DELIVERABLE

Project Acronym: EMMA
Grant Agreement number: 621030
Project Title: European Multiple MOOC Aggregator

D2.1 Aggregator Description

Final version

Authors Rosanna De Rosa
Name (Organization) Unina
Contributors: Jorge Civera UPV
Eleonora Pantò (CSP)
Kairit Tammets (TLU)

Due Date of Deliverable: Month 8
Start date of project: 1/2/2014
Duration 30 Months
Leading organization for this document Unina

PROJECT CO-FUNDED BY THE EUROPEAN COMMISSION WITHIN THE ICT POLICY SUPPORT PROGRAMME		
	Dissemination level	
PU	**Public**	**v**
CO	**Confidential, only for members of the consortium and the Commission Services**	
PP	**Restricted to other programme participants**	

This project has been funded with the support of the Competitiveness and Innovation Framework Programme (CIP) – ICT Policy Support Programme (ICT PSP) of the European Union

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Revision History

Revision	Date	Autho	Organisation	Description
V01	14/09/2014	RDR	UNINA	First draft document
V02	10/10/14	EP	CPS	Revision and contribution
V03	16/11/14	KT	TLU	Contribution
V04	20/11/14	JC	UPV	Contribution
V05	25/11/14	PR	ULEIC	Revision
V06	25/11/214	CPN	ATOS	Revision
Final	26/11/14	RDR	UNINA	Sent to the Commission

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of Contents

1 Executive summary 5

2 Introduction 5

3 Introduction: general principles 6

3.1 The technology 7

3.2 Learning content 7

4 The technical project 8

4.1 Types of users and Cases analysis 10

4.2 List of general functions 16

4.3 Production environment 20

4.4 Deployment Diagrams 21

4.4.1 Architecture 21

4.4.2 User level 21

4.4.3 Logical level 21

4.4.4 Data Level 21

4.5 Entities and relationships – Database 22

4.5.1 Diagram of db v 1.0 22

4.5.2 Diagram of Db v 1.2 23

4.6 External Components Integration Process 24

4.6.1 Tracking System 24

4.6.2 Learning analytics Dashboard 28

4.7 Translectures and translation Services 29

4.7.1 The transLectures Platform (TLP) 29

4.7.2 The Document Translation Platform (DTLP) 34

Table of Figures

Figure 1 - Production environment 20
Figure 2 - Deployment diagram 22
Figure 3 - Db diagram – v. 1.0 23
Figure 4 – Db Diagram – v. 2.0 24
Figure 5 - Tracking system diagram 25
Figure 6 - Instructors' dashboard enrollment activities in pilot phase 28
Figure 7 - Students' dashboard accessed materials 29
Figure 8 - Main components of the transLectures Platform and a simplification 30
Figure 9 - TransLectures Ingest Service workflow for an uploaded media package 33
Figure 10 - TransLectures player for translation editing 34
Figure 11 - Document Translation Web Editor 36

1 Executive summary

The main output of WP2 is to have the EMMA platform in place with the translation/transcription as well as learners behavior tracking and monitoring systems. As described in previous deliverables, the platform is expected to be launched in **beta release** at the end of month 8 (September 2014).

This deliverable concerns the technical characteristics of the Emma aggregator – the various components and learning environments and how they interact. It includes a detailed description of the aggregator's architecture and design.

Since deliverables D2.2 and D2.5 were about Interface specification, platform navigation and learning environment, this deliverable is related only to software and technical specification, including wireframe, case analysis and used codes.

2 Introduction

The purpose of the EMMA (European Multiple MOOC Aggregator) project is to showcase excellence in innovative teaching methodologies and learning approaches through the large-scale piloting of MOOCs (Massive Open Online Courses) on different subjects. To achieve this, EMMA will provide a system for the delivery of free, open, online courses in multiple languages from different European universities to help preserve Europe's rich cultural, educational and linguistic heritage and to promote real cross-cultural and multi-lingual learning.

The project will last for 30 months and began in February 2014. During this period, the 12 partners will collaborate together to provides 16 MOOCS as well as experience and expertise in the field of e-learning, learning analytics and innovative translation technology, supporting the implementation of the platform and its deployment and exploitation. The first MOOCs will be offered to the public in September 2014 and there will be a continuous flow of MOOC offerings in the following 22 months of the project. Learning data will be collected and analysed during the project, while an extensive communication campaign will disseminate the project results and outcomes.

The EMMA aggregator is an e-learning system that provides the functionality of a Learning Management System (LMS) – in providing features such as downloadable content, quizzes and learner activity tracking – but incorporates additional features, such as content translation and learners' increased control over their learning direction. In so doing it goes beyond the features of a typical MOOC.

EMMA will operate in two main modes. Firstly, EMMA is an aggregator[1] and hosting system of courses produced by European universities. To capitalize on this, the EMMA team is taking a deliberate multi-lingual, multi-cultural approach to learning by offering inbuilt translation

1 This document do not provide detailed information on how the aggregation facilities work. According to definition of the Emma policy access, the aggregation features will be described in the next deliverable on User Manual.

and transcription services for courses hosted on the platform. This removes from learners the necessity to study only MOOCs written in a language they understand and frees them to study any EMMA MOOC, regardless of the authoring language[1].

The second mode is that of a Personal Learning Environment (PLE), where learners may construct their own learning pathways using units from EMMA MOOCs as building blocks[2]. This extends the cross-cultural opportunities, allowing learners to select individual lessons (or part-lessons) from one or more MOOCs and add their own notes and bookmarks, as well as content from other sources. Thus, for example, a learner in one European country may study a MOOC from a different country, learning the subject from that culture's perspective, while studying selected parts of a MOOC on the same topic from their own country to compare viewpoint.

This deliverable concerns the technical characteristics of the Emma aggregator – the various components and learning environments and how they interact. It includes a detailed description of the aggregator's architecture and design.

3 Introduction: general principles

The conditions which are necessary to make an e-learning system viable are the following:

- network connection for access to learning content and development of learning activities on the Learning Management System (LMS);
- use of a personal computer, and other associated devices, as the main tool for participating on the course and in learning activities and events;
- a course which poses few constraints in terms of timetable and presence;
- continuous monitoring of learning activity and outcomes, through tracking devices as well as specific evaluation and self-assessment activities;

These conditions make it possible to evaluate:

- the multimediality: whether the different media codes have been effectively integrated to facilitate understanding of the content;
- the human interaction: to encourage the creation of shared learning spaces for groupwork and socialisation;
- the interaction with the materials: to encourage personalised learning pathways and optimise learning.

The term learning object can be used to define any object that forms part of the learning process and can be (re) used at different times and in different places, as long as the metadata is shared. The division of courses into stand-alone units (learning objects) makes it possible for individual units to be reassembled as part of other courses.

[1] The project will implement translation from Italian, Spanish, French, Portuguese, Dutch and Estonian

[2] This document do not provide information on how the PLE works. For detailed information, refer to the User manual.

into English, and from English into Italian and Spanish

Different classification systems are currently available for the organisation of content, but the two that are most commonly used in education are: AICC (Aviation Industry CBT Committee) and SCORM (Sharable Courseware Objects Reference Model). Various working groups are also involved in trying to define international specifications for learning objects. Examples of these are the Alliance of Remote Instructional Authoring and Distribution Network for Europe (ARIADNE) and PROmoting Multimedia access to Education and Training in EUropean Society (PROMETEUS). Another important point is the use of XML language (Extensible Markup Language) which allows for better atomisation of content and makes exportation to other devices more effective (e.g. handheld or wearble devices).

3.1 The technology

The basic component in any e-learning experience is the technological platform (LMS) that manages the delivery and exploitation of the learning. This type of management system allows for tracking of numbers of participants on each course and enables us to follow learner activity: which contenti s accessed, how long is spent on an activity or page; test results.

All the course and user information is stored and indexed on the platform database: this feature enables any user to access their own courses and learning space from any computer as long as it is connected to the Internet. Generally speaking, there is no need to download any special software to access the courses nor, indeed, is it always necessary to accept or save cookies on the user browser. Users are thus completely detached from a specific location, making it much easier for them to access their learning content modeled on the server, anytime and anywhere.
If the platform is the most important component of online learning, the virtual classroom (or shared learning space) is the methodology that allows for interaction between the learners and content. It comprises a set of tools that encourage collaboration and help to organise asynchronous conversation: discussion forums, documentary repository, access to teaching materials and supplementary study materials.

3.2 Learning content

There is no one format that the learning content has to follow, but generally speaking the approach adopted is to use the authoring software to create new, multimedia materials or to revamp existing digital content.

Whether created specially for EMMA or reworked, the learning content had to guarantee four main features:

- **Modularity:** the teaching materials need to be structured in learning modules or units so that users can devote smaller chunks of time to their course (approx. 15/20 minutes at a time) thus personalising both the way the learner approaches the content and the time they spend on it.

- **Interactivity:** any user needs to be able to interact with the learning material, and the material needs to respond effectively to the requirements for motivating human/machine interaction.
- **Totality:** each Learning Object needs to correpsond to a specific learning objective and should help learners achieve that objective.
- **Interoperability:** the teaching materials need to be structured so that they can be delivered, and the different learning activities can be tracked, on any platform, in line with the main standards (AICC, SCORM, IMS).

From a technical point of view, Learning Objects are described using XML specifications and/or specialist XML dialects, like EML and others, and these are interpreted by the browser as it interacts with the LCMS server to create the multimedia object, with the right features and the right sequencing in the course. A learning object can also have specifications as regards entry level, prerequisites and context for application.

