



## Report concerning the infrastructure design and testing of modules

<b>Project Title</b>	<b>Multichannel Adaptive System Training for micro, small and medium Enterprises</b>
<b>Project Acronym</b>	<b>MASTER</b>
<b>Grant Agreement Number</b>	<b>2010-1-PL1-LEO05-11470</b>
<b>Deliverable Type</b>	<b>Report</b>
<b>Deliverable Number</b>	<b>D9 Draft of 31 August</b>
<b>Date of Delivery</b>	<b>30/11/2011</b>
<b>Author(s)</b>	<b>Bálint Tillman (Corvinno), Anita Pintér (Corvinno)</b>
<b>Editor</b>	<b>dr András Gábor (Corvinno)</b>
<b>Related Work Package</b>	<b>WP3</b>
<b>Availability of Deliverable</b>	<b>Internal Document</b>



This project has been funded with support from the European Commission under the Lifelong Learning Programme. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

## Table of Contents

1. Introduction.....	3
2. Studio overview.....	4
2.1. Educational Ontology.....	4
2.2. Content Authoring .....	8
2.3. Repository.....	8
2.4. Test Bank.....	8
2.5. Packaging.....	9
2.6. Process of Content Development.....	9
2.7. Adaptive Testing Engine.....	10
2.8. Results and learning statistics .....	11
2.9. Mobilized content delivery system (user perspective).....	11
2.10. Non-functional Capabilities.....	11
3. Architecture.....	13
3.1. Connections to External Systems.....	13
3.2. Internal design.....	14
4. User specifications.....	16
5. GUI Specifications.....	18
6. Use cases.....	21
6.1. External system.....	21
6.2. Studio – User.....	23
6.3. Studio - Trainer .....	26
6.4. Emphasized tasks for Administrator.....	36

## 1. Introduction

The purpose of this Master project deliverable is to describe the main concepts of the ontology based electric learning environment (Studio) highlighting its infrastructure design and its functionalities in details. It helps better understanding the philosophy of the system background and gives detailed instructions for the further project activities and for the Master pilot implementation.

The document consists of 3 pillars. At first it gives an overview of the learning system with defining the system components, through their roles and functionalities. It also focuses on the learning infrastructure with describing the repositories and their main characteristics.

The second section contains detailed information of the system architecture including the external accesses, authentication and design components.

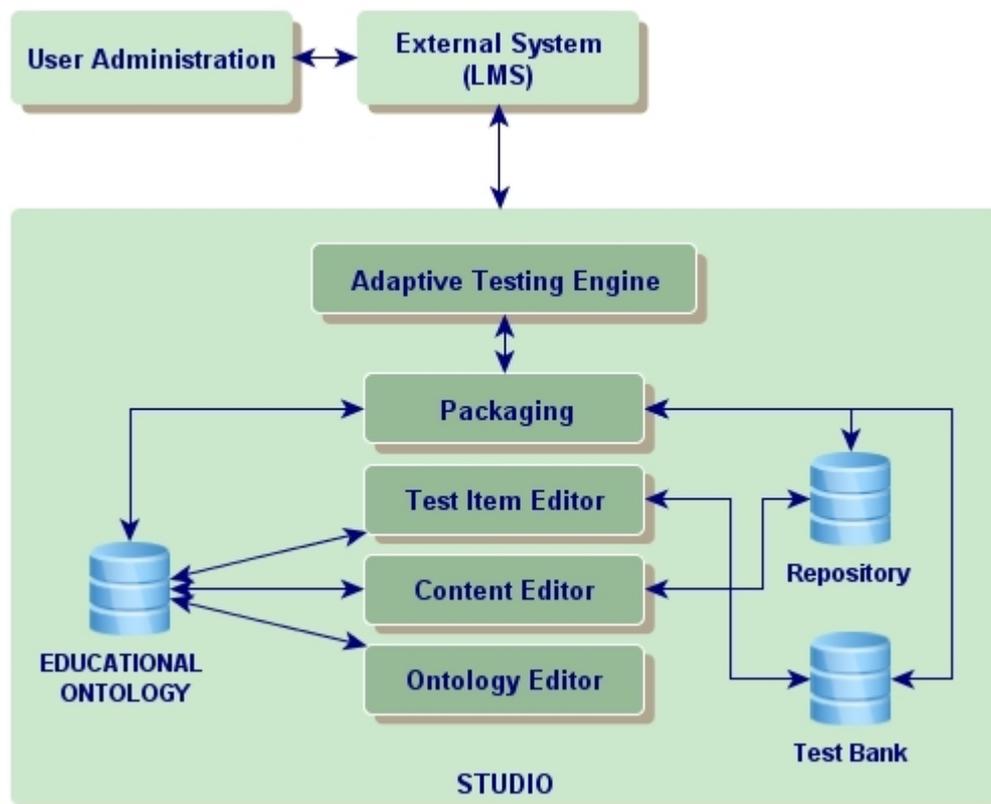
Finally the biggest part describes the specifications including user and GUI specifications and gives an overview of use cases and tasks with using illustrations to help better understanding the main context of the Studio system.

This document is offered for those experts and involved partners who are working on the content development, building up the ontology model, and having administrator rights to the ontology based electric learning system.

## 2. Studio overview

The main goal was to create a web-based learning infrastructure which supports the whole learning cycle, independently from its form (e.g. workstation- or mobile phone-based learning). For the proper learning experience Studio is needed to be attached to a Learning Management System (LMS), but because of the authentication processes Studio works with basically any type of web-based system with user administration module.

The Studio main components are the educational ontology, the repository, the adaptive testing engine and the related editors. The connection between components can be seen in Figure X. This chapter contains the detailed description of the components, their functionalities and the platform's non-functional capabilities.



### 2.1. Educational Ontology

The scope of curricula taught in a certain training program can even be rather broad and curricula in general are substantively different in nature, which clearly poses a challenge on the ontology model building process. It should be also taken into consideration that the structure and content of a subject might be at least partly different in different institutions. Major classes of the ontology that were developed in the first cycle should meet these challenges. The following section gives a description of all of the classes in the ontology.

### ***“Scope of Activities” Class***

The “Scope of Activities” class contains all of those professions, employments and activities that can be successfully performed with the acquisition of those competencies that are provided by the given training program.

### ***“Task” and “Competence” Class***

A job consists of numerous tasks that should be executed in the course of everyday work. At the same time the employee must possess certain competences to be able to accomplish tasks relating to her position. So each task should be in “requires” relation with competences.

On the other hand one scope of activities should be in direct “specified by” – “served by” relation with tasks. This way the given scope of activities prescribes a number of concrete tasks that will define concretely required competences.

### ***“Group of Task” and “Competence Module” Class***

By defining separate classes for tasks and competences, not sets (competence modules, group of tasks), but their elements are connected to each other. At the same time the “Group of Tasks” and the “Competence Module” classes should be entered to the model to enable the definition of sets of tasks and competences as well and ensure further ways of comparison.

### ***“Knowledge Area” Class***

Knowledge areas and competences are connected directly with the “requires” and “ensures” connection. (A competence requires the knowledge of a given knowledge area and the good command of a knowledge area ensures the existence of certain competence(s).)

The class of “Knowledge Area” is an intersection of the ontology, where the model can be divided into two parts:

- One part of the model describes the relation of knowledge areas and labour market requirements with the help of the above-described elements.
- The other part will depict the internal structure of knowledge areas.

