DELIVERABLE D4.5

FINAL RELEASE OF VISUALISATION AND OLM WEB SERVICES AND TOOLS.

Document Version:	2.5 – October 2, 2016
Document Status:	Final
Document Type:	Report
Diss. Level:	Public
Lead Partner:	UoB
Delivery Date:	M31

Authors: Blandine Ginon (UoB),
Drew Masci (UoB),
Matthew Johnson (UoB)

EXECUTIVE SUMMARY

D4.5) Final release of visualization and OLM web services and tools: This deliverable will cover the final release of LA/EDM Web services and algorithms, including manuals. The components will include improvements to and extensions of the system based on the results of pilot studies and user trials.

The purpose of this document is to accompany the final release of the Visualisation Service along with the OLM software package. This document covers an in depth research and development process that is meant to clearly depict a clear picture of how the responsibilities of UoB have been addressed. After multiple studies were conducted the feedback and data were used to better improve the OLM software package to in turn answer the key questions of the LEA's Box project.

Section 2 mainly focuses on the Open Learner Modelling process along with the different Visualisations employed by the final version of the OLM software. Section 3 covers the technical development of the software package and Section 4 covers how to use and configure the OLM within LEA's Box.

Also during the course of the third year of the project, UoB contributed to the following publications:

- LAK 2016, long paper: "Introducing Learning Visualisations and Metacognitive Support in a Persuadable Open Learner Model", Susan Bull, Blandine Ginon, Clelia Boscolo
- LAK 2016, workshop organisation: "LAL Workshop: Learning Analytics for Learners", S. BULL, B. GINON, J. KAY, M. KICKMEIER-RUST
- ITS 2016, short paper: "Persuading an Open Learner Model in the Context of a University Course: An Exploratory Study" Blandine Ginon, Clelia Boscolo, Matthew D. Johnson, Susan Bull
- EC-TEL 2016, poster: "Helping Teachers to Help Students by using an Open Learner Model" Blandine Ginon, Matthew D. Johnson, Ali Turker, Michael Kickmeier-Rust
- EC-TEL 2016, workshop organisation: "Fourth International Workshop on Teaching Analytics" Ravi Vatrapu, Michael Kickmeier-Rust, Blandine Ginon and Susan Bull
- EC-TEL 2016, workshop paper: "an open learner model used by teachers to monitor speed reading learners", Blandine Ginon, Matthew D. Johnson, Ali Turker, Michael Kickmeier-Rust

TABLE OF CONTENTS

1. OPEN LEARNER MODEL

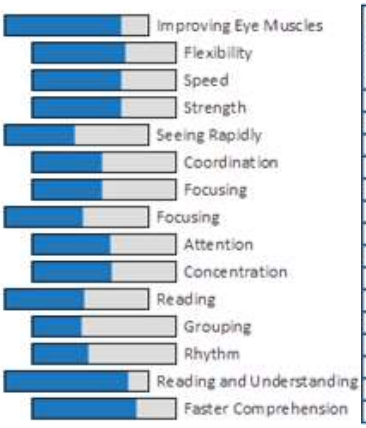
The final release of the LEA's BOX open learner model builds on the work reported in D4.3. In this section we report revisions and extensions to the visualisation set. We first provide a summary of the final visualisations (Section 2.1.i), consider the rationale behind the reduction of the visualisation set (Section 2.1.ii) and the visual properties of those retained (Section 2.1.iv).

1.1. VISUAL REPRESENTATION OF THE LEARNER MODEL

1.1.1. FINAL VISUALISATION SET

The LEA's BOX OLM contains 7 visual representations of the underlying learner model (shown Figure 1). Skill meter, table, radar plot, and network are carried forward from Release I of the system. The across time and heatmap are carried forward from the proto-types in Release II. Level of activity is new in Release III. Much of the work of Year 3 has focussed on aspects of stabilisation, scalability and improving the consistency of these visual methods. Aspects of accessibility are covered in Section 2.1.v. Each of the visualisations makes use of a specifically designed visualisation service, which is described further in Section 2.1.iii. The informational content of the visualisation is calculated as per the learner modelling algorithm described in Section 2.3.

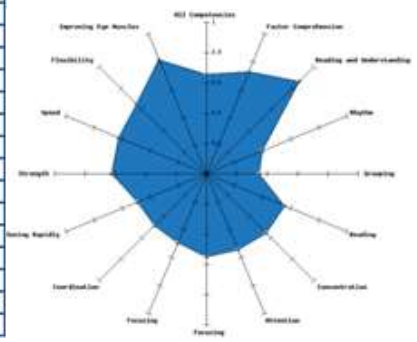
SKILL METER



TABLE

Very Weak	Weak	OK	Strong	Very Strong	
				●	Improving Eye Muscles
			●		Flexibility
			●		Speed
			●		Strength
		●			Seeing Rapidly
		●			Coordination
		●			Focusing
		●			Focusing
		●			Attention
		●			Concentration
		●			Reading
	●				Grouping
	●				Rhythm
				●	Reading and Understanding
			●		Faster Comprehension

RADAR PLOT



NETWORK

ACROSS TIME

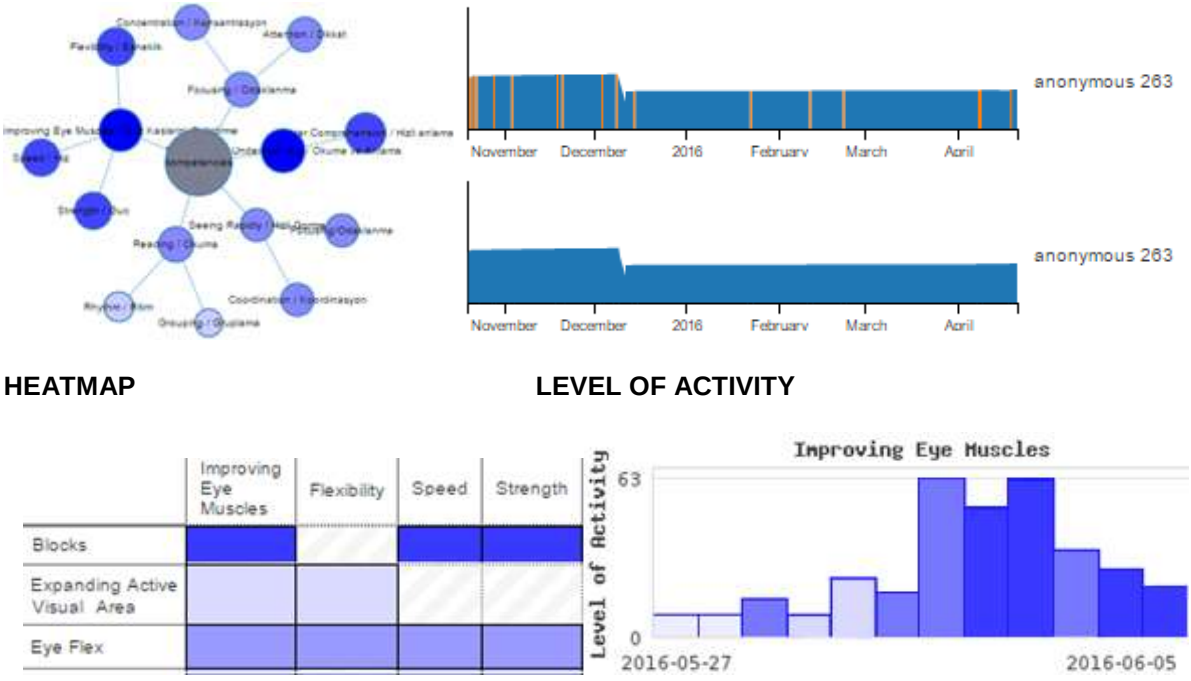


Figure 1: Release III Open Learner Model Visualisations

The final 3 visualisations in Figure 1 are the most recently added:

The **across time** visualisation shows how the learner model has changed across time. (x-axis=time; y-axis=learner model value). As a result from testing with end users, the orange bars within the graph indicate that the change in the learner model at this point in time is the outcome of an episode of persuasion persuasion (see also Section 2.2). Scales are consistent and aligned between graphs.

The **heatmap** visualisation shows a collision matrix between two perspectives from which the learner model may be opened. The default is set to competencies against activities, but this is configurable. The intensity of colour indicates the extent to which the student holds the competency in each activity. Improvements are made following end user evaluation, including adding a background of grey stripes and no border, if the activity cannot give information about a specific competency (i.e. there is no relationship between items on either axis).

The **level of activity** visualisation, as per the heatmap, uses intensity of colour to indicate the state of the learner model: the stronger the colour, the greater the extent to which the competency is held. The height of the bars (y-axis component) shows the number of pieces of evidence added on a particular day, from which the learner model is built. The x-axis component is time, with a single bar existing for each day. As per the across time visualisation, the learner model is cumulatively calculated, and so the open learner model representation on the bars is that of the model at that point, and not just a learner model of information relating to a specific day. This visualisation was developed to support the evaluation with Hizligo in Turkey in June 2016, based on teacher driven requirements (see deliverable D5.6 for further information).

For descriptions of skill; meter, radar plot, table, and network please refer to D4.3. Visualisations are also covered in the user manual in the APENDIX – USER MANUAL.

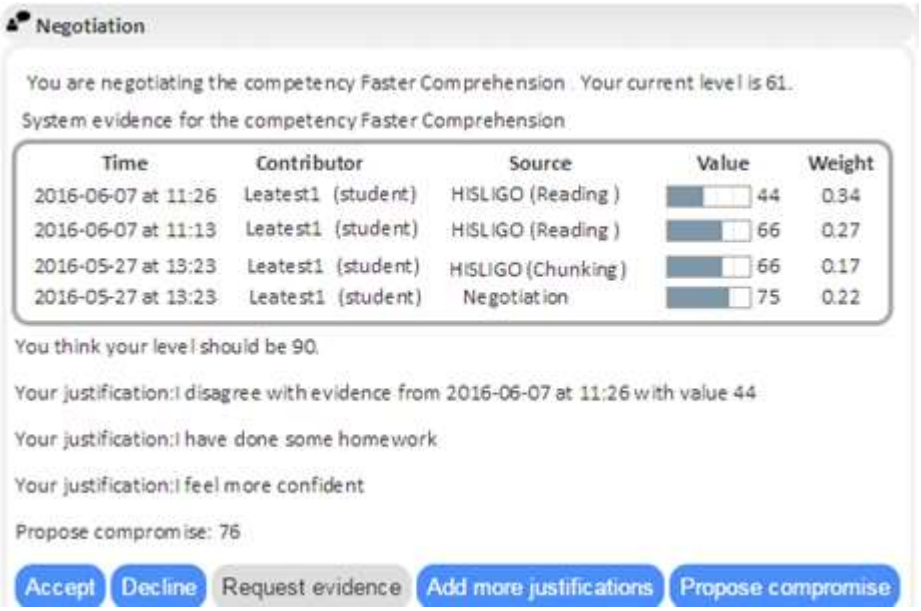


Figure 2: Learner Model Persuasion Dialogue: Showing the Learner Model Process and Evidence Layer

Additionally, we consider opening up of the evidence layer as an additional open learner model method, as it shows the evidence values and the weightings in terms of the learner model. This is available as part of the persuasion method which forms part of the open learner model interactive maintenance facilities. Whilst this is not a full aggregation of all information to produce a single value for the learner model, it shows a more detailed representation of the learner model process and evidence layer (see Figure 2). We therefore include it in our descriptions in the latter part of this section.

1.1.2. CRITICAL EVALUATION OF VISUALISATION SET

Following feedback from end users, consultation of literature and engagement with advisory members, the visualisation set available to students and teachers was reduced to 7 visualisations. This is in part to reduce aspects of cognitive load, and to remove some of the visual methods that were rated as confusing or were not used, as indicated by our evaluations. We have discontinued support smiley faces, histogram, word cloud, gauges, stars and the treemap. We summarise rationale in Table 1:

Table 1: Reduction of the Visualisation Set

Visualisation	Decision	Rationale
Skill Meter	Retain	Well used and well liked. Easy to interpret. There are many examples of OLMs that implement this method successfully.
Table	Retain	Well used and liked. Easy to interpret. Teachers indicated that they, and students, were used to dealing with information in forms similar to this.
Smiley	**Remove**	These were not used and not rated as particularly useful, with the high-school age students with which LEA's BOX was piloted. There is some evidence to suggest that students may interpret them as condescending. There are further issues that emoticons/smilies could be interpreted differently across cultures, adding to the complexity of the issue.
Histogram	**Remove**	Low level of use and usefulness rating. Evaluation indicated that it was difficult to locate items within the structure. Other methods were indicated as more easily interpretable.
Word Cloud	**Remove**	Overall learners did not understand the information presented. The representation of Weak items was thought to be especially misleading (given students' prior knowledge). Greater support was able to be given using other visualisation types.
Stars	**Remove**	Learners indicated these were understood, but were difficult to interpret and compare. They were thought to be overly complex graphically and the alignment makes it look really messy and overwhelming (visually). Information of the same type could be more easily discerned from the table or skill meter.
Gauges	**Remove**	Not well used or understood. There is an increased cognitive load in interpreting this visual metaphor in a visual-spacial dimension. Likewise these are space consuming and can be difficult to interpret when there are many in alignment; there were issues of scalability and performance.
Network	Retain	The network was understood and also shows the additional component of the relationships in the structure. Nodes are collapsible allowing it to be more readable. This worked best for showing student competency.

Heatmap/ Collision Matrix	Retain	This was rated as relatively easy to interpret, and useful for identifying gaps, particular when configured to show activities against competencies, and identifying areas where there is insufficient data.
Across Time	Retain	Indicated as useful and important to display the temporal aspects of the open learner model.
Treemap	**Remove**	Not used and indicated as difficult to use. It was stated that these are difficult to use to gain overviews of information, and in this context it was difficult to indicate precisely the state of the learner model. Identification of areas for improvement was difficult, as a treemap would no show these.
Radar	Retain	Rated as useful and familiar to users, even if not interpreted consistently. Indicated as being useful in allowing students to see the relative strength of certain nearby competencies.
Level of Activity	Retain	Easily interpretable visual form. Shows additional activity-based analytic information in addition to a representation of the open learner model. Requested and designed in collaboration with end users.
List of Evidence (in persuasion)	Retain	This is core to the workflow of persuasion, and a key information source for decision making within this (see Section 2.3.)