4 The technical project

The EMMA platform is an evolution of an already existing platform called Federica developed by the University of Naples, which has been developed and extended to adapt its functions to the delivery of MOOCs.

The technical work on EMMA was structured as explained below:

1. ***Analysis of platform usability and accessibility****:* the aim of this was to author a set of user-guidelines, detailing functions relating to user experience. Methodology and results of this analysis were reported in **D.2.5 Interface Specification** authored by UNINA.

2. ***Definition of user functions****:* the aim was to identify the set of functions and features that would be avaialable to each user group, and what sequence of actions would be allowed.

3. ***Creation of graphic interface****:* using a wireframe (pencil sketch) as a starting point and specialised for each user group, mapping the individual functions and sequence of possible actions that would be available to each..

4. ***Creation of software:*** the software management, is an LCMS (Learning Content Management System) and comprises the following logic modules, which are grouped according to specialist function:

   a. **DataBase Module (DB)**: comprises the set of functions for accessing the database and the related document repository, as well as for implementing the data forms included in the DB;

   b. **Public Interface SW module**: refers to the set of functions and layout that are available to any anonymous visitor viewing the site;

   c. **Authentication SW module**: this constitutes the set of functions that allows users to be recognised either directly or indirectly (via social media), as well as the collateral functions, like forgotten password, modify registration data, etc.

   d. **Student Interface SW module**: the set of functions and layout that can be seen and used by Student users accessing the platform, who can register on a course, follow courses,

do exercises and quizzes, add their own notes and bookmarks, incorporate content from other sources;

e. **Teacher interface SW module**: the set of functions and layout that can be seen and used by Teacher users accessing the platform; it enables them to author courses and assessment modules, upload video files even in other formats, write and comment on blogs and communication features.

f. **Editor interface SW module:** the set of functions and layout that can be seen and used by Teacher users accessing the platform, but with the added extra that it is possible to create a course from materials that the official course teacher has provided; the functions are course authoring assessment module authoring, uploading of video files, even in different formats, editing blogs and communication.

g. **SysAdmin interface module**: regards overall platform configurations, adding or cancelling users and courses; configuration of various services offered by the server.

h. **Skill Assessment SW module**: the module that oversees the creation of new quizzes by teachers and manages student completion of these tests and exercises, elaborating and storing the results on the student dashbaord.

i. **Natural Language Processing SW module**: the module that handles finding results when a search string has been sent, comprising one or more words to make a phrase or sentence. Once the request has been made, other concepts resulting from the user's latest interctions, profile, the topics he/she follows on the blogs etc are associated to the search, using a *Knowledge Base*[2] approach. The engine will work progressively better as time goes on, thanks to the self-learning function. Once it is fully functional, the platform will offer a specific function making logical correlations between topics and learning content, and will allow for semantic searches across the learning content itself, expressed through phrases in natural language.

j. **Content tagging module**: the module that enables weighting to be given to specific concepts expressed in different courses, and to associate meaning, taxonomy, and ontology to them, either automatically – following certain rules – or manually. Once it is fully functional, the system will be able to deal with disambiguations and homonyms.

5. ***Platform installation, configuration and testing with a sample of users***: the system, which is made up of the modules mentioned above, was installed and configured on the test platform; a sample group of users in the different partner-countries were invited to test the platform to assess its effectiveness and note any bugs or weaknesses. This procedure was

[2] Da Wikipedia: A **knowledge base** (**KB**) is a technology used to store complex structured and unstructured information used by a computer system. The initial use of the term was in connection with expert systems which were the first knowledge-based systems. The original use of the term knowledge-base was to describe one of the two sub-systems of a knowledge-based system. A knowledge-based system consists of a knowledge-base that represents facts about the world and an inference engine that can reason about those facts and use rules and other forms of logic to deduce new facts or highlight inconsistencies.[1]

carried out for all types of prospective EMMA users. After, It has been implemented the roll-out during the production phase.

4.1 Types of users and Cases analysis

The type of user we expect to see on the EMMA platform are:

1. ***Anonymous visitor:*** The kind of general visitor who goes to an e-learning portal to look for a course. They usually arrive via a search engine because they were looking for something specific and they usually have a look at a course and its description. If they want to look at the course content, they are asked to register on the platform and enroll on the course. Anonymous visitors only have access to the platform content that is public and OPEN.

2. ***Identified user:*** Identified users have their own personal space on the platform, allowing them to maximise its potential and use all the learning tools available for registered users. As explained for anonymous visitors, users need to enroll to access the course content, but, unlike the anonymous visitor, they can insert the selected course in their list of courses they want to study and thus receive notifications when the course starts, new lessons are published or any changes are made.

3. ***Teacher:*** Teachers are identified users who have permission to create a course. They can access the backend of the platform to structure their courses (Course, Lessons, Units) and author the content. They can act as moderators, responding to user comments, and they can send official communications as comments. Teachers have to register on the platform first, and then ask UNINA for accreditation as a teacher, as this is a manual process that is handled directly by the platform administrators.

Cases Analysis

Use Case ID:			
Use Case Name:	**Course authoring**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Teacher**
Description:	This function allows for management of the course and its components, currently divided into Lessons and Units
Preconditions:	Registration and authorisation of Teacher account
Postconditions:	None
Special Requirements:	

Assumptions:	The teacher has authorisation to author and publish a course
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Course authoring information**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Teacher**
Description:	This function serves to manage information for introducing any course. The introduction is visible to all users, and serves to present the course through a video, text and additional materials.
Preconditions:	Registration and authorisation of Teacher account
Postconditions:	None
Special Requirements:	
Assumptions:	The teacher has authorisation to author and publish a course
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Authoring course lesson**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Teacher**
Description:	This function is for dividing the lessons into units, and for each lesson a title, abstract and learning objectives can be inserted, and the publication dates for each lesson can be planned.
Preconditions:	Registration and authorisation of Teacher account
Postconditions:	Nessuna
Special Requirements:	
Assumptions:	The teacher has authorisation to author and publish a course
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Authoring course unit**		
Created By:	Admin	Last Updated By:	Admin

Date Created:		Date Last Updated:	

Actor:	**Teacher**
Description:	This feature is for managing text and multimedia content, so adding videos, pictures, external links, tags, documents, pdfs etc as well as inserting title, text xontent and deciding whether the unit should be published or not.
Preconditions:	Registration and authorised teacher account
Postconditions:	None
Special Requirements:	
Assumptions:	The teacher has authorisation to author and publish a course
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Exploitation of course**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user, NON registered user**
Description:	You can see the course introduction, the lessons making up the course, and the suggested supplementary materials, the start date and, in some cases, the finish date of the course.
Preconditions:	None
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:			
Use Case Name:	Lesson exploitation		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user, User enrolled on a course**
Description:	The information regarding the lessons is only visible to users registered on the platform and enrolled on the course that contains the specific lesson.
Preconditions:	Registration and authorised teacher account
Postconditions:	None
Special Requirements:	
Assumptions:	The teacher is authorised to author and publish a course
Notes and Issues:	

Use Case ID:			
Use Case Name:	Exploitation of units		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user; User enrolled on a course**
Description:	The learning content can only be exploited by users who have enrolled on the course and previously registered on the platform.
Preconditions:	None
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:			
Use Case Name:	Add a comment		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user, Users enrolled on course**
Description:	A user who is only registered on the platform can comment on the course, while users who enrolled on the course, can comment on individual lessons and units too.
Preconditions:	None
Postconditions:	None
Special Requirements:	

Assumptions:	The teacher is authorised to author and publish a course
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Self-assessment tests**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user; User enrolled on course**
Description:	Any user who is enrolled on a course can do the self-evaluation tests, be they multiple choice or open questions. Results are visible immediately.
Preconditions:	Enrollment on course
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Annotation**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user, User enrolled on a course**
Description:	Stduents who are enrolled on a course can access their personal space and add notes to the unit they are studying, or insert pieces of text from the unit content in the *annotations* section.
Preconditions:	User has to be enrolled on a course
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:	

Use Case Name:	**Add TAG**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user, User enrolled on course**
Description:	Students can add a kind of twitter tag to their comments, replies and notes. This is visualised as a hypertext link that takes you to a selection of courses and other conversations with the same tag.
Preconditions:	
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Blog**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user**
Description:	Students have a personal mini-blog where they can share information or create discussions with their peers. Students can post text or multimedia content as well as tags.
Preconditions:	
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:			
Use Case Name:	**Profile**		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user, User enrolled on a course**

Description:	In the profile section, learners get their own dashboard and information regarding themselves (native language, name, email etc.) the courses they are following, and information about their learning activity on EMMA can be found here. From here, learners can access the notification system and also change certain personal settings.
Preconditions:	
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

Use Case ID:			
Use Case Name:	Assignement		
Created By:	Admin	Last Updated By:	Admin
Date Created:		Date Last Updated:	

Actor:	**Registered user, User enrolled on a course**
Description:	Once a learner has finished studying a lesson unit, they may be asked to complete an assignment. They can do the assignment or opt to carry on with the next unit instead.
Preconditions:	
Postconditions:	None
Special Requirements:	
Assumptions:	
Notes and Issues:	

4.2 List of general functions

Course enrollment	Any user can decide to enrol on a course even if the course has already started. An email and internal message system will inform the user on how the course is progrssing.It is possible for users to cancel their registration if they no longer wish to follow the course.
Users: Registered users, teachers	