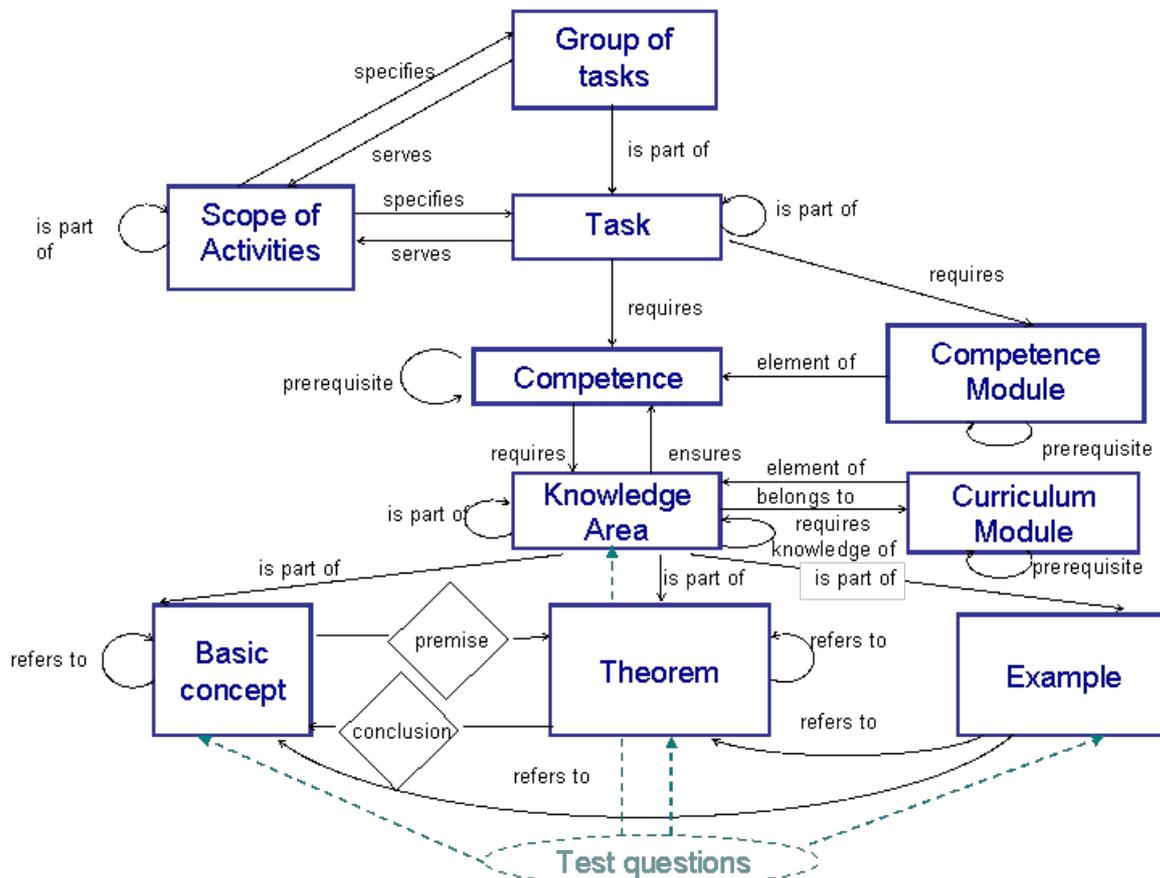
The internal structure of knowledge areas must be refined to allow of effective ontology construction and the efficient functioning of the adaptive knowledge testing system.

“Knowledge Area” is at the very heart of the ontology, representing major parts of a given curriculum. Each “Knowledge Area” may have several Sub-Knowledge-Areas through the “is part of” relation. Not only internal relations, but relations connecting different knowledge areas are also important regarding knowledge testing. This is described by the “is part of” relation. At the same time another relation has to be introduced, namely the “requires knowledge of” relation. This relation will have an essential role in supporting adaptive testing. If in the course of testing it is revealed that the student has severe deficiencies on a given knowledge area, then it is possible to put questions on those areas that must have been learnt in advance.

For the sake of testing all of those elements of knowledge areas are also listed in the ontology about which questions could be put during testing. These objects are called knowledge elements and they have the following major types: “Basic concepts”, “Theorems” and “Examples”. In order to precisely define the internal structure of knowledge areas relations that represent the connection between different knowledge elements also must be described.

This way the ontology model that provides the base for the application (the adaptive testing system) is completed. The model of Educational Ontology is depicted by Figure 6 using the following notation:

- Rectangles sign classes.
- Arrows depict 0-N relations (so a competence may have several prerequisites, scope of activities may specify more tasks at the same time and it is also possible that a competence those not have any prerequisites).



**Figure: Educational Ontology**

### Ontology editor

By using ontology editor that is based on the educational ontology developers are able to work more efficient. The learning time of this editor is shorter than other general ontology editors. These were some aspects at the development of Studio's editor:

- Extensible – The training system has to meet the labour markets requirements, so it will require constant maintenance and development.
- Treatment of high volume data – Even one curriculum may consist of several hundred ontology elements, like knowledge areas, basic concepts, theorems etc. So the modelling of all the curricula of a given training program will require even greater capacity.
- Interoperability – This learning management solution includes several different systems and applications. So it must be ensured that all part, even including the ontology, can easily and efficiently communicate with each other.

- User friendly interface – a simple but useful interface helps user to work faster and more effectively. The main goal is to create a view of the educational ontology in an understandable way.

## **2.2. Content Authoring**

As it was described in the previous section, the ontology provides the underlying structure of the curriculum based learning materials. Every other content developed during the curriculum development process is attached to this structure. This way, all subsystems of the Ontology-based Authoring Environment should be integrated with the ontology layer.

The creation of a new curriculum begins with the selection of the relating domain ontology. Curriculum related learning content, created in the system, should follow the underlying ontology structure that defines the domain of discourse. Since the structure of the curriculum has already been defined, the only task of the content developer is to assign content elements to adequate nodes of the ontology. Content elements can be found in the Repository or they can be created there while attaching them to the ontology.

## **2.3. Repository**

The central element of content development and management is the Repository. This component stores every content element that can be useful in composing a curriculum. Its content can be an image, an article, short texts like a useful paragraph or a famous quote or even audio and video materials. The role of the Content Repository is to store and manage these content elements while maintaining a rich set of metadata describing the contained elements. Each content element can be described with Dublin Core metadata (ISO, 2003) and other useful descriptors, like tags or categories. This rich description enables that stored elements can be easily found and retrieved by curriculum developers.

## **2.4. Test Bank**

In order to provide adequate support for knowledge testing several theoretical foundations and conceptions must be laid down concerning the structure of test bank and test items as well. One pillar of the testing system is the set of test questions. Accordingly all test questions must have the following characteristics:

- All questions must be connected to one and to only one knowledge element or knowledge area in the ontology. On the other hand a knowledge element or knowledge area may have more than one relating test question. This way the Test Bank is structured by the Educational Ontology.
- All questions should be weighted according to their difficulty.

Test questions will be provided in the form of multiple-choice questions. Therefore parts of a question are the following:

- Question
- Correct answer
- False answers

The Test Bank does not form an integral part of the ontology. This means that questions do not have to form a part of the ontology if we want to represent correctly a given curriculum. That is the very reason for connecting the Test Bank to the elements of the ontology only with dashed lines on Figure 6.

### **Test Item Editor**

As it was seen in the previous section and also in the case of the Content Authoring, Test Authoring is also based on the underlying ontology. The process of Test Development is also similar: questions have to be written and attached to various nodes of the ontology. Naturally, more questions can (and should) be attached to one node. The final phase of the test development process is test generation that means the setting of appropriate testing and evaluation algorithms for a test instance. Multiple tests can be generated using the same question set by setting the parameters differently.

## **2.5. Packaging**

The final step in the curriculum development process is packaging. During this step the curriculum material is packed in concept groups. The concept group contains a part of the ontology, basically the domain or a part of a domain which users should learn. The adaptive testing engine runs the tests based on concept groups also they are used during evaluation. Using this approach content developers are able to create multiple tests for the same domain with different nodes.

