colMOOC – an Innovative Conversational Agent Platform to Support MOOCs

A Technical Evaluation

Anastasios Karakostas[1], Efstathios Nikolaidis[1], Stavros Demetriadis[2], Stefanos Vrochidis[1] and Ioannis Kompatsiaris[1]

[1]Centre for Research & Technology Hellas, Thessaloniki Greece
{akarakos, stathis.nikolaidis, stefanos, ikom}@iti.gr
[2]Aristotle University of Thessaloniki, Thessaloniki, Greece
sdemetri@csd.auth.gr

Abstract— **The main goal of this document is to describe the implementation of each component of the colMOOC platform and the integration stage of the operational platform as a whole. Additionally, a first simple operational scenario that has been implemented is also being described, in order to demonstrate the interoperability of the platform. Finally, this paper presents a positive technical assessment of the 1st version of colMOOC platform through performance benchmarking experiments performed on system's components in order to verify its correct operation and interoperability of the different integrated modules to meet the requirements of the use cases for the end users.**

Conversational Agents; MOOCs; Learning Analytics

I. INTRODUCTION

Massive Open Online Courses (MOOCs) are becoming increasingly popular as an important alternative supplier of knowledge, training and accreditation, aiming principally at large-scale interactive participation and open access via the web [1]. Initially, they adopted mainly a 'knowledge transmission' perspective emphasizing teacher-led individual learning, rather than learning through peer interaction [2]. Conversational agents are software programs engaging learners in a conversation through natural language. Researchers provided evidence of the impact of having a conversational agent interact with peers in computer-supported collaborative learning (CSCL) settings (e.g. [3], [4]). Based on the above background, the colMOOC platform aims to perform research leading to the development of a system that supports the Conversational Agents (CA) supported collaborative activities in MOOCs together with Learning Analytics capabilities (LA). The envisioned system covers areas of a collaboration lifecycle. Namely, there are provisions and components that help the student collaboration and others that support the teacher and the designer. Finally, there are components that accumulate knowledge of past events to help handling future events, as such colMOOC can be viewed as a learning platform which makes use of knowledge from previous activities to help support better a current activity. The main goal of the current paper is twofold: a) to describe the colMOOC platform and b) to present the first simple operational Use Case scenario that has successfully been implemented in order to test the interoperability of the different integrated modules so far, and the main outcomes of this testing.

II. ARCHITECTURE AND PLATFORM COMPONENTS

The architecture consists of: **Ingestion layer** – Serves as the input mechanism into the platform. Different kinds of information can serve as input to the system. Typically, once a new piece of data has been ingested into the platform, it is stored temporarily in a raw storage system, and a proper notification is sent via the service bus to the interested parties. **Internal services** – These services are used internally for the proper functioning of the capabilities provided by the various components. These services are used mostly for data storage and communication. Some generic data analytics and processing services may be used as well. **Business layer** – This layer encompasses the components that perform the actual platform specific capabilities. The bulk of the platform is concentrated at this layer, which interacts in turn with all additional layers.

A particular group of components in this layer tackles the analysis of different kinds of data flowing into the platform. Among this group we can find the following components: a) Conversational agent editor, b) MOOC authoring. A second group of components deal more specifically with the analysis. Among these components are: a) Learning analytics and b)Communication with the MOOCs.

External layer – This layer handles the interaction of the platform with external entities, both as input providers and output recipients. There are two main groups of components making up this layer, namely: Conversational agent player; MOOC student environment.

The REST architectural pattern is used to model the services used by the different services. REST proposes a very lightweight, HTTP-based method of stateless operations between resources in the Web. Due to the distributed approach of the design and implementation of the colMOOC platform, according to which different partners develop independently and maintain ownership over different components of the system, combined with the complexity of the platform, which is comprised of many different components, we opted to follow a micro-services approach to make the entire process of design, implementation, and integration more manageable, in a distributed manner.

Using this approach, we maintain the independence of the different components, and the integration focus is on the exposed capabilities of each component. The integration between components is being realized via two main mechanisms, first, inter-communication via the platform communication service api, and second through exposed Representational state transfer (REST) interfaces of various components, potentially aided by a discovery service. In addition, date is being exchanged between components by using shared data stores, and having links to data items incorporated into messages exchanged between components.

For communicating through the communication api, the components need only to agree on the call via which they will be interacting, and the format of the messages published on that call. For communicating via a REST interface, there needs to be a way for one micro-service to find another micro-service it wishes to invoke. For that purpose, a service discovery component may be deployed. Such a component serves as a central registry of available micro-services, responding to queries about the location of a certain micro-service.