Comments	Comments can be added, or made in response to existing comments, at the level of course, lesson or unit. Once

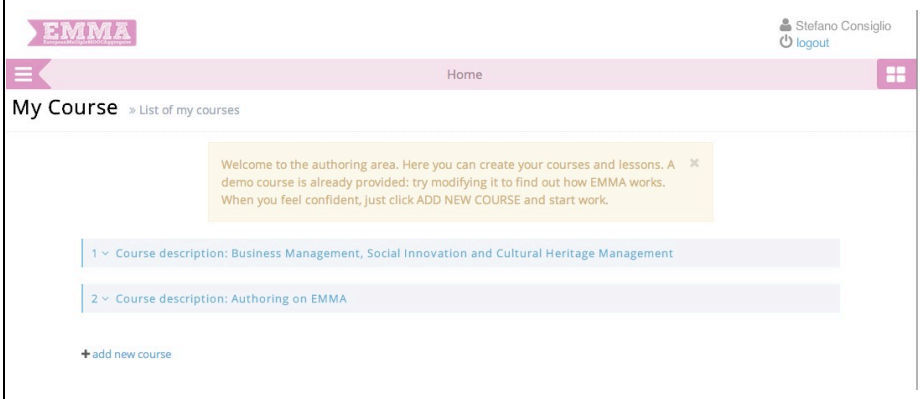
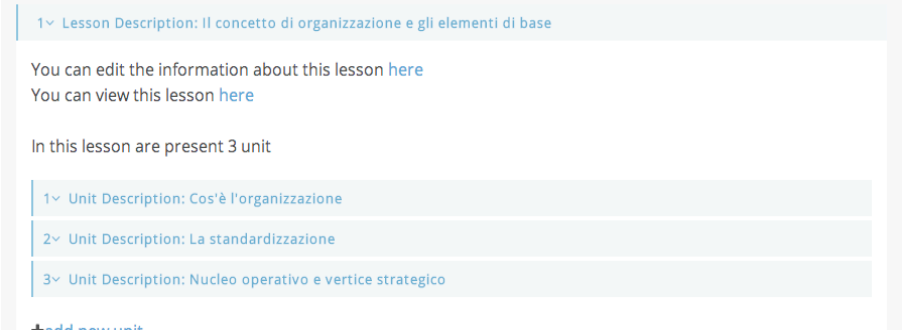
	fully functional, the conversations will also have a hashtag feature.
Users: Registered users, teachers	

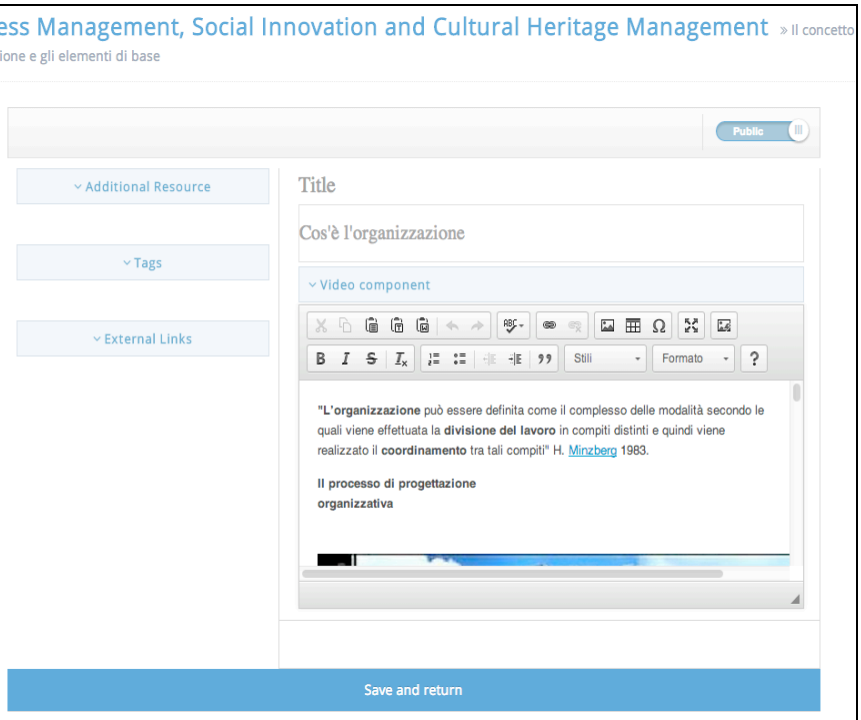
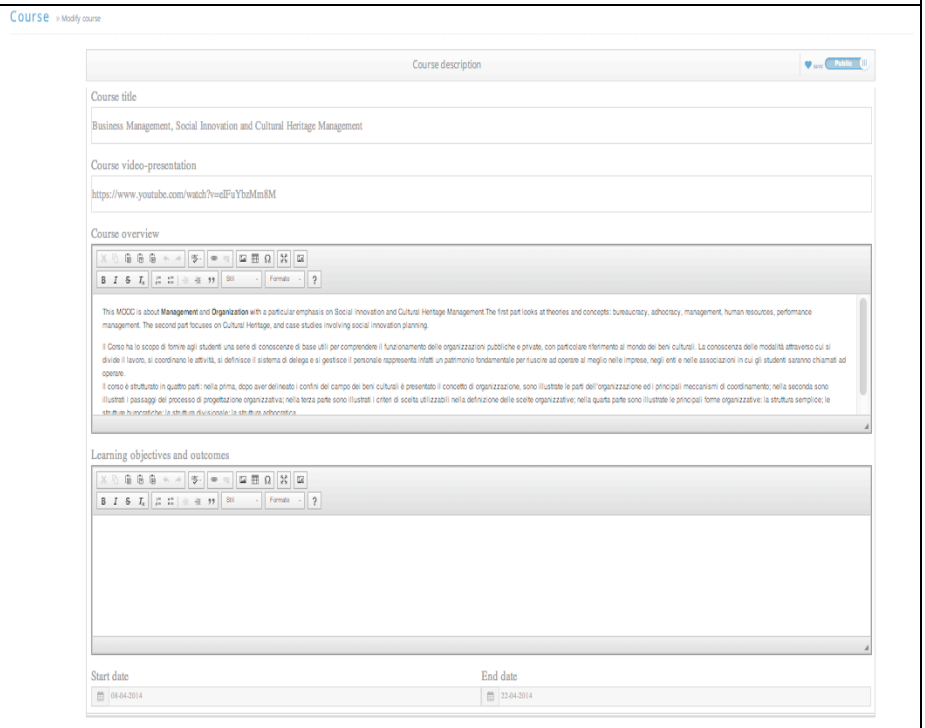
Notifications	A learner is mapped into the notification system any time they interact with any of the objects in the system (write a comment, comment a lesson, ecc.). A notification is reported by the sistem. Teacher communications to the class are also sent out via mail.
Users: Registered users, teachers	

Toolbook	The term *Toolbook* refers to the set of functions included on the platform toolbar that enable users to create their own personal list of courses they would be interested in doing, and a noticeboard to pin resources, links or specific content from the lessons or on the platform *(ref. to User Manual for detailed description)*.
Users: Enrolled users, teachers	

Course search	Any course search can be filtered by university or institution providing the course. Once fully functional, it will be possible to search for courses via specific content, based on the tags that each lesson includes.
Users: Anonymous users, Enrolled users, Teachers	

Course management (authoring)	This function enables teachers to manage their courses and its compnents. Course – Here a course can be added, as well as an introductory section, explaining the course content and some practical details. Lesson – Any lesson can contain various units. A description of the lesson and practical details (like publication date of lesson) can be added here too, and videos can also be included. Unit – This is where the actual learning content is found, in the form of videos, text, supplementary materials and external resources.
Users: teachers	

List of teacher courses	
Lesson structure within a course	
Unit structure within the lesson	
Inside the unit. This is where teachers can select a unit to add content or modify.	

	
Modifying course information	

4.3 Production environment

The whole of the platform has been developed using PHP (5.3.6) as far as the underlaying software is concerned, and the interoperability with DBs which was developed using MySql. Figure 1 illustrates the framework structure.

