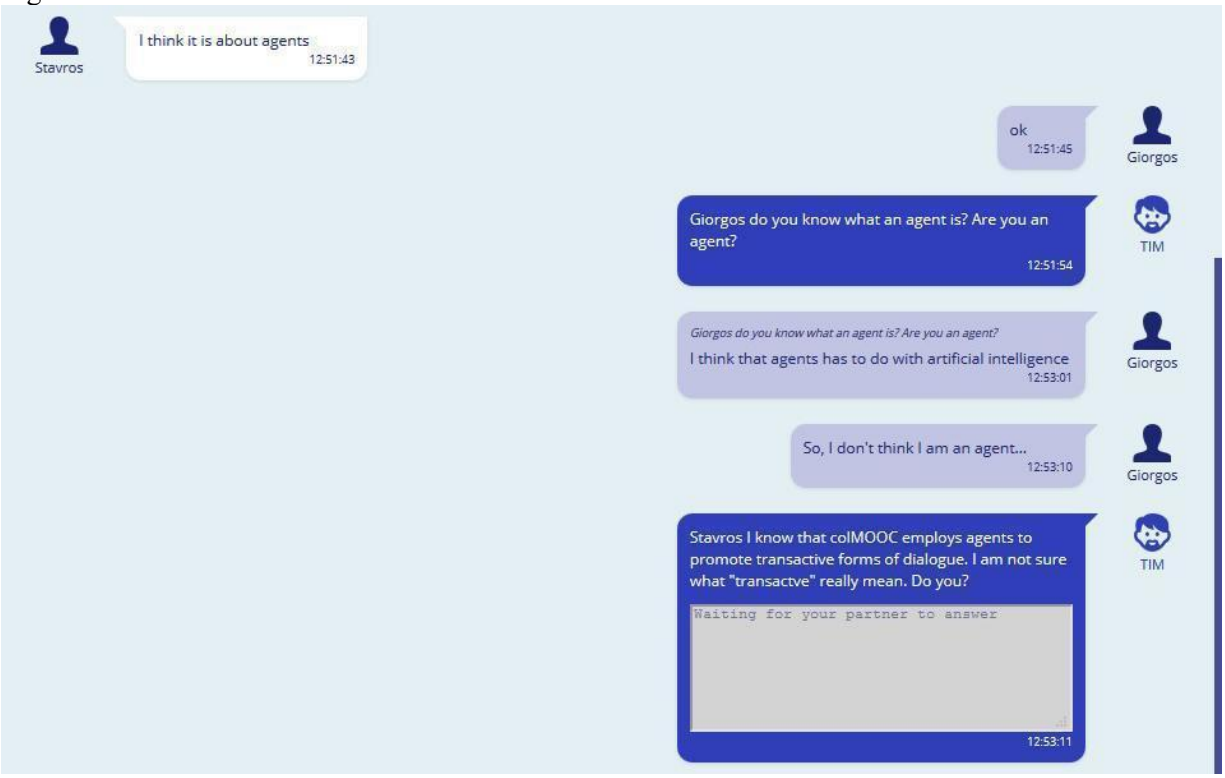


**Figure 11:** Building on prior knowledge dynamic intervention strategy 2nd example (triggered by the concept "agent" – Giorgos' participant view)

Another example showing an agent intervention triggered by Stavros' utterance of "MOOCs".

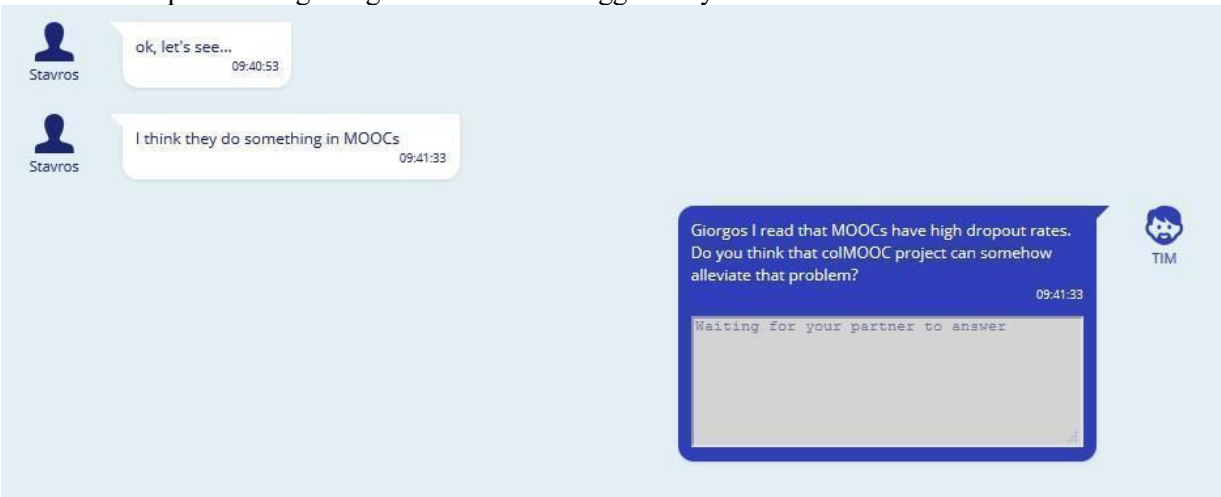


**Figure 12:** Building on prior knowledge dynamic intervention strategy third example (Stavros' participant view)

### 2.3.4 Verifying

**Table 11:** Verifying dynamic intervention strategy description

| | |
|---|---|
| Intervention Name | Verifying |
| Intervention Strategy Version | 1.0 |
| Development State | Implemented in colMOOC player v.0.6. |
| Intervention Description | If both concepts of a conceptual link are simultaneously introduced in the discussion, the agent intervenes asking students to comment on the relation the two concepts. |
| Purpose | - Encourage students to verify or clarify a relevant agent contribution, thus helping both learners to engage more profitably in the conversation.<br>- Support accountability to the learning community. |
| Triggers When (Transaction Pattern) | Both concepts of a conceptual link are detected in a student utterance. |
| Requires Modelling Of | - Domain concepts detected in last student utterance<br>- Domain concepts already discussed (chat history) |
| Requires Conceptual Link(s) | With 2 concepts<br>*[Concept A + Relationship + Concept B]* |
| Transaction Pattern Example | *"[10:38:04] Student A: The [Concept A] and [Concept B] are somehow connected here."* |

| | |
|---|---|
| Intervention Message Example | *"[10:38:05] Agent: "Do you both agree with the following statement: [Concept A + Relationship + Concept B]? Why?"* |

In the example presented below, the agent intervention is triggered by Giorgos' utterance of "project + collaborative learning".

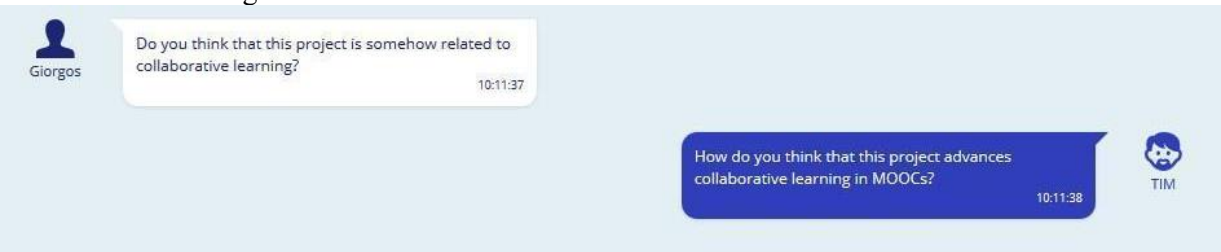


**Figure 13:** Verifying dynamic intervention strategy example

### 2.3.5 Ask for explanation

**Table 12:** Ask for explanation dynamic intervention strategy description

| | |
|---|---|
| Intervention Name | Ask for explanation |
| Intervention Strategy Version | 0.1 |
| Development State | To be implemented. |
| Intervention Description | Agent intervenes after detecting that a single concept had been introduced by a single student while the other student has only expressed a single agreement/disagreement (without providing a further explanation). |
| Purpose | - Supports accountability to rigorous thinking.<br>- Push for clear statements of claims (positions and explanations) and sound reasoning in backing up claims with evidence. |
| Triggers When<br>(Transaction Pattern) | After a single concept was introduced by a student, there is a simple agreement or disagreement - consisting of 3 or less words - expressed by their partner. |
| Requires Modelling Of | - Domain concepts detected in last student utterance<br>- Domain concepts already discussed (chat history)<br>- 10-second monitoring of partner's actions (simple agreement/disagreement detection) |
| Requires Conceptual Link(s) | With 1 or 2 concepts<br>*[Concept A]* or *[Concept A + Relationship + Concept B]* |
| Transaction Pattern Example | *"[10:41:20] Student A: The first principle of [Concept] says that ...*<br>*[10:41:29] Student B: I agree"* |
| Intervention Message Example | *"[10:41:30] Agent: [Student B], why do you agree with what your partner said about [Concept A/B]?"* |

### 2.3.6 Agree/disagree

**Table 13:** Agree / disagree dynamic intervention strategy description

| | |
|---|---|
| Intervention Name | Agree / disagree |
| Intervention Strategy Version | 0.1 |
| Development State | To be implemented. |
| Intervention Description | Agent intervenes after detecting that a student explains a concept, while the other student remains silent or provides an oversimplified response. |
| Purpose | - Support accountability to the learning community |
| Triggers When<br>(Transaction Pattern) | Following a domain concept is introduced and explained by a learner (in a long message), their partner remains silent or provides a very short response, consisting of 3 or less words. |
| Requires Modelling Of | - Domain concepts detected in last student utterance<br>- Domain concepts already discussed (chat history)<br>- 10-second monitoring of partner's actions (detection of utterance size) |
| Requires Conceptual Link(s) | With 1 or 2 concepts<br>*[Concept A]* or *[Concept A + Relationship + Concept B]* |
| Transaction Pattern Example | *"[10:45:00] Student A: [Concept A/B] has to be or usually is to be followed, or can be desirably followed, or is an inevitable consequence of something …"* |
| Intervention Message Example | *"[10:45:10] Agent: [Student B], what do you think of what [Student A] said about [Concept A/B]? Do you agree or disagree?"* |

### 2.3.7 Reminder

**Table 14:** Reminder dynamic intervention strategy description

| | |
|---|---|
| Intervention Name | Reminder |
| Intervention Strategy Version | 0.1 |
| Development State | To be implemented. |
| Intervention Description | The agent intervenes before the group submits their final answer to the task and reminds peers that there are still some concepts not discussed. |
| Purpose | Conceptually enrich students' discussion and answers by encouraging students to expand their dialogue on other domain concepts as well. |
| Triggers When<br>(Transaction Pattern) | One or more concepts were not discussed yet and a student decides to initiate the group answer submission process by clicking on the available Submit button. |
| Requires Modelling Of | - Task answer submission<br>- Domain concepts already discussed (chat history) |
| Requires Conceptual Link(s) | With 1 or 2 concepts<br>*[Concept A]* or *[Concept A + Relationship + Concept B]* |
| Transaction Pattern Example | The user clicks on the SUBMIT button in an attempt to submit the group answer to the task. |
| Intervention Message Example | *"Agent: I would like to remind you that you could also to talk about [Concept A/B] … Please note that your team answer has not been submitted yet. If you want to submit your team answer, click again the Submit button and confirm your submission."* |

# 3 Conversational Agent Module Deployment

## 3.1 Software development life cycle methodology

In software engineering, a system development methodology is regarded as a framework that is used to structure, plan, and control the process of developing an information system.