## **2.6. Process of Content Development**

As it was described before, development of the curriculum content begins with the construction of the appropriate ontology. Ontology and domain experts determine the structure and concepts of the domain of the curriculum and with the chosen editor tool the ontology is built.

As the ontology is finalized, domain experts extend the bare structure with textual and multimedia content elements. Content elements reside in the Repository. Domain experts can search the repository for already existing content or create new elements if needed. Selected content elements are attached to the appropriate nodes of the ontology. This process is basically the establishment of assignments or relationships between ontology nodes and content elements.

Content developer has to design the curriculum material carefully to maintain a balance between the core and illustration material. Core material is related strongly to the ontology concepts, building the most important and basic elements of the curriculum, while the purpose of illustration elements is to

help understanding the material. Core elements are usually textual ones, while illustration material can involve large variety of content elements, like pictures or video clips.

After finishing the content assignment, the Test Bank has to be filled. The domain expert can use the Test Item Editor to edit questions and assign them to the appropriate node of the ontology.

Result of this process is the finished ontology structure with attached content elements and questions. The last phase of content development is the packaging, when the concept groups are made and the result is passed to user for usage.

## **2.7. Adaptive Testing Engine**

The only task that must be accomplished before starting the construction of the adaptive testing system is to lay down the main principles of our own adaptive testing methodology and work out its process.

### **Adaptive Knowledge Testing Approach**

In contrast with traditional examination the number of test items and order of questions in an adaptive test is only determined during writing the test itself with the goal of determining the knowledge level of the test taker as precisely as possible with as low number of questions as possible (Linacre, 2000). Adaptive testing is not a new methodology and despite the fact that it has many advantages compared to traditional testing, its application is not widespread yet. This research has focused on the computerized form of adaptive testing; whose main characteristics – independently from the methodological approach – are the following:

- The test can be taken at the time convenient to the examinee; there is no need for mass or group-administered testing, thus saving on physical space.
- As each test is tailored to an examinee, no two tests need be identical for any two examinees which minimize the possibility of copying.
- Questions are presented on a computer screen one at a time.
- Once an examinee keys in and confirms his answer, he is not able to change it.
- The examinee is not allowed to skip questions nor is he allowed to return to a question which he has confirmed his answer to previously.
- The examinee must answer the current question in order to proceed onto the next one.
- The selection of each question and the decision to stop the test are dynamically controlled by the answers of the examinee (Thissen and Mislevy, 1990).

A methodology of adaptive testing has been elaborated that provides help in determining the knowledge level of the student with asking as few questions as possible.

## **2.8. Results and learning statistics**

Studio offers different approaches to follow the users' activities and results. These statistics are created to help users, give feedback about their progress and to help content developers assess the curriculum. The following options are available currently:

- After each test completed by users the system shows the results, and users gain access to the related contents from the repository.
- Content developers can use some basic statistics implemented in the Studio to follow user activity.
- Content developers can use the Studio's built in query language to write customized query, that can be exported in a comma-separated value file format (CSV file) for further processing. The Studio's query language is based on SQL, so for writing queries the basic knowledge of SQL is useful.
- The Studio's statistics module is reachable for external systems too. In this case external system calls the Studio's module with an HTTP request, which contains a query. The results can be provided in different formats. This way results and user activities can be reached in the external system without manually exporting data from Studio.

## **2.9. Mobilized content delivery system (user perspective)**

Students, who log into the system from a mobile phone, using the mobile interface of the external system, can access and read content from modern mobile devices like smart phones or tablet pc-s.

The mobile interface have the same functionality as the desktop version, but some additional content like Flash or executable files will not work on every device. Because of this issue content developers need to avoid these types of content, and work with the built in content manager. If some of the mentioned content appears, content developers should warn the users and give them information about the proper usage of those files.

In case using a WAP browser of a mobile phone, it is probable that the browser is not capable to access the application. To avoid this problem we recommended students to use the Opera Mini browser, which is a free internet browser application for wide range of mobile phones. This Java based browser runs on almost all commonly used devices. The downloading instructions for the Opera browser are available at Opera's official homepage.

## **2.10. Non-functional Capabilities**

- Multilingual environment: The Studio supports translations on the whole system. This means that items of the ontology, repository and test bank can be translated. With Studio

content developers have a great tool to create and manage multilingual contents. Studio offers the possibility to translate the interfaces' texts too. Every time the content or interface text is not available on the required language Studio returns with the default language (English).

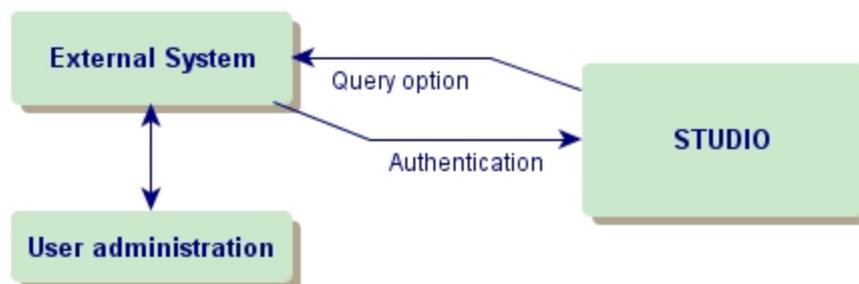
- Scalability: The architecture of the Studio is scalable by using interfaces that runs on the client's browser and databases which were designed with great scalability. The interfaces generate only a reasonable number of requests with small data traffic. For a big number of users it is possible to add additional servers or database server for the architecture.
- Platform independency: One major skill of the Studio to easily adapt to different platforms. The web-based design allows us to create interfaces that will work on different platforms, operating systems and browsers until they support web standards. If the current interfaces do not work for some reason, but it is possible to access Studio server on the platform, developers can build a platform specific interface without disturb the existing ones.
- Loose coupling: From the previous point comes that every interface can be rebuilt without changing the main server. There are only some pattern and communication form that has to be followed for proper working (like the HTTP based communication in JSON or XML format). Work with external systems is the same: Studio works with multiple systems, just the connection patterns has to be implemented.
- Privacy: Studio does not store any additional user information just the user's nick name (or something similar) and the external system's name. This way unauthorized person cannot reach user's identity from Studio, to do that he has to have proper rights in the external system and Studio as well.

### 3. Architecture

There are two points of view of the Studio's architecture: the connection with other systems and the Studio's inner design. In the first case we are looking at Studio as a black box and only describe the available communication interfaces. After that Studio's inner design will be described with its inner communication model that allows us a better understanding of Studio's background processes

#### 3.1. Connections to External Systems

Studio built as an independent application and runs on its own but requires an external system to authenticate users. After the authentication process Studio will recognize users and allows them to work in the system. This will only allow access to Studio for a limited time and after the session timed out, users need to identify themselves. The session only ends if the user ends interactions with the system. As a web application users need to authenticate at every sign in.



Studio is able to send back information to the external system as well. This includes some basic statistics, user activity, results. This type of feedback is especially important working with LMS to complete the learning infrastructure but will work with other systems too.

#### **Authentication**

For authentication and to enter Studio there are two basic actions which have to be implemented: the background communication of the servers and creating the entry links for Studio.