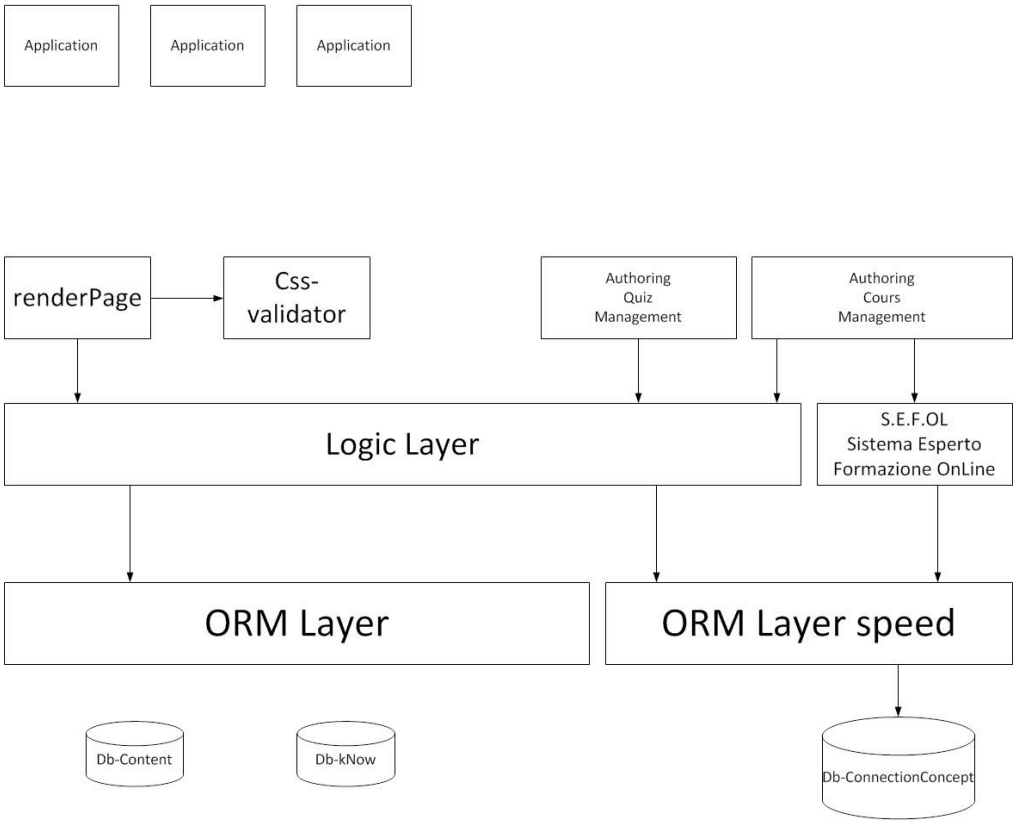


Figure 1 - Production environment

ORM Layer: The ORM layer acts as a wrapper for the data in the database, and highlights the functioning logic related to selection, modify and cancel data.

ORM Layer speed: Some functions are present at this level that use a faster provider for fetching data

Logical Layer: All the functions are present at this level, organised according to object

Below we list the different user interactions with EMMA. Not all of these have been developed for the 1.0 release of the platform. Certain other functions make explicit reference

to the Federica[3] platform where components have been taken from there and then optimised for EMMA.

4.4 Deployment Diagrams

4.4.1 Architecture

The software architecture is designed according to a Level structure. This gave us three basic levels for organising the functions and making sure that data is secure.

4.4.2 User level

This is the level concerning all functions based on the initial case analysis

4.4.3 Logical level

This is the level describing the logical functioning

4.4.4 Data Level

The data level concerns the management of data and information

[3] Note that Emma is in part the result of pre---existing software components integration, plus new component developed by other partners (es. Tracking system, dashboard).

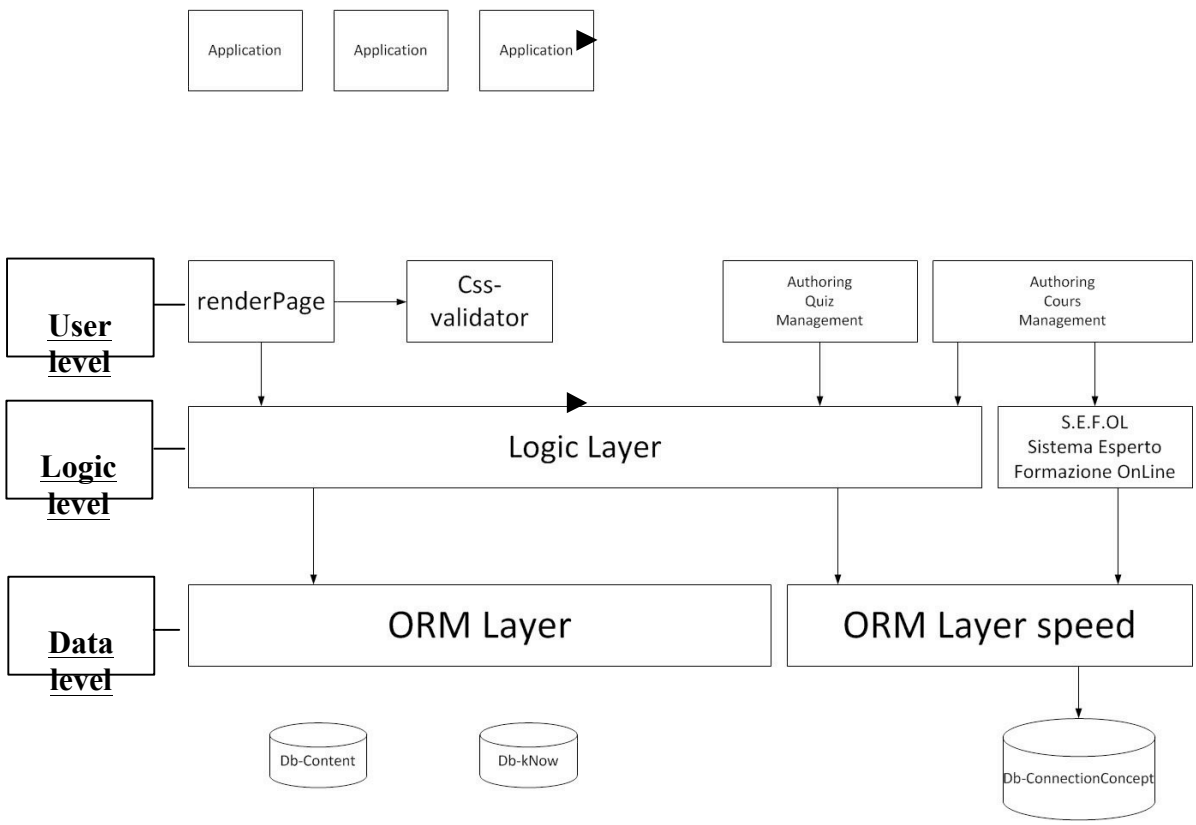


Figure 2 - Development diagram

4.5 Entities and relationships – Database

4.5.1 Diagram of db v 1.0

The first version of the data framework shows all the general functions present on the platform and illustrates the overall concept of how the platform works. This first version is a quick protype version which will be refined in future releases. The aim of this framework is to provide support for the decision-making process regarding which functions and features to implement and critical issues surrounding their inclusion.

This prototype diagram is a starting point and only includes the early functions and features adopted. It does include all the functions that have been requested yet, which is why there are no triggers, normalisation or data optimisation function. It does not take the scalability of the data or its visualisation into account to enable us to focus on "what we want from the platform" rather than "what we would like from the platform". The transition, which is fundamental for this first version of the database diagram, is linked to the model of engeeniring chosen for the project, which is prototypal with spiral diagram.

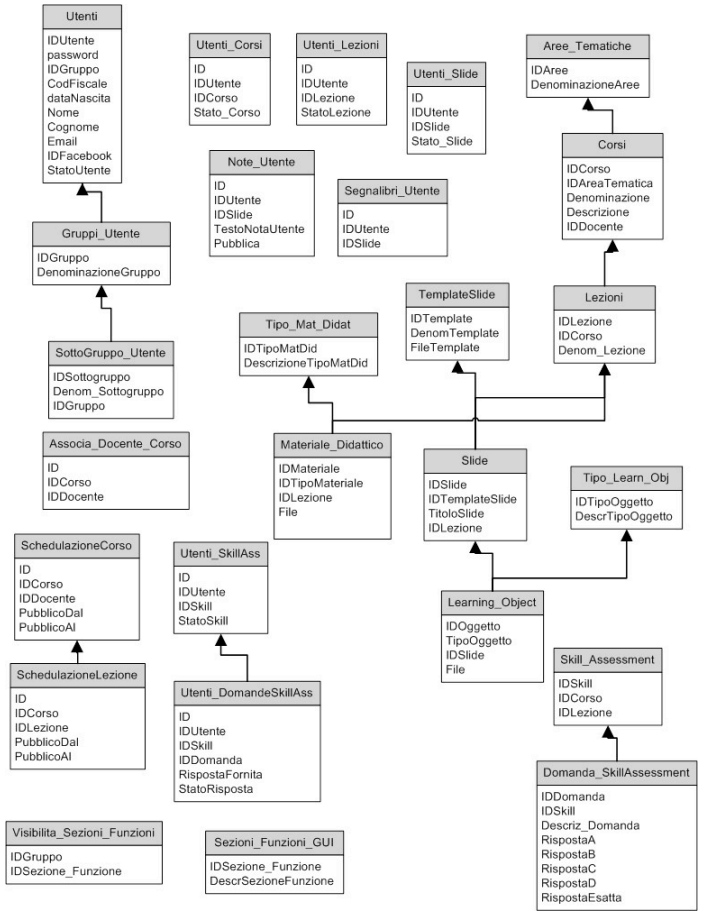


Figure 3 - Db diagram – v. 1.0

4.5.2 Diagram of Db v 1.2

The second DB release is an example of diagram integrated with more functions. Until the conclusion of project more functions and integration will be possibile, so the diagrams represented here are only representative of the current state of the platform (09/2014)