To design and develop the colMOOC conversational agent module, the project research methodology was based on an agile software development life cycle model (SDLC), which is often used in the learning sciences to design technological or pedagogical interventions (Isaias & Issa, 2015). Many alternative approaches have been developed in the past in order to counter the rigidness and lack of flexibility found in waterfall-like SDLC models. Along with the rise of the incremental and rapid application development (RAD) models, a manifesto for agile software development was introduced by software developers, aiming to bring together the best traits of other agile-like models into one framework. Following this, agile methods of development have become increasingly popular (Bhalerao et al. 2009). Nowadays, there are numerous variations of the agile model sharing the same principles and traits, such as the emphasis on simplicity, the prioritization of user satisfaction, the frequent requirements updates and software deliveries, the existence of self-organizing project teams and the regular discussions that take place emphasizing team improvements.

The agile SDLC was selected because of its strengths, such as the increased flexibility and development speed offered. In general, most agile methodologies have emerged from the increasing need to match the speed at which technology evolves. A key aspect setting it apart is its dexterity in developing software at high speed, with software products being deliverable in weeks instead of months. This is possible due to the model emphasis on collaborative efforts and documentation. Another main advantage of the agile model is its increased flexibility since it can be combined with other existing models. It has the capacity to deliver systems whose requirements go through constant changes while, at the same time, demanding strict time limits.

Furthermore, the agile model is often praised for its high degree of user satisfaction and user-friendliness, reduced error margins, and the ability to incorporate solutions to address the needs of mutable requirements. In that manner, it is considered a client-centric model, which advocates “short iterations and small releases” in order to obtain feedback on what has been accomplished. With the feedback that is received, improvements can be made that will have positive repercussions on the quality of the end product (Bhalerao et al. 2009).

Despite the variations in timescales or stage description, the general path that an agile development process will take can be outlined in concrete steps. Most agile methods attempt to minimize risk by developing software in short timeboxes, called iterations, which typically last one to four weeks. Each iteration is like a miniature software project of its own and includes all the tasks necessary to release the mini-increment of new functionality: planning, requirements analysis, design, coding, testing, and documentation. While iteration may not add enough functionality to warrant releasing the product, an agile software project intends to be capable of releasing new software at the end of every iteration. At the end of each iteration, the team reevaluates project priorities.

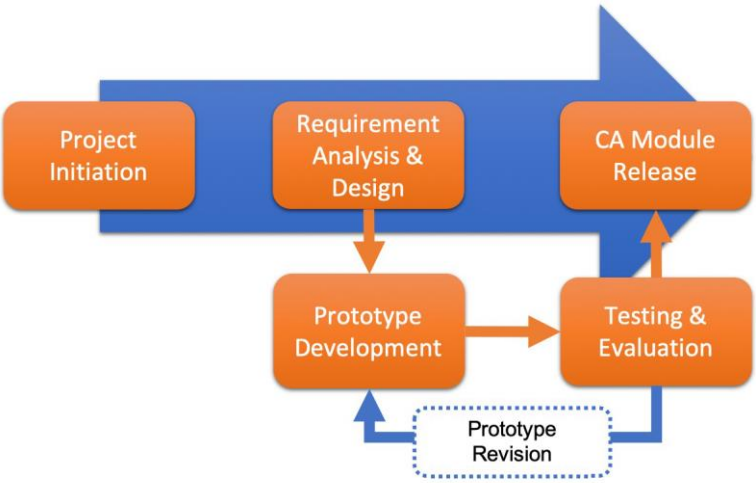


**Figure 14:** The agile SDLC process

In the heart of our Agile SDLC (Figure 14), there was also an evolutionary prototyping internal process. This process was selected as a means of keeping the risks as low as possible while striving for a high probability of success. The evolutionary approach describes development in successive versions, similar to the incremental and iterative SDLCs. In our project, the primary goal of our methodology was to accommodate the eventual changes in requirements and needs, addressing potential technological and scientific risks. With each iteration, or version of the colMOOC agent, the project prototype was tested in order to provide valuable input on its traits and requirements. In that manner, the conversational agent module has evolved throughout the entire development phase and, at the same time, there was a continuous effort for keeping the time frame short, in accordance to the project roadmap.

## 3.2 colMOOC agent editor

Following the system architecture introduced in D.2.1, this section presents details about the colMOOC editor. The editor constitutes a core system component, which is used by teachers to create chat-based activities that can be incorporated into a MOOC. Teachers can deploy a colMOOC chat activity in the MOOC of their choice. In case their MOOC is colMOOC-compatible, the aforementioned process is very similar to inserting other course activities, such as videos, forum discussions, and quizzes.

The colMOOC editor consists of the following steps: (A) Agent Creation/Editing, (B) Topic and Task Setup, and (C) Domain Configuration.

In the initial step (Figure 15), teachers have the option to choose between editing a previously developed agent by giving the agent ID and creating a new one. All activity configuration files of the agents that have been created are stored on the colMOOC server and can be accessed by their unique IDs. If teachers choose to edit a previously developed agent, all the fields of the editor are pre-populated with data that emerge from the activity configuration file. This data can be further altered by updating the agent details.

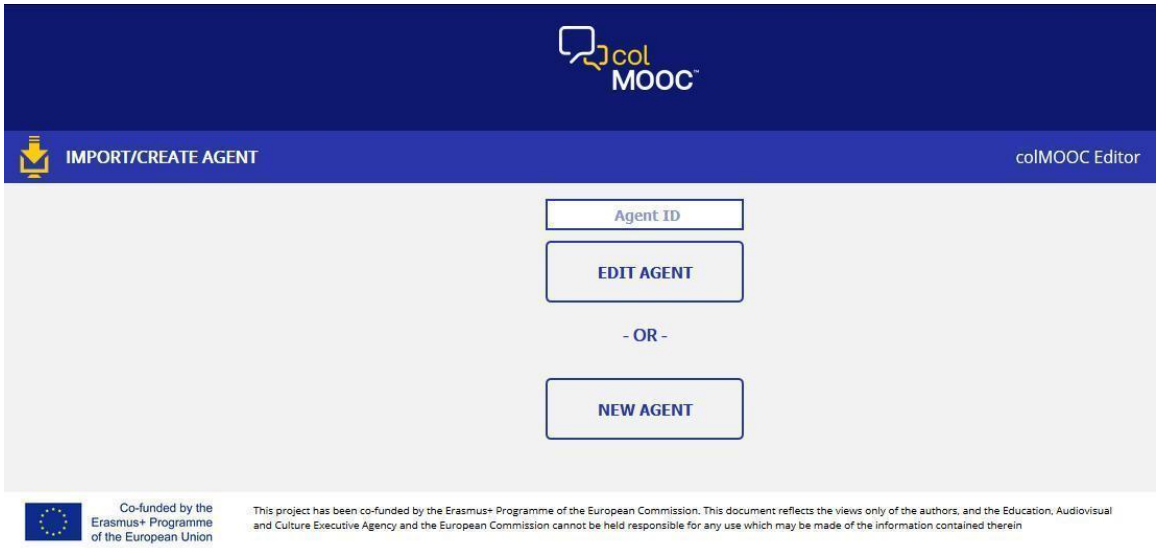


**Figure 15:** The first step of the editor wizard: Edit agent or create a new agent

In the following step, teachers can set up the appearance of the agent, the task/topic details, and enter some information as an activity guideline (Figure 16). The appearance of the agent includes details regarding the agent name, the corresponding avatar, and the language of the agent. The colMOOC project currently supports 4 languages to customize the interface: English, German, Greek, and Spanish The language setting is further used by the player to customize all user interface messages to the corresponding language. Also, the language setting affects the process of producing the correspondent questions for each conceptual link that teachers create on the last step of the activity creation process.

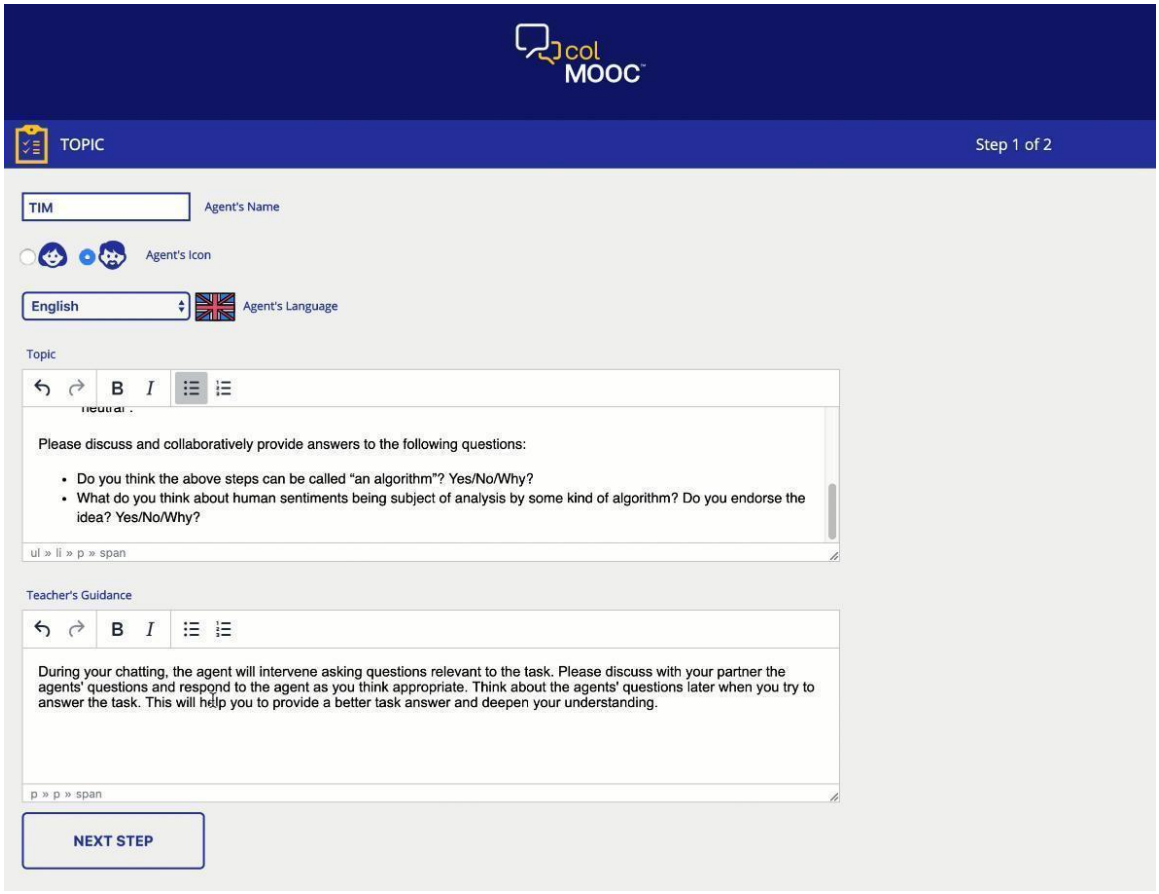


**Figure 16:** The second step of the editor wizard: Agent settings, topic and teacher guidance

Teachers can use the rich-text editor in order to configure the topic of the chat-based activity. This discussion topic is essentially an open-ended domain question, which encourages students to discuss with each other during their respective course activity. After discussing, students should collaboratively try to answer this question in the associated answer area, which can be located in the colMOOC player interface. To assist students with the chat activity, teachers can also provide some additional information in the Teacher's Guidance field, serving as learners' guidance in the activity. This information can be accessed by the students in a popup window, which can be displayed in the colMOOC player.