- Studio needs to recognize the external systems server for this the system has built in function which only needs some information about the new system to connect with. The major work of this process is to create the authentication functions in the external system which usually needs development. Studio offers patterns and support for this process. There are test servers to try out the communication and test interfaces to work with. The external system gives only some basic information about users like the user's name, nick name, rights and selected language. Rights may be overwritten by Studio.
- The entry links need to be built by external systems, these links could be a simple HTML link (or button that points to the same URL, etc.) or a URL for an iframe. In each scenario the links are basically built in the same way, just with iframes it is possible to add Studio's interface to

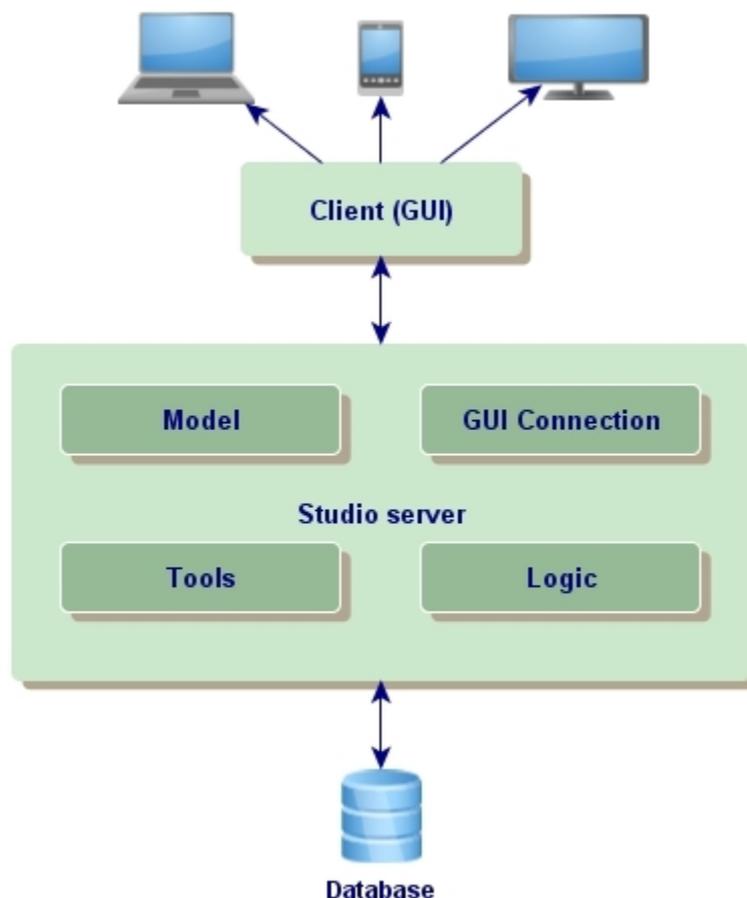
an existing web page. Studio offers a pattern to creating links, like selecting interfaces or selecting concept groups for tests. Link has to contain a token which has to be implemented in the external system, and it is used for authentication. The token is the only variable in the link, but it depends on the external system to make the other parameters variable too,

### **Query option**

Studio can send information back to other systems for further evaluation. These can be posted from the external system via HTTP requests. Requests have to contain a query written on Studio's own query language or a basic statistic type and a format parameter. The external system has to implement some major functions to process the data that will send the Studio. Data types can be JSON, XML or some character-separated format like CSV.

### **3.2. Internal design**

Studio has a web-based three-tier architecture which contains graphical user interfaces that can run in user's browsers, a main server and a database layer.



### **Client**

The most important issue that Studio's interfaces are not web pages; they are web applications running in web browsers. The major benefits from this design is that users do not need to install software on computers and they only need to download the GUI once, there are no additional page loadings. The interfaces gain their data from the Studio server by sending AJAX requests, and process them.

This approach allows us to implement interfaces on different platforms or browsers. GUI only needs to follow the communication protocols and patterns by Studio.

### **Studio Server**

The main component of the Studio architecture is the Studio Server which handles everything from user authentication to database writing. The GUI can be downloaded from this server after user is authenticated. The server processes HTTP requests and sends the results back to a GUI or other external system like it has been discussed in the previous section.

Server code can be divided into four major types:

- **Connection:** this part of the code is responsible for the process of HTTP requests; check user rights and parameters; send the results back.
- **Logic:** It contains all core functions of the Studio (ontology editing, adaptive test engine, statistics module, etc.)
- **Model:** model contains all classes that are used in Studio like ontology classes, users etc.
- **Tools:** tools are used by the server's other three main part, mainly these are functions that are used for content encoding, handle databases.

### **Database**

Database runs in the third tier of the architecture, it cannot be reached directly from interfaces, only system administrators can work with them in that way. The database used by Studio is well scalable and easy to use.

## 4. User specifications

### Administrator role

The major roles of the administrator are the followings:

- System development
- System configuration, maintenance
- Server monitoring and control
- Assign privileges for access
- Verify that peripherals are working properly
- Support for general technical problems which closely related to the system
- Backups and maintenance

In our case the administrator is both of a system administrator and user administrator. This role has access and general overview of the whole system. It has access to change, modify, delete or create in each segments of the system for the Studio development and maintenance.

### Trainer role

The major roles of the trainer are the followings:

- Content delivery or development
- Ontology development
- Multiple choice tests generation
- Content and ontology maintenance
- Users' activity guidance
- Users' activity monitoring

In our case the trainer role is strongly related to the content and ontology creation and to the users verifying. It has limited access for the whole system (for the ontology, test, and the content editor and for the statistics module).

### User role

The major roles of the user are the followings:

- Using Studio
- Access to the tests
- Access to learn from the multimedia materials
- Access to the test results and relating statistical data

In our case the users' role are using the electronic educational environment (Studio) to improve their knowledge in case of predefined learning modules through multiple choice questions and relating learning materials based on multimedia format.

## 5. GUI Specifications

The Studio's main goal is to provide access to different platforms, to reach that goal we choose web standards in communication and implementing interfaces. The used frameworks are built on HTML, CSS and JavaScript; with these fundamental components we are able to support a wide range of web browsers. In this section you can read more about the interfaces, their roles, building blocks and browser support. (We focus on the five major browsers on today's browsers market)

Student GUI - Desktop	
<b>Definition</b>	This interface guides users through the test and gives the possibility to see the results, and access to Studio's learning materials where users can improve their knowledge.
<b>Main components list</b>	<ul style="list-style-type: none"> <li>• Welcome Panel</li> <li>• Question Panel</li> <li>• Completed Tests List</li> <li>• Suspended Tests List</li> <li>• Ontology Visualization (Tree Structure)</li> <li>• Results Panel</li> <li>• Learning Material Panel</li> </ul>
<b>Supported browsers</b>	<ul style="list-style-type: none"> <li>• Internet Explorer 6+</li> <li>• Firefox 3.6+ (PC, Mac)</li> <li>• Safari 3+</li> <li>• Chrome 6+</li> <li>• Opera 10.5+ (PC, Mac)</li> </ul>

Administrator GUI – Desktop	
<b>Definition</b>	This interface contains all the necessary panels and functions for ontology and content development. The statistics and administration is also available from this GUI.
<b>Main components list</b>	<ul style="list-style-type: none"> <li>• Welcome Panel</li> </ul>

- Ontology Editor
- New Node Form
- Ontology Visualization (Tree Structure)
- Description Editor
- Question List for a Node
- Question Editor
- Ontology Relation Editor
- Statistics panel for an ontology node
- Learning Material Editor
- Search for an ontology node
- Concept group Editor
- Archive concept groups list
- Main Statistics Panel
- Filtering By Date Toolbar
- Query Writer Panel
- Administration Panel
- Change Monitoring Panel

**Supported browsers**

- Internet Explorer 6+
- Firefox 3.6+ (PC, Mac)
- Safari 3+
- Chrome 6+
- Opera 10.5+ (PC, Mac)

**Student GUI – Mobile**

**Definition**

This interface guides users through the test and gives the possibility to see the results, and access to Studio's learning materials where users can improve their knowledge. Basically the same as Student GUI – Desktop, but this interface is optimized for tablet pc-s and

smart phones.	
<b>Main components list</b>	<ul style="list-style-type: none"> <li>• Frame Panel</li> <li>• Welcome Panel</li> <li>• Question Panel</li> <li>• Completed Tests List</li> <li>• Suspended Tests List</li> <li>• Ontology Visualization (Nested list) with results information</li> <li>• Learning Material Panel</li> </ul>
<b>Supported platforms</b>	<ul style="list-style-type: none"> <li>• Apple iOS 3+</li> <li>• Android 2.1+</li> <li>• BlackBerry 6+</li> <li>• Runs on Desktop WebKit browsers too, like Chrome or Safari</li> </ul>

Users can choose between mobile and desktop interfaces by setting the interface in a predefined URL, or by using the Studio's platform identification. In the first case the system administrators can handle different platforms, operating systems, browsers, by selecting the appropriate interface. Otherwise Studio will decide to show an interface based on the platform identification's result.