1.1.3. VISUALISATION SET IN TERMS OF ACTIVITY-BASED ANALYTICS

Following on from work outlined in D4.3, we consider each of our visualisations in terms of additional aspects of activity-based information that is able to be generated from the underlying learner model (Table 2). The level of activity visualisation is designed to present multiple aspects of these, in addition to the current state of the learner model. Most specifically this indicated the level of information from which the model is built, identifying intense or absent periods of interaction, and also when the last updates occurred. Likewise, the list of evidence is presented as part of the learner model persuasion dialogue also is able to show many aspects of these by externalising information about the information from which the model is built - the notable exception is that is does not combine the evidence in terms of a model. Similarly the across time visualisation is able to show temporal aspects, which includes recent updated. The remainder of the visual methods show primarily only OLM information, although when multiple instances of the visualisations are co-located on the interface, it is possible to compare between them to identify distinctions between data sources, and potentially missing information. Future work may wish to consider how OLM visualisations can support more activity-based and temporal aspects of learning analytic information in commonly open forms of the learner model.

Table 2: Visualisation Properties in Terms of Presenting More Activity-Based Information in the Open Learner Model

Attribute	Skill Meter	Table	Radar Plot	Network	Across Time	Heatmap	Level of Activity	List of Evidence (in persuasion)
Level of activity							●	●
Level of information								●
Intense interaction							●	○
Last update					●		●	●
Distinction between sources	○	○	○	○	○	○	○	●
Recent activity change							●	○
Process/sequence					○		○	
Missing information	○	○	●	○	○	●	○	
Redundant/outdated evidence								●
Durations of interaction					○		●	○
O.L.M.	●	●	●	●	●	●	●	
Recent informational change					●	●	●	●

Key: ● attribute is present ○ attribute is present when stacked/configured

1.1.4. VISUALISATION SET IN TERMS OF VISUAL PROPERTIES

In D4.3 we summarised our visualisation set in terms of a series of visual properties. In Table 3 we summarise the updated version of this, highlighting that the diversity of representations presented is to support individual differences, user preferences, different contexts of use and styles of interaction, different visual densities of information and different levels of complexity.

All visualisations are available to both the student and the teacher; no prescribed use for any is given. Both stakeholder types may configure which visualisations they wish to use.

Table 3: LEA's Box OLM Visualisation Set, Visual Properties

Attribute	Skill Meter	Table	Radar Plot	Network	Across Time	Heatmap	Level of Activity	List of Evidence (in persuasion)
OLM Perspectives	1	1	1	1	1	2	1	1
Graphical	●		●	●	●	●	●	●
Textual		●						●
Quantised		●		●			●	
Continuous	●		●		●	●	●	●
Structured	●	●		●		●		
Interactive			○	●				
Text Labels	○	○	○	○	○	○	○	●
Shape			○	○	○			
Colour				●		●	●	
Size				●			●	
Area	●		○		●		●	●
Pattern								
Position				○				
Proximity				○				
Line thickness								
Quantity								●
Image								
Animation								

Hyperlinking								
Historical					●		●	●
Current	●	●	●	●	●	●	●	
Multi-dimensional						●		
Value		●						●
Orientation								
Texture								
Depth								
Hierarchical	○	○		○	○		○	
Network/arcs				●				

Key: ● attribute ○ present to some extent in the visualisation, but not a core element

1.1.5. ACCESSIBILITY TESTING

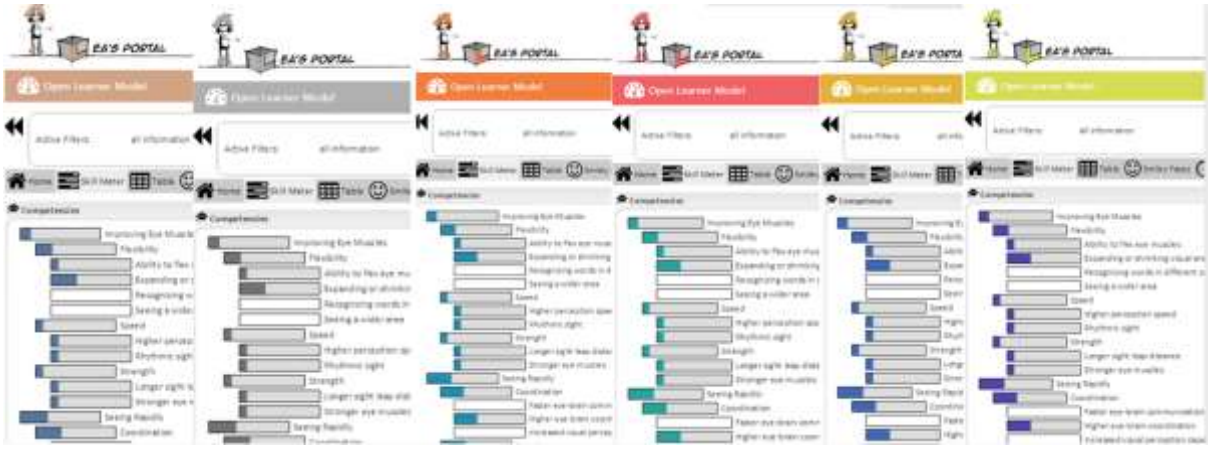


Figure 3: Accessibility Testing: Visual Impairment

Part of our development process required UoB to do an in depth analysis of the OLM software package given different forms of visual disabilities. A part of the reviewers concern was that elements of the OLM wouldn't be distinguishable from each other for students that suffered from various different forms of colour blindness. To best assess the potential problem, various bits of literature was read about the different forms of Colour Blindness. After the initial attempts to find potential users with any of the various different forms of colour blindness it was decided that a faster and more effective way of testing the software was to find a way of generating a visual output simulating the various different forms of the colour blindness.

Table 4: How Different Forms of Colour Blindness Perceive the Same Skill Meter Visualisation

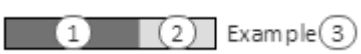
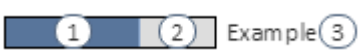
Name	Skill Meter Visualisation	Symptoms
No Colour Blindness		Able to see every colour
Protanopia		Complete absence of red
Protanomaly		Altered spectral sensitivity to red
Deuteranopia		Complete absence of green
Deuteranomaly		Altered spectral sensitivity to green
Tritanopia		Complete absence of blue
Tritanomaly		Blue-green and yellow-red/pink hue discrimination
Achromatopsia		Complete absence of any colour
Achromatomaly		Complete absence of any colour

Figure 3 above shows the OLM seen through multiple different forms of colour blindness and it can be seen that it's still possible to distinguish all the elements of the OLM. Table 4 shows how the same skill meter visualisation can be perceived by each of the different forms of colour blindness and each one can distinctly see the context of the visualisation clearly. In the development process of the OLM each

visualisation was designed and build with the same visual scheme, the level of the competency would be represented by the colour blue with the Red, Green, & Blue (RGB) values of 31, 119, and 180 respectively. Any visusalisation that represented incorrect values used the colour grey with the RGB values 221, 221, and 221 respectively. Any visualisation that aimed to represent model with no evidence would do so with a white area with the RGB values 255, 255, 255 respectively. With these RGB values for correct, incorrect, and no evidence the relative luminance can be calculated as 42%, 87%, 100% respectively. This means that even with Achromatopsia or Achromatomaly where there is a complete absence of colour in both cases, with these distinct values in relative luminance it's easy to identify each independent section.

1.2. INTERACTIVE MAINTENANCE

Interactively maintained OLM are OLM that allow students to be involved in the maintenance of their model. The aim is to help students to make their model more accurate, to give students more control over their model and to promote reflection. Several categories of interactively maintained OLM exist, notably the negotiated OLM and the persuadable OLM. In a negotiation, both the stakeholders (usually the system and the student) have the same possible moves. In case of unresolved negotiation, i.e. when they can't find an agreement, both beliefs are keeps in two separated models, each stakeholder has the control of its own model. In a persuasion, the stakeholders don't necessary have the same possible moves and the same control over the model. Most of the times, like in LEA's persuadable OLM, the system have the control on the model and the model is modified only in the case of a resolved persuasion, i.e. when a stakeholder succeed in persuaded the other. Studies have showed that students feel more confident to contribute to the OLM maintenance when there is a control of the system, when the model is updated only if the system agrees with it. For this reason, we chose to implement a persuasion facility in LEA's Box OLM.

The possible moves for the system and the learner are illustrated in Table 5. As the current implementation is a persuasion feature rather than a negotiation, we can observe two main differences between the moves available to the learner and the system. First, only the learner can try to persuade the system to change a value in the learner model – the system does not have the facility to request that the learner revise their own learner model value. Secondly, the discussion can only be initiated by the learner, and only they have the option to challenge the evidence used by the system to calculate the value in the model – i.e. the system cannot make a corresponding challenge. The statement, only available for the system, is not exactly a move but a step between two moves to sum up the current state of the discussion, such as reminding of the student's current level, their self-assessment and any justification that they have already provided to try to persuade the system to change their model.

The persuasion workflow is given in Figure 4. When a discussion of a given competency is initiated by the student, the system displays the student's current level for this competency as a statement. Then, the student can either request evidence or self-assess. The move "request evidence" is available for the student during all the discussion. The evidence explains how the current level of a student is calculated for the competency being discussed. It takes into account all pieces of evidence

directly associated with this competency and the student's current level in its sub-competencies. A direct piece of evidence can, for instance, be a score in a quiz, a teacher assessment, or the result of a past discussion of this competency. Each piece of evidence has a weight: the more recent a piece of evidence, the higher its weight.

The student's self-assessment is followed by a statement by the system that reminds the student of their current level and their self-assessment. Then, the system requires justifications in order to increase or decrease the student's level to fit the student's self-assessment. In order to either accept or decline the student's self-assessment, the system uses the discussion parameters defined by the teacher. These parameters (described below), can take into account the student's justifications, current level and self-assessment.

Table 5: Moves for Each Stakeholder, with Examples

	Student	**System**
Accept/agree	Agree with the system's evidence; Accept a compromise	Agree with the student's justifications; Accept a compromise
Decline	Decline a compromise proposed by the system	Decline a discussion (e.g. "Your last persuasion is too recent")
Compromise	Propose a compromise between the system's compromise and the student's self-assessment	Propose a compromise between the current level and the student's self-assessment
Request evidence or justifications	Request evidence for current level	Request justifications for a self-assessment
Provide evidence or justifications	Provide justifications (e.g. "I did some homework", "I had a class")	Provide evidence (e.g. "Your level in Focusing is 68 and this is a sub-competency of Seeing rapidly")

Self-assess	e.g. “I think my level should be 80”	×
Challenge evidence	e.g. “I disagree with this quiz result”	×
Statement	×	e.g. “Your level for Concentration is 75. You think it should be 80”

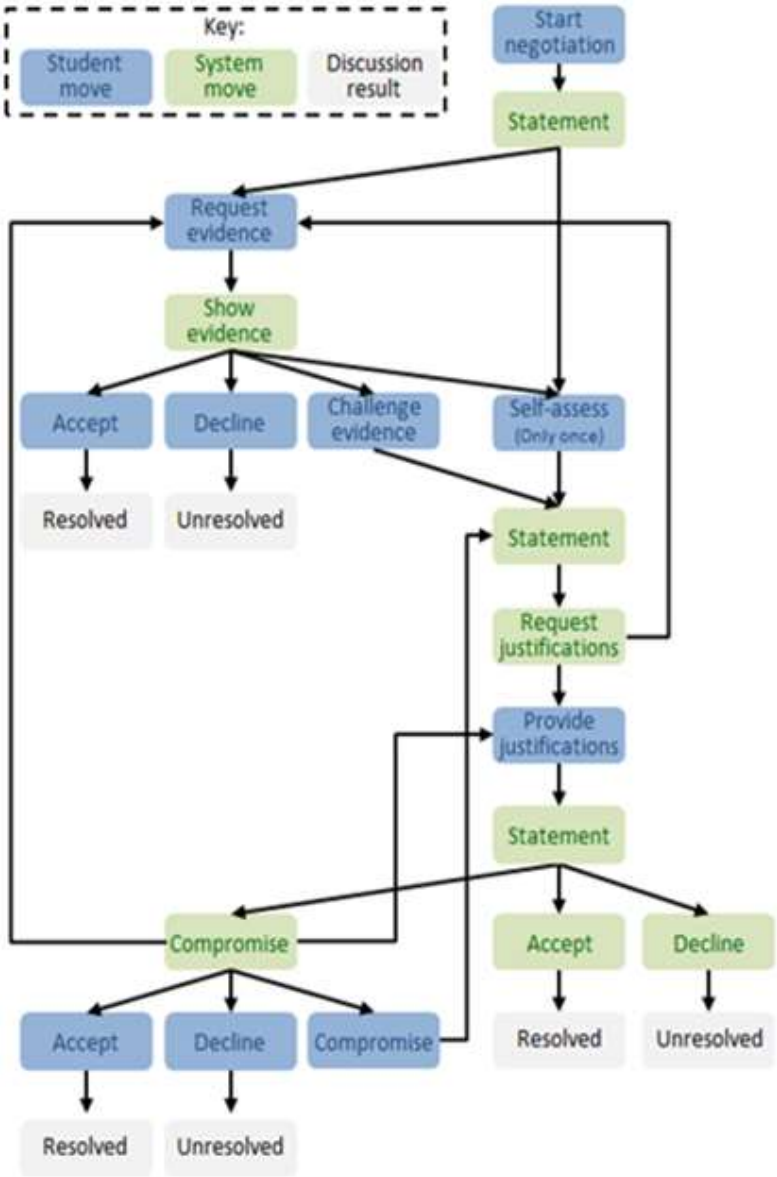


Figure 4: Persuasion Workflow

The system can also propose a compromise between the student self-assessment and their currently represented level. In the case that either the system or the student accepts a self-assessment or a compromise, the discussion is considered as resolved and the model is updated with a level that both the student and the system agreed. This leads to the generation of a new piece of evidence in the model. All evidence that is older will no longer contribute to the modelling process for the persuaded competency, but it will remain in the system in case the student or system wishes to access it later. The modelling process is presented in details in the next section. If new evidence is added after a successful persuasion, the outcome is treated the same as any other piece of evidence in the modelling process (see example in Figure 6). If a self-assessment or a compromise is declined, the discussion ends but the model is not updated as the system, parameterised by teacher, ultimately retains the control (as in other persuadable models, e.g. [7]; [21];[22]). In both cases, the discussion is recorded.