A key function of the agent editor is that teachers can use conceptual links (CLs) to shape the domain model that this specific activity refers to. Conceptual links serve as a representation of the domain, which comprises concepts related to the task, and linking verbs or modifiers that associate concepts (Figure 17). In general, conceptual links are strictly related to the task terms that the teacher considers essential for students to argue on before answering the task.


**Figure 17:** The last step in the colMOOC editor wizard allows teachers to shape the agent domain model by creating conceptual links

Two types of conceptual links are supported at the moment: (a) single-concept and (b) two-concept links, consisting of two concepts (nodes) that are connected using a 'verb' or 'modifier'. Regarding the verb or modifier part, teachers can decide to either select one of the pre-defined relations or enter a new one by typing it. Multiple conceptual links can be created in order to shape the agent domain for the specific activity.

Each conceptual link created dynamically generates one or more questions, which can be further edited by the teacher. Let us take an example where a teacher enters the conceptual link: "computer program can measure text sentiment". The two concepts are "computer program" and "text sentiment", while their connection is expressed by "can measure". Teachers can press the "down arrow" link, residing at the left of the conceptual link in order to review all the respective questions that are generated by the system automatically. These are basically the questions that the colMOOC agent may display during a peer dialogue via its intervention mechanism (Figure 18).

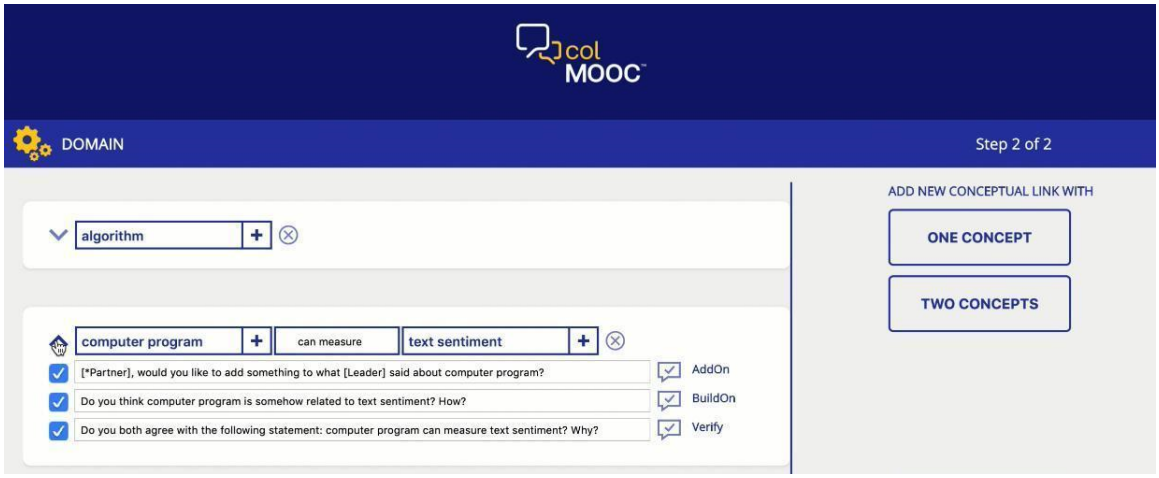


**Figure 18:** An example of agent questions generated following the creation of a conceptual link

Teachers can exclude a question from the list of the derived ones by deselecting the correspondent check button. In case they do so, the system deactivates the specific question, which will not appear in colMOOC player even when the associated intervention pattern is detected. Also, there are some situations where the generated questions might seem linguistically incorrect requiring some

improvement. To overcome similar issues, teachers can proceed to edit the questions proposed by the system (Figure 19).


**Figure 19:** Excluding and editing the automatically generated agent questions (interventions), and adding [*Parner] and [Leader] annotations

Furthermore, teachers can choose whether an answer box will be visible to students or not when the specific intervention fires, by toggle the corresponding bubble image at the end of each question. This answer box is displayed in the colMOOC player and asks students to write down their answer to the agent question. In case the answer box is disabled, such as in the case where the agent displays a comment instead of a question, students can take into consideration the agent message without having to answer directly to it.

Teachers can also use some simple markup language to further enhance the functionality of the agent intervention mechanism. More specifically, they can enter the asterisk wildcard or the square brackets annotation in the agent questions, which arise from the conceptual links. In this manner, the agent messages can be directed to either a specific learner of the team or the whole team. In a peer discussion, the system makes the assumption that two kinds of users can be identified (Figure 19):

- The “active” users, which are the users that usually trigger the intervention because they mention a key concept from a conceptual link. Teachers can use the “[Leader]” square bracket annotation to refer to this kind of users.
- The “inactive” users, which are the users that are not very talkative in the chat or their participation is very limited. Teachers can use the “[Partner]” square bracket annotation to refer to this kind of users.

For instance, the first question of the “computer program can measure text sentiment” conceptual link is: “[*Partner] would you like to add something to what [Leader] said about computer program?”. In this question, two bracket annotations are used. During a chat activity, these brackets are replaced in real-time by the actual nicknames of the “inactive” and “active” users respectively. The asterisk wildcard indicates the direction of the question. In the example mentioned here, the question is addressed to the inactive user ([Partner] annotation). This means that the answer box inside the intervention will be active only for the “inactive” user, while the “active” user ([Leader] annotation) will see the answer box disabled.

A student role may change multiple times during a chat activity; it may change each time an intervention pattern is detected. If a student triggers an agent intervention, he/she becomes the

"Leader", but only for the specific agent intervention. In case their partner triggers the next agent intervention their role instantly changes to "Partner".

There are some situations that students during their dialogue could refer to a concept by a similar expression in a peer dialogue. In this case, the intervention will not be fired because there is not an exact match between the concept of the conceptual link that teachers constructed and the concept that peers mentioned in their dialogue. To overcome this issue, teachers can enter a list of synonyms for each concept (Figure 20). To provide an example, let us take for example the term "computer program". For this concept, a synonym could be the concept of "software". If the peer dialogue parser detects that peers are discussing "software", the system may trigger the corresponding intervention. It should be noted that although an agent intervention may be triggered by the synonym concept, the main concept entered by the teacher will be used in the agent question. Teachers can enter as many synonyms as they want separated by commas.

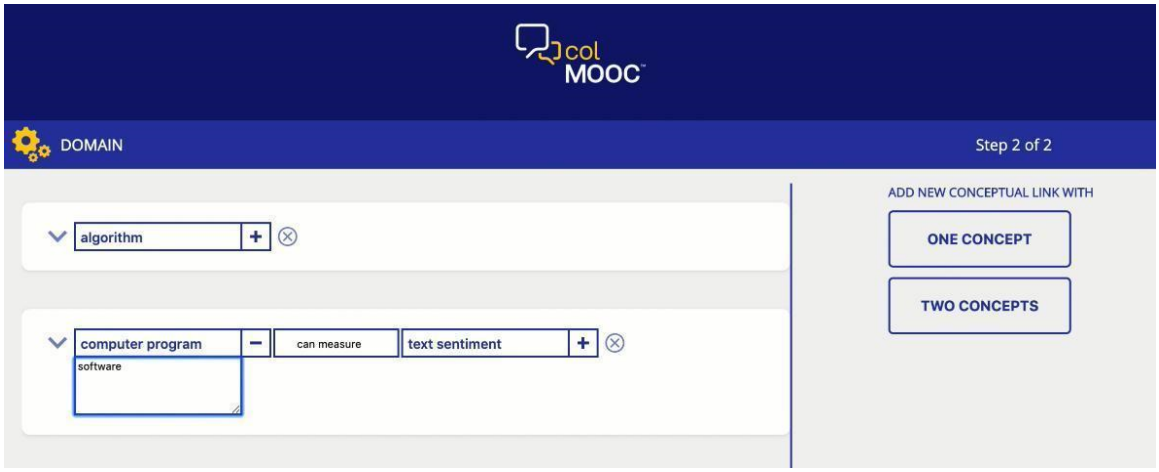


**Figure 20:** Entering synonyms for a concept in a conceptual link

The conceptual links created by teachers serve as an input to the "generator" system component of the colMOOC editor (Figure 21). The generator consists of the "APT Q-Stems" and the "language adapter" sub-components. The APT Q-Stems sub-component contains the blueprint to construct T-Questions (transactive questions). For example, an APT Q-Stem could be "[Student B], would you like to add something to what your partner said about [Concept]?". Having this as a blueprint, the generator creates the T-Qs that the agent can pose to the peers during their discussion, by replacing the placeholders in the Q-Stem with the actual concept. The 'Language Adapter' sub-component is responsible for linguistic processing, which is necessary to produce grammatically and syntactically correct questions in any supported language.

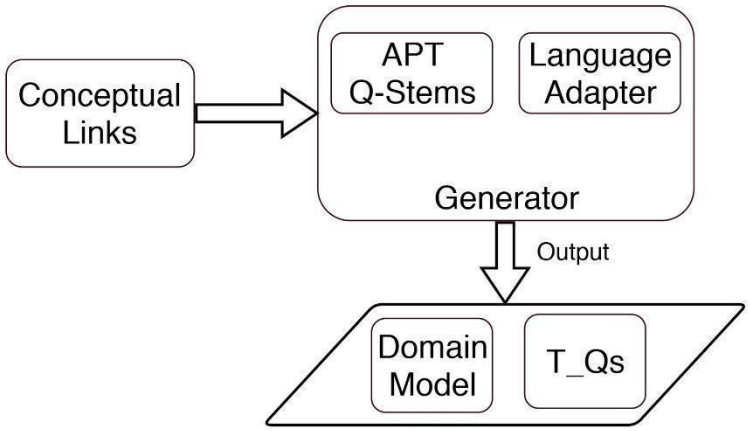


**Figure 21:** The generator component (part of the editor)

In the next few tables, we will analyze the use case examples of the colMOOC editor. We will describe how teachers can interact with the colMOOC editor to set up the activity.