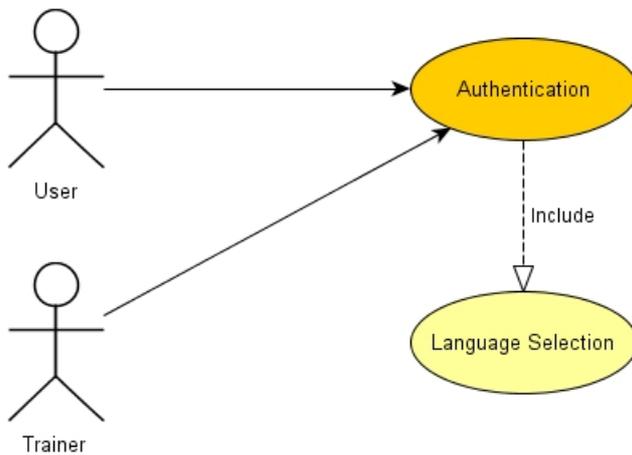
Studio Mobile interface is still a web application and this allows us for supporting wider range of platforms than native applications. In the future we are planned to add optimized interfaces for more mobile devices, meanwhile users can use the desktop version of the Studio.

## 6. Use cases

This section of the document contains the necessary use cases for Studio. It is a higher level view of the system, but this is the same as the implementation itself. Use cases were divided into four groups: external system, user side, trainer side and tasks for administrators.

### 6.1. External system

The two use case discussed here is not part of the Studio but have to be implemented for proper usage.



#### **Authentication**

<i>Use Case:</i>	Authentication
<i>Goal in Context:</i>	Setup of the user and trainer roles; redirection into the Studio
<i>Primary Actors:</i>	Users, trainers
<i>Secondary Actors:</i>	None
<i>Stakeholders &amp; Interests:</i>	Setup of general data (name, ID); Setup selected language in case of the primary actors into the Studio.
<i>Trigger:</i>	The user or trainer clicks to the link which redirects into the Studio.
<i>Preconditions:</i>	The primary actors are registered members of the external system; Permissions and access settings of Studio are created; primary actor has a selected language in the external system; the relation and communication between Studio and the external system are solved.
<i>Success End Condition:</i>	The primary actors enter to the appropriate interface of the Studio environment.
<i>Failed End Condition:</i>	An error message shown if the login is not succeeded into the system.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"> <li>1. The primary actors click onto the Studio link.</li> <li>2. Data processing (through the server).</li> </ol>

3. The appropriate Studio interface loads in the browser (in a new page or iframe)

*Extensions:* Language selection

**Language selection**

*Use Case:* Language selection

*Goal in Context:* Selected language is shown for the primary actors.

*Primary Actors:* Users, trainers

*Secondary Actors:* None.

*Stakeholders &* Studio menu is available in the selected language for the primary actors.

*Interests:*

*Trigger:* The appropriate language can be selected in the external system by the primary actors.

*Preconditions:* Studio disposes with localization of the appropriate language.

*Success End Condition:* Studio loads in the appropriate language.

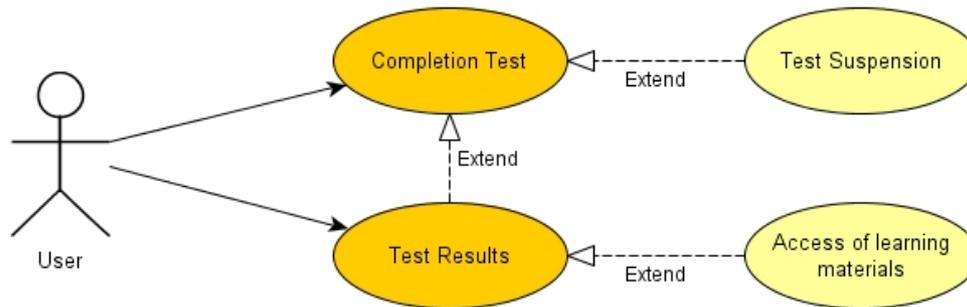
*Failed End Condition:* Texts of the system elements are shown in the default language.

*Main Success Scenario:*

1. The required language is selected in the external system by the primary actor
2. After the language selection Studio loads automatically in the required language.
3. According to the language settings the text of system is localized.

## 6.2. Studio – User

The user side of the Studio covers testing, evaluation and provides the learning materials. These use cases were used for implementing both desktop and mobile interfaces for Studio and it is suggested to follow these use cases by implementing new interfaces.



### Completion Test

<b>Use Case:</b>	Completion test
<b>Goal in Context:</b>	The user discovers own knowledge relating to the predefined curriculum.
<b>Primary Actors:</b>	Users
<b>Secondary Actors:</b>	None
<b>Stakeholders &amp; Interests:</b>	Primary actors – Through the completed test's result can be discovering the user's knowledge (gaps) and those parts which have to improve more.
<b>Trigger:</b>	Primary actors can decide to start a new test or continues the suspended one (if it exists) by clicking one of above mentioned options.
<b>Preconditions:</b>	The relating questions are already prepared by the trainer.
<b>Success End Condition:</b>	The filled test results are available on the platform.
<b>Failed End Condition:</b>	An error message shown, if there is problem with a test completing.
<b>Main Success Scenario:</b>	<ol style="list-style-type: none"> <li>1. Primary actor selects the new test option or chooses from suspended test items (if these exist).</li> <li>2. On the main page (panel) questions are shown one by one.</li> <li>3. Primary actor selects an answer for the question.</li> <li>4. By clicking to the 'Send' button the selected answer will be confirmed and sent to the server.</li> <li>5. In response, the server sends a new question or finishes the current test.</li> <li>6. After finishing the test its result is available for checking or a new test can be started.</li> </ol>
<b>Extensions:</b>	Suspension, Test results

## **Test Suspension**

<i>Use Case:</i>	Test suspension
<i>Goal in Context:</i>	During the test it is possible to suspend the questioning.
<i>Primary Actors:</i>	Users.
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	The tests can be suspended at any time. Suspended test can be found and continued later under the 'Suspended tests' panel.
<i>Trigger:</i>	Primary actor has decision that which options are chosen: continues a suspended test or closes the main page.
<i>Preconditions:</i>	User already answered for the first question.
<i>Success End Condition:</i>	The last filled test results are saved automatically on the server.
<i>Failed End Condition:</i>	If the back ups are failed, the suspended test cannot be continued.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. Click to the 'Suspend test' button; refresh the browser, close the main page or other errors happen.</li><li>2. The last suspended test is available in the list of suspended tests. After resetting the system or reopening the Studio environment the suspended tests are available to continue.</li></ol>

## **Test results**

<i>Use Case:</i>	Test results
<i>Goal in Context:</i>	Give support in exploring the users' knowledge based on predefined curriculum. Give help and feedback for users to complement their educational deficiencies.
<i>Primary Actors:</i>	Users
<i>Secondary Actors:</i>	None
<i>Stakeholders &amp; Interests:</i>	Gives support to the user improving the missing knowledge areas based on the test result.
<i>Trigger:</i>	After completing a test, the result is automatically shown. In other cases all detailed results of the filled tests are available by clicking on the selected item.
<i>Preconditions:</i>	At least one test is completed by the actor.
<i>Success End Condition:</i>	After completing a test, the result is automatically shown on a new form and the offered learning materials are also available there.
<i>Failed End Condition:</i>	Error message shown if there are any problems with database availability.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. After completing a test, the result is automatically shown on a new form. In other case all detailed results of the filled tests are available through the</li></ol>

'Completed tests' panel by clicking on the selected item.