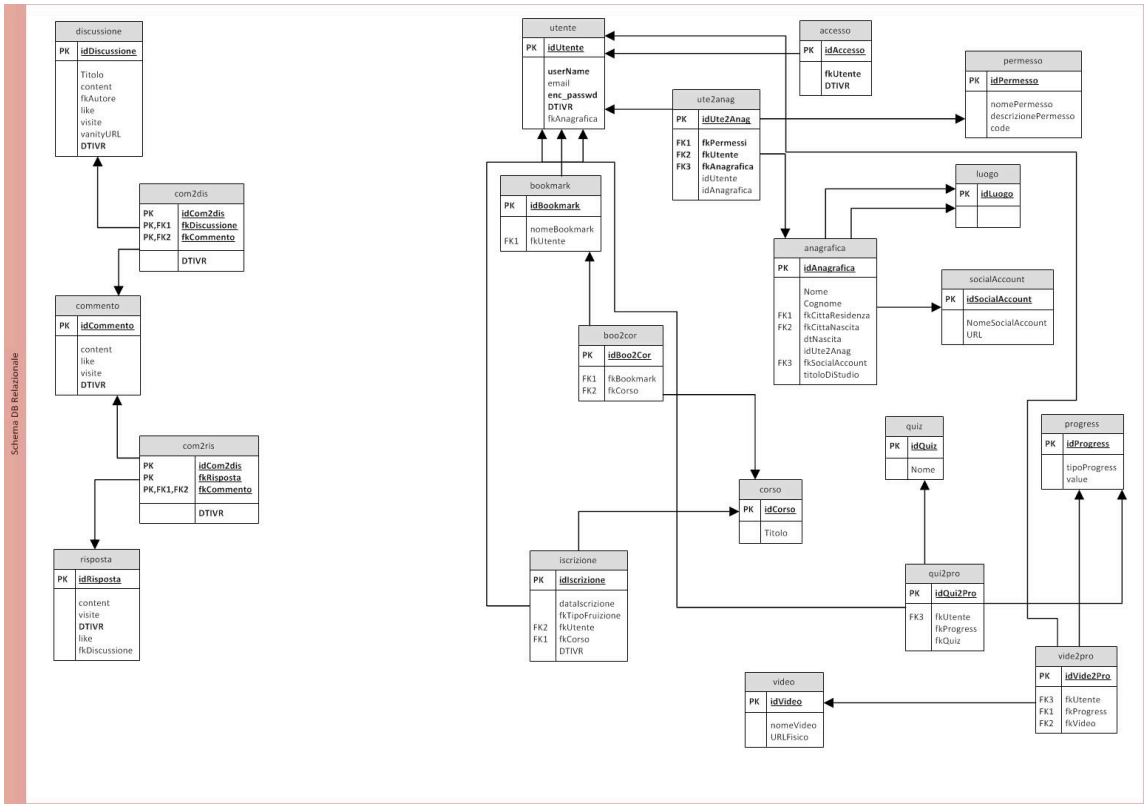


Figure 4 – Db Diagram – v. 2.0

4.6 External Components: the Integration Process

4.6.1 Tracking System

The tracking system is developed in PHP and Javascript. The PHP is used during the page construction to insert some code (HTML and Javascript) into the pages, and is also used to implement a web service that actually saves the user actions (events) to LearningLocker (the system which will be used to analyze user behaviour and build learning analytics). The javascript is used to actively detect implicit actions (such as enter or exit from a page), or explicit user actions on a page such as clicks, mouse moves, etc...

When the javascript code detects one of these actions is on the EMMA platform, a request is sent to the tracking system backend which analyzes the information and call the web service to send these data to the Learning Locker instance.

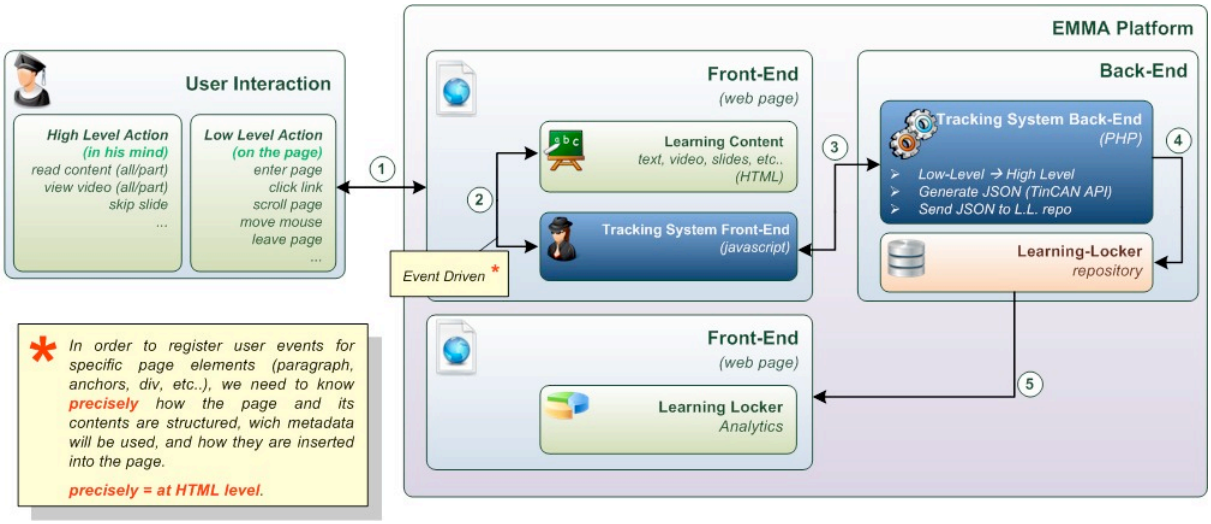


Figure 5 - Tracking system diagram

The tracking system has been designed in order to minimize the amount of work needed for its integration into the EMMA platform. To achieve this result, we added a specific HTML attribute (`ets_action`) in the HTML tags of the elements (buttons, links) that performs the functions that have to be tracked.

When a page is loaded, the tracking system analyzes its content looking for elements with that attribute, and bind the click/touch event with a specific function of the tracking system that is in charge of sending the data to the learning locker instance.

In some particular pages an additional function automatically tracks when the user enter and/or exit from a page and it can also analyzes whether the user is active or inactive during the time spent on the page. This last information can be combined with additional information about the number of words/characters included in the page, and the duration of a video presentation, to detect actual reading speed or to estimate the idleness of a user.

All the various features can be enabled or disabled on individual pages depending on the specific needs. At this stage, the main features are:

- Record when the user entrer in a page
- Record when the user leave a page
- Capture internal events/operations
- Detect user idleness

Once these operations are carried out, the information is sent to LearningLocker in a JSON format string which include all the information needed to identify the user, the action, the object and the context.

This is an example of the information sent to LearningLocker, that will be used to build learning analytics:

```
{
    "actor": {
        "name": "User Name",
        "mbox": "mailto:username@csp.it ",
        "objectType": "Agent"
    },
    "verb": {
        "id": "http://activitystrea.ms/schema/1.0/visited",
        "display": {
            "en---US": "visited"
        }
    },
    "object": {
        "id": "http://platform.europeanmoocs.eu/course.php?cor=28",
        "definition": {
            "type": "http://adlnet.gov/expapi/activities/course",
            "name": {
               "en---GB": "Alteracoes Climaticas: o contexto das experiencias de vida"
            },
            "extensions": {
               "http://id.tincanapi.com/extension/datetime": "2014---09---08 08:56:20",
               "http://id.tincanapi.com/extension/duration": "186878",
               "http://id.tincanapi.com/extension/idleDuration": "116135",
               "http://id.tincanapi.com/extension/videoDuration": "0",
               "http://id.tincanapi.com/extension/tags": {
                   "2": "#imocac_TF",
                   "3": "#imoocac14_s6",
                   "1": "imooac14_s2",
                   "0": "lesson 1"
               }
            }
        }
    },
    "context": {
        "instructor": {
            "name": "Teacher Name",
            "mbox": "mailto:teachername@csp.it",
            "objectType": "Agent"
        },
        "contextActivities": {
            "parent": [
               {
```

```
                "id": "http://platform.europeanmoocs.eu/course.php?cor=28",
                "objectType": "Activity"
            }
        ],
        "grouping": [
            {
                "id": "http://wp.europeanmoocs.eu/course.php?cor=28",
                "objectType": "Activity"
            }
        ]
    }
},
"authority": {
    "name": "",
    "mbox": "mailto:user@csp.it",
    "objectType": "Agent"
},
"id": "b3a6ceec---d3a9---4dc0---a837-
--217d5fa5be36", "stored": "2014---09-
--08T09:14:20.674600+00:00", "timestamp":
"2014---09---08T09:14:20.674600+00:00"
}
```


In this specific example, the tracked action can be summarized with the following sentence:

```
User Name visited "Alterações Climáticas: o contexto das experiências de vida" | 5 minutes ago (Wed, Sep 8, 2
014 9:14 AM)
```

All these functions are completely transparent to the user. In particular, the user does not need to do anything to inform that he is leaving a page, because this is automatically detected by the tracking system (even if the browser/app is closed).

The system is designed to be multi platform and has been tested on all the major operating systems, browser, and devices (included Android and Apple smartphones and tablets).

Since the tracking system has been developed separately from the platform, some special attention has been taken into consideration in order to avoid conflicts. For examples in both PHP and Javascript, as well as in the database, all the variables, classes, functions and tables, use a particular naming convention where a prefix (`ets_`, standing for emma tracking system) was added to the name.

4.6.2 Learning analytics Dashboard

xAPI statements will be retrieved from Learning Locker to develop dashboards by using Highcharts.com. EMMA platform provides different dashboards for two types of users - students and instructors. The aim of implementing the learning analytics dashboard is to support awareness and sense-making of the learning experience in online learning settings. Students' dashboards enable to raise students' awareness of their learning activities by providing overview of the progress or social structures in the course context. Instructors' dashboard provides feedback to course designers about the activities during the course.

In the pilot phase instructors' dashboard visualizes enrollment activity (see Figure 6) where instructor can see the enrollment and unenrollment activities during specific time (week, month).