**Table 15:** Create a new activity use case

| | |
|---|---|
| Use case name | Create a new activity |
| Brief description | The teacher creates a new chat-based colMOOC activity to embed it in a specific MOOC |
| Triggers | The teacher presses the "New Agent" button located at the first step of the editor's wizard |
| Actors | Teacher |
| Main flow | 1. The teacher accesses the MOOC platform and chooses to insert a new chat-based activity in the MOOC<br>2. The teacher presses the "New Agent" button to start creating a chat-based activity from scratch |
| Alternative flow | - |

**Table 16:** Load an activity use case

| | |
|---|---|
| Use case name | Load an activity |
| Brief description | The teacher loads a previously created agent into the current activity |

| | |
|---|---|
| Triggers | The teacher presses the "Edit Agent" button located at the first step of the editor's wizard |
| Actors | Teacher |
| Main flow | 1. The teacher accesses the MOOC platform and chooses to insert a new chat-based activity in the MOOC<br>2. The teacher enters the ID of the agent that wants to load to the activity<br>3. The teacher presses the "Edit Agent" button to alter the settings of the agent |
| Alternative flow | - |

**Table 17:** Setup an activity use case

| | |
|---|---|
| Use case name | Set up an activity |
| Brief description | The teacher navigates through the "topic" and "domain" steps of the agent and fills all the required fields to set up the agent behavior |
| Triggers | The teacher presses the "New Agent" or "Edit Agent" buttons located at the first step of the editor's wizard |
| Actors | Teacher |
| Main flow | 1. The teacher presses the "New Agent" or "Edit Agent" buttons to create or load respectively a chat-based activity<br>2. The teacher chooses the avatar, the agent name, and the agent language.<br>3. The teacher fills the "topic" text area with the main subject that students should discuss during the activity<br>4. The teacher fills the "teacher's guidance" text area with instructions regarding the activity<br>5. The teacher presses the "Next step" button to move to the next step<br>6. The teacher sets up the "Conceptual links"<br>a. The teacher adds synonyms to each concept (if necessary)<br>7. The teacher presses the "down arrow" button to see the produced questions based on the specific conceptual links<br>8. The teacher edits the questions and/or selects which questions will be used by the agent to interventions<br>a. The teacher clicks inside the questions and makes the appropriate changes<br>b. The teacher uses the "[Leader"] and "[Partner"] annotations as placeholders inside the questions |

| | |
|---|---|
| | c. The teacher uses the asterisk annotation to direct the intervention to a specific peer<br>d. The teacher uses the toggle textbox choice to enable or disable the answer box of the intervention<br>9. The teacher presses the "One concept" or "Two concepts" buttons to add other conceptual links into the domain<br>10. The teacher presses the "Save agent" button to save the configuration to the server |
| Alternative flow | 1. The teacher moves to the previous step<br>a. The teacher presses the "Previous step" button to move to the previous step of the wizard<br>2. The teacher does not fill the "topic" text area<br>a. The teacher does not fill the "topic" area and presses the "Next step" button<br>b. The editor component displays an error message that this field should be filled before continuing to the next step and does not navigate to the next page<br>3. The teacher does not fill the "teacher's guidance" text area<br>a. The teacher does not fill the "teacher's guidance" area and presses the "Next step" button<br>b. The editor component displays an error message that this field should be filled before continuing to the next step and does not navigate to the next page<br>4. The teacher does not fill the "agent name" text field<br>a. The teacher does not fill the "agent name" text field or provides a name consisted of 2 or fewer letters, and presses the "Next step" button<br>b. The editor component displays an error message that this field should be filled before continuing to the next step and does not navigate to the next page<br>5. The teacher tries to delete a conceptual link<br>a. The teacher presses the "delete" choice next to a conceptual link<br>b. The editor component displays a message to confirm the intent<br>c. The editor deletes the conceptual link on the positive confirmation or returns to the previous screen on the negative confirmation. |

### 3.3 colMOOC Agent Player

The colMOOC player component is responsible for providing a chat-like environment for the learners. To construct this environment, the player takes as input an 'activity configuration file', which includes all the relevant activity setup information entered in the colMOOC editor. The 'peer dialogue parser' component (Figure 22) interferes between the domain and intervention models, and parses the conversation between peers, trying to identify single concepts or linked concepts included in the domain model and compute a probability that peers discuss a relevant to a conceptual link concept. When this probability exceeds a hardcoded in the agent threshold a trigger

is sent to the "intervention model". The intervention model is aware of the rules to intervene n the peer's chat. Such rules may be relevant to time trackings, such as time between interventions or idle time between chat messages, relevant to chat history, or relevant to domain concepts identification and tracking. If all conditions for intervention are met, the intervention model chooses a question from the T_Qs and post it to the chat dialogue. Intervention model decides which question is the most dominant to be used by checking some criteria. Those criteria are previous negative feedback on the question, the direction of the question, and question's complexity (more preferable questions that include more concepts).

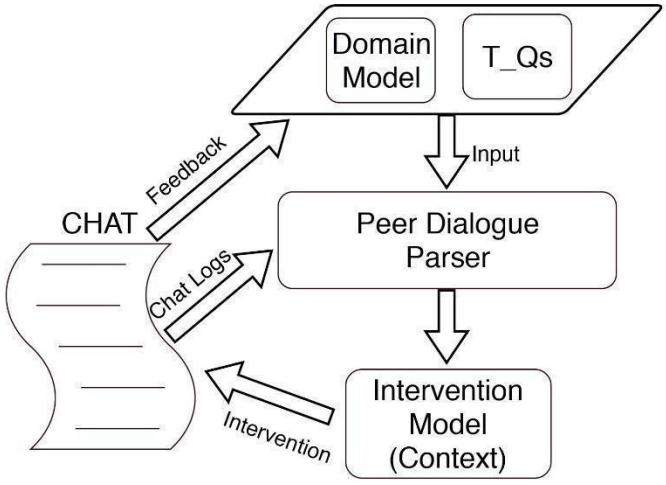


**Figure 22:** The player component

Chat activity can be found in the context of a MOOC, in line with other activities in the course. Students can visit the chat activity in the MOOC and start it from the link in the description page of the activity (Figure 23).

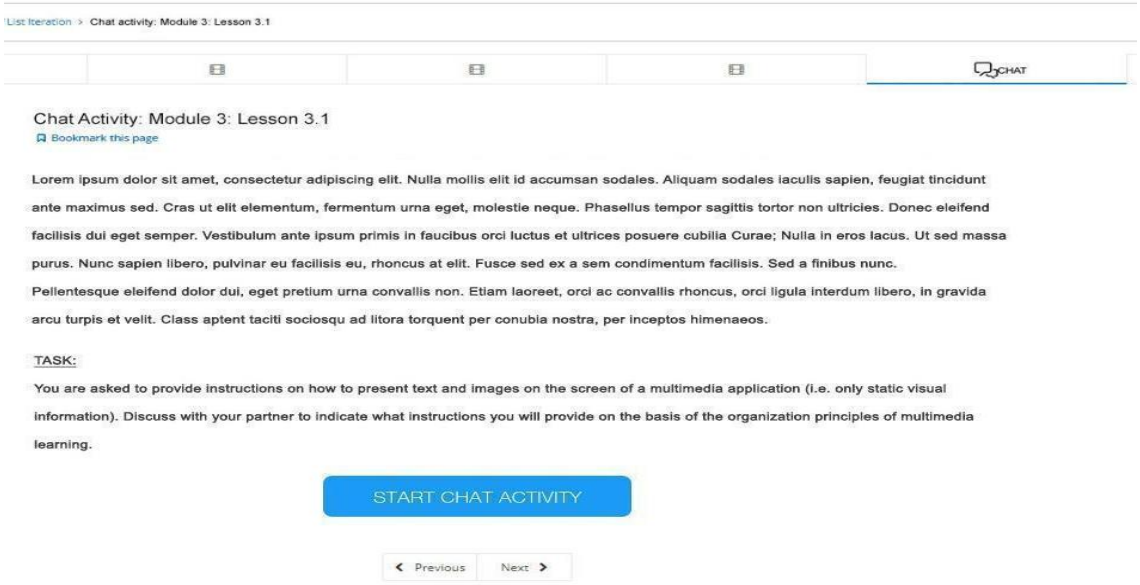


**Figure 23:** A link in the MOOC to start the chat activity

While entering the activity, students enter a system queue, waiting to be paired with a peer in order to initiate the collaborative activity. After peer matching is completed, both students enter the chat-based activity that displays an open-ended question (Figure 24). Students can read their task and begin chatting to collaboratively answer to the aforementioned question.

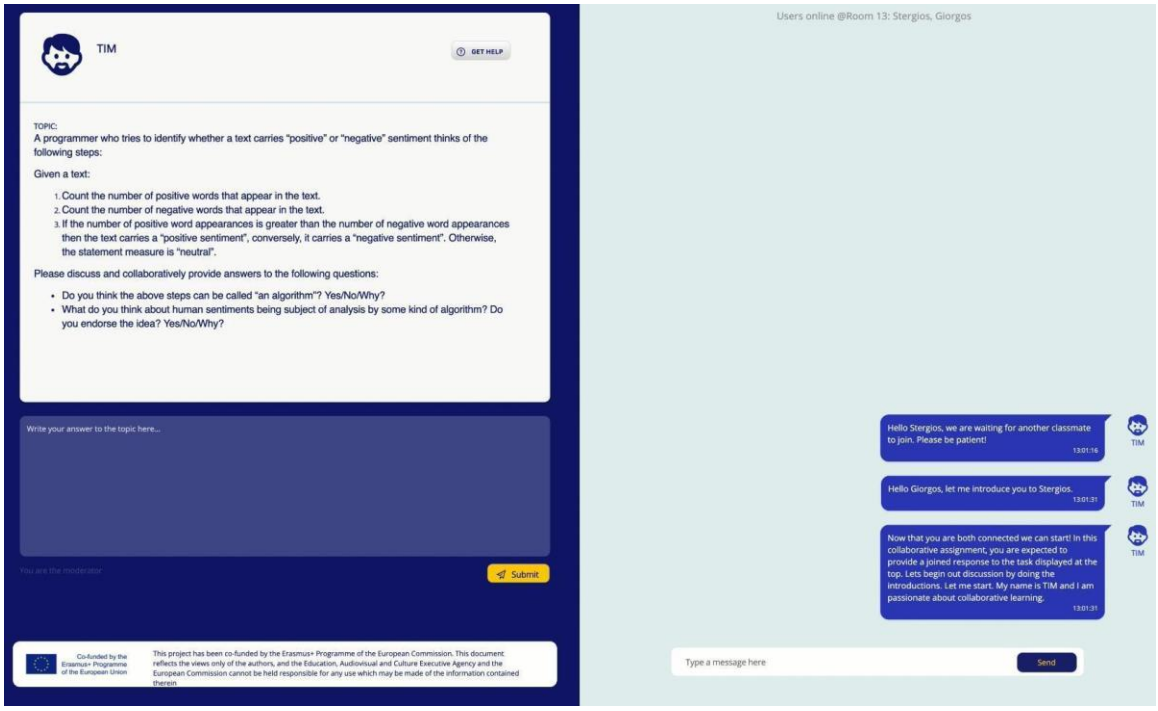


**Figure 24:** Chat activity home screen (player component)

The colMOOC player interface consists of the following parts:

- The colMOOC agent avatar. This avatar, which can be used to customize the agent appearance, was set by the teacher component during the activity set up in the colMOOC editor. Students can click on the "Get help" button, next to the agent avatar, to display the guidelines and instructions entered by the teacher for the specific activity.
- The task activity. The teacher has set the task activity during the agent setup in the editor by defining an open-ended question. This basically sets the main topic of discussion for the chat activity.
- The task answer box. Students can use this area to submit their answer to the task. The users who are assigned the 'moderator' role are responsible for writing and submitting their team answer, while their partners are able to audit what is written in realtime. After discussing, agreed on, and completing the answer to the task, moderators can press the "submit" button to submit their result.
- The discussion area. This area is used for peer chatting as well as for the interaction with the conversational agent. Every chat message (bubble) includes the user avatar and the corresponding message. Students compose their message using the text area at the bottom of their screen.

Figure 25 demonstrates the user interface of the colMOOC player component. The figure illustrates a chat-based activity with various messages between peers in the chat area.