III. The Platform

Briefly, the demonstration has 2 phases:

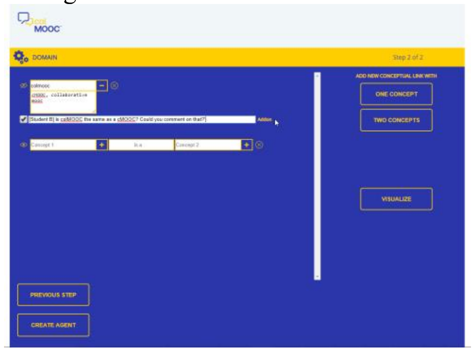
1. The creation of a new agent by the teacher
2. A small collaborative activity between two students based on the editor that has just been created.

Editor – steps

1. Teacher enters colMOOC Editor and creates new Agent
2. Teacher fills in Agent's Name/Icon/Language and Topic for intended activity

Example: Name: TIM Icon: Male Language: English Topic: How well do you know the colMOOC project? Please explain how the colMOOC project is related to MOOCs and collaborative learning. In your answer you could report some major objectives and deliverables of the project.

3. Teacher creates conceptual links that trigger agent's intervention



colmooc

cmooc, collaborative mooc

[Student B] is colMOOC the same as a cMOOC? Could you comment on that?]

Figure 1. Teacher creates conceptual links that trigger agent's intervention

Player –steps

1. Student writes his name in order to enter colMOOC Player
2. Student enters colMOOC Player, while Agent welcomes him by providing the Help message.
3. Another Student enters colMOOC Player so Agent makes the introduction.
4. Students start chatting to each other.
5. Agent intervenes with a question when a concept appears to the conversation and students comment on Agent's intervention

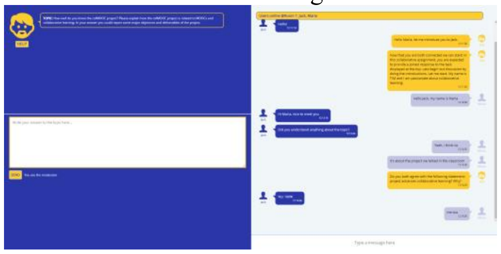


Figure 2. Agent intervenes with a question when a concept appears.

IV. Technical Evaluation of the System

Following the guidelines of the colMOOC technical evaluation strategy, technical meetings were regularly organized and several test cases were performed so as to validate and verify 1st version's achievements. The evaluation focused on covering as many Key Factors as possible with respect to the technical work.

A. *Testing and Evaluation Methodology*

In order to conclude about the technical evaluation, it is noteworthy to refer to the extended efforts of testing that we committed upon, through teleconferences with other partners, so as to solve intra-communication problems with other components as well as internal functionality misbehaviours. At project integration level, we used weekly plenary meetings (more than 20) to provide updates on the current status of the different parts of the project in order to accomplish with specific deadlines following a SCRUM testing methodology. In the following is described this testing methodology which was tailored to the integration of the 1st version of colMOOC platform. Our testing framework was designed with reference to the requirements. Within colMOOC testing framework meetings were organized and facilitated by the scientific manager assisted by the technical manager of the project. Every week a different sequence of the storyline from the Use-Cases was set by the scientific manager to be tested, creating a backlog – a list of tasks to perform during the meeting. These tasks were forwarded before meetings via email to all participants and were populated on colMOOC's DokuWiki under the page "Technical Telcos – Minutes and action points". By this manner and assigning roles to the participant "actors" the testing procedure was well tuned and all members had a clear idea about their responsibilities.

In this integration of the 1st prototype of the platform all testing scenarios were consisted of two main phases. The phase of the design-editor and the phase of the activity-player. The main "acting roles" in the sequence of testing scenarios were those of the teachers and students. These roles are incarnated in the context of the External facing layer of the platform. The aforementioned tools were serving as the principal criterion for the stakeholders to monitor and to evaluate the process. Furthermore, an additional tool serving for internal purposes as a message logger was used for the graphical representation and monitoring of the messages exchanged between the different components. The main idea of integration testing was to combine gradually and test the interface between the key components and eventually to expand in order to test all the integrated components of the platform fulfilling a complete flow that describes a Pilot Use Case scenario. Thus, the scientific manager, orchestrating the whole process and with respect to the predefined testing plan, was requesting for actions from technical partners.This approach emphasizes the value of collaboration and the rapid feedback that iterative usability evaluation can provide, its effectiveness typically relies more on the involvement of End Users - usability experts.