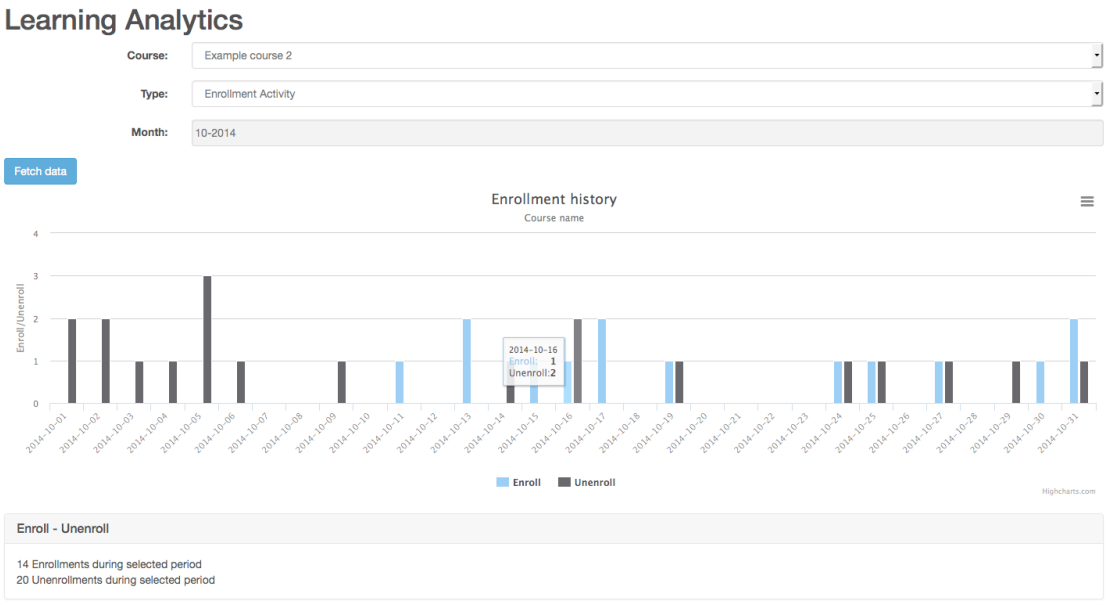


Figure 6 - Instructors' dashboard enrollment activities in pilot phase

In future the overview of accessed learning materials, performed assignments and emerged social structures will be also visualized to instructors.

In pilot phase students' dashboard will visualize:

- Accessed learning materials. Student X can see what are ten the most popular learning resources used by him/her. Dashboard (see Figure 7) also visualizes ten the most popular learning resources of the course (does not count the views of Student X) by peers on that MOOC. Highlighted (orange) indicates the resources that Student X has not yet accessed.

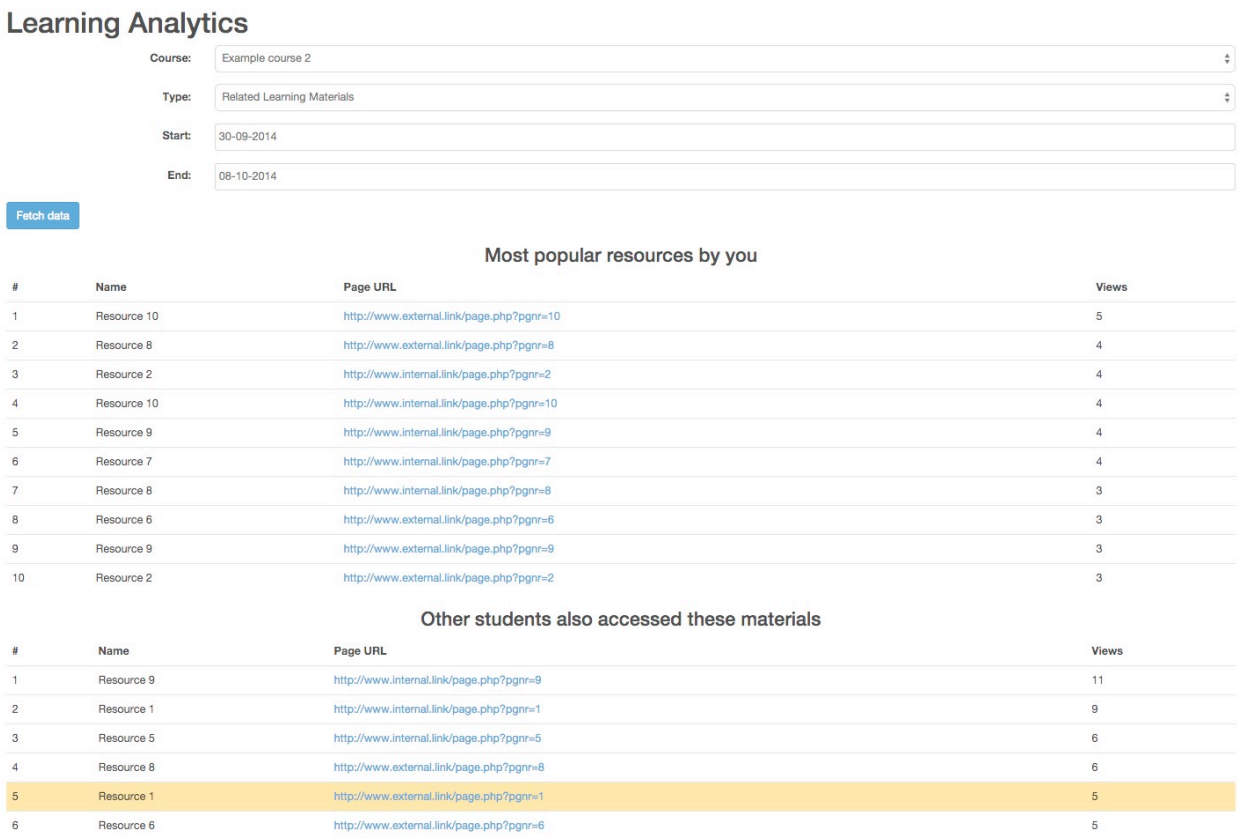


Figure 7 - Students' dashboard accessed materials

4.7 Translectures and translation Services

In order to make accessible EMMA MOOCs to audiences of different countries and languages, the EMMA consortium has made use of the transLectures Platform (TLP) hosted at the UPV to generate transcriptions and translations of all video contents from these MOOCs, as well as the Document translation Platform (DTLP), which is actually an adaptation of TLP developed specifically for the EMMA project and devoted to translate MOOC textual contents into several languages. The following sections explain in detail both platforms.

4.7.1 The transLectures Platform (TLP)

The transLectures Platform (TLP) is an open source, self-contained piece of software developed by the UPV which includes everything that is needed in order to integrate transcription and translation technologies into a remote video lecture repository. It includes also a set of interfaces and web tools to allow casual users to transcribe and translate video and audio files under a cloud-based service.

TLP has been developed as part of the European research project transLectures. transLectures (acronym of Transcription and Translation of Video Lectures) was an EU (FP7-ICT-2011-7) STREP project in which advanced automatic speech recognition (ASR) and machine translation (MT) techniques were tested on large educational video repositories ("video lectures"). The project began in November 2011 and finished in October 2014. Further information about this project can be found at the official web page: http://translectures.eu. TLP is freely available and its latest release (TLP 1.0.1) can be downloaded from the transLectures official web page: http://translectures.eu/tlp.

TLP is divided mainly in four components: Database, Web Service, Ingest Service and Player. Figure 8 shows the main components of TLP and a simplification of all possible interactions between them.

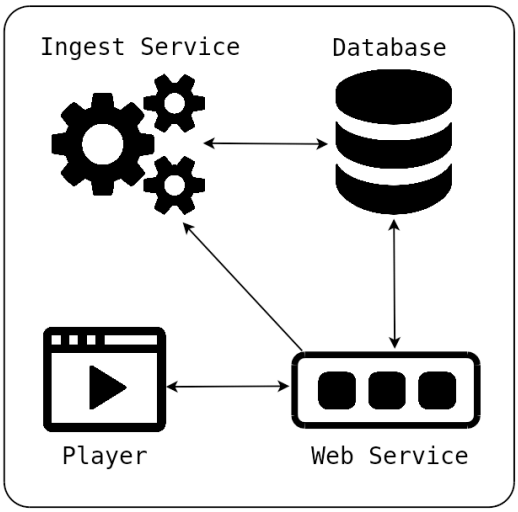


Figure 8 - Main components of the transLectures Platform and a simplification

Database

The transLectures Database is a PostgreSQL relational database which stores all the data required for the Web Service and the Ingest Service. The main categories of data stored in the Database are as follows:

- **Video lectures:** All the information related to a specific lecture is stored in the database, including language, duration, title, keywords and category. In addition, an external ID, provided by the client repository, is stored and used for lecture identification purposes in all transactions performed between the client and the transLectures Player and Web Service.

- **Speakers:** Information about the lecture speaker can be used by the ASR system to adapt the underlying models to the unique characteristics of a given speaker and, in doing so, improve the quality of the resulting subtitles.
- **Subtitles:** All subtitles generated via the Ingest Service are stored in the Database and retrieved by the Web Service.
- **Uploads:** Every time an ***/ingest*** operation is performed, a new upload entry is stored in the Database.

Media and subtitle files are stored on the hard drive separately to the relational database. These files, together with the relational database, are what we refer to as the transLectures Repository.