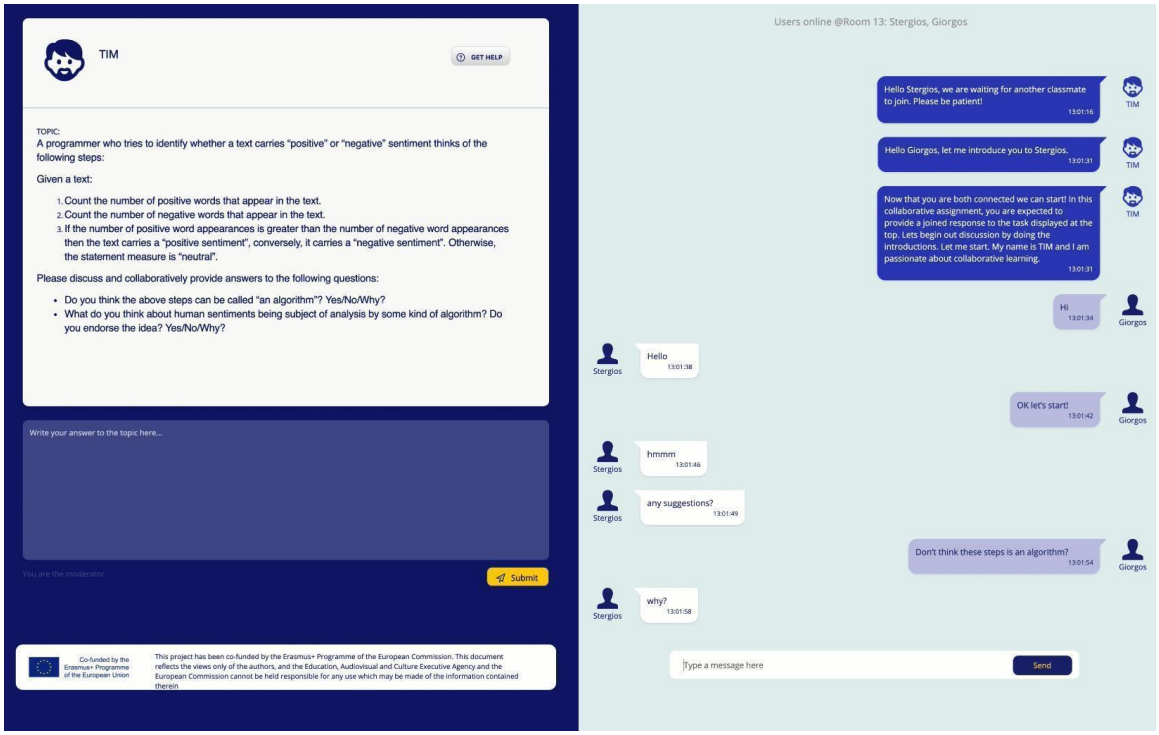


**Figure 25:** Several messages in the chat-based activity

As we mentioned earlier, the player component continuously parses users' dialogues to identify concepts that have been set by the teacher in the activity domain. When an intervention pattern is identified (for example, one or more domain concepts are detected by the peer dialogue parser), the intervention model delivers the associated agent message (prompt) in the chat area (Figure 26). An agent intervention message consists of:

- An agent avatar, to distinguish the agent intervention from the peer chat dialogues
- An intervention text. In the case of a dynamic intervention, the agent prompt/question arises from the domain concepts discussed by the peers.
- An answer box for providing a response to the agent question.
- A drop-down menu for providing feedback for each agent intervention.

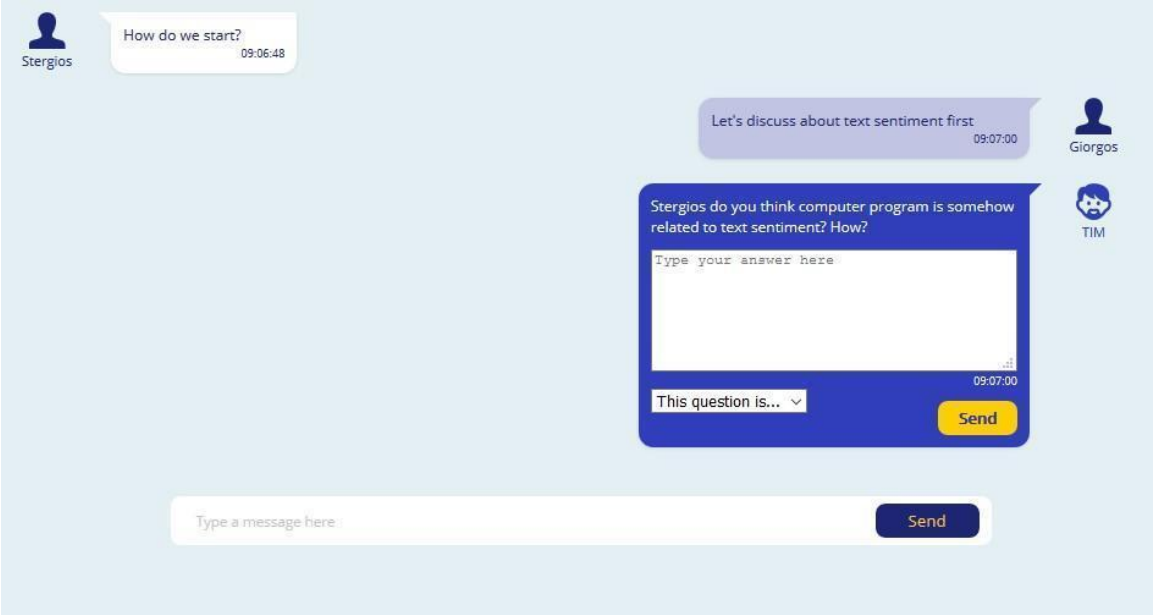


**Figure 26:** Agent intervention message with textbox enabled

If users press the drop-down menu, they can choose from some predefined feedback choices or write down their own (Figure 27). The feedback that users provide to agent interventions is stored in the question metadata. All these can be further used to affect the decision models residing in the agent intervention model in order to select and display questions having the highest score.

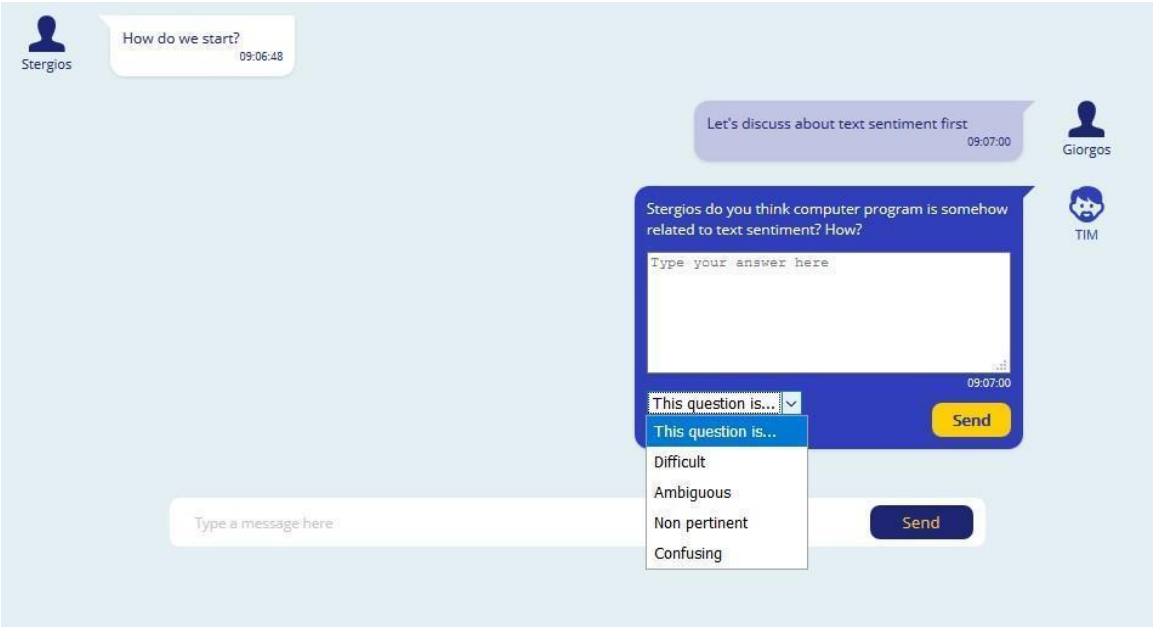


**Figure 27:** Feedback choices to agent intervention

When an intervention is displayed and its answer box is enabled, users can immediately respond to it or try to answer after a short period of time. Following an intervention, since the conversation between peers can continue, this may cause the disappearance of the intervention bubble out of the screen. To overcome this issue, the agent interventions remain pinned at the top of the chat area and, thus, remain available to the peers. Meanwhile, if another intervention with an answer box

emerges and the previous one has not been answered, a bubble icon appears at the top of the interface indicating the number of unanswered (unhandled) interventions. Students can switch between the available interventions and select the one they would like to answer.

As we mentioned earlier, MOOC instructors can choose the direction of the intervention. A question can address either a single user or the whole team. In the case of a question is directed to a single user, only the specific user can reply to the intervention (Figure 28). The other user views a disabled answer box (greyed out) with no reply button. This box displays the following group awareness message: "waiting for your partner to answer" (Figure 29). In case an intervention does not address a particular team member, both learners can provide a response to the agent question.

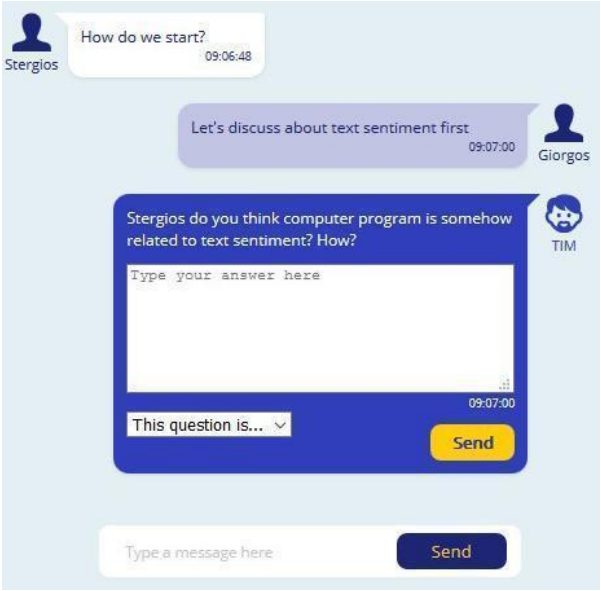


**Figure 28:** Agent question targeting a single user (addressed student view)

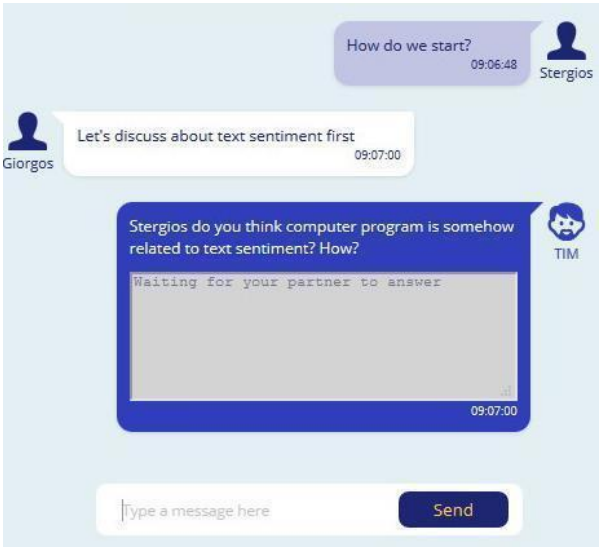


**Figure 29:** Agent question targeting a single user (answer box remains disabled for the partner of the addressed student)

Another type of intervention refers to the interventions that do not expect a response by the students and can serve as a tip or comment displayed by the agent during the students' discussion. In this type of interventions, there is no dedicated answer box below the agent intervention text (Figure 30). As illustrated in the image below, students are not required to submit a direct response to the agent and can continue their peer discussion, after taking into consideration the agent message.