2. A tree of knowledge structure is found on the new form. It contains the asked ontology nodes. If users want to expand the whole structure is it possible with the '+' sign in front of the nodes.

3. After selected an ontology node (knowledge area) its basic information is available on the form.

4. In addition, the related learning material is also available there.

*Extensions:* Access of learning materials.

**Access of learning materials**

*Use Case:* Access of learning materials

*Goal in Context:* Users improve their knowledge through the learning materials.

*Primary Actors:* Users.

*Secondary Actors:* None.

*Stakeholders & Interests:* After completed a test the actors improve their knowledge using the offered and appropriate learning materials.

*Trigger:* Under the result the learning materials can be found and selected by the actors.

*Preconditions:* The selected ontology node has at least one completed test and learning material.

*Success End Condition:* The related learning material of the selected ontology node is shown for the actor.

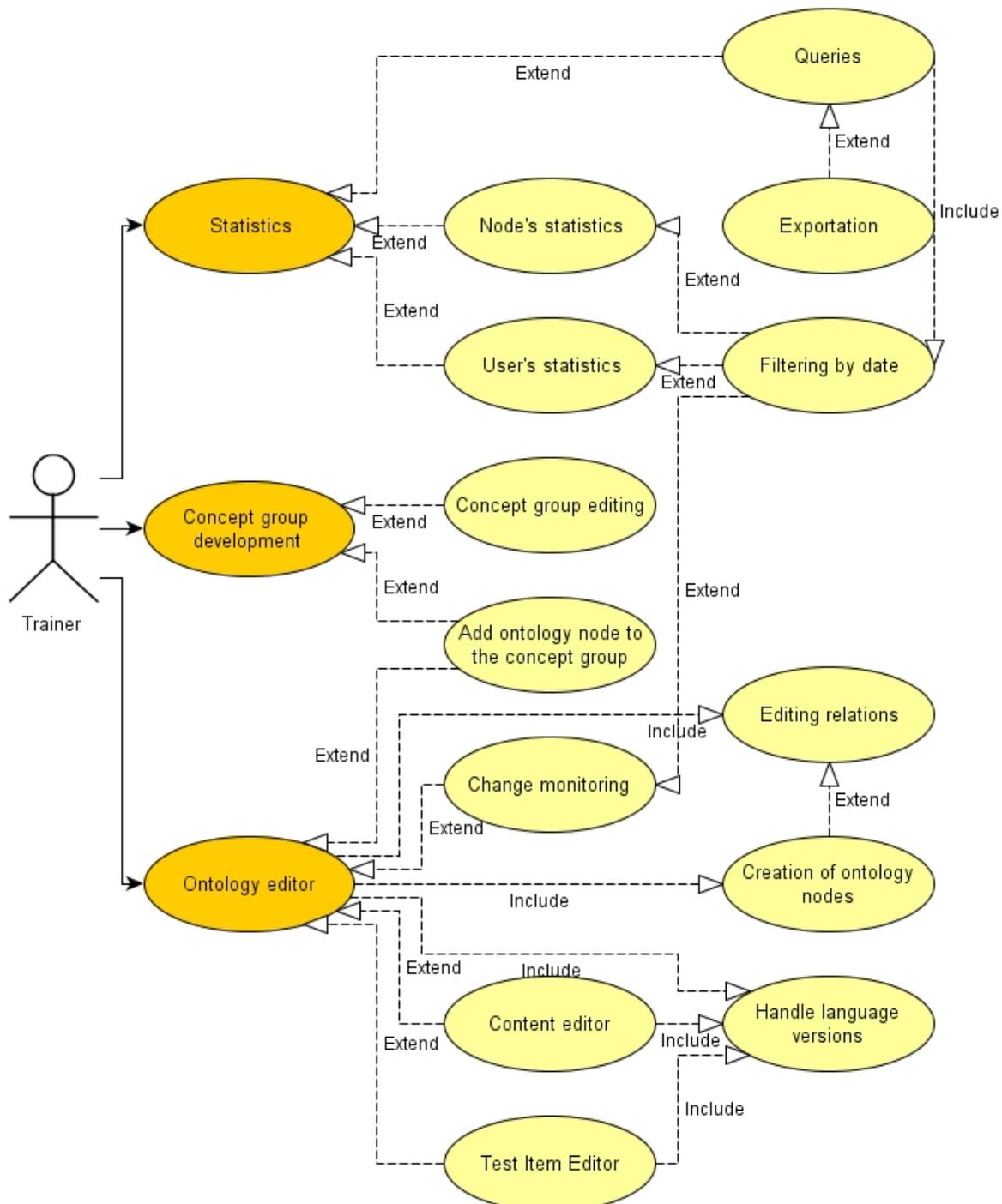
*Failed End Condition:* Error message shown if the appropriate learning material does not exist.

*Main Success Scenario:*

1. The users select an ontology node.
2. The users click to the 'Related learning material' option.
3. Finally the learning material appears.

### 6.3. Studio - Trainer

Trainer side of the Studio describes the background functionalities of the Studio like ontology editing, content development, concept group creation and statistics. These use cases are only available to trainers. As it was mentioned earlier Statistics can be reached from external systems, still the trainer will be the actor who uses the gathered information.



## **Statistics**

<i>Use Case:</i>	Statistics
<i>Goal in Context:</i>	Trainer get inform how to use the Studio environment.
<i>Primary Actors:</i>	Trainers.
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	Users' activities can be monitored through their completion tests.
<i>Trigger:</i>	The primary actors chooses the 'Statistics' option.
<i>Preconditions:</i>	At least one completed test exists.
<i>Success End Condition:</i>	The primary actors get general feedback from the users' activities (e.g.: platform activities, test results or which learning materials are appeared)
<i>Failed End Condition:</i>	Error message: Filled test does not exist.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. Actors click to the Statistics module.</li><li>2. The selected information is shown.</li></ol>
<i>Extensions:</i>	Users' statistics, Node's statistics, Queries.

### **Users' statistics**

<i>Use Case:</i>	Users' statistics
<i>Goal in Context:</i>	The actors get general feedback from each user (e.g.: platform activities, test results or system usage).
<i>Primary Actors:</i>	Trainers.
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	Users' activities can be monitored and scored through their completion test results.
<i>Trigger:</i>	Users' statistics can be selected in the Statistics module.
<i>Preconditions:</i>	At least one test is already completed by a user.
<i>Success End Condition:</i>	The actor gets information about the user results.
<i>Failed End Condition:</i>	Error message: There are not any completed tests.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The trainer selects the users' statistics.</li><li>2. The statistical results of each user are shown in systematic way.</li></ol>
<i>Extensions:</i>	Filtering by date

### **Node's statistics**

<i>Use Case:</i>	Node's statistics
------------------	-------------------

<i>Goal in Context:</i>	Receive feedback about the learning process.
<i>Primary Actors:</i>	Trainers
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	It can be evaluated each of learning materials, related issues, and questions.
<i>Trigger:</i>	The node's statistics is selected in the Statistics module.
<i>Preconditions:</i>	At least one test is already completed by a user.
<i>Success End Condition:</i>	The actors get information about the content.
<i>Failed End Condition:</i>	Error message: There are not any data in the selected time period.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The actors select the node statistics.</li><li>2. The statistical results of the nodes are shown in systematic way.</li></ol>
<i>Extensions:</i>	Filtering by date

