colMOOC™

D4.1

Technological roadmap

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

D4.1 - Technological roadmap

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:	Report
Deliverable reference number:	D4.1
Deliverable title:	Technological roadmap
WP contributing to the deliverable:	WP4
Delivery date:	M6

WP Leader:	CERTH
Responsible organization:	CERTH
Abstract:	This deliverable presents the technological roadmap towards the implementation of the colMOOC platform. First, this roadmap presents a high-level view of the envisioned architecture for the colMOOC platform, illustrating the conceptual architecture and a draft of the complete technical architecture. Then, the deliverable describes a summary of the supporting development technologies, input/output specifications, the increasing functionality over time, the development timeline, as well as the detailing schedule and resource allocation. Finally, a global project timeline is presented, describing the platform's scheduled iterations and

	the levels of functionality at the project milestones.
Keywords:	Technological roadmap, development, components

Dissemination level:	Public

Authors and Reviewers

Stathis Nikolaidis, Centre for Research and Technology Hellas (CERTH), Greece, stathis.nikolaidis@iti.gr

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

REVIEWERS

Thrasyvoulos Tsiatsos, Aristotle University of Thessaloniki (AUTH), Greece, tsiatsos@csd.auth.gr

Document Change Log

Version	Date (mm/dd/yyyy)	Author (s)	Sections Changed
0.1	03/01/2018	See author list	ALL
0.2	04/20/2018	See author list	ALL
1.0	06/15/2018	See author list	ALL

Executive Summary

This deliverable presents the technological roadmap towards the implementation of the colMOOC platform.

First, this roadmap presents a high-level view of the envisioned architecture for the colMOOC platform, illustrating the conceptual architecture and a draft of the complete technical architecture. Then, the deliverable describes a summary of the supporting development technologies, input/output specifications, the increasing functionality over time, the development timeline, as well as the detailing schedule and resource allocation. Finally, a global project timeline is presented, describing the platform’s scheduled iterations and the levels of functionality at the project milestones.

Table of Contents

List of Tables

List of Figures

List of Acronyms

Acronym	**Description**
AB	Advisory Board
DLP	Deliverable Lead Partner
EB	Ethics Board
IPG	Intellectual Property Management Group
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

This document presents a roadmap of how the Consortium plans to develop the colMOOC platform. The objective of the deliverable is to specify in detail: (i) the progressively increasing functionality (in particular at the time points of the milestones) of the platform as a whole, (ii) the resources that will be needed to achieve this functionality; (iii) the technical infrastructure specifications.

To this end, and based on a model of iterative development, the roadmap includes:

1. A high-level view of the envisioned Platform architecture, illustrating the conceptual architecture.
2. A functional description of the integration, describing:
    a. A summary of its scientific and technical objectives, and supporting technologies;
    b. Its progressively increasing functionality over time;
    c. Its timeline, detailing schedule and resource allocation;
3. A global project timeline, describing the platform's scheduled iterations and the levels of functionality at the project milestones;

Therefore, this deliverable is structured as follows. Section 2 presents the architecture. Section 3 provides the functional description of the integration in alignment with the project. Finally, Section 5 provides the overall project timeline and the milestones, while Section 5 concludes the deliverable.

1.2 Architecture

The forming architecture is formulated with the ultimate goal of providing accuracy and functionality for complex collaborative learning scenarios supported by a conversational agent and integrated in MOOC platforms.

The architecture is roughly made up of the following layers:

1. Ingestion layer, containing mechanisms and channels through which data is brought into the platform;
2. Internal services, generic data repositories and communication services being used by the different components;
3. Business layer, containing the components that perform the actual platform-specific capabilities;
4. External facing layer, including the CA, LA modules, interacting with people and entities outside the platform (end-users of the platform).

The first step in a generic flow of the platform consists of new data being pushed into the platform. The new data can originate from specific MOOC. Note that these MOOC may be bi-directional as they can serve to communicate back with the users upon need. Moreover, incoming data to the platform can originate from other sources as well, such as stored agents.

Once a new piece of data (e.g. student id in the MOOC) is successfully ingested into the system, the data is stored in a temporary raw data store and the availability of the new piece of data is broadcasted to all interested components.

All interested parties will receive the information, which includes a pointer enabling to access the data, and shall perform their specific analysis on the new data. Note, that the result of such an analysis may in turn create a new piece of data which is of interest to other components like LA, that once again will be made aware and access the data in the same manner explained above, providing services for analysis.

The final results of the analysis can be added to the data base, and if required will notify MOOC of the identification of a new state.

1.2.1 Platform components

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:

- Conversational agent editor
- MOOC authoring

A second group of components deal more specifically with the analysis. Among these components are:

- Learning analytics
- 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 will be 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.

1.3 Technology stack

In order to keep the architecture coherent and manageable, and simplify hosting and maintenance, a controlled set of technologies will be used to implement the shared components of the platform.

As a general principle, open source technologies will be used to implement the platform ("open source" defined as software licensed under an OSI[1]-approved license that is included in the list of

[1] Open Source Initiative, see http://opensource.org/.