To summarize, plenary meetings were serving as weekly reviews for testing the platform, during which the teams demonstrate the new functionality to the end-users-experts who provide feedback that could influence the next meeting. Technical overview combined to the feedback from the stakeholders was resulting in changes to the delivered functionality, but also in revising or adding items to the product backlog. Defects, change requests, new tasks were accumulated after session to feed the Issue Management tool that is described in the following subsection. Various ad-hock sessions were also scheduled jointly with the regular meetings. The main goal of those sessions was to fill the software gaps between components.

B. *Issues Management*

There is no doubt that issue tracker, is one of the key factors for evaluating the software process development. It provides a platform to support software development and maintenance activities such as issue reporting, tracking and resolution. Issue tracking (such as a bug report or a feature enhancement request) is a social and collaborative process in which an issue tracker serves as a communication hub and channel between the developers and QA (Quality Assurance) team. For the issues found during development and technical tests of the colMOOC platform, the consortium partners are using GitLab embedded issues management approach to store issues, create milestones and distribute the technical work needed to overcome these issues. GitLab issue tracker serves the following purposes; Safe and reliable method for the team :to raise issues.;Track and assign responsibility to specific people for each issue.;Analyze and prioritize issues easily.;Record issue resolution for future reference and project learning. And Monitor overall project health and status. Finally, in order to impose the time plan for addressing the reported issues, specific Milestones are used with reference to the technical weekly meetings The management of issues, labels and milestones is open to all technical partners and is done at repository level, thus changes are made easily when needed. The view of issues however can be aggregated at the root level colMOOC-project in GitLab, therefore it is easy to have an overview of the issues of whole framework.

C. *Evaluation of Key Factors*

For the evaluation of the quality of the 1st version of the platform we selected to present some quality metrics of the modules which are considered as the key factors of the implementation. There are many scalability dimensions in the integration and mechanisms used which affect favorably the overall performance and throughput of the system. Currently the deployed system can comfortably accommodate the anticipated load of the colMOOC pilots, and has the capacity to support a higher load, given the current installation and deployment. In addition, there are various scalability factors, affecting performance that can be applied when the system load gets considerably larger.

Number of API Calls: For scalability and fault tolerance the api can run with a number of servers acting as cooperating message brokers; cooperating for providing continuous service. Running multiple brokers means that for each partition there shall be a single broker acting as the designated leader, and a list of brokers acting as replicas. Currently colMOOC's api is deployed over 5 brokers. The number of brokers can be scaled up based on need, but for the foreseeable future there is no expectation that the platform would require more brokers to be deployed.

Number of Students and groups: 136 Students logged in the colMOOC Player module at the same time, as pairs of two. The 68 groups of Students ran colMOOC Player in two phases with two different versions of the same Agent.

Number of Chat posts: By running a total number of 136 different sessions (68 groups with 2 sessions each), there was 30.692 chat posts (average number of 225,6 chat posts per session and 112,8 chat posts per student per session).

Number of Agent interventions: The first version produced 286 Agent interventions (average number of 4,2 interventions per group), while the second version (the more intervening Agent) produced 761 interventions (average number of 11,1 interventions per group). By increasing the total load to the upper limits we documented above, there was no significant difference regarding the overall response of the system and all individual modules (colMOOC Editor, Player or LA).

Acknowledgements

This research has been funded by the Erasmus+ Programme of the European Commission (project No 588438-EPP-1-2017-1-EL-EPPKA2-KA - colMOOC)

References

[1] A. Littlejohn, “Understanding massive open online course,” CEMCA EdTech Notes, 2013, 1–12. Retrieved from http://cemca.org.in/ckfinder/ userfiles/files/EdTech.

[2] G. Conole, “MOOCs as disruptive technologies: strategies for enhancing the learner experience and quality of MOOCs,” RED - Revista de Educación a Distancia, 39, 1–17, 2013.

[3] R. Kumar, and C. P. Rosé, “Architecture for Building Conversational Agents that Support Collaborative Learning,” IEEE Transactions on Learning Technologies, 4(1), 21-34, 2011

[4] E. Walker, N. Rummel, and K. R. Koedinger, “Designing automated adaptive support to improve student helping behaviors in a peer tutoring activity,” International Journal of Computer-Supported Collaborative Learning, 6(2), 279-306, 2011