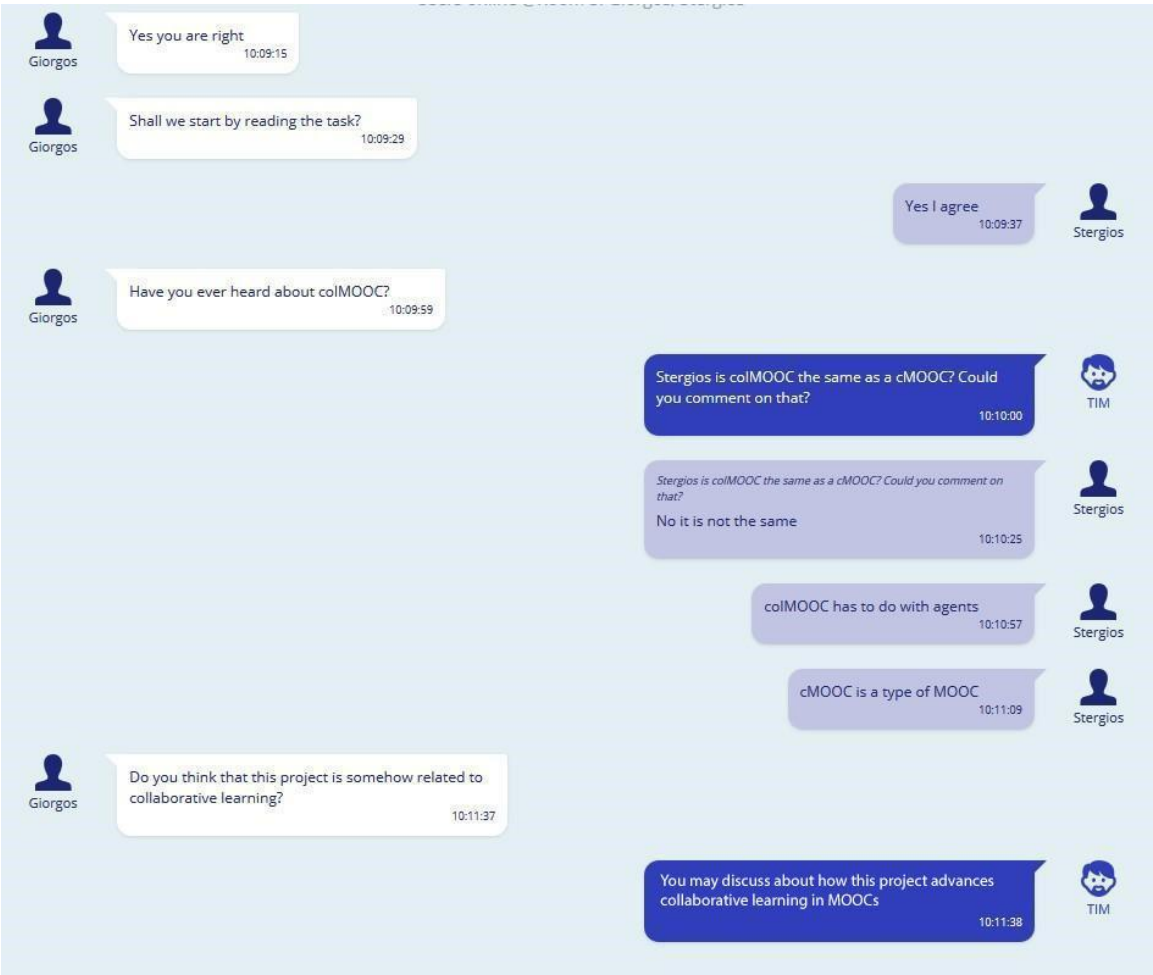


**Figure 30:** Agent tip displayed during students discussion

The overall aim of the chat activity is answering an open-ended domain question collaboratively. Peers can jointly work and provide a team response to the task question using the respective task answer box. Students who enter first the chat activity are given the role of the moderator. Only students having the role of moderator can write and submit their team answer; their partner can view in real-time what is written in the text area but cannot submit the answer of the team. The latter action is only available for the activity moderator. After the "Submit Task" button is clicked and a task answer has been submitted, the agent informs students by posting a relevant message in the chat (for more information, please check the intervention strategy 'Task Completion' presented in the 2.2.6 section). Following the completion of the task answer submission, the specific activity is considered completed, and peers can leave the chat activity by closing the associated tab (Figure 31).

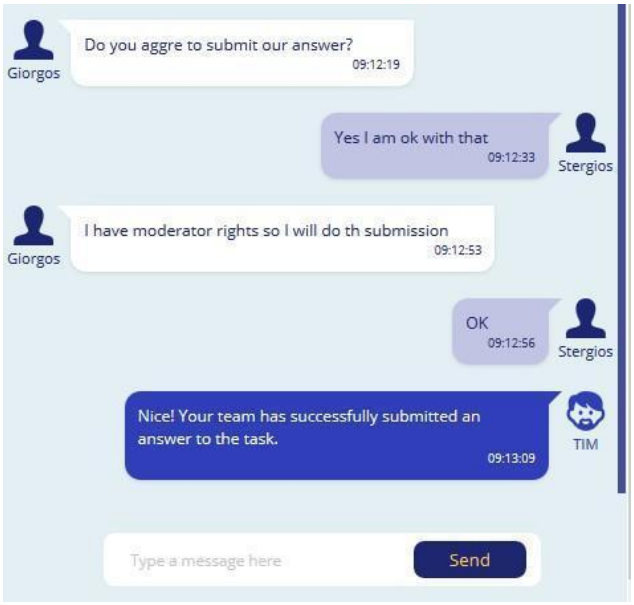


**Figure 31:** Task submission static intervention strategy message

In the next few tables, we will analyze a number of use case examples of the colMOOC player. We will describe how students can interact with the colMOOC player to complete an activity.

**Table 18:** Access an activity use case

| | |
|---|---|
| Use case name | Access an activity |
| Brief description | The student starts the chat-based activity |
| Triggers | The student presses the “Start Activity” link, located in the MOOC |
| Actors | Student |
| Main flow | The student starts the chat-based activity by pressing the relevant link located in the MOOC |
| Alternative flow | - |

**Table 19:** Peer matching use case

| | |
|---|---|
| Use case name | Peer matching |

| | |
|---|---|
| Brief description | The student enters the colMOOC chat activity and waits for a peer to be paired with |
| Triggers | The student presses the "Start Activity" link, located in the MOOC |
| Actors | Student |
| Main flow | 1. The student starts the chat-based activity located in the MOOC<br>2. The student waits for peer matching<br>3. After successfully peering, both students enter the chat activity |
| Alternative flow | 1. Peer matching takes very long to complete<br>a. A message that the peer matching it taking very long to complete is shown to the students after 10 minutes have passed without matching, asking whether the students want to wait more time or complete the activity at a later time<br>b. The student chooses to connect to wait for a peer to be found or chooses to abandon the activity and connect at a later time respectively |

**Table 20:** Chatting use case

| | |
|---|---|
| Use case name | Chat |
| Brief description | The students start to communicate through text messages by using the dedicated chat area |
| Triggers | Both students have entered the chat room |
| Actors | Student |
| Main flow | 1. The student writes the message on the "text message area"<br>2. The student presses "enter" or the "send" symbol next to the text message area to post a message to the activity chat window |
| Alternative flow | - |

**Table 21:** Submit the task activity use case

| | |
|---|---|
| Use case name | Submit task activity |

| | |
|---|---|
| Brief description | The team has completed discussing about the topic and they are ready to post their answer through the "Team Answer" text area |
| Triggers | The student who has the moderator role the "Submit" button |
| Actors | Student |
| Main flow | 1. The student with the moderator role write the answer to the task in the "Team Answer" area<br>2. The moderator presses the "Submit" button<br>3. A confirmation message appears that the task has been successfully submitted |
| Alternative flow | - |

**Table 22:** Answer to an intervention use case

| | |
|---|---|
| Use case name | Answer to an intervention |
| Brief description | The agent can intervene in the group chat by posting some questions to foster discussion. A student can answer to agent intervention |
| Triggers | An agent intervention appears in the chat area |
| Actors | Student, Agent |
| Main flow | 1. The agent intervenes in the discussion by posting a question<br>2. The student writes the answer to the agent intervention<br>3. The student submits the answer to the agent intervention |
| Alternative flow | 1. The student chooses not to answer to the intervention<br>a. Agent intervention is shown in the chat area. The student keeps texting or chooses not to immediately answer to the intervention. The agent intervention stuck on the top of the chat area, remaining available for the student to answer to it.<br>2. A second intervention appears while the first one remains unanswered<br>a. Both interventions are stuck on the top of the screen and a bubble message appears with the total number of unanswered interventions<br>b. The student chooses the intervention to answer by pressing on the bubble message |

| | |
|---|---|
| | c. The student answers to the agent intervention<br>3. There is no answer box in the agent intervention<br>a. Users can comment on the agent intervention in the chat area |

**Table 23:** Giving feedback to an intervention use case

| | |
|---|---|
| Use case name | Giving feedback to an intervention |
| Brief description | Students can provide feedback to an agent intervention. This feedback can be further taken into account to suppress agent intervention questions that received a lot of negative feedback |
| Triggers | Agent intervention inside the chat area |
| Actors | Student, Agent |
| Main flow | 1. Agent intervenes in the discussion by posting a question<br>2. The student chooses from the dropdown list the corresponding predefined feedback to the agent intervention |
| Alternative flow | 1. The student does not want to use one of the predefined feedback<br>a. The student chooses the "other" option<br>b. The student writes his feedback |

# 4 Modifications to the Conversational Agent Module

## 4.1 Introduction and note to reviewers

In this section, we will present some modifications made to the editor and player components of the colMOOC. Some software components were redesigned and altered during the software development lifecycle. These included an updated user interface, some usability-related improvements, and the addition of some new functionalities.

The first version of the agent components was offered by the design team to the development team and served as a common ground for developing the main agent functionality. During the development, the user interface of the components of the colMOOC was continuously being audited for potential improvements. In the next section, we describe the evolution of the graphical user interface between the first version of the system components (player and editor) and the second version, which included several changes focusing on the resolution of some usability issues.

Please note that: *the first prototype version of the Conversational Agent player (baseline version 04) is presented in deliverable D4.2. The current deliverable demonstrates a next updated version 0.6 of the colMOOC platform (editor and player). Although these modifications were generally effective after M12 (which is the due date of the current deliverable) we decided to include them herein to provide an updated overview of the project software interface during interim evaluation reviews. Final modifications of the project software platform will be in detail presented as "2nd version of the colMOOC platform" in deliverable D4.4 (due on M24).*

## 4.2 The evolution of the graphical user interface and usability

During the development of the colMOOC agent, several mockups and prototypes were created. This process required continuous collaboration between the design and development teams. The role of the prototypes were to align both teams to have a common understanding of the final product. A series of optimization and modification stages emerged from the agile development methodology, which was followed throughout the project (see 3.1 section).

The initial prototype (baseline version 0.4) included a basic user interface, which served as the basis to discuss with the development team a series of interface-related elements of the conversational agent. The initial prototype can be seen in Figure 32, where the different areas of the colMOOC player are being shown.