EUPL-compatible licenses[2]). Exceptions will be proprietary or non EUPL-compatible licenses for software belonging to a partner in the colMOOC consortium, or specific products, where no EUPL-compatible alternatives exist.

1.3.1 Programming languages

A preliminary list of the selected programming languages for module implementation is provided in Table 1.

Language/framework	Versions allowed	Notes
HTML/CSS	ANSI-C 99 / ISO C++	For colMOOC Editor & colMOOC Player components development
JavaScript	Any	Node.js 11.0+
PHP	7.0	For colMOOC API component development

Table 1: Programming languages

1.3.2 Databases/Repositories

A preliminary list of selected database/storage solutions is provided in Table 2.

Data store	Versions allowed	Notes
MySQL	8.0	Storage for colMOOC Editor & colMOOC Player components
MongoDB	3.4.2	Storage for colMOOC Analytics component

Table 2: Database and repository packages

1.3.3 Other technologies

A preliminary list of miscellaneous technological packages (application servers, special purpose stacks, etc.) is provided in Table 3.

Product	Versions allowed	Notes
Slim	3.0	PHP framework for colMOOC API component
Node.js	11.0+	JavaScript application server
Apache	2.4.39	HTTP server

Table 3: Other technologies

1.3.4 Exchange formats

The preferred exchange format for structured data will be **JSON**[3]. **UTF-8** encoding will be used in order to ensure correct and consistent handling of multilingual content.

[2] See https://joinup.ec.europa.eu/software/page/eupl/eupl-compatible-open-source-licences for complete list of EUPL-compatible open source licences.

2 System development, integration and evaluation

Due to the distributed approach of the design and implementation of the colMOOC platform, in which different partners 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 REST interfaces of various components, potentially aided by a discovery service.

Micro-services is an architecture form and methodology for breaking up a large and complex system into small units, each fulfilling a unique well-defined task, in an independent manner. Each such micro-service is deployed separately and can interact with its peer micro-services using standard communication protocols. Such an approach eases the task of long term maintenance and evolution of a large system, compared to a monolithic system. In addition, internal changes within a micro-service do not influence any other system component as long as the external contract of the micro-service is adhered to. Moreover, each component can be implemented in a different programming language, using different middleware.

As mentioned earlier, there are two main mechanisms for micro-services to interact, namely, the communication service api and a REST interface (possibly with the help of a discovery service). For communicating through the api the components need only to agree on the topic via which they will be interacting and the format of the messages published on that topic. 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.

2.1.1 Twelve-Factor Applications[4]

The platform will be composed of multiple micro-services and aspires to seamlessly deliver services to its users, thus we are encouraged to use the following guidelines:

1. **Codebase**

Maintain a one-to-one relationship between codebase and the application (one codebase multiple deploys).

2. **Dependencies**

They should not rely on system-wide packages or functions. Dependencies must be declared inside the scope of the application

[3] http://json.org/

[4] https://12factor.net/

3. Configuration

Separation between configuration and code, as configuration varies across deployments.

4. Back-end Services

A micro-service should make no distinction between local and third party services.

5. Build, release, run

Separate between build, release, and run stages.

6. Processes

Stateless. Any persistent data is stored using external services.

7. Port binding

Application should be standalone, rather than relying on a running instance of an application server. Services are provided to externals via a certain port bound to the application.

8. Concurrency

Should easily be scaled horizontally and having the workload distributed amongst multiple instances of the same micro-service.

9. Disposability

Support failures by easily being started and stopped.

10. Development / production parity

Designed for continuous deployment. Therefore, the difference between development and production environments should always be minimal.

11. Logs

Provide a continuous stream of logging information, which is retrieved by a separately configured service.

12. Admin processes

Run admin/management tasks as one-off processes, on an environment similar to production.

2.1.2 System architecture

This will involve the definition of the global architecture for the colMOOC platform, from its high-level component design to the server layout definition. A roadmap will also be defined, to drive the process of implementation.

Dependencies	User requirements, specification of pilot use cases and validation plan: the Use Cases will drive the processes and ultimately the architecture will be built to service those use cases, so a clear definition is essential.
Critical Factors	• Incomplete Use Case definition: UCs must be clearly defined and well understood to ensure the appropriate systems, processes and capabilities are put in place in the architecture. UCs must include comprehensive scenarios and validation plans to ensure the architecture is built correctly to specifications. • Incomplete service definition: each of the technical

	development WP is responsible for defining the service API for components related to its research. Failure to provide the API specification in a timely and thorough manner can lead to delays or errors in definition of the architecture.

Table 4: Architecture definition summary

2.1.3 Technical infrastructure

This implies the provisioning and operation of the infrastructure, on which the colMOOC system will run. This includes the internal services, business services, associated repositories and external entities (mobile applications, control centers).

It easily follows that some subsystems of the colMOOC platform cannot be deployed to the platform, due to licensing reasons. Others may be dependent on secondary systems that cannot be easily replicated in a different environment, or may require concentrated computing power that is provided by a highly specialized expert system in a partner's data centre. In such cases, proxy services will be built by partners and exposed to the rest of the platform. Proxy services will delegate work to the actual services and do any extra work as required in a transparent way.