The discussion parameters are defined in the teacher's preferences page. Thus, the teacher can define a *minimum time between two discussions*, e.g. "no minimum time", "30 minutes" or "1 week". If, for instance, the time has been defined as "1 week", it means that if a student attempts to persuade the system to change a level for which a successful discussion has already happened during the week, the system will decline the student's self-assessment. The teacher can also define a *minimum number of pieces of evidence* with a source other than discussion between two discussions, such as evidence from a teacher assessment or the result of a pedagogical activity. The teacher can also define a *maximum threshold to increase* and a *maximum threshold to decrease* the level. For instance, with a maximum increase of 10 out of 100, if a student has a level of 65 and self-assesses with more than 75, then the system will offer a compromise between 65 and 75. Finally, the teacher can define the *justifications* that the student can provide during the discussion, each associated with a maximum weight. When a student self-assesses with a level higher than their current level, they will be able to provide the system with one or more justifications which are assigned a positive weight. In this case, if the student's self-assessment is higher than their current level plus the sum of their justification weights, then the system will offer a compromise between the student's current level and their current level plus the sum of their justification weights. Likewise, when a student self-assesses with a level below their current one, they will be able to provide the system with justifications, and these will be assigned a negative weight.

```
if(lastNegoTooRecent || notEnoughOtherEvidence)
{
   Decline();
}
else if(selfAssessement > currentLevel)
{
   offerCompromise = false;
   if (selfAssessement > max) offerCompromise = true;
   if (selfAssessement > calculatedValue) offerCompromise = true;
  if(offerCompromise) Compromise(min(max, calculatedValue));
  else Accept(selfAssessement);
}
else
{
   offerCompromise = false;
   if (selfAssessement < min) offerCompromise = true;
   if(selfAssessement < calculatedValue) offerCompromise = true;
   if(offerCompromise) Compromise(max(min, calculatedValue);
   else Accept(selfAssessement);
}
```

Figure 5: Persuasion Algorithm

The system's decision algorithm using these parameters is given in Figure 5, where *lastNegoTooRecent* is a Boolean that is true if the time since the last discussion of the same competency is within the teacher's parameter; *notEnoughOtherEvidence* is a Boolean that is true if the number of pieces of evidence with a source other than discussion since the last discussion is below the teacher's parameter; *max* is an integer equal to the student's current level plus the maximum threshold to increase a level defined by the teacher and *min* is an integer equal to the student's current level minus the maximum threshold to decrease a level defined by the teacher.

In the example of Figure 6, the student proposed a change from 61 to 90 after viewing evidence, and stated that extra reading has been completed and challenging the most recent piece of evidence with value 44. A compromise of 76 is offered by the system. If the student agrees, the model will be updated with this value. However, the student can also decline the persuasion, or try to continue to persuade the system by providing additional justifications and eventually by offering a compromise between his/her initial self-assessment of 90 and the compromise offered by the system of 76.

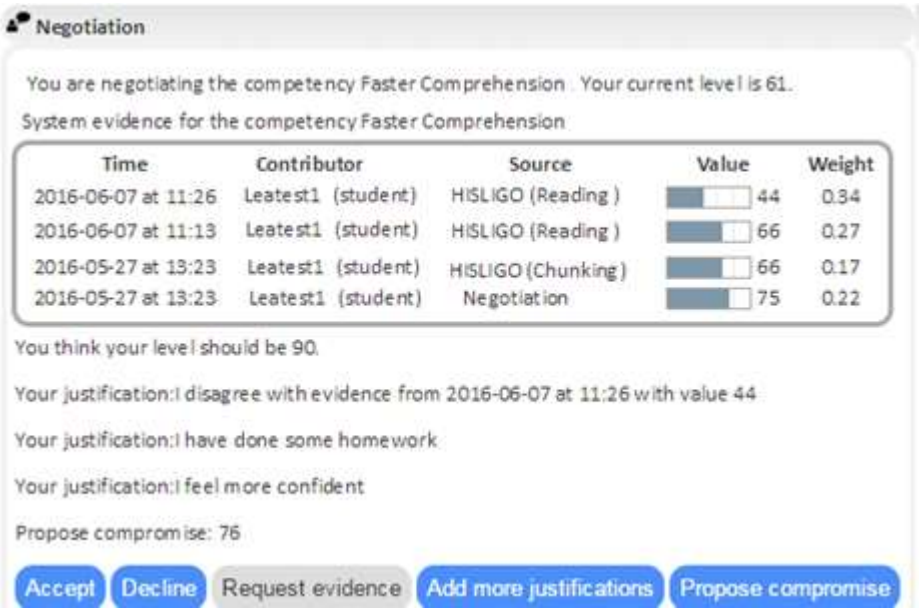


Figure 6: Persuasion example for the competency "Faster Comprehension"

1.3. FINAL LEARNER MODELLING ALGORITHM

The LEA's BOX OLM learner model calculation is based on earlier work in the Next-TELL project, taking an active learner modelling and an evidence based approach. Key amendments are in terms of the impact that the outcomes of persuasion have on the calculation of the learner model. The simplified algorithm, at the heart of the LEA's BOX OLM is shown in

Figure 7.

If the system is persuaded that the update should take place, then this is added as a new piece of information. All previous information for the given competency is retained, but does not contribute to the model calculation. The modelling process has been amended in order to take into account outcomes of persuasion. The learner modelling algorithm gives a weighted average, based on the age of the information, and this is opened to the learner during persuasion (top right of

Figure 7). It also takes into account the relevant importance of competencies and activities on an individual basis (Step 3 of

Figure 7), however in our use case this information is not known, introducing inaccuracy into the modelling process, through the omission of contextual information that allows refinement.

1. ***Evidence.*** *Retrieve all evidence within the chosen scope. Break down by student and then*

by competency. For each combination: (steps 2-5)

2. ***Basic influence.*** *Order evidence newest to oldest. Assign a relative influence to each item, with newest receiving a greater weight. Where b=basic influence, d=depreciation factor:*

$$b_i = b_{i-1}\,(1.0 - d_i) \qquad (1)$$

If the item of evidence is from persuasion, subsequently $b_i = 0$

3. ***Refined influence.*** *For the basic influence (b) of each item of evidence (e), take into account the levels of influence the competency (c), activity (a) and the competency as part of the activity (f).*

$$e_i = b_i\,(0.5 + f_i/2)(0.5 + c_i/2)(0.5+a_i/2) \qquad (2)$$

Normalise all influences so that they $\sum e_i = 1.0$

4. ***Competency Value.*** *Combine the influence* e_i *with the evidence item* v_i *where v>=0.0 and v<=1.0, to give value for the competency node (n)*

$$n_i = \sum ev \qquad (3)$$

5. ***Combine Framework.*** *If part if a structured competency framework, combine the competency with its sub-competencies (s) as per the following, where D indicates the presence of sub-competency data:*

$$n_i = \begin{cases} 0.5n_i + 0.5n_s & \text{if } D_S \wedge D_i \\ n_i & \text{if } \neg D_S \wedge D_i \\ n_s & \text{if } D_S \wedge \neg D_i \\ 0.0 & \text{if } \neg D_S \wedge \neg D_i \end{cases} \qquad (4)$$

6. ***Combine for students.*** *Combine frameworks with equal weighting given for each participant*

student.

Figure 7: The Learner Modelling Process

2.LEA'S OLM AS AN TECHNICAL AND INTEGRATED COMPONENT

This section of the report covers an in depth detailing of all the changes made to the LEA's OLM software package since the D4.2. Each subsection contains a detailed explanation as to how the improvements were made while generally explaining why they were required. These are the final implementations of each technical aspect of the OLM and integration between the OLM and LEA's Box.

2.1. FINAL OLM SOFTWARE IMPLEMENTATION

2.1.1. FINAL OLM ARCHITECTURE

During the use of the OLM in a study for University level Italian students, it was deemed inappropriate for use due to hidden information. The original database structure had the visual render of their model only appear to the user when evidence was present within the search_knowledgelevelraw table.

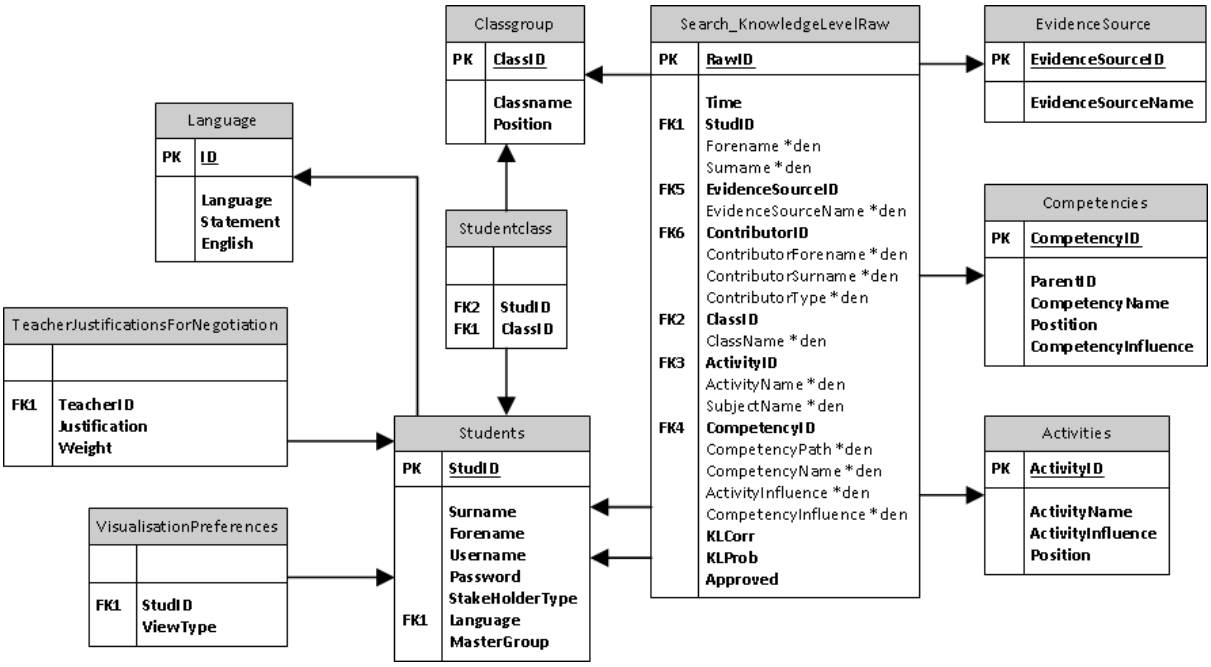


Figure 8: LEA's OLM database structure (original)

Figure 8 above is the original database structure for LEA's OLM. It was designed that all information is denormalised to the Search_KnowledgeLevelRaw table, meaning that each piece of evidence is entered into the table with all relating pieces of evidence duplicated from their corresponding tables. With this structure relationships were only visible to the user when a link between Group, Competency, Activity and Evidence Source was identified. This was problematic because if a student hadn't done any work towards a specific competency, it wouldn't be rendered as in the output and thus a student may never know about it.

After the initial study it was decided but TUG and UoB that the database structure needed to be redesigned to accommodate for any missing relationships and to mimic a closer representation to the structure TUG had already implemented in LEA's Portal.

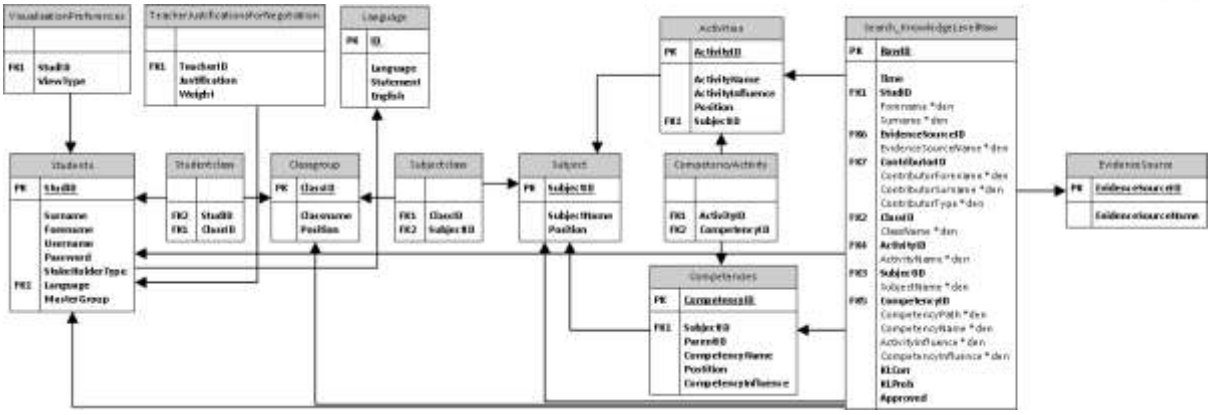


Figure 9: LEA's OLM database structure (revised)

Figure 9 above is the new database structure design. Relationships can be established between a student and nearly any other piece of information without evidence being required. The changes to the structure include a many to many relationship between Activities and Competencies, and Subjects and Groups, and a one to many relationship between Subjects and Activities, and Subjects and Competencies. Evidence still held in the Search_KnowledgeLevelRaw table is still denormalised like in the previous version of the database structure.