#### **Filtering by date**

<i>Use Case:</i>	Filtering by date
<i>Goal in Context:</i>	Get information of the selected statistics for a given time period
<i>Primary Actors:</i>	Trainers.
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	Get detailed information of the selected statistics for a given time period.
<i>Trigger:</i>	Select the time period (start and end date) to query the required statistical data.
<i>Preconditions:</i>	The statistical data is available if the start and end dates are in a correct format.
<i>Success End Condition:</i>	The statistical data is available in the required time interval.
<i>Failed End Condition:</i>	Error message: Incorrect date interval, incorrect date format, or there are not any data in the required time period.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The actors select a statistics.</li><li>2. The actors select an interval.</li><li>3. Refresh the statistical database.</li><li>4. The statistical results are shown in the selected period in the default date format.</li></ol>

#### **Queries**

<i>Use Case:</i>	Queries
<i>Goal in Context:</i>	There is a possibility to use different queries from the database instead of

---

	the predefined statistical options.
<i>Primary Actors:</i>	Trainers
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	Based on the individual needs it is possible to use customized queries in the Studio.
<i>Trigger:</i>	Choosing 'Create' queries in the Statistics module.
<i>Preconditions:</i>	The trainer knows the required syntax for writing queries, and knows the structure of the database. Information of completed tests exists in the database.
<i>Success End Condition:</i>	The results of the individual queries are available in tabular format.
<i>Failed End Condition:</i>	Error message: using incorrect syntax or there are not any data.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The actors select the system element which is related to the required query.</li><li>2. Write a query which can be filtered by date</li><li>3. Executing query.</li><li>4. The server processes the query and sends the results in readable format back.</li><li>5. On the client side the results are processed and appeared in tabular format.</li></ol>
<i>Extensions:</i>	Exportation
<b>Exportation</b>	
<i>Use Case:</i>	Exportation
<i>Goal in Context:</i>	The results of the quires are exported in another (external) system.
<i>Primary Actors:</i>	Trainers
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	With this option there are not limited barriers of analysing results in another programme, software.
<i>Trigger:</i>	The actors select the export of queries.
<i>Preconditions:</i>	The user writes the query in a correct way. Information of completed tests exists in the database.
<i>Success End Condition:</i>	The results of the individual queries are exported in easy readable format.
<i>Failed End Condition:</i>	Error message: using incorrect syntax or there are not any data.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The actors select the system element which is related to the required query.</li><li>2. Write a query which can be filtered by date</li></ol>

---

3. Export data based on the query execution.
4. The server processes the query and sends the results back in appropriate file format.
5. On the client side the file can be downloaded.

### **Ontology editor**

<i>Use Case:</i>	Ontology editor
<i>Goal in Context:</i>	Selection of the domain and domain ontology building (which is the 'backbone' of the content).
<i>Primary Actors:</i>	Trainers
<i>Secondary Actors:</i>	None
<i>Stakeholders &amp; Interests:</i>	The tests and the learning materials are based on the ontology implementation.
<i>Trigger:</i>	The actor selects the ontology editor.
<i>Preconditions:</i>	The actor is aware of ontology concept, how it builds up, and how ontology classes and relations can be applied in the system.
<i>Success End Condition:</i>	The actors can successfully compile ontology.
<i>Failed End Condition:</i>	The actors cannot successfully compile ontology without help (help from the developer).
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The actor selects the ontology editor from the menu.</li><li>2. The actor compiles the ontology with using the tool bar of the ontology editor (add new element, edit relations etc.)</li><li>3. The created ontology has to be saved.</li><li>4. Depend on the needs here can be managed the languages' localization.</li></ol>
<i>Extensions</i>	Test editor, Content editor, Change monitoring

### **Creation of ontology nodes**

<i>Use Case:</i>	Creation of ontology nodes
<i>Goal in Context:</i>	Add new element to the ontology model (which is the 'backbone' of the content).
<i>Primary Actors:</i>	Trainers.
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp; Interests:</i>	The ontology is build up with ontology elements (e.g.: learning materials knowledge areas- examples, connections etc.-)
<i>Trigger:</i>	The trainer adds the new element to the ontology in the ontology editor.
<i>Preconditions:</i>	The trainer is aware of ontology concept, how it builds up, and how ontology classes and relations can be applied in the system.
<i>Success End Condition:</i>	The trainer can successfully add a new element to the ontology.

*Failed End Condition:* If this element already exists in the system does not allow creating it again.

*Main Success Scenario:*

1. The trainer selects the 'Add' button to create a new element or a learning material in the ontology.
2. The pop-up window requires the necessary parameters (id, name, class, language).
3. The server confirms the addition and displays the changes in the ontology visualization.

### ***Editing relations***

*Use Case:* Editing relations

*Goal in Context:* Edit relations and links between ontology nodes in the domain ontology model.

*Primary Actors:* Trainers

*Secondary Actors:* None

*Stakeholders &* Relations between ontology nodes are strongly connected to each other.

*Interests:*

*Trigger:* After selecting the ontology node choose the 'Edit' option.

*Preconditions:* The actor is aware of ontology concept, how it builds up, and how ontology classes and relations can be applied in the system.

*Success End Condition:* The actors can successfully edit the relation of the selected ontology element.

*Failed End Condition:* Error message: the relation of the selected ontology node cannot be edited.

*Main Success Scenario:*

1. The actor selects an ontology node after the 'Relation' button.
2. The actor edits the required relation.
3. The actor adds, deletes, or modifies the test items of the selected ontology node.
4. After saving the modification is activated.

### ***Test item editor***

*Use Case:* Test item editor

*Goal in Context:* Create test items (questions, answers) to the ontology nodes in the domain ontology model.

*Primary Actors:* Trainers.

*Secondary Actors:* None.

---

<b>Stakeholders &amp; Interests:</b>	The testing process is based on the existing test items.
<b>Trigger:</b>	After selecting the ontology node choose the 'Edit' option under the 'Questions' button.
<b>Preconditions:</b>	The trainer has to choose those kind of ontology nodes for which a question can be added
<b>Success End Condition:</b>	The trainer can create, delete or modify the test item of the selected ontology node successfully.
<b>Failed End Condition:</b>	Error message: the test item of the selected ontology node cannot be edited.
<b>Main Success Scenario:</b>	<ol style="list-style-type: none"><li>1. After selecting an ontology node click to the 'Question' button.</li><li>2. The actor adds, deletes, or modifies the test items of the selected ontology.</li><li>3. The pop-up window requires the edited question.</li><li>4. After confirmation the test item is sent to the server and saved.</li><li>5. After refreshing the test bank is updated with the modification.</li></ol>
<b>Content editor</b>	
<b>Use Case:</b>	Content editor
<b>Goal in Context:</b>	Create content (learning materials) to the ontology nodes in the domain ontology model.
<b>Primary Actors:</b>	Trainers.
<b>Secondary Actors:</b>	None.
<b>Stakeholders &amp; Interests:</b>	The trainer uses these contents, learning materials to improve the users' knowledge.
<b>Trigger:</b>	After selecting the ontology node choose the 'Edit' option under the 'Learning material' button.
<b>Preconditions:</b>	The actor is aware of ontology concept, how it builds up, knows Wiki format and HTML code.
<b>Success End Condition:</b>	The actor can create, delete or modify a learning material of the selected ontology node.
<b>Failed End Condition:</b>	Error message: the learning material of the selected ontology node cannot be edited.
<b>Main Success Scenario:</b>	<ol style="list-style-type: none"><li>1. After selecting an ontology node choose the 'Learning material'.</li><li>2. The actor can add, delete, or modify the test items of the selected ontology node.</li><li>3. After confirmation the learning material is sent to the server and saved.</li><li>5. The content bank is updated with the editions.</li></ol>