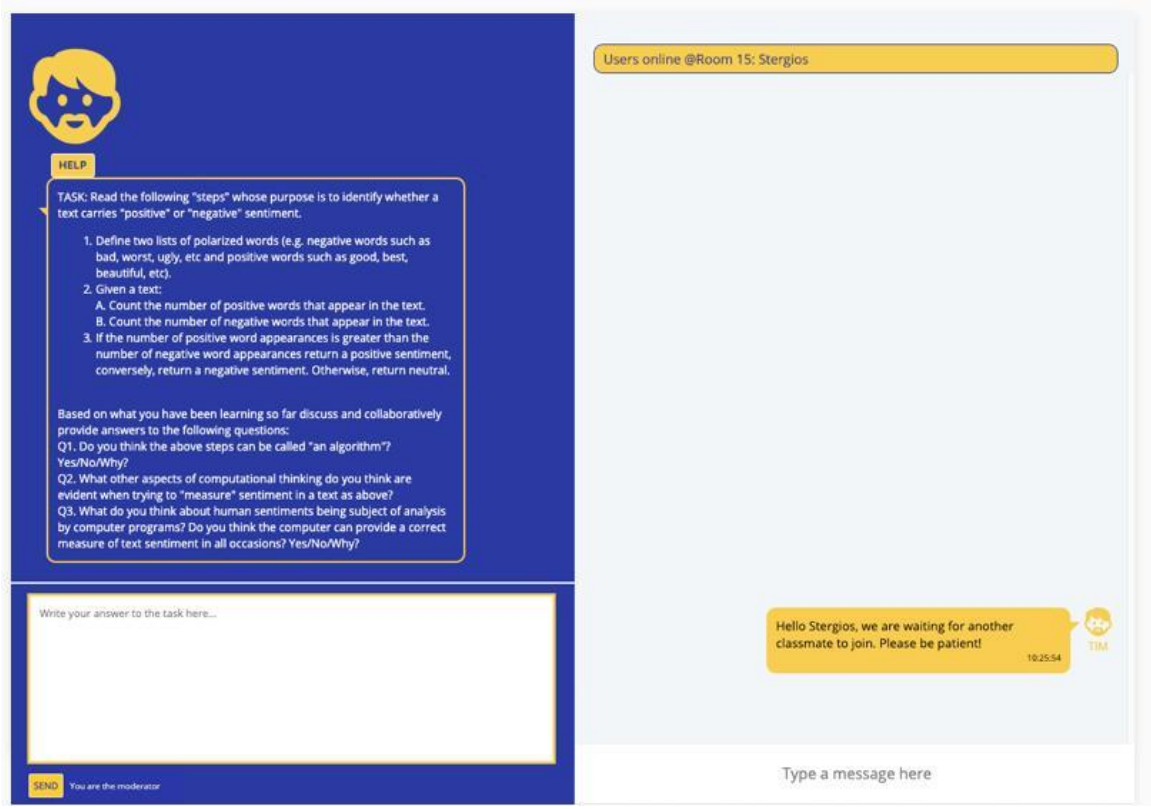


**Figure 32:** The mockup of the colMOOC player interface (initial baseline version 0.4)

The agent consists of two different areas: (a) a left area including the task in a text form along with a task answer box and (b) a right area featuring the exchange of all chat messages.

After reviewing this initial prototype, several changes were proposed regarding the graphical user interface. These proposals were in line with the "Brand Toolkit Guide", developed for the colMOOC project. The suggested modifications focused on the consistent usage of a specific styling and color schema in the user interface of the whole system.

Moreover, some other issues regarding the color palette were detected (fig 35). For instance, the visibility of the text and graphics inside specific areas were found to be problematic. The proposed modifications included the following changes:

- Avoidance of bright yellow color as a base in big areas
- Contrast in some areas needs adjustment for accessibility and readability (Figure 33)
- Addition of extra margin so that elements do not overlap (Figure 34)

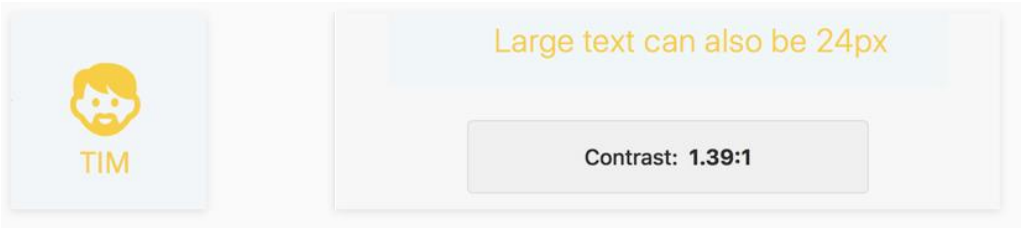


**Figure 33:** Detected color palette issues

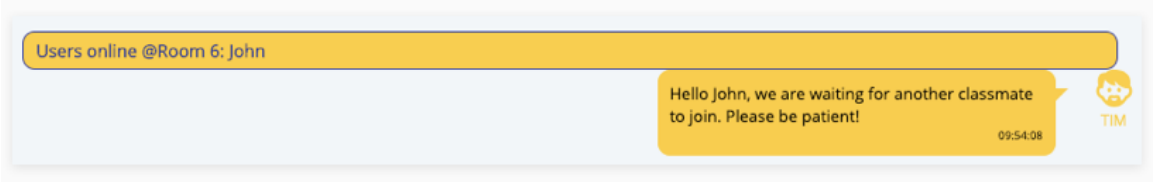


**Figure 34:** Add extra margin to overlapped elements

The usage of a responsive grid was also proposed in order to resolve reflow issues, which were caused when a teacher used a long text for the task description. The new responsive layout was designed to adapt to different screen sizes and orientations in order to ensure consistency across layouts. The interface was redesigned to respond to the user's behavior and environment based on screen size, platform, and orientation. This was achieved through a mix of flexible grids and layouts, images, and intelligent use of CSS media queries.

The new proposals of the design team also contained fixes for some usability-related issues identified. For example, a proposal was made for the improvement of the message composer functionality, which is used by the students for sending new chat messages. The message composer had some visibility and overflow issues. As illustrated in Figure 35, when a user entered a long sentence exceeding the text area's boundaries, the text floated left, causing the disappearance of the text.

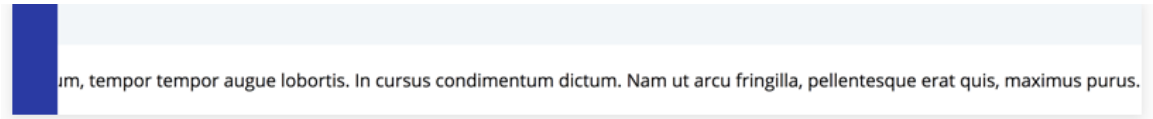


**Figure 35:** Message composer overflow issues

The design team recommended the following:

- the usage of left alignment for the text inside the message composer
- the replacement of the input text with a text area that supports multiple lines
- the addition of a send icon on the right

Regarding the task panel, it was made clear that some changes should be made regarding the message displayed inside the answer box area. Considering that only the 'moderator' can post a team answer to the task, it was suggested that the system should display proper messages informing the students of their current activity role (Figure 36).


**Figure 36:** Task answer box

After discussing the aforementioned proposals with the design team, the development team of the project proceeded to construct a final, improved version of the colMOOC player (**Figure 37**)

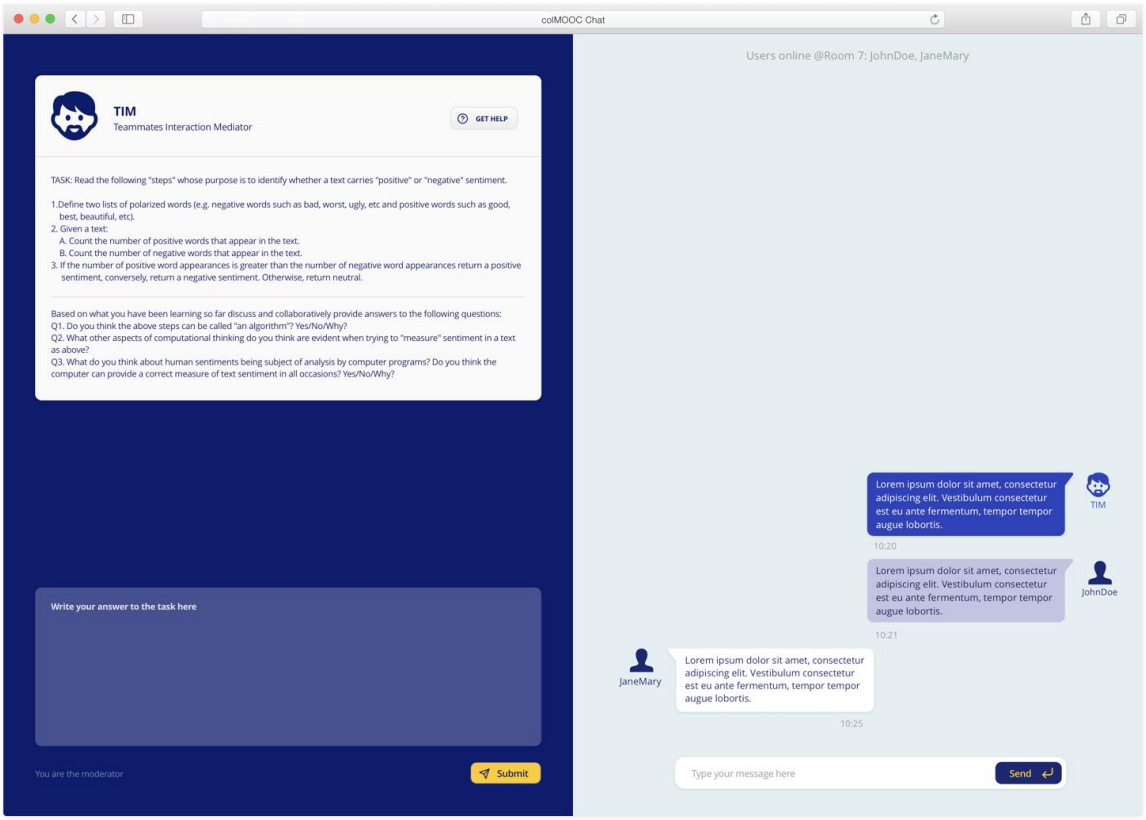


**Figure 37:** Proposed design and color palette for the player component user interface

Modifications and optimization suggestions were also proposed for the colMOOC editor component. Figure 38 depicts an initial prototype of the editor (baseline version 0.3), which allowed the agent domain model configuration by the course instructor. More specifically, the teachers could use this interface to shape the domain model of the agent by entering conceptual links.