Subtitle files are kept in DFXP format, which is an extension of the TTML format that allows to keep track of user modifications. Below is shown an example of a DFXP file, in which two different users supervises and automatic segment.

```
<?xml version="1.0" encoding="utf---8"?>
 <tt xml:lang="en" xmlns="http://www.w3.org/2006/04/ttaf1"
 xmlns:tts="http://www.w3.org/2006/10/ttaf1#style" xmlns:tl="translectures.eu">
  <head>
    <tl:d aT="automatic" aI="UPV---v1.0" wS="UPV---v1.0" tS="2012---10---03T21:32:52" aC="0.56"
    cM="0.75" videoId="00505---Profesores_Alcoy.M03.B01" b="0.0" e="12.50"/>
  </head>
  <body>
    <tl:a>
      <tl:s  sI="1" cM="0.75" b="0.0" e="3.20">
        <tl:w cM="0.85" b="0.0" e="0.75">most</tl:w>
        <tl:w cM="0.89" b="0.75" e="0.95">of</tl:w>
        <tl:w cM="0.63" b="0.95" e="1.15">you</tl:w>
        <tl:w cM="0.40" b="1.15" e="1.35">are</tl:w>
        <tl:w cM="0.90" b="1.35" e="1.50">probably</tl:w>
        <tl:w cM="0.85" b="1.50" e="1.75">ventured</tl:w>
        <tl:w cM="0.55" b="1.75" e="2.00">the</tl:w>
        <tl:w cM="0.98" b="2.00" e="2.75">problem</tl:w>
        <tl:w cM="0.60" b="2.75" e="3.20">that</tl:w>
      </tl:s>
      <tl:s sI="1" aT="human" aC="0.81" cM="1.0" aI="John" b="0.0" e="3.20" tS="2012---10---04T13:31:45">
        most of you probably ventured into the problem set
      </tl:s>
      <tl:s sI="1" aT="human" aC="0.97" cM="1.0" aI="Carlos" b="0.0" e="3.20" tS="2012---10-
        --07T21:12:43"> Most of you have probably ventured into the problem set.
      </tl:s>
    </tl:a>
    <tl:s sI="2" cM="0.19" b="8.5" e="12.50">
```

```
      <tl:w cM="0.1" b="8.5" e="9">To</tl:w>
      <tl:w cM="0.2" b="9" e="10">solve</tl:w>
      <tl:w cM="0.1" b="10" e="10.7">on</tl:w>
      <tl:w cM="0.1" b="10.7" e="12.5">this</tl:w>
    </tl:s>
  </body>
</tt>
```

Web Service

The Web Service is the interface for transfering information and data between the client and the transLectures Platform. It also enables the subtitle visualisation and editing capabilities of the transLectures Player. This Web Service is implemented as a Python Web Server Gateway Interface (WSGI) and defines a set of HTTP interfaces related to subtitle delivery and media upload:

- ***/ingest***: POST request which allows the client to upload audio/video files and other related material, such as slides and other text resources that can be used to adapt the ASR system, together with other metadata in a Media Package File (MPF).
- ***/status***: GET request to check the status of a video lecture uploaded via the /ingest interface.
- ***/lecturedata***: GET request that returns basic metadata and file locations for a given video lecture.
- ***/langs***: GET request that provides the client with a list of subtitles and languages available for a given video lecture.
- ***/dfxp***: GET request that returns the subtitles in DFXP format for a given lecture and language.
- ***/mod***: POST request that sends and commits changes made by a user when editing a transcription or translation.

Ingest Service

The transLectures Ingest Service is the tool devoted to handling and properly processing the Media Package Files (MPF) uploaded via the ***/ingest*** interface of the Web Service. It is implemented as a Python module that should be executed periodically (typically every minute) to check for new lecture uploads and their processing and to assess whether existing uploads are being processed correctly. The uploads table of the Database is used to keep the status of every upload up-to-date. This information is also accessed by the Web Service's ***/status*** interface.

As previously mentioned, MPFs are uploaded to the transLectures Platform via the Web Service's ***/ingest*** interface and are stored in the Database. Then the Ingest Service reads the uploads table of the Database and starts processing each MPF. An upload will typically follow the sequential steps shown in Figure 9.

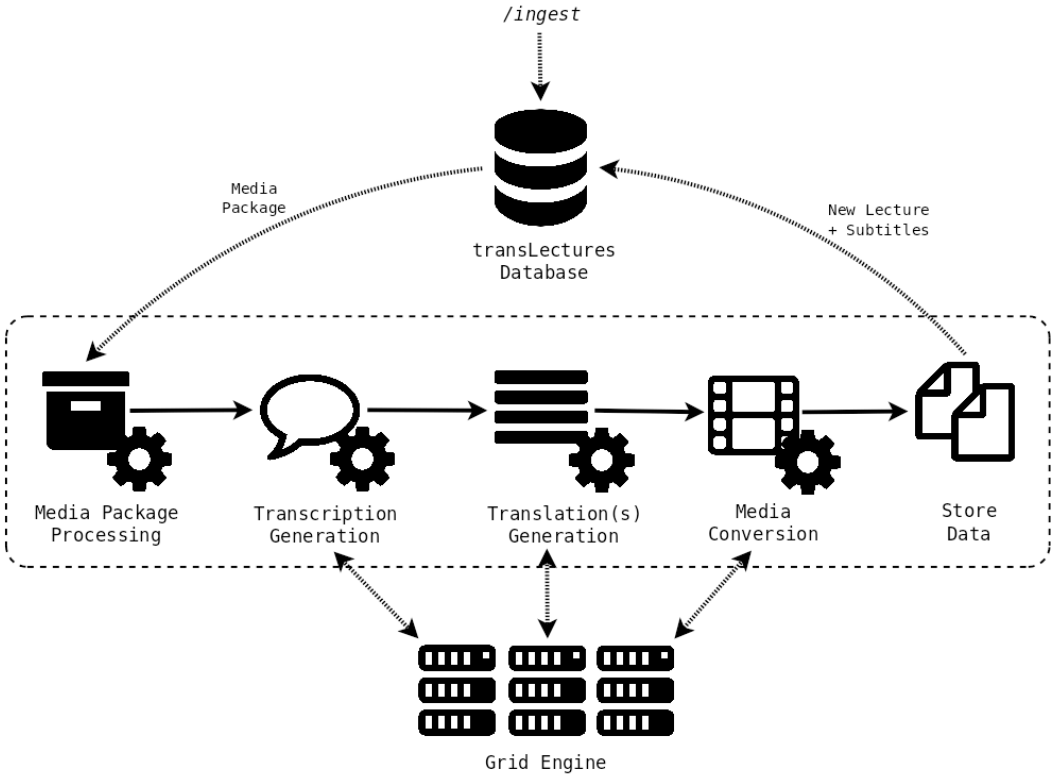


Figure 9 - TransLectures Ingest Service workflow for an uploaded media package

Uploaded media files by EMMA partners has been transcribed and translated by using the proper Automatic Speech Recognition (ASR) and Machine Translation (MT) systems developed by the UPV which are described in detail in the Deliverable D2.3.1.

Player

The transLectures Player is an HTML5 tool which allows users to review and modify the automatic transcriptions and translations of a given media file. Please see Figure 10.

The main features of the transLectures Player are:

- HTML5 technology.
- User authentication system including multiple user confidence levels.
- Multiple editing screen layouts.
- Simultaneous transcription and translation post-editing.
- Keyboard shortcuts to enhance user experience.
- Designed to be easily integrated in any corporative/institutional web layout.

Figure 10 - TransLectures player for translation editing

4.7.2 The Document Translation Platform (DTLP)

The Document Translation Platform (DTLP) is an adaptation of the transLectures Platform (TLP) which allows the translation of text documents in multiple languages. This adaptation was created by the UPV under the EMMA project framework.

Being an adapted version of TLP, DTLP is also compounded by four main components, which have been adapted to the requirements of this particular task. These components are the Database, Web Service, Ingest Service and Web Editor (Player in TLP). The basis and the workflows involved are almost the same as in TLP. Therefore, we will focus on the description of the changes of DTLP with respect to TLP.

Database

The main categories of data stored in the Database are the following:

- **Documents:** All the information related to a specific document is stored in the database, including the source language and the number of sentences. An external ID, provided by the client, is stored and used for document identification purposes in all transactions performed between the client and the Web Editor or Web Service.
- **Translations:** All translations generated via the Ingest Service are stored in the Database and retrieved by the Web Service.
- **Uploads:** Every time a ***/upload*** operation is performed, a new upload entry is stored in the Database.

Text documents along with their translations are stored in a custom XML file format called DTLX (Document TransLation XML) clearly inspired in the TLP's DFXP format, in which the history of user editions is tracked for every sentence of the document. Below is shown an example of a DTLX file in which a single sentence has been automatically translated and afterwards amended by a user.

```
<?xml version="1.0" encoding="utf---8"?>
<tt xmlns:tl="translectures.eu">
  <head>
    <tl:d dI="Document_1" sL="it" tL="en" st="fully_human" />
  </head>
  <body>
    <tl:s sI="1">
      <tl:o> Spero di avere presto tue notizie. </tl:o>
      <tl:a>
        <tl:t aT="automatic" aI="MT_UPV" tS="2014---05---21T13:17:10"
          cM="0.472"> I hope hear you soon.
        </tl:t>
        <tl:t aT="human" aI="Jorge Civera" tS="2014---05---30T09:12:23"
          aC="0.7"> I hope to hear from you soon.
        </tl:t>
      </tl:a>
    </tl:s>
  </body>
</tt>
```

Full specifications of the DTLX format can be found in Annex A.