After the completion of the database migration, implementation of the LEA's API, and vigorous testing of the link between the Central Executive and OLM, it was found that because of the manually set IDs for the evidence data in the search_knowledgelevelraw table of the OLM that many of the IDs were duplicates. This was due to the SQL query used to insert the new evidence into the table; originally it would get the next available ID from a lookup table and then set the ID of the evidence to that. However because multiple concurrent requests can be submitted, causing many pieces of evidence would have the same ID which would cause conflicts in the OLM. The solution to this was to change the ID field of search_knowledgelevelraw to an automatic incrementing primary key and stop the OLM from manually defining it.

2.1.2. FINAL OLM DATABASE IMPLEMENTATION

During the development of the LEA's OLM package, multiple studies were being held in Turkey. During these studies it was found that the OLM was under performing and this could be directly linked to the Apache Derby database used to hold the data. While observing the database processing times when the server is under load it could be seen that getting the information caused intense stress to the server. Apache Derby is a Java based Database and as such the Java virtual machine only empties out memory when it is full. However if the host machine runs out of memory before the Java virtual

machine has reached its maximum allocated memory it makes the memory management aspect of the virtual machine struggle with the situation.

Due to difficult memory management, limited system resources, and the impossibility of accessing the Derby database outside of the University of Birmingham's intranet, it was decided to mimic TUG in their use of MySQL. The OLM database structure would be identical from one platform to the other as both MySQL and Derby run off a Structured Query Language. Development time for the OLM to support MySQL was minimal and nearly all the code worked exactly as originally intended, the only difficulty lay in transplanting the data held in Derby as it doesn't support exporting data to CSV or SQL files and any attempt at using a GUI to do so ended in failure due to the size of the data.

After the migration was completed the performance could be seen to be greatly improved, the database could be accessed from outside of the host server, and database backups could be taken as CSV files making them take less storage space and thus could be held for longer.

2.1.3. INTEGRATION WITH VISUALISATION SERVICES

The purpose of the visualisation service is to create a lightweight web service allowing for many of the LEA's Box applications to visually represent the graphical outputs to its users in the same familiar appearance.

```
{
    "name" : "String",
    "data" : "Double",
    "title" : "String",
    "id" : "Integer",
    "parent" : "Integer",
    "dismensions" : "Integer",
    "source" : "Integer"
}
```

Figure 10: Original JSON structure used by Visualisation Service

The OLM is built to load its content asynchronously; it builds the model for the user after the initial page has loaded. Because of this the OLM communicates with the Visualisation Service, as the last part of the rendering process, using multiple AJAX HTTP Post requests. The model is packaged up and sent as a JSON object that resembles the structure in Figure 10 and is sent in a Post method as it is too large for a HTTP Get request.

```
{
    "name" : "String",
    "data" : {
        ["value" : "Double",
            "time" : "String",
            "negotiation" : "Boolean"]
    },
    "title" : "String",
    "id" : "Integer",
    "parent" : "Integer",
    "dismensions" : "Integer",
    "source" : "Integer"
}
```

Figure 11: Revised JSON structure used by Visualisation Service

Figure 10 was the JSON object originally used by the Visualisation Service to render a One Dimensional model. As development of the OLM improved the capabilities of the model rendering process, the OLM became able to produce Two Dimensional models. As such both the Visualisation Service and the JSON structure needed to be adapted to allow for support of a two dimensional model, scaling value against time. For this to work the JSON object had to be able to associate the value of the student's model against specific times in their progress towards any competency. As such the JSON object needed to accommodate an array of these data points which lead to the development of Figure 11.

During the third year of the project, a study run by SEBIT introduced the Negotiation system of the OLM to the study's students. This granted the students the opportunity to alter their model by convincing the OLM software that their understanding of the competency was different from how the system rendered it. Upon a successful negotiation, the OLM will accept the evidence provided by the student and the OLM will update its model to accommodate the negotiation. For the one dimensional visualisations it isn't possible to distinguish the difference between negotiated evidence and evidence from other data sources because the model calculates the value to be a single point of data, so there is no comparison between data sources. The only instance when a user would be able to identify when points of Negotiation had occurred was when they were observing the Across Time visualisation.

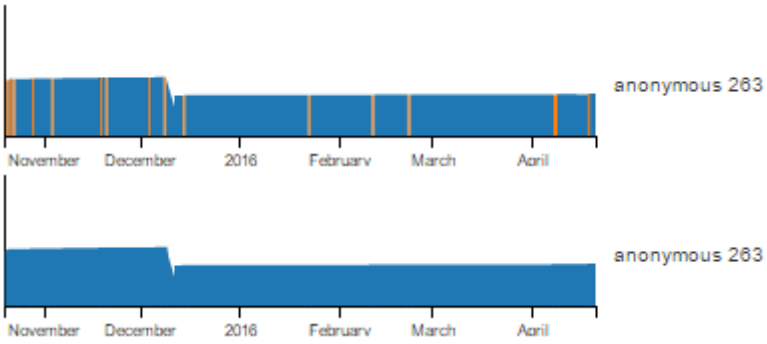


Figure 12: Across Time visualisation with and without Negotiation points.

Figure 12, depicts two sets of the same data using the Across Time visualisation for a single user. The top data indicates where on the student's progress they succeeded in using the Negotiation system to influence their model and the bottom data has this indication removed. Because of the difficulty in understanding how the results of the negotiation can impact the model, it was advised the model indicate the points of negotiation. In Figure 11: Revised JSON structure used by Visualisation Service, it can be seen that within the "data" array, one of the elements is a Boolean associate with the index "negotiation". This Boolean is used to indicate to the Visualisation Service that the point of data it is inspecting is a point of negotiation and thus to treat it differently.

2.2. INTEGRATION WITH LEA'S BOX AND EXTERNAL APPLICATIONS

2.2.1. INTEGRATION AND LEA'S API

The aim of LEA's Box was meant to be a lightweight package that multiple external data sources could submit their data to for educational purposes. The problem however is that this data has to be freely given by the external packages, or alternatively the data needs to be extracted by the box from the source. The method for extracting the data is commonly referred to as Scraping or Data Mining if the data is used for different purposes other than the original intention. Unfortunately the development time to build a tool to extract the information can be extremely time consuming and one needs to be built for each individual datasource. The alternative method was to allow for the opportunity to have third party extensions of the software created allowing the data to be freely given to the box, this would be accomplished using an API.

Property	Type	Description	Required
id	UUID	UUID assigned by LRS if not set by the Activity Provider.	Recommended
actor	Object	Who the Statement is about, as an Agent or Group Object. Represents the "I" in "I Did This".	Required
verb	Object	Action of the Learner or Team Object. Represents the "Did" in "I Did This".	Required
object	Object	Activity, Agent, or another Statement that is the Object of the Statement. Represents the "This" in "I Did This". Note that Objects which are provided as a value for this field should include an "objectType" field. If not specified, the Object is assumed to be an Activity.	Required
result	Object	Result Object, further details representing a measured outcome relevant to the specified Verb.	Optional
context	Object	Context that gives the Statement more meaning. Examples: a team the Actor is working with, altitude at which a scenario was attempted in a flight simulator.	Optional
timestamp	Date/Time	Timestamp (Formatted according to ISO 8601) of when the events described within this Statement occurred. If not provided, LRS should set this to the value of "stored" time.	Optional
stored	Date/Time	Timestamp (Formatted according to ISO 8601) of when this Statement was recorded. Set by LRS.	Set by LRS
authority	Object	Agent who is asserting this Statement is true. Verified by the LRS based on authentication, and set by LRS if left blank.	Optional
version	Version	The Statement's associated xAPI version, formatted according to Semantic Versioning 1.0.0.	Not Recommended
attachments	Array of attachment Objects	Headers for attachments to the Statement	Optional

Figure 13: xAPI specification (reproduced from http://tincanapi.com)

Figure 13 above is a sample of the Experience API (xAPI), also known as TinCanAPI; this API call is used in the creation of a statement. The purpose of the xAPI is meant to allow Learning Management Systems to communicate with other educational software packages over a standard of communication that's predefined and well established. However the statement in Figure 13 can be used to define nearly anything for the developer's intended purpose and because of the xAPI's design the level of complexity in using it is high and many different calls must be used to complete a single task because it is very discrete.

```
http://.../leas-
portal/api/pushin.php?secrettoken=*****&studentid=1&datasourceid=39
&activityid=85&value=3&minvalue=2&maxvalue=4
```

Figure 14: LEA's API Data Import Method

Because of the impact on development time would jeopardise any potential studies, it was decided that the Box would use its own API instead of the xAPI standard. LEA's Box API is a small lightweight API that works in tandem with the LEA's Configuration Tool. The API allows for submission of data using the HTTP request in Figure 14. Each argument, represented by underline, is crucial in the request and is used to ensure that the data is valid and is being associated to the right account.

Once the LEA's API has received the data submitted to it from the educational/professional software package, it is processed by the Central Executive and then handed on to the OLM. The OLM's database serves as a duplicate of the Box's database; all the records held by the Box are duplicated into the OLM and controlled exclusively by the Box. Though it is not good practice to have duplicate data housed across multiple databases and servers, it was the most viable solution with the resources available. LEA's OLM was initially developed to utilise a Derby database (variant of SQL) whereas the Portal was developed in PHP and designed to use MySQL. Though both database platforms function similarly in querying them, the PHP Portal didn't support the Derby database hosted for the OLM and the OLM couldn't communicate to the MySQL database hosted for the Portal due to security restrictions. In addition to database communication issues, another problem with the OLM and the Portal getting its data from the same database was the increasingly complex queries to the database by the OLM where a query can take several seconds preventing the Portal from getting any of its queries resolved. All these issues lead to the OLM having its own independent database that is configured from the Configuration Tool developed by TUGraz via the OLM API.

2.2.2. STRESS TESTING THE LEA'S API WITH OLM INTEGRATION

Figure 15: Sample of data being submitted to the Central Executive via the Stress Test Tool

After the development of the LEA's API that allowed the feeding of data between the Portal and the OLM, it required stress testing to determine how much information wouldn't arrive at the OLM. A JavaScript web page was built to send randomly generated data to the Portal as both integer values and floating point values as quickly as for five minutes or until the person operating the test manually shut it down, this is shown in Figure 15. It was found that the test did not operate as intended as the near constant stream of HTTP GET requests caused the JavaScript to inaccurately determine the time and the trigger used to stop the requests was never activated. 10,359 records were submitted to the LEA's API through the stress test and 10,356 were received by the OLM, this means that there's roughly a 0.029% chance that the data won't be received. To accommodate for this possibility the OLM API notifies the LEA's API if it successfully received the data submitted, otherwise the LEA's API will assume it's a failure and log it within the portal that the data wasn't successfully submitted, this allows for an opportunity to resubmit the data at a later point. In addition to the stress test tool identifying the percentage change of the data failing to be submitted, it also allowed us to determine a fundamental flaw that the Database architecture causing multiple pieces of evidence to have the same primary key. This is better explained with solution in section 3.2.ii Final OLM Database Implementation.

2.2.3. EXPANSION OF THE OLM API

As revealed by the Stress Test documented in section 3.2.ii of this document, it was found that approximately 1 in 3,453 records never reached the OLM. In order to resolve this issue the data is kept by the Central Executive to be later sent again to the OLM but the OLM's API needed to be altered.

Table 6: Improved OLM API method for submitting evidence

Arguments	Description	Returns
sharedsecret **leasid** **method** **competencyid** **groupid** **userid** **datasourceid**	access password id number of the user logged in ***"addinformation"*** id of competency id of the group id of user to be updated id of the datasource	On success of adding information: 'information added: "<time of addition>" value:"<value>"'. Else 'competency does not exist in the database', 'group does not exist in the database', 'user does not exist in the database', 'datasource does not exist in the database', 'user is not a member of the group', 'user is a teacher', 'value is not a number', 'value should be in a range of 0 to 1',

value **activityid** **[timestamp]**	inference value (range: 0 to 1) id of the activity time of evidence added (optional)	'adding information failed'.
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=addinformation&competencyid=198&groupid=72&userid=1001&datasourceid=20&value=0.756&activityid=911		

Table 6 above is the new appended API method for submitting evidence data to the OLM. The alteration required to accommodate for missing evidence is the timestamp argument. This argument is completely optional but allows the developer to submit the timestamp with a piece of evidence, if the timestamp is present then the OLM will submit the evidence to its database with that explicit timestamp, alternatively the OLM will submit the evidence with the current time as the timestamp of entry.

2.2.4. IMPORTING DATA FROM EXTERNAL APPLICATIONS

During the third year of the project multiple studies were done using the external education software package Hizligo. Part of the studies involved the students using the OLM to reflect on their progress within the competencies they were associated to. In order for this to happen the software package Hizligo needed to submit data to LEA's Box where it is then processed. When the data is received by the box it can generate large volumes of relationships between a single piece of data from an activity and any competencies associated with that activity. So when the data is received and processed by the Central Executive it is then stored in a rawdata table held by TUGraz. This table is to prevent data loss as with large volumes of data a single push from the Central Executive to the OLM can result in approximate a 0.01% chance of the data not arriving. This is identified by the Central Executive by a successful receipt of the data, if so the data in the rawdata table is marked with a success; else it is marked with a failure. From there the failed attempts can simply be resubmitted to the OLM without any issue.

id	datasourceid	activityid	value	minvalue	maxvalue	timestamp	studid	token	status
22742	41	235	5	1	7	2016-02-16 17:02:27	b5259c750fa2c2c63b3307a53f224f65	ofby31Wqr4	True
22743	41	235	5	1	7	2016-02-16 17:02:53	b5259c750fa2c2c63b3307a53f224f65	ofby31Wqr4	True
22744	41	237	5	1	7	2016-02-16 17:03:47	b5259c750fa2c2c63b3307a53f224f65	ofby31Wqr4	True
22745	41	247	2	1	3	2016-02-16 17:06:02	b5259c750fa2c2c63b3307a53f224f65	ofby31Wqr4	True
22746	41	250	5	1	20	2016-02-16 17:08:21	b5259c750fa2c2c63b3307a53f224f65	ofby31Wqr4	True
22747	41	252	100	0	100	2016-02-16 17:15:02	b5259c750fa2c2c63b3307a53f224f65	ofby31Wqr4	True
22748	41	233	3	1	7	2016-02-16 18:13:32	bae1a65e5ab6e33714aaee6f48e8ce4c	ofby31Wqr4	True
22749	41	235	4	1	7	2016-02-16 18:13:57	bae1a65e5ab6e33714aaee6f48e8ce4c	ofby31Wqr4	True

Figure 16: data import, an example

Figure 16 is a segment of live anonymous data within the rawdata table. DatasourceID, and ActivityID have a direct relationship with ID's held between the Portal and the OLM. Within the rawdata table the StudID field is used as a reference of the student for external applications, in a relationship table held by the Central Executive exists the link between external ID and internal ID. From here the Central Executive can look up relationships between Activities and Competencies and then for each combination of Activity, and Competency for the Student with predefined Group and Subject a request is made to submit the data to the OLM.