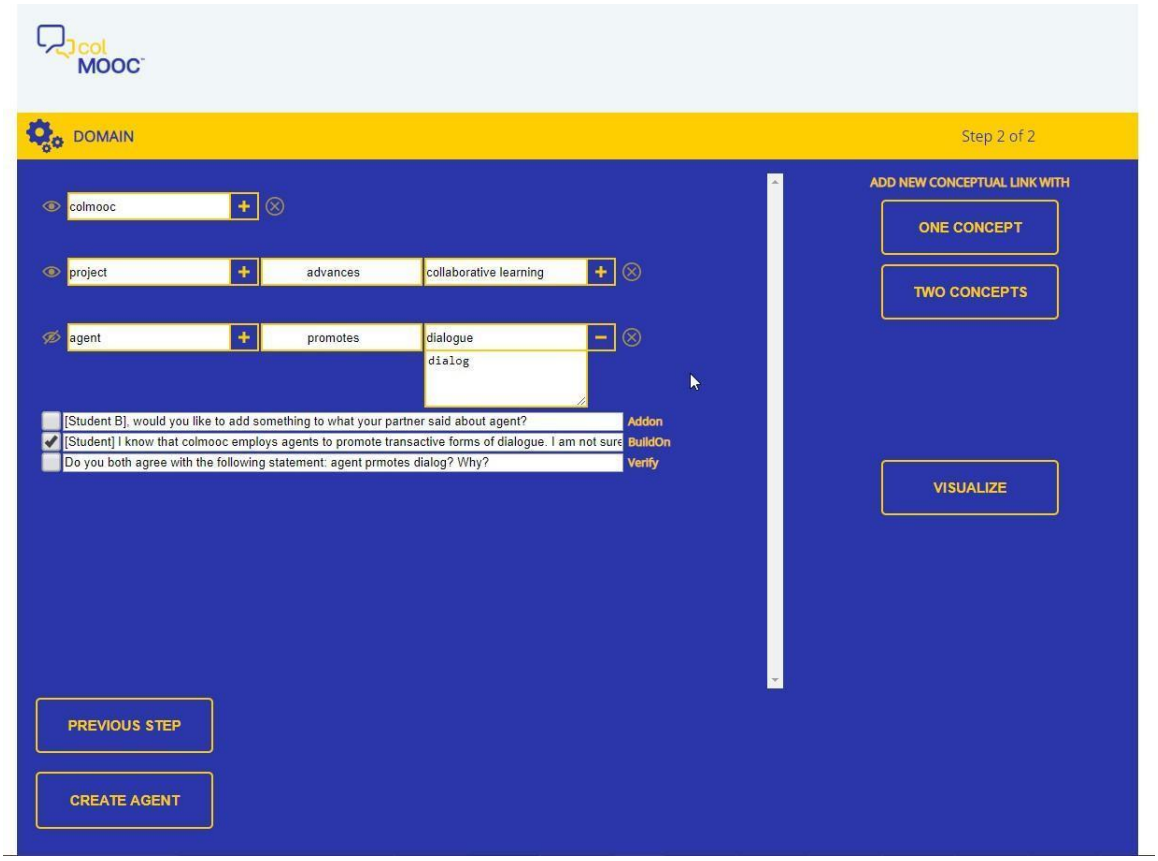


**Figure 38:** The initial prototype of the colMOOC editor allowing the teacher to construct a series of conceptual links

Teachers could press the "eye" icon to access the default questions produced for the associated conceptual link. Teachers could alter the text of the question by clicking inside the respective input text box. Prior to the agent domain model configuration, there was a step calling the course instructor to set the agent name, the agent avatar, the agent language, and the task open-ended question that would serve as the activity discussion topic (Figure 39).

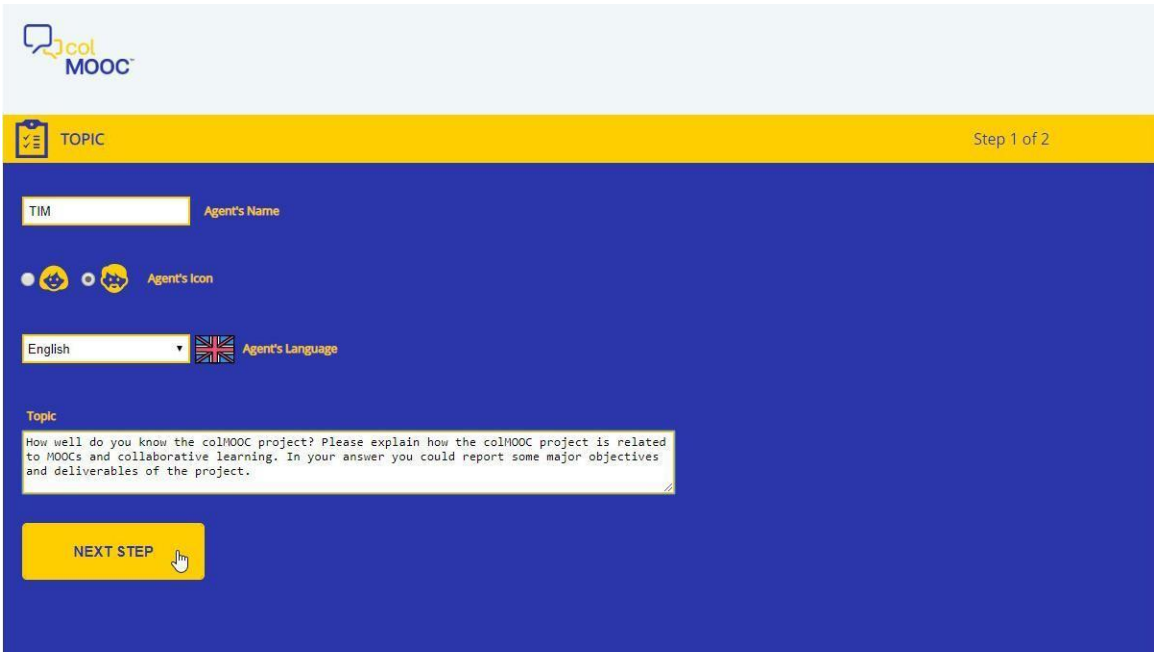


**Figure 39:** A first draft of the editor component illustrating the creation of a chat activity

Both the editor and player components of the colMOOC needed to use the same color scheme for seamless navigation between those different components. The modifications proposed by the design team included:

1. Change of the color scheme to match the player component
2. Change the styling of the different buttons used to navigate between the different screens
3. Have new conceptual link selection buttons placed below all conceptual links
4. Each conceptual link should be placed in a dedicated expandable area with a toggle button
5. Placement of a remove button on each conceptual link
6. The “synonyms” selection should be placed on the right of each conceptual link
7. Addition of the “teacher guidance” text area on the first step of the editor component
8. Add the ability to the expandable area of each conceptual link to be moved to change the priority in the domain model
9. Add the ability to import the configuration of a previously developed agent

Based on the above suggestions, a second version of the colMOOC editor component was created. Figure 40 shows the first wizard step in the editor. This includes a new text area called “teacher’s guidance” text area as well as an “Import config” button to upload the configuration of a previously created agent.

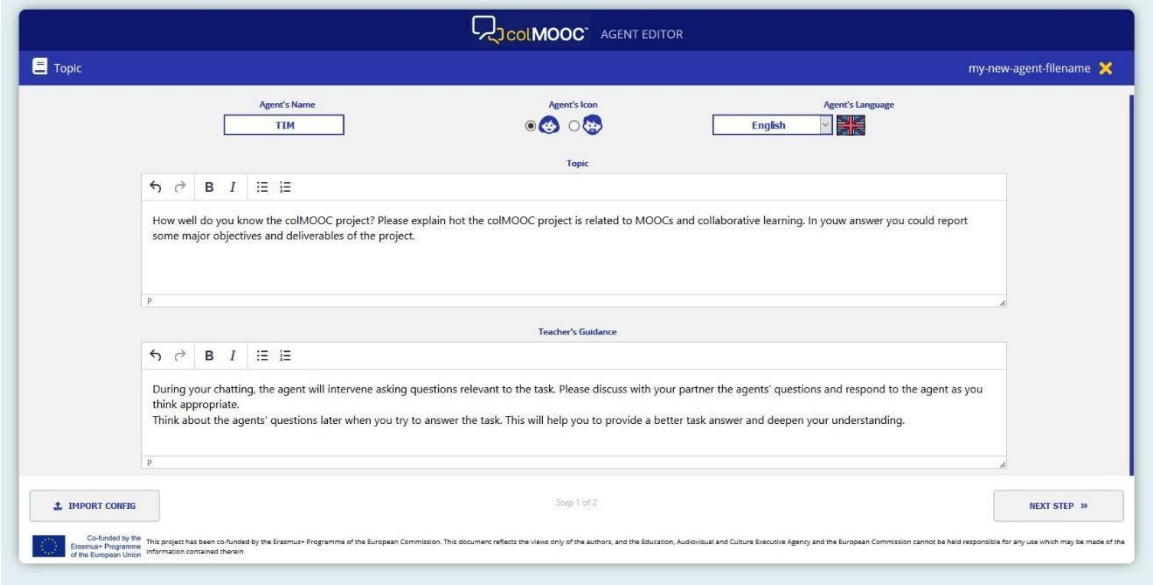


**Figure 40:** The first step of the second version of the editor wizard

Figure 41 depicts the second step of the editor component with the aforementioned recommendations applied to the user interface.

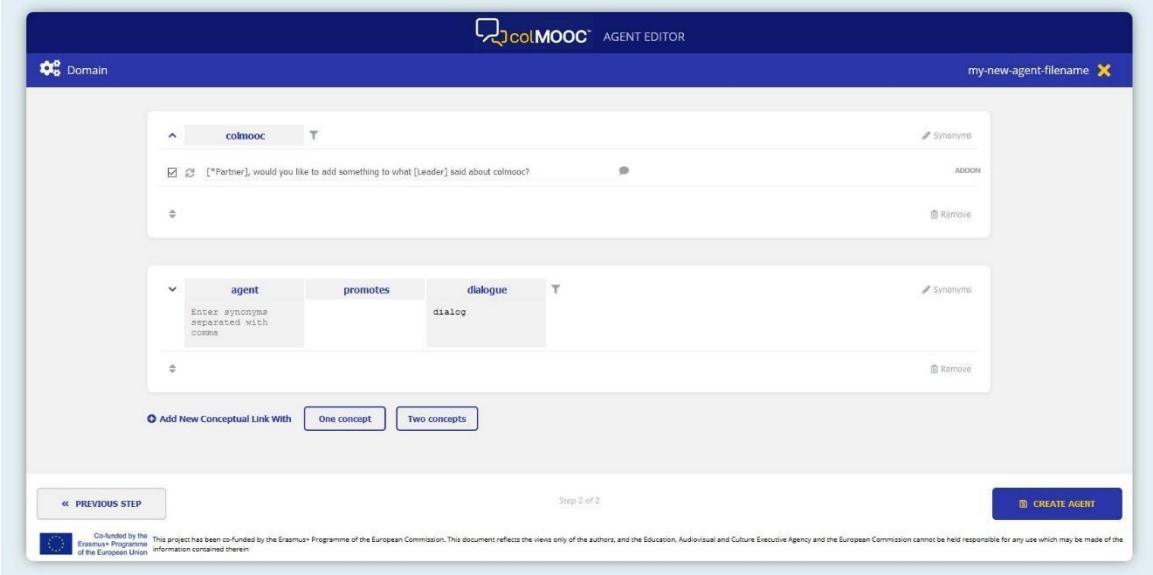


**Figure 41:** The second step of the second version of the editor wizard

## References

Bhalerao, S., Puntambekar, D., & Ingle, M. (2009). Generalizing Agile software development life cycle. International Journal on Computer Science and Engineering, 1(3), 222–226.

D2.1 (2018). colMOOC (Integrating Conversational Agents and Learning Analytics in MOOC) project, Deliverable 2.1 "CA model design", project co-funded by the European Commission within the program Erasmus+ Knowledge Alliances: Cooperation for innovation and the exchange of good practices (2018-2020), Ref.: 588438-EPP-1-2017-1-EL-EPPKA2-KA)

Executive Brief (2008). Which Life Cycle is best for your project? Retrieved from http://www.executivebrief.com

Isaias, P., & Issa, T. (2015). Information system development life cycle models. In High Level Models and Methodologies for Information Systems (pp. 21-40). Springer, New York, NY.