Web Service

The Web Service is the interface used for transfering information and data between the client and DTLP. It features the following interfaces related to document upload and retrieval:

- ***/upload***: POST request which allows the client to upload untranslated document files in DTLX format.
- ***/status***: GET request to check the status of a document uploaded via the /upload interface.
- ***/langs***: GET request that provides the client with a list of translations available for a given document.
- ***/get***: GET request that returns a translated document in DTLX format.
- ***/mod***: POST request that sends and commits changes made by a user when manually editing the translation of a particular document.

Ingest Service

The Ingest Service is the tool devoted to process the uploaded documents via the ***/upload*** interface and translate them into the requested target language. In DTLP the upload workflow becomes very simple: after validating the format of the uploaded DTLX document, it is

translated into the target language by making use of the proper MT system (which are described in Deliverable D2.3.1), and finally it is stored in the database.

Web Editor

A new Web DTLX Editor has been neatly designed from scratch to allow users to manually review and correct translation errors from the translated documents. The saved amendments are sent back to the Web Service using the ***/mod*** interface. Figure 11 shows a capture of the Web Player when editing a Spanish into English translation.

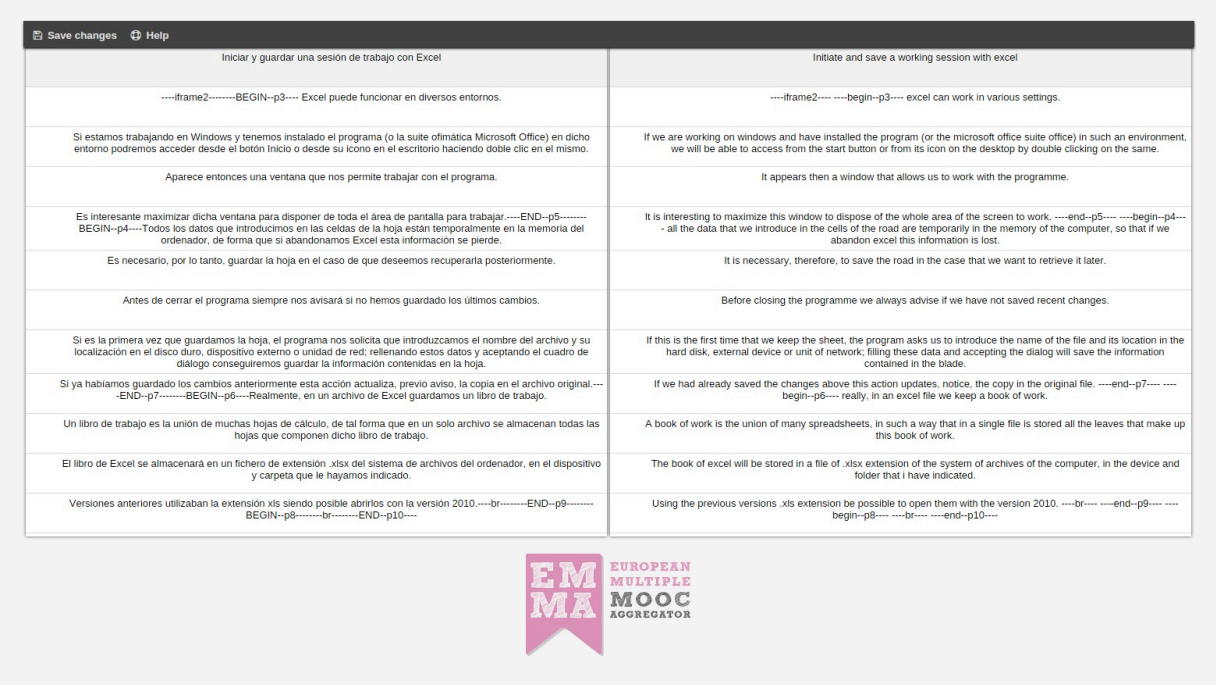


Figure 11 - Document Translation Web Editor

Annex A: The DTLX Format

1. Introduction

This Annex defines an extension to the XML format for storing text translations in the EMMA project. This format will be named Document TransLation XML (DTLX).

2. Namespace

A new namespace has been created to reflect the XML changes, so the new tags will be called <*tl:XXX*> where tl is the new header and *XXX* is the tag. A reference to a non-existing XSL file is added for XML validation at the opening <*tt*> tag:

<*tt xmlns:tl="translectures.eu"*>

This is needed to make this extension valid for any XML parser.

3. Use cases

1) A user generates a DTLX file, defining the source language, the target language and sentences that will be translated.

2) A user uploads a DTLX file to the transLectures ingest service, which is processed and translated into the requested language. Translations are included in the DTLX file.

3) A user supervises automatic translations. Amended translations are appended to the DTLX file.

4. Tags

A set of tags is proposed to store the required data:

4.1 Document tag *(<tl:document>)*

It defines metadata and translations at the top level. Abbreviature: ***<tl:d>.*** Attributes:

- ***documentId***: Document identifier. Abbreviature: ***dI***.
- ***srcLang***: Source language tag in ISO 639-1 format. Written language of the input sentences. Abbreviature: ***sL***.
- ***trgLang***: Target language tag in ISO 639-1 format. This is the language into which source sentences will be translated. Abbreviature: ***tL***.
- ***status***: It provides information of the status of the document. Abbreviature: ***st***. Possible values:

- *not translated* → DTLX file contains only source sentences, without any translation.
- *fully automatic* → All sentences have been translated by an automatic translation system.
- *partially human* → Some sentences have been supervised by a person.
- *fully human* → All sentences have been supervised by a person.

4.2 Sentence tag *(<tl:sentence>)*

It declares a sentence, including the source language text and all possible translations (both automatic or manually edited). Abbreviature: ***<tl:s>***. Attributes:

- ***sentenceId***: Sentence identifier. Abbreviature: ***sI***.

4.3 Source sentence tag *(<tl:origin>)*

Sentence text in the source language. Please avoid multiple sentences in one long sentence (more than 50 words). Abbreviature: ***<tl:o>***

4.4 Alternative tag *(<tl:alt>)*

The objective of this tag is to maintain all alternative sentences (source or target) and a log of modifications made to each sentence. Abbreviature: ***<tl:a>***.

4.5 Translation tag *(<tl:translation>)*

Translation for the source text in the target language. This tag contains some attributes used to identify the author of the translation, among others. Abbreviature: ***<tl:t>***. Attributes:

- ***authorType***: Type of author. Their values are automatic or human. Human for those translations generated by human experts or completely supervised by human experts. Automatic for those translations fully generated by an MT system. Abbreviature: ***aT***.
- ***authorId***: Author identifier. For example: UPV, Jorge Civera, etc. Abbreviature***: aI***.
- ***authorConf***: Confidence measure of the author, from 0 to 1, when the authorType is human. This attribute is associated with an authorId. This tag could be useful for non-native users supervising a foreign language. Abbreviature: ***aC***.
- ***confMeasure***: Confidence measure of the sentence, from 0 to 1. These values are generated by MT systems. It can be understood as the degree of confidence that the automatic system has about the correctness of the translation. Abbreviature: ***cM***.
- ***timeStamp***: Instant of creation or modification. The timestamp format is a combination of data and time of day in Chapter 5.4 of ISO 8601. The format is [-]CCYY-MM-DDThh:mm:ss[Z|(+|-)hh:mm]. Abbreviature: ***tS***.

5. Use cases examples

1) A user generates a DTLX file, defining the source language, the target language and sentences that will be translated.

```
<?xml version="1.0" encoding="utf---8"?>
<tt xmlns:tl="translectures.eu">
  <head>
    <tl:d dI="Document_1" sL="it" tL="en" st="not_translated" />
  </head>
  <body>
    <tl:s sI="1">
      <tl:o> Spero di avere presto tue notizie. </tl:o>
    </tl:s>
    <tl:s sI="2">
      (...)
    </tl:s>
    (...)
  </body>
</tt>
```

2) A user uploads a DTLX file to the transLectures ingest service, which is processed and translated into the requested language. Translations are included in the DTLX file.

```
<?xml version="1.0" encoding="utf---8"?>
<tt xmlns:tl="translectures.eu">
  <head>
    <tl:d dI="Document_1" sL="it" tL="en" st="fully_automatic" />
  </head>
  <body>
    <tl:s sI="1">
      <tl:o> Spero di avere presto tue notizie. </tl:o>
        <tl:t aT="automatic" aI="UPV" tS="2014---05---21T13:17:10"
          cM="0.472"> I hope hear you soon.
        </tl:t>
    </tl:s>
    <tl:s sI="2">
      (...)
    </tl:s>
    (...)
  </body>
</tt>
```

3) A user supervises automatic translations. Amended translations are included in the DTLX file.

```
<?xml version="1.0" encoding="utf---8"?>
<tt xmlns:tl="translectures.eu">
  <head>
    <tl:d dI="Document_1" sL="it" tL="en" st="partially_human" />
  </head>
  <body>
    <tl:s sI="1">
      <tl:o> Spero di avere presto tue notizie. </tl:o>
      <tl:a>
        <tl:t aT="automatic" aI="UPV" tS="2014---05---21T13:17:10"
          cM="0.472"> I hope hear you soon.
        </tl:t>
        <tl:t aT="human" aI="Joan Albert Silvestre" tS="2014---05---30T09:12:23"
          aC="0.7"> I hope to hear from you soon.
        </tl:t>
      </tl:a>
    </tl:s>
    <tl:s sI="2">
      (...)
    </tl:s>
    (...)
  </body>
</tt>
```