Additionally during the third year of the project another study was undertaken using the Hizligo software package. There was however a change to the case study requiring that the OLM had a different competency architecture from that of other applications in LEA's Box. This was artificially done using the API to remove the lowest level of the hierarchy and thus when the Central Executive attempted to pass the data to the OLM, this discrepancy caused the error checking to always fail making it impossible to know any point in the study if any data failed to be submitted.

2.2.5. IMPORTING DATA AND PATCHING BETWEEN FORMATS: CANVAS IMPORT TOOL

During a study with University level Italian students, students were asked to use the Canvas teaching platform in order to run quizzes to assess their progress. This information was put into the OLM database manually as Apache Derby has no import facility, and the CSV export from Canvas was in an incompatible format. As such an import tool was constructed to specifically accommodate the Canvas CSV files. Seven pieces of information were required in order to import the data into the OLM directly, Student ID, Group ID, Competency ID, Activity ID, Evidence Source ID, Subject ID, Value. The only pieces of data that were available in the Canvas CSV were an anonymous Student ID, the Activity Name, and the value. From this a GUI was constructed to help assist the importing of student data.

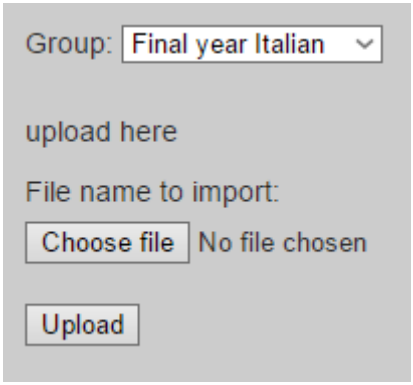


Figure 17: Canvas Import Tool, initial interface

Figure 18: Canvas Import Tool, upload URL

Figure 17 is an image depicting the simplicity of the Canvas Import Tool. Figure 18 is the end of the address bar requesting the Canvas Import Tool. It shows that in order to access the Canvas Import Tool you need a valid teacher ID, once one is found the user is given access to the tool as shown in Figure 17. Upon uploading a CSV file to associate with a specific group, the user is then given the option to implement a relationship between the anonymous Student ID held within the CSV file and the students, by name, in the selected group. These choices are remembered by the tool to reduce repeated work. Additionally the user is able to associate Activities held in the CSV and Activities associated to the group via the Subject relationship they would both share, and lastly relationships that exist between Activity and Competency can be used to activate the relationship for the CSV file. After all the settings are made the Canvas Tool displays what evidence is going to be entered into the OLM and the user is then given an opportunity to proceed.

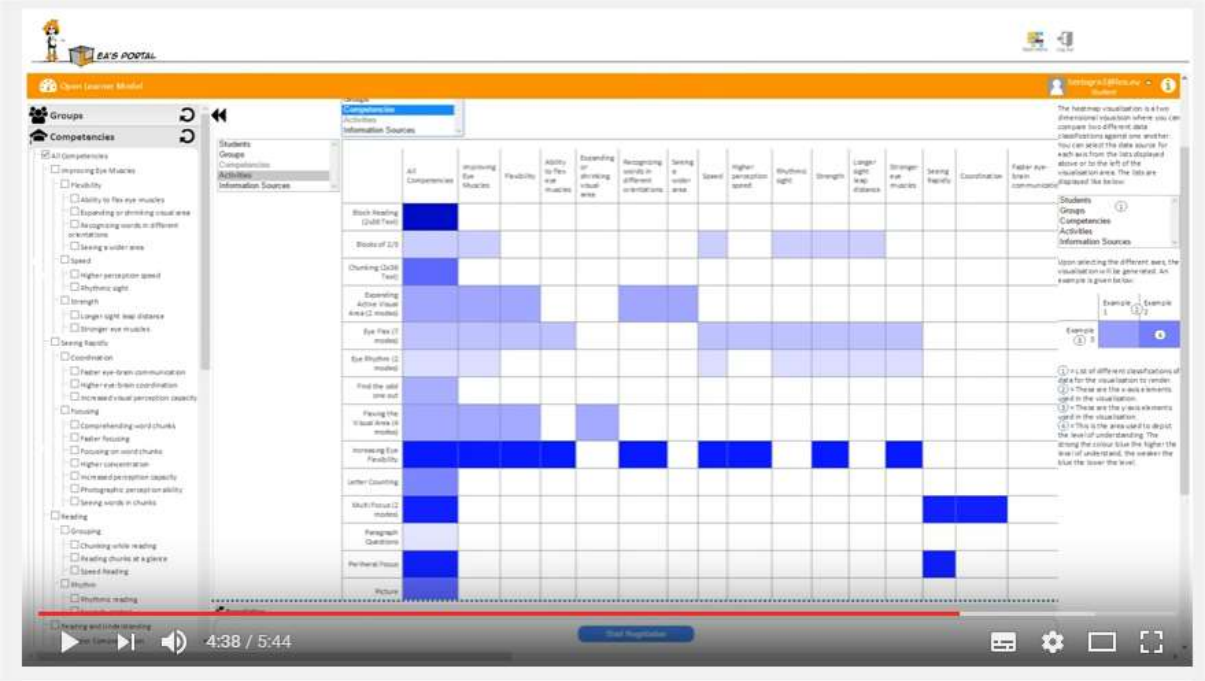
Initially it would take approximately three hours to manually add student data into the OLM database, after the initially set up of the Canvas Import Tool this time was reduced to under five minutes.

The development process for the Canvas Import Tool was estimated to take between six hours, it actually took six days. The reason behind this was due to the incompatibility of the information. As stated earlier, the OLM required seven pieces of information in order to submit the evidence and the Canvas CSV file didn't supply it. Much of the time in development was spent building the interface between user and the Import Tool, crucial information was missing and had to be created at the same time as the final implementation of the OLM database was being designed. The Canvas Import Tool suffered in the development time because by this point LEA's API didn't exist and the design of the Import Tool had to accommodate for a changing OLM Database structure. With the development of the LEA's API, see section 3.2.i, software packages like the Canvas Import Tool could be developed

much faster regardless of the state of the internal design changes of various components. This is alone is why the integration between the Central Executive and HISLIGO took less than 8 hours.

3.USING LEA'S OLM

3.1. ONLINE RESOURCES



https://www.youtube.com/watch?v=Snfi_qsutxc

https://www.youtube.com/watch?v=JsX_H8J-d3g

3.2. USER MANUAL – UPDATED FROM D4.3

Included as Appendix X

3.3. HOW TO CONFIGURE AND USE THE OLM

The purpose of this section of the document is to aid users on how to both configure and use the OLM. This set of step by step instructions aim to remove the ambiguity behind the multiple different ways of how to configuring the OLM

3.3.1. REGISTERING TO LEA'S BOX AS A TEACHER

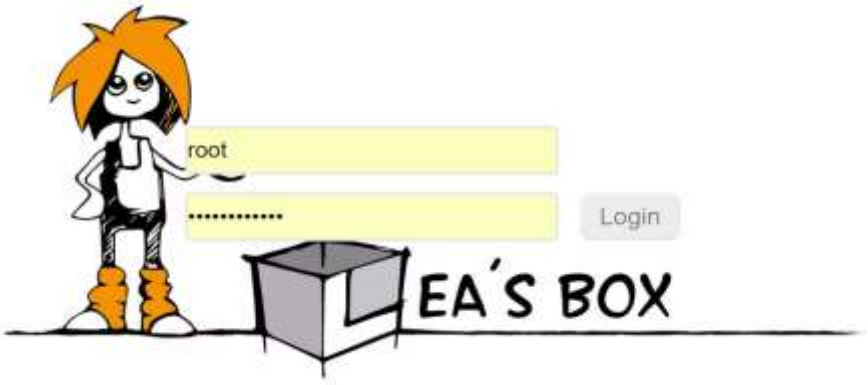
Because LEA's Box is a research project, it was decided not to allow self-registration of users. In order for a Teacher to have an account, it needs to be set up for them by an Administrator of LEA's Box. This process may include setting up a new school since the Box uses a static role based security system utilising the school id of a user. The administrator can log into the configuration tool and start with adding teachers.

Teachers are created by defining a username, and password and specifying a firstname, and surname. Eventually they will also need to be assigned to groups (i.e. classes) and subjects but for account creation this isn't required. The assignment of groups and subjects is important for the rights and access management. Teachers can access all information for any group(s) or subject(s) they are assigned to.

3.3.2. LOGGING INTO LEA'S BOX

To log into the LEA's Box, simply go to the following web page and enter your details and then using your mouse, click on the Login button:

http://css-kmi.tugraz.at/mkrwww/leas-portal2/index.php

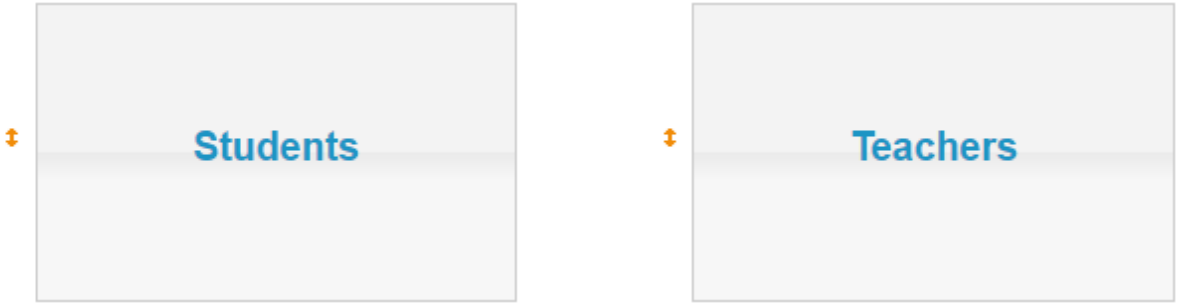


3.3.3. FINDING THE CONFIGURATION TOOL (SETTINGS)

To locate the Configuration Tool (Settings), simply log into the LEA's Box as a Teacher and look for the following icon:

3.3.4. CREATING TEACHER AND STUDENT ACCOUNTS

As a Teacher you can create both other teacher and student accounts. Simply navigate to the Configuration Tool (Settings) and you'll be able to find the following options:

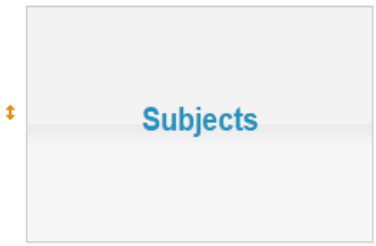


In order to create new students, select the Students option within the Configuration Tool (Settings). You'll be welcomed by a list of different students already created by yourself or other Teachers within your school. Before creating a new student, make sure they don't already exist within the aforementioned list. To create a new student simply press the new button located towards the top left of the list, once you do you'll be presented with the following pop up window:

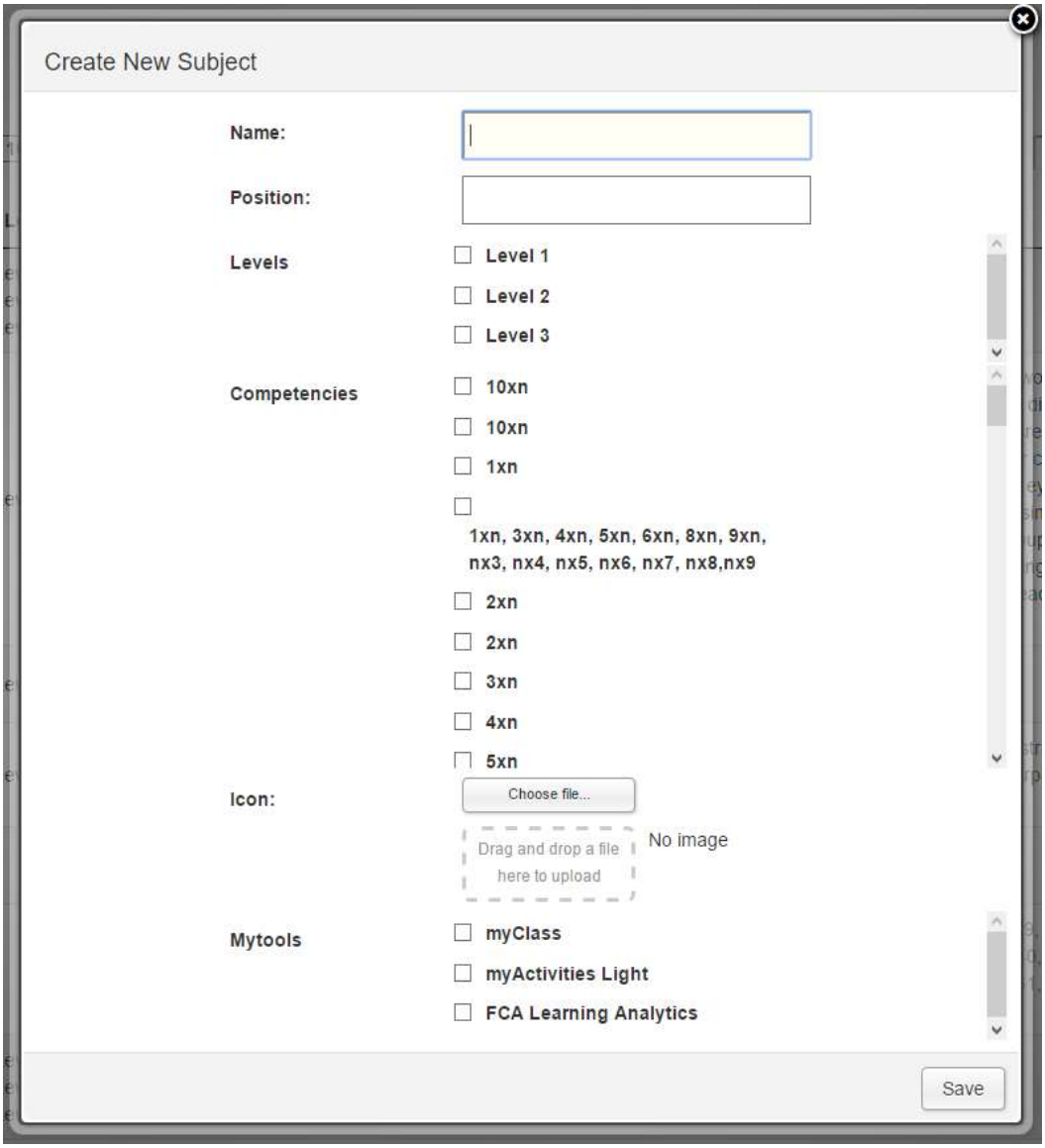
To define a student a Teacher needs to specify a Student's User ID, this is what is used to log into the LEA's Box, First name, Last name, and Password. Once these are entered into the system and saved the student is then able to log into LEA's Box. There are additional options that may be applicable to your institution: Level, Language and Icon.