---

### **Handle language versions**

<i>Use Case:</i>	Handle language versions
<i>Goal in Context:</i>	Created contents (knowledge areas, learning materials) and test items (questions and answers) can be localized into an appropriate language.
<i>Primary Actors:</i>	Trainers
<i>Secondary Actors:</i>	Users
<i>Stakeholders &amp; Interests:</i>	Trainer - Created contents (learning materials) and test items (questions and answers) can be translated into the appropriate language by the trainer. User - The user selects from the language versions. After the language selection contents and the test items are shown in the appropriate language.
<i>Trigger:</i>	The primary actor can setup the language settings.
<i>Preconditions:</i>	Created contents (knowledge areas, learning materials) and test items (questions and answers) are translated into the selected language.
<i>Success End Condition:</i>	The contents and the test items are shown in the selected language for the secondary actors.
<i>Failed End Condition:</i>	The contents and the test items are shown in the default language for the secondary actors.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The primary and the secondary actors meet with any ontology nodes during their platform activities.</li><li>2. The contents and the test items are appeared in the selected language.</li></ol>

### **Change monitoring**

<i>Use Case:</i>	Change monitoring
<i>Goal in Context:</i>	Any changes in the ontology can be follow-up.
<i>Primary Actors:</i>	Trainers
<i>Secondary Actors:</i>	Administrators
<i>Stakeholders &amp; Interests:</i>	Trainer – Trainer can follow-up the other primary actors' modifications.
<i>Trigger:</i>	The primary and the secondary actors select the change monitoring.
<i>Preconditions:</i>	Modifications already exist which are correctly saved.
<i>Success End Condition:</i>	The current and archive changes are available for the primary and the secondary actors in tabular format.
<i>Failed End Condition:</i>	Error message: Data no exist.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The primary and the secondary actors select the menu item regarding to change monitoring.</li></ol>

2. Set of data can be decreased using date filter.
3. Information is received from the server which data appear in tabulator format.

*Extensions* Filtering by date

### **Concept group development**

*Use Case:* concept group development

*Goal in Context:* After the domain ontology development the actors can select sets of knowledge areas from the ontology model for which have existing test items.

*Primary Actors:* Trainers.

*Secondary Actors:* None.

*Stakeholders & Interests:* The actor collects those sets of knowledge areas which want to share with the users in a testing process.

*Trigger:* The actor selects the 'Edit concept groups'. Under this menu item it can be created a new concept group.

*Preconditions:* -

*Success End Condition:* The actor creates a new concept group.

*Failed End Condition:* If this concept group already exists, the system does not allow creating it again.

*Main Success Scenario:*

1. The actor selects the 'Edit concept groups'. Under this menu item the actors click to the 'New' button.
2. The actor gives an ID.
3. A new concept group is developed later can be modified.

*Extensions* Concept group editing, Add ontology element to the concept group

### **Concept group editing**

*Use Case:* Concept group editing

*Goal in Context:* The actor can localize the name of concept group, edit its thresholds or delete the concept group.

*Primary Actors:* Trainers

*Secondary Actors:* None.

*Stakeholders & Interests:* The changes can be implemented by the actor

*Interests:*

*Trigger:* The actor selects the 'Edit concept groups'. Under this menu item it can be done any modifications by clicking the each button.

*Preconditions:* At least one concept group exists.

*Success End Condition:* The actor edits the concept group.

*Failed End Condition:* Error during the modification.

*Main Success Scenario:*

1. The actors select the 'Edit concept groups' menu item.
2. Translate the name of the item.
3. Give threshold to the item.
4. Can be deleted it.

***Add ontology node to the concept group***

*Use Case:* Add ontology node to the concept group.

*Goal in Context:* The actor selects those knowledge areas which add to the concept group.

*Primary Actors:* Trainers

*Secondary Actors:* None

*Stakeholders &* The test is built up from selected ontology nodes.

*Interests:*

*Trigger:* The actor clicks to the 'Edit concept groups' menu. The actor selects the appropriate node from the ontology visualization.

*Preconditions:* At least one concept group exists and the ontology is also completed and available for the actor.

*Success End Condition:* The actor creates a test from the selected nodes of the ontology visualization.

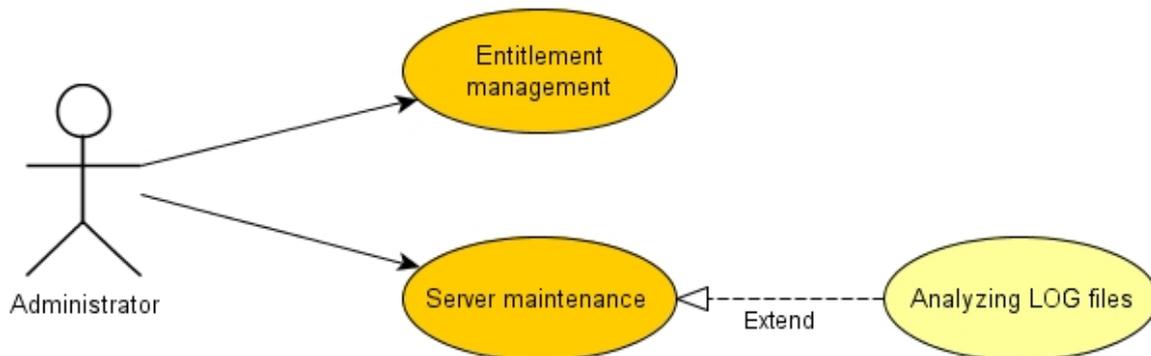
*Failed End Condition:* Error during the modifications.

*Main Success Scenario:*

1. The actors click to the 'Edit concept groups'.
2. The actors select nodes from the visualization and the start node.
3. The actors save the modifications.

#### 6.4. *Emphasized tasks for Administrator*

These tasks are for administrators only to maintain Studio. Entitlement management may need some additional rights in external systems; however administrators could set the rights from the Studio too.



##### **Entitlement management**

*Use Case:* Entitlement management

*Goal in Context:* Setup the users' and trainers' role in the Studio.

*Primary Actors:* Administrators.

*Secondary Actors:* None.

*Stakeholders & Interests:* The actor setups the permissions and access rights.

*Trigger:*

Make changes in one of the users' or trainers' profile.

*Preconditions:* The selected user or trainer has already registered in the external system.

*Success End Condition:* The selected user or trainer can log in into the Studio system.

*Failed End Condition:* -

*Main Success Scenario:* 1. The Administrator selects a user or trainer to setup the access rights to the Studio.

##### **Server maintenance**

*Use Case:* Server maintenance

*Goal in Context:* Ensure continuous Studio operations.

*Primary Actors:* Administrators.

*Secondary Actors:* None.

*Stakeholders & Interests:* Studio is accessible.

*Interests:*

*Trigger:* Studio is installed on a server.

---

<i>Preconditions:</i>	-
<i>Success End Condition:</i>	Monitoring server operations, and making database backups.
<i>Failed End Condition:</i>	There are not database backups, and the server is out of order.
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. The actor monitors the server regularly.</li><li>2. The actor makes database backups regularly.</li></ol>
<i>Extensions</i>	Analyzing LOG files
<b>Analyzing LOG files</b>	
<i>Use Case:</i>	Analyzing LOG files
<i>Goal in Context:</i>	debug and data mining
<i>Primary Actors:</i>	Administrators.
<i>Secondary Actors:</i>	None.
<i>Stakeholders &amp;</i>	In case of an error the reason has to be defined. The useful information is
<i>Interests:</i>	collected for possible solutions.
<i>Trigger:</i>	The writing has to be done into the running log file on the server.
<i>Preconditions:</i>	-
<i>Success End Condition:</i>	Monitoring the system operations regularly with using data mining.
<i>Failed End Condition:</i>	-
<i>Main Success Scenario:</i>	<ol style="list-style-type: none"><li>1. Reading log files in case of an error.</li><li>2. Using data mining tools to follow-up the system usage.</li></ol>