Dependencies	System architecture: technical infrastructure will be provisioned to implement the system architecture.
Critical Factors	• Distributed infrastructure: parts of the infrastructure are foreseen to be distributed and include systems which are run in-house by some partners. Issues with firewalls, authentication, etc. are to be expected, and acted upon as needed. • Performance: the scope and complexity of the colMOOC processes and the size of the data to be handled is not yet known and is likely to evolve as research progresses. Performance problems may arise, and resizing and/or provisioning of extra capacity may be required.

Table 5: Infrastructure definition summary

2.1.4 System development

This is the task for all development activities related to the technical implementation of the Use Cases and any other work required for the completion of the objectives.

The system development will be iterative, from an initial dummy prototype to the final version, with the following cycle planned:

Operational Prototype

Dummy end-to-end architecture; all service dummies in place, operating on test/mock data.

First Prototype

Using real services from WP2-WP6; First Prototype milestone (MS3).

Second Prototype

Using real services from WP2-WP6; Second Prototype milestone (MS4).

Final System

Complete colMOOC platform; Final System milestone (MS5).

Dependencies	• All technical WP services • WP2 Use Cases and Evaluation
Critical Factors	• Changing requirements: Use Cases should remain stable during the project. While minor changes are always to be expected, major breaking changes may cause throwing away work that has already been done and impact on delivery capabilities.

Table 6: System development summary

2.1.5 colMOOC API

The colMOOC API serves as a central point of communication between different system components and external platforms. Its main mode of operation is that of create / edit / delete entities, which supports different parts of a composite application to be unaware of each other but still manage to communicate upon need.

The proposed API is in charge of notifying and updating interested and registered components when new entities (such as Activities, Agents, Students and Sessions) which are of interest to them have been created or edited by another component.

The colMOOC will be configured, upon deployment, with the necessary set of calls as agreed upon between the different components.

Dependencies	• Slim – a PHP micro framework for creating powerful web applications and APIs
Critical Factors	• Needs to support the number of calls required along with the aggregated throughput. Requires cooperating components to agree upon call format and abide by them.

Table 7: Communication bus summary

2.1.6 Activity table and workload

Activity	Subactivity	Description	Dependencies
System architecture	Technological roadmap	Current document	N/A
	System requirements and architecture	Description of technical requirements derived from Use Cases and detailed design of platform architecture	WP2 user requirements
Technical infrastructure	N/A	Provisioning and operation of the infrastructure, on which the colMOOC system will run	System architecture

System development	Operational Prototype	Initial dummy prototype	WPs 3-6
	First Prototype	First iteration of colMOOC platform	WPs 3-6
	Second Prototype	Second iteration of colMOOC platform	WPs 3-6
	Final system	Final version of colMOOC platform	WPs 3-6
API	N/A	Central point of communication between different system components	• Slim
End-users applications	N/A	Mobile application used by first responders and the general public	• WP2 user requirements • Integrated system backend (T7.2) • Communication infrastructure (T7.4)

Table 8: WP7 activity table

3 Project Timeline and Milestones

The platform will be built incrementally and iteratively, starting with a system made up of dummy services for formal end-to-end validation and delivering increasingly enriched functionality and further capabilities on each milestone.

The figure below illustrates the timeline for the technical milestones in the evolution of the system.

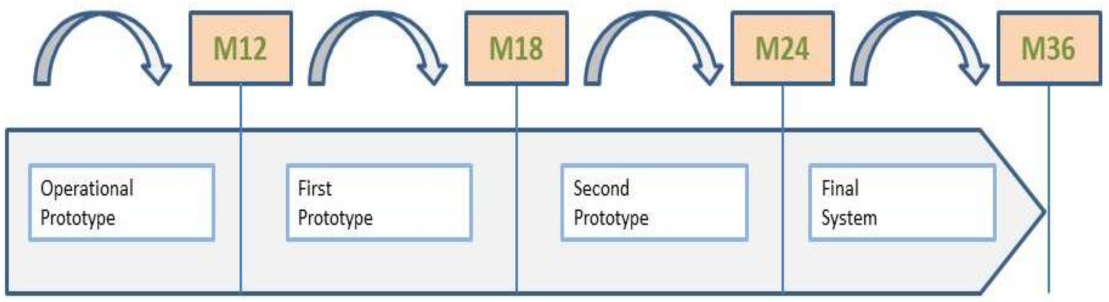

Figure 1: colMOOC walking skeleton

4 Summary

In this deliverable, the technical roadmap of the colMOOC platform was presented. This document will guide the development of the project with a view to attaining the scientific and the technical objectives envisaged.

However, since the project is following an iterative development procedure (use cases-requirements-development-evaluation), it is expected that small adaptations and deviations from the initial technical specifications and the module functionalities foreseen by this document will be needed to satisfy the final user requirements. Such changes/adaptations might be required at component or subcomponent level. The platform architecture together with the detailed specification of the modules (including any updates required) will be reported in D7.2 (System requirements and architecture).