3.3.5. CREATING SUBJECTS AND GROUPS

In Lea's Box, subject refer to a typical school subject such as maths, biology or any other set of competencies (e.g., cross subject meta abilities). In order to create new subjects, simply click on the Subjects button located in the Configuration Tool (Settings):

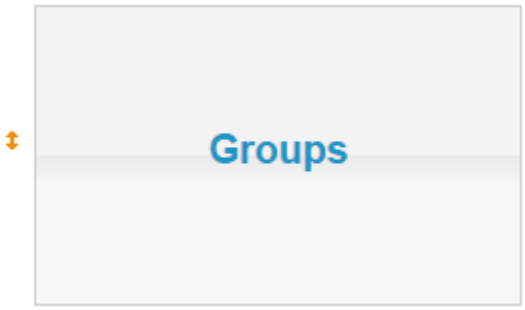


When entering the Subjects section of the Configuration Tool (Settings), you'll be introduced to a list of all the existing Subjects. To create a new subject simply click on the New button located to the top left of the list. When you click the New button you'll be presented with the pop up window below:

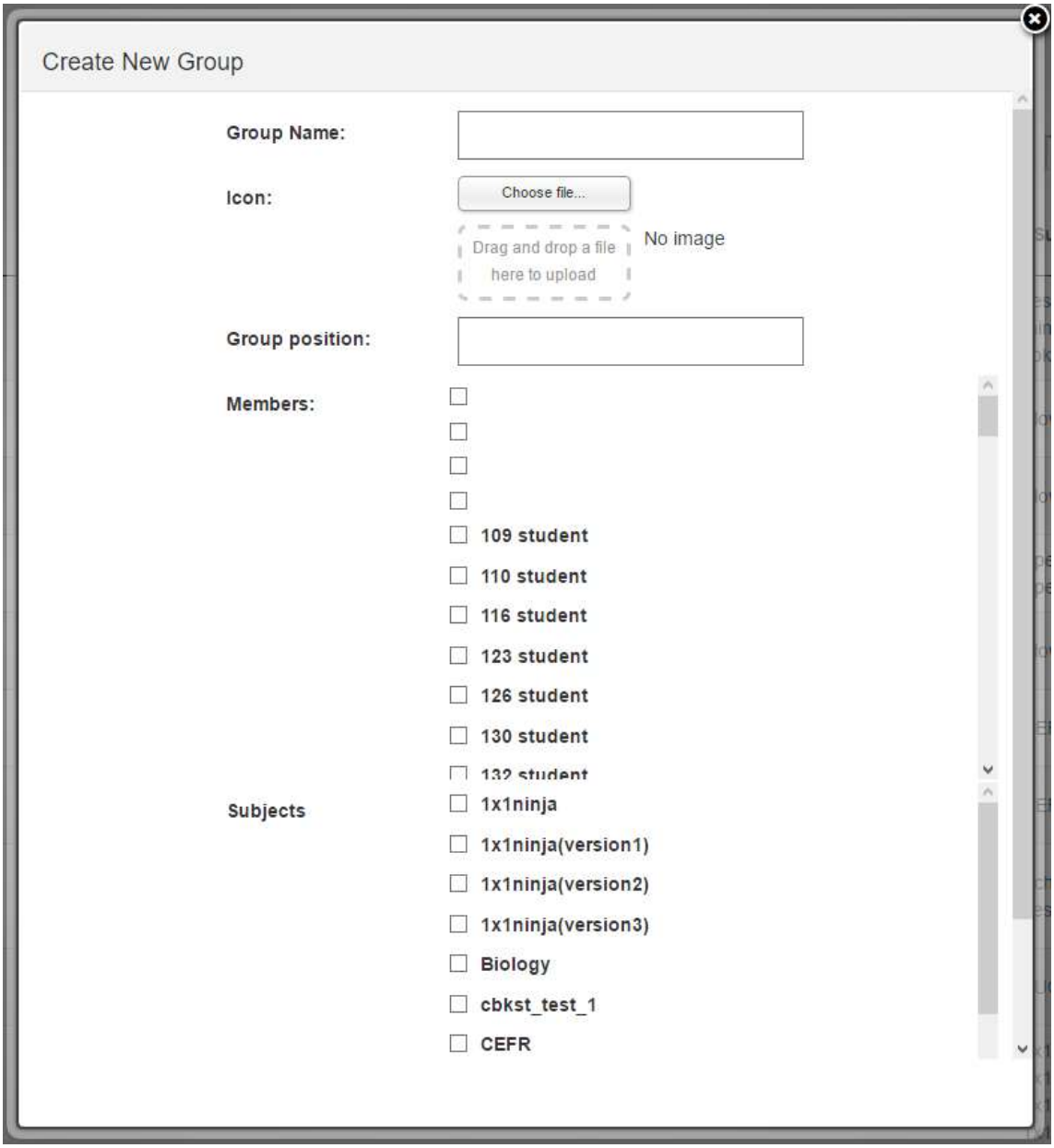


Subjects are defined by their name, all the other information is optional. A teacher can select one or more school levels with which the subject is associated. Optionally a teacher can select the related competencies. This however, is likely not the most user friendly approach, thus competencies can be assigned to subjects in competency module.

Groups define a set of students; this can be a regular class or certain sub-groups. To create a Group simply click on the Groups button found in the Configuration Tool (Settings):



From there you'll be presented with a list of groups. To create a new group simply click on the New button located to the top left of the list. You'll be presented with the following pop up window:



To create the group you only need to enter the Group Name, all the other options are entirely optional. From the same panel you are able to create the relationships between the groups and subjects but if you haven't created your subjects yet you can come back to this group by editing it and adding the at a later date. You can also add other users to the group, including other teachers, but you're not forced to do it immediately, this can be done in the future also.

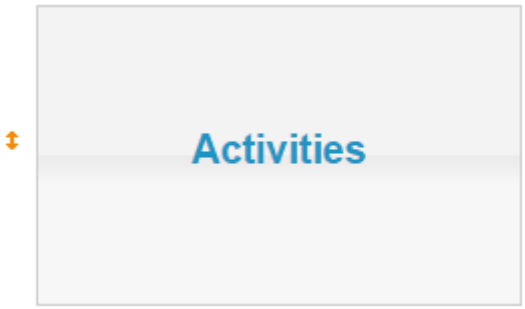
3.3.6. CREATING DATA SOURCES AND ACTIVITIES

A data source is simply the term used to define where the evidence is coming from that would go towards a student's understanding, for example a test or piece of homework would be considered a data source as well as specialist software. To create a Data Source simply click on the button in the Configuration Tool (Settings):

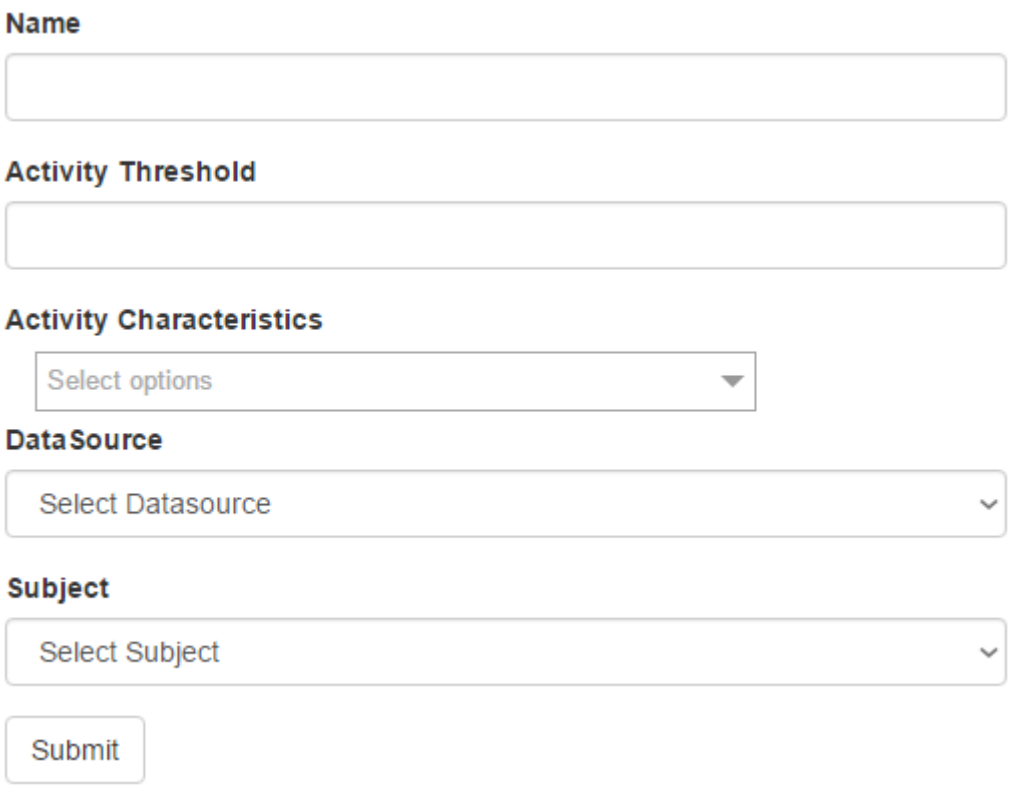
Upon entering the Data Source section of the Configuration Tool (Settings), you'll be presented with a list of already existing Data Sources. To create a new data source simply click on the Add button located to the top left of the list. Once you do the following screen will appear:

A data source is created by defining a name and linking it with a subject.

Activities refer to sub-part of a data source; imagine an external source is Moodle. This Learning Management System may have a course including several quizzes. These, in turn, may be linked to different subsets of competencies of the particular course. To create new activities click on the Activities button on the Configuration Tool (Settings):



Upon entering the Activities section of the Configuration Tool (Settings) the user is presented with a list of all existing activities. To create a new one simply click on the Add button located to the top left of the list. Once you have done that the page will change to look like the following:



In Lea's Box, activities are defined by a name, a Data Source, and a Subject.

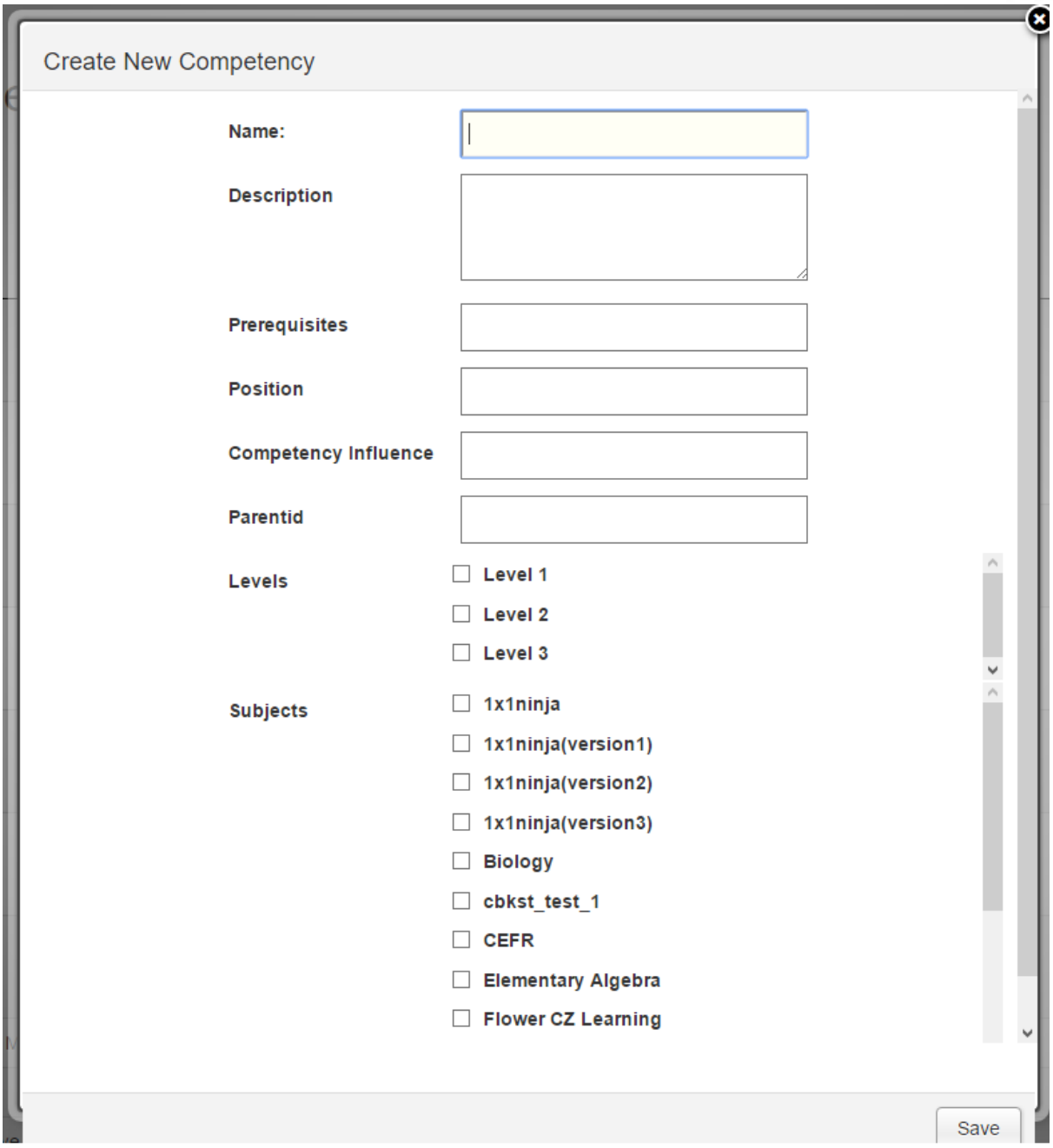
Please note that activities can be also defined as internal entities that can be recorded by teachers. This is one of the key functionalities of myClass, for example.

3.3.7. CREATING COMPETENCIES

Competencies and competence models are likely the most complex concept in this Learning Analytics system. Specifically since the Lea's Box system is dealing with two distinct types of competence models. On the one hand, the tree-like hierarchical models of the OLM, and on the other hand, the combinatorics structures and spaces of the CbKST approach.

To create a new competency click the Competencies button of the Configuration Tool (Settings):

Upon entering the Competencies section of the Configuration Tool (Settings) you'll see a list of all the currently existing competencies. In order to create a new one simply click on the New button located to the top left of the list. Once you've done so a pop up window will appear that will look as follows:



Competencies are defined by a name, a position in the hierarchy (defaults to 1), the ID of a parent competency (can be found in the list on the previous screen, defaults to 1), and the Subjects it's related to. As previously stated, a user can define a hierarchy by adding the id of the parent competency. The following diagram illustrates the concept. This is, of course, not a very user friendly approach, but in the context of this research project and given the scope of the project, a functional solution.

Top Level Competence (id: 1)
Child Competence (id: 2) > parentid = 1
Child Competence (id: 3) > parentid = 1
Competence (id: 4) > parentid = 3
Competence (id: 5) > parentid = 4
Child Competence (id: 6) > parentid = 1

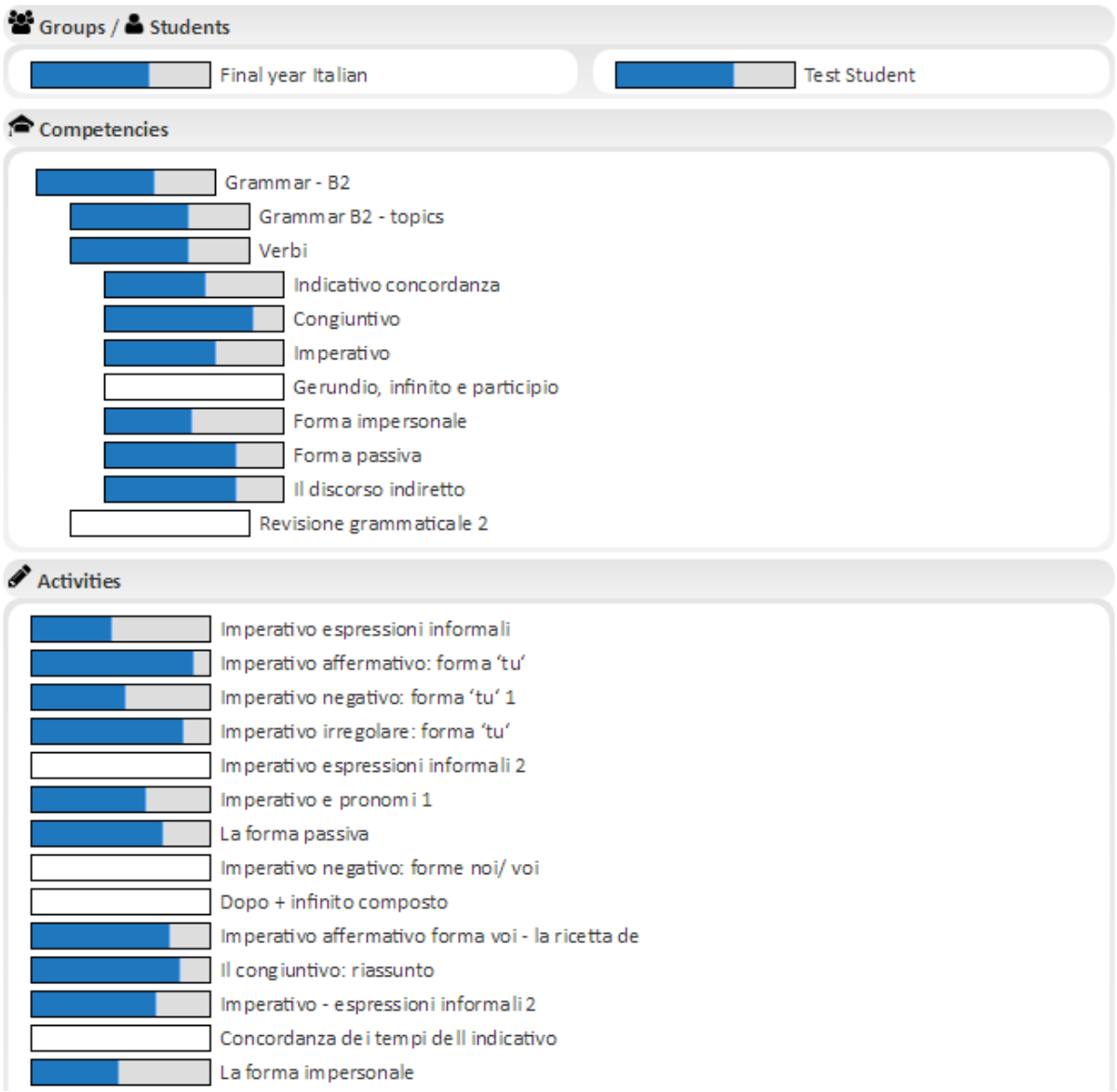
3.3.8. FINDING THE OLM WITHIN LEA'S BOX

To locate the OLM, simply log into the LEA's Box as a Teacher and look for the following icon:

3.3.9. UNDERSTANDING YOUR LEARNER MODEL

When a student or teacher submits work towards a student's model, the model is actively updated in real time. Both student and teacher can observe the student's model in the OLM. There are seven different visualisations that are used to represent a student's model. Below is an example of a single Visualisation Service, the Skill Meter:

When you select the Skill Meter visualisation you'll be presented with a lot of different instances of each of these Skill Meters. The student's model can be observed from many different angles, from the perspective of the competencies individually, the activities that have contributed towards the model, the data sources that feed the activities, each group that the student is in or as the student as a whole.



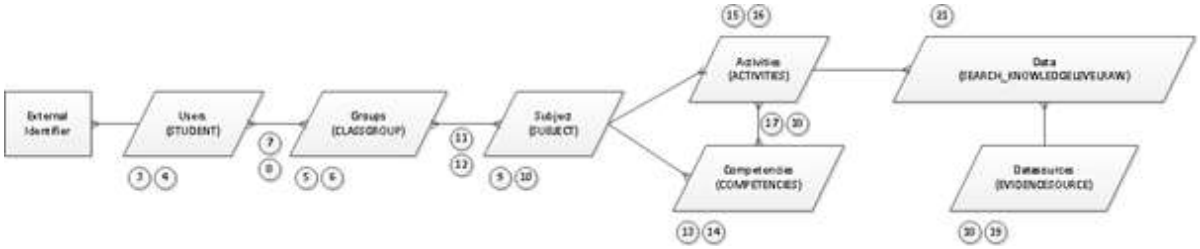
Each section will show the exact same evidence but in different ways. The above screenshot shows the majority of a model with demo data. This Test Student belongs to a single group, which means that the student's model visualised as everything contributing to the student perspective and the group perspective is 100% identical, if the Test Student account belonged to multiple groups then the student's perspective of the model would be the average of all the groups' perspectives. Likewise this

can be better understood with all the competencies displayed. Grammar - B2, which is at the head of the hierarchy, is the average of all the evidence that contributes to the model in the sub tiers of the hierarchy, and just like the group and student perspective of the model the Grammar - B2 competency is 100% identical with the student perspective.

However the competencies are made from all the evidence within the sub tiers and any evidence provided directly to the competency. So from the screenshot above, the Verbi competency is fed all the evidence from its seven sub competencies (including empty ones) and is averaged out with any evidence directly fed to the Verbi competency. After that the Grammar - B2 competency is fed all the evidence of its 3 sub competencies (including empty ones) and is averaged out with any evidence directly fed to the Grammar - B2 competency.

4.APPENDIX – API SPECIFICATION

The revised API calls for the learner modelling software are included in this document. These bring the configuration into line with the following architectural description, as defined during the November 2015 meeting.



The OLM will display a message if it is not logged in. The ability for the user to log in direction has been disabled for the purposes of integration. (Access should be through the portal).

A lightweight API is included for the purposes of LEA's BOX integration work. These are designed to take ID numbers as parameters, so that the portal (and in particular the configuration tool) is the only entity that needs to define them. The ID numbers are thus the same across systems based at TUGraz and UoB. The same function call will complete both the *add* and *update* actions. If an entity does not exist, then it will be automatically created.

4.1. LOG IN ("LOGIN")

Arguments	**Description**	**Returns**
sharedsecret **leasid** **password**	access password id number of the user user's password	On successful log in: 'LOGIN SUCCESSFUL FOR "<user name>".' Else: 'user could not be found', 'password encryption was not successful' or 'password is incorrect'.
Example http://.../leas-olm/api/login?sharedsecret=********leasid=1011&password=12345		

4.2. LOG OUT ("LOGOUT")

Arguments	**Description**	**Returns**

sharedsecret	access password	On successful log out: 'log out was successful.' Else: 'user is not logged in', 'log out failed'.
Example http://.../leas-olm/api/logout?sharedsecret=********		

4.3. ADD/UPDATE USER ("UPDATEUSER")

Arguments	Description	Returns
sharedsecret **leasid** **password** **username** **forename** **surname** **school** **type**	access password id number of the user user's password username of the user first name of the user last name of the user id of the user's school *"student"* or *"teacher"*	On success of adding user: 'user was added to the database: "<user name>" (id:"<user id>")'. On success of updating user: 'user was update: "<username>" (id: "<leasid>")'. Else 'Please enter the parameter "sharedsecret/leasid/password/username/forename/surname/school/type"', 'Lea's ID number ("<leasid>") must be an integer', '"<type>" was not recognised. This should either be "teacher" or "student"'.
Example http://.../leas-olm/api/updateuser?sharedsecret=********&leasid=1011&username=bbrown&forename=Bob&surname=Brown&password=12345&school=masterGroup&type=teacher		

4.4. DELETE USER ("DELETEUSER")

Arguments	Description	Returns
sharedsecret **leasid** **method** **userid** **[override]**	access password id number of the user logged in ***"deleteuser"*** Id of the user to be deleted (Optional) true	On success of adding user: 'user deteled'. Else 'user cannot be found', 'user cannot be deleted as there is data associated. Override needed', 'deleting user failed'.

Example
http://.../leas-olm/api/masterapi?sharedsecret=******&leasid=1&method=deleteuser&userid=1**

4.5. ADD/UPDATE GROUP ("UPDATEGROUP")

Arguments	**Description**	**Returns**
sharedsecret	access password	On success of adding group: 'group created'. On success of updating group: 'group updated'. Else 'creating group failed', 'updating group failed'.
leasid	id number of the user	
method	***"updategroup"***	
groupid	id of the group	
groupname	name of the group	
position	the number in the sequence that the groups are ordered by	
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=updategroup&groupid=75&groupname=the%20api%20updated%20this&position=1		

4.6. DELETE GROUP ("DELETEGROUP")

Arguments	**Description**	**Returns**
sharedsecret	access password	On success of adding group: 'group deleted'. Else 'deleting group failed', 'group cannot be deleted as there is data associated. Override needed.'
leasid	id number of the user	
method	***"deletegroup"***	
groupid	id of the group	
[override]	(Optional) true	
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deletegroup&groupid=1		

4.7. ADD USER TO GROUP ("ADDUSERTOGROUP")

Arguments	Description	Returns
sharedsecret **leasid** **method** **userid** **groupid**	access password id number of the user logged in ***"addusertogroup"*** Id of the user to be added to the group id of the group	On success of adding user to group: 'user added to group'. Else 'adding user to group failed', 'user is already a member of the group'.
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=addusertogroup&userid=1011&groupid=75		

4.8. DELETE USER FROM GROUP ("DELETEUSERFROMGROUP")

Arguments	Description	Returns
sharedsecret **leasid** **method** **userid** **groupid** **[override]**	access password id number of the user logged in ***"deleteuserfromgroup"*** Id of the user to be deleted from the group id of the group (Optional) true	On success of deleting user from group: 'user removed from group'. Else 'user cannot be found', 'group cannot be found', 'user is not a member of this group', 'removing user from group failed'.
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deleteuserfromgroup&userid=2&groupid=1		

4.9. ADD/UPDATE SUBJECT ("UPDATESUBJECT")

Arguments	Description	Returns
sharedsecret **leasid** **method** **subjectid** **subjectname** **position**	access password id number of the user who is logged in ***"updatesubject"*** id of the subject name of the subject the number in the sequence that the subjects are ordered by	On success of adding group: 'subject created'. On success of updating group: 'subject updated'. Else 'creating subject failed', 'updating subject failed'.
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=updatesubject&subjectid=75&subjectname=the%20api%20updated%20this&position=1		

4.10. DELETE SUBJECT ("DELETESUBJECT")

Arguments	Description	Returns
sharedsecret **leasid** **method** **subjectid** **[override]**	access password id number of the user ***"deletesubject"*** id of the subject (Optional) true	On success of adding subject: 'subject deleted'. Else 'deleting subject failed', 'subject cannot be deleted as there is data associated. Override needed.'
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deletesubject&subjectid=1		

4.11. ADD SUBJECT TO GROUP ("ADDSUBJECTTOGROUP")

Arguments	Description	Returns

sharedsecret **leasid** **method** **subjectid** **groupid**	access password id number of the user logged in ***"addsubjecttogroup"*** Id of the subject to be added to the group id of the group	On success of adding subject to group: 'subject added to group'. Else 'adding subject to group failed', 'subject is already in the group'.
Example		http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=addsubjecttogroup&subjectid=1011&groupid=75

4.12. DELETE SUBJECT FROM GROUP ("DELETESUBJECTFROMGROUP")

Arguments	**Description**	**Returns**
sharedsecret **leasid** **method** **subjectid** **groupid** **[override]**	access password id number of the user logged in ***"deleteuserfromgroup"*** Id of the subject to be deleted from the group id of the group (Optional) true	On success of deleting subject from group: 'subject removed from group'. Else 'subject cannot be found', 'group cannot be found', 'subject is not associated with this group', 'removing subject from group failed'.
Example		http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deletesubjectfromgroup&subjectid=2&groupid=1

4.13. ADD/UPDATE COMPETENCY ("UPDATECOMPETENCY")

Arguments	Description	Returns
sharedsecret	access password	On success of adding competency: 'competency created'. On success of updating competency: 'competency updated'. Else 'updating competency failed', 'creating competency failed'.
leasid	id number of the user logged in	
method	***"updatecompetency"***	
competencyid	id of the competency	
competencyname	name of the competency	
position	the number in the sequence that the competencies are ordered by	
competencyparentid	the competencyid of the parent competency	
subjectid	the subject that the competency comes under	
competencyinfluence	influence of the competency (value 0 to 1)	
Example	http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=updatecompetency&competencyid=75&competencyname=the%20api%20updated%20this&position=1&competencyparentid=0&subjectid=911&competencyinfluence=5	

4.14. DELETE COMPETENCY ("DELETECOMPETENCY")

Arguments	Description	Returns
sharedsecret	access password	On success of deleting competency: 'competency deleted'. Else 'competency cannot be found', 'deleting competency failed', 'competency cannot be deleted as there is data associated. override needed', 'competency cannot has sub-competencies that are associated.
leasid	id number of the user logged in	
method	***"deletecompetency"***	
competencyid	id of the competency	
[override]	(Optional) true	

		override needed'.
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deletecompetency&competencyid=1		

4.15. ADD/UPDATE ACTIVITY ("UPDATEACTIVITY")

Arguments	**Description**	**Returns**
sharedsecret	access password	
leasid	id number of the user who is logged in	
method	***"updateactivity"***	
activityid	id of the activity	
activityname	name of the activity	
position	the number in the sequence that the activities are ordered by	
activityinfluence	for the modelling procedure. a number between 0 and 1	
subjectid	id of the subject the activity belongs to	
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=updateactivity&activityid=75&activityname=the%20api%20updated%20this&position=1&activityinfluence=5&subjectid=911		

4.16. DELETE ACTIVITY ("DELETEACTIVITY")

Arguments	**Description**	**Returns**
sharedsecret	access password	
leasid	id number of the user	
method	***"deletesubject"***	
activityid	id of the activity	

[override]	(Optional) true	
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deleteactivity&activityid=1		

4.17. ADD COMPETENCY TO ACTIVITY ("ADDCOMPETENCYTOACTIVITY")

Arguments	**Description**	**Returns**
sharedsecret **leasid** **method** **competencyid** **activityid**	access password id number of the user logged in ***"addcompetencytoactivity"*** Id of the competency id of the activity	
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=addcompetencytoactivity&competencyid=1011&activityid=75		

4.18. DELETE COMPETENCY FROM ACTIVITY ("DELETECOMPETENCYFROMACTIVITY")

Arguments	**Description**	**Returns**
sharedsecret **leasid** **method** **competencyid** **activityid** **[override]**	access password id number of the user logged in ***"deletecompetencyfromactivity"*** Id of the competency id of the activity (Optional) true	

Example	http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deletecompetencyfromactivity&competencyid=2&activityid=1

4.19. ADD/UPDATE DATA SOURCE ("UPDATEDATASOURCE")

Arguments	**Description**	**Returns**
sharedsecret **leasid** **method** **datasourceid** **datasourcename**	access password id number of the user logged in ***"updatedatasource"*** id of the datasource name of the datasource	On success of adding competency: 'datasource created'. On success of updating competency: 'datasource updated'. Else 'updating datasource failed', 'creating datasource failed'.
Example		http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=updatedatasource&datasourceid=20&datasourcename=api%20test

4.20. DELETE DATASOURCE ("DELETEDATASOURCE")

Arguments	**Description**	**Returns**
sharedsecret **leasid** **method** **datasourceid** **[override]**	access password id number of the user logged in ***"deletedatasource"*** id of the datasource (Optional) true	On success of deleting datasource: 'datasource deleted'. Else 'datasource cannot be found', 'deleting datasource failed', 'datasource cannot be deleted as there is data associated. override needed'.
Example		http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1&method=deletedatasource&datasourceid=1

4.21. ADD DATA AND COMPETENCY INFORMATION ("ADDINFORMATION")

Arguments	Description	Returns
sharedsecret **leasid** **method** **competencyid** **groupid** **userid** **datasourceid** **value** **activityid** **[timestamp]**	access password id number of the user logged in ***"addinformation"*** id of competency id of the group id of user to be updated id of the datasource inference value (range: 0 to 1) id of the activity time of evidence added (optional)	On success of adding information: 'information added: "<time of addition>" value:"<value>"'. Else 'competency does not exist in the database', 'group does not exist in the database', 'user does not exist in the database', 'datasource does not exist in the database', 'user is not a member of the group', 'user is a teacher', 'value is not a number', 'value should be in a range of 0 to 1', 'adding information failed'.
Example http://.../leas-olm/api/masterapi?sharedsecret=********&leasid=1011&method=addinformation&competencyid=198&groupid=72&userid=1001&datasourceid=20&value=0.756&activityid=911		

5. APPENDIX – USER MANUAL

5.1. INTERFACE STRUCTURE AND COMPONENTS

The interface constitutes one primary webpage, which acts as a browser for the open representation of the learner model (Figure 25). This browser is then embedded within the LEA's BOX portal. It is the same for both teachers and students, with the exception that students see only their own data, whilst teachers can see data for all students with whom they share a group. For the ease of showing the general layout of the screen Figure 25 to Figure 38 use test data to show where information will appear on the screen. Visualisations are covered in Section 0.

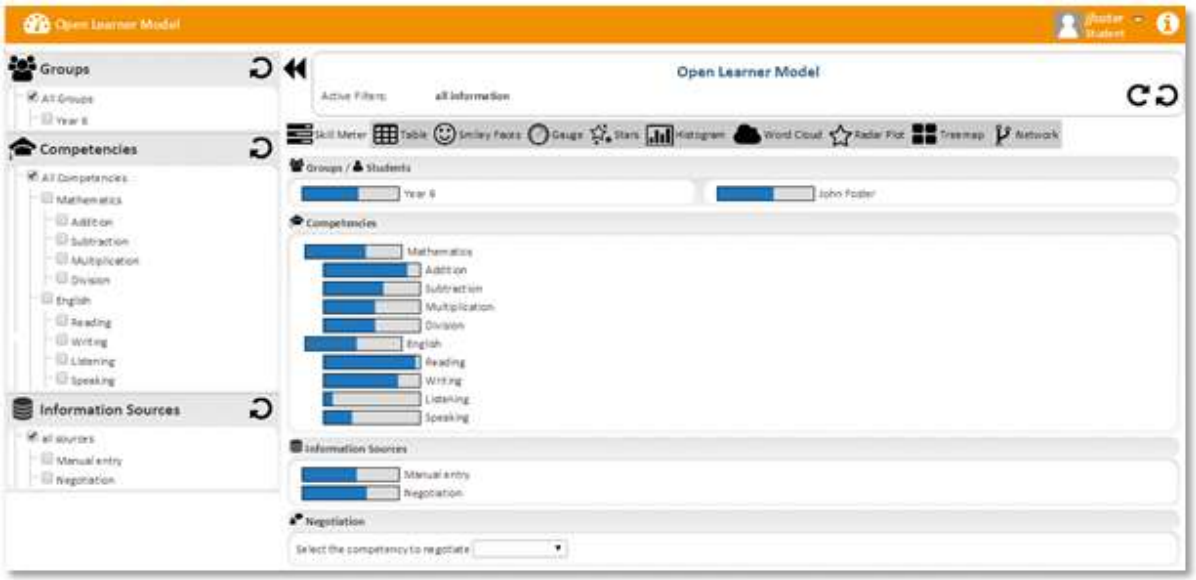


Figure 25: OLM browser interface.

The following key facilities are included:

- **Information filters** (left of Figure 25). These allow criteria to be specified to narrow down the scope of the information presented in the visualisations. These may be added in any combination or permutation. Specific groups, competencies or information sources may be specified. For the case of teachers, individual students may also be specified. Visualisations are automatically updated when criteria are amended. The filters may be hidden to allow more space for the visualisations.

- **Open learner model visualisations** (centre and right of Figure 25). Different visual methods are used to display the same underlying learner model information. These may be switched between using the tab structure. Each set of visualisations is broken down into a learner model opened from the perspective of groups, (students,) competencies and information sources. Each of the sections are collapsible, to allow greater space for individual visualisations. The visualisations are rendered by posting the relevant modelled dataset to the visualisation service and displaying the returned graphic or HTML content.

- **Breadcrumb and functions** (top centre and right of Figure 25). The filters currently applied, and the nature of the information in the visualisations, are described using a breadcrumb to show where the user currently is within the 'browser'. To the right hand side of this section there are also refresh and reset functions. Additionally affordances such as the loading symbol will appear here whilst the visualisations are loading or updating.

- **Customisation functions** (very top right of Figure 25). The menu which is headed by the user's username allows the browser to be customised. The language may be localised to English, French, German, Czech, Turkish or Norwegian. The visualisations that are displayed in the browser may also be turned on and off using the preferences page.

- **Help** ('i' icon, very top right of Figure 25). This will display basic guidance on how to operate the browser.

- **Negotiation** (bottom of Figure 25). This facility, described in section, is only available for students.

5.2. VISUAL METHODS

The OLM set of visualisations consists of twelve visualisations that are graphical and textual, some which show structure, some which are interactive, and some that quantise the data, whilst others use a continuous scale (Table 9).

Table 9: LEA's Box OLM visualisation set.

Visualisation	Graphical	Textual	Quantised scale	Continuous scale	Structure	Unstructured	Interactive
Skill Meter	✓			✓	✓		
Table		✓	✓		✓		
Radar Plot	✓			✓		✓	
Network	✓		✓		✓		✓
Across time	✓			✓	✓		

Heatmap	✓			✓		✓	✓
Level of Activity	✓		✓	✓		✓	

The visualisations coloured on Table 9 have been added since the last release. Since the last release, four visualisations have been added: stars, gauge, across time and heatmap. The two first are very simple but answer to a need of customisation. The stars have characteristics equivalent to smiley faces, but are less “children connoted”. The gauge have characteristics equivalent to skill meters that is one of the most popular visualisation, but with a very different design. The ‘across time’ visualisation answers to a need to represent the model evolution across time. It is not possible with the other LEA’s Box OLM visualisations even if this kind of visualisation is frequent in OLM. The heatmap visualisation is a multidimensional visualisation, it answers to a need to represent on a same graphic two kinds of information. For instance a heatmap can represent on a same graphic the data coming each information sources for each competency.

SKILL METERS

Student competency is represented using a bar with a continuous scale. The proportion of colour is analogous to the extent to which the student is competent in the area. Indentation is used to show hierarchical structure.

TABLE

Each element is a separate line in the table and hierarchical structure is shown using indentation. Competency is quantised into five categories, ranging from *very weak* to *very strong,* with a dot being placed in the appropriate table column to indicate this.

RADAR PLOT

Each axis displays a competency or data item. The further away from the centre the data point is, the greater the competency. Again, the structure of the information is not shown, however items are ordered clockwise.

NETWORK

The network visualisation shows competency through the size of the nodes on the network. Nodes are quantised into 5 different sizes and shades of green; the larger the node, the greater the competence. Structure is shown by arcs between the nodes. The visualisation is a force-directed network and the nodes may be moved, and sub-nodes collapsed to increase readability. This visualisation is interactive.

ACROSS TIME

The across time visualisation presents an area graph for each item to be visualised in the OLM. This is the state of the model across time. Competency is shown on the y axis, and time on the x axis. All scales are the same between graphs, and graphs are shown in alignment for ease of comparison.

HEATMAP

The heatmap visualisation allows any two information types within the OLM to be compared. Select a data type for the x axis and y axis and the heatmap matrix will be displayed. This visualisation is able to display more data at once than the others, and allows different relationships to be compared. For example, in Figure 38, the open learner model shows the different levels of competency for information coming from each datasource. The intensity of the (red) pigmentation shows the extent of competency.