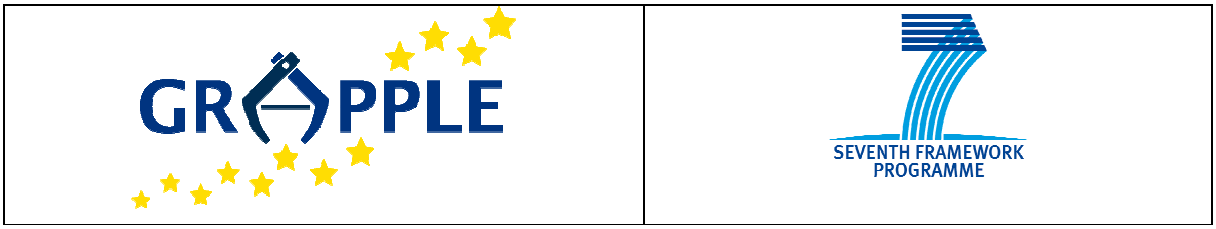


GRAPPLE

D4.5b Version: 1.0

Implementation of interactive visualization of models and students data

Document Type	Deliverable
Editor(s):	Riccardo Mazza (USI), Luca Mazzola (USI)
Author(s):	Riccardo Mazza (USI), Luca Mazzola (USI), Christian Glahn (OUNL), Alexander Nussbaumer (UniGraz), Dominique Verpoorten (OUNL)
Reviewer(s):	Noaa Barak (ATOS), Maurice Hendrix (Warwick).
Work Package:	WP4
Due Date:	31 Oct. 2009
Version:	1.0
Version Date:	01-02-2010
Total number of pages:	31

Abstract: This document reports the implementation design of the interactive visualizations of open learner models that will be performed in GRAPPLE. The visualizations will be based on data stored in the domain model, conceptual adaptation model, and user model, and aim at supporting learners to be more engaged in the learning process, and instructors in assisting the learners.

Keyword list: open learner models, model representation, interactivity, privacy, smart indicators, learning interaction cycle,

Summary

The GRAPPLE Project (Generic Responsive Adaptive Personalized Learning Environment), aims at delivering to learners a technology-enhanced learning (TEL) environment that guides them through a learning experience, automatically adapting to personal preferences, prior knowledge, skills and competences, learning goals and the personal or social context in which the learning takes place. The system will include a user model infrastructure that keeps track of the learner's knowledge and skills acquired during the learning process. This knowledge will be made available to the learners and instructors by means of interactive visualizations that will be performed in GRAPPLE. The visualizations will consider the learner model, the domain model, and the adaptation model, and aim at helping instructors in assisting the learners. The deliverable also discusses how visualizations may support learners to be more engaged in the learning process.

This deliverable reports the outcomes of the second phase of the Task 4.5 of the GRAPPLE project. In the first phase we analyzed: of the state of the art in the field of user modelling, the requirements analysis undertaken with potential users of the systems through a series of interviews and meetings, an initial description of user model (UM) and domain model (DM) components (which provide the input data for the visualizations) some possible scenario of usage to guide the following phases of the work, and the technical infrastructure. In this phase of the project, reported in this document, we describe the list of data that we extracts from other GRAPPLE components, the aggregations we make with this data, the visualizations that we will implement, and the architecture of the software.

Authors

Person	Email	Partner code
Riccardo Mazza	mazzar@usi.ch	USI
Luca Mazzola	mazzolal@usi.ch	USI
Christian Glahn	Christian.Glahn@ou.nl	OUNL
Alexander Nussbaumer	alexander.nussbaumer@uni-graz.at	UniGraz
Dominique Verpoorten	Dominique.Verpoorten@ou.nl	OUNL

Table of Contents

Tables and Figures

List of Figures

List of Tables

List of Acronyms and Abbreviations

GALE	GRAPPLE Learning Environment
GAT	GRAPPLE Authoring tool
GEB	GRAPPLE Event Bus
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
GUMF	GRAPPLE User model framework
GVIS	Grapple Visualization Module
LMS	Learning Management System

1 Introduction

In GRAPPLE we want to support learners with the acquisition of meta-cognitive skills, and the instructors with a tool to monitor the current status of knowledge of their students. One of the main objectives of WP4 is hence to explore the usage of open learner models in GRAPPLE. In order to "open" model to learners, peers, and instructors, we have implemented a module that we called **GVIS** (Grapple VISualizations), which make use of graphical representations of user data distributed in different GRAPPLE modules. These representations will provide the learner's individual progress over the domain of knowledge, showing the learner's state in relation with concepts, course's goals, and peers. Research on open student models showed that the visualization of selected parts of learner models supports meta-cognition and helps to raise awareness on collaborative learning processes in informal learning processes. In GRAPPLE we want to explore this approach with the GVIS module. Deliverable D4.5a describes a preliminary study that includes an analysis of the state of the art in the field of user modelling, the requirements analysis undertaken with potential users of the systems through a series of interviews and meetings, an initial description of data sources, and some ideas for graphical representation of data.

This deliverable, as follow up of Deliverable D4.5a, starts from these considerations, and provides some details about the GVIS module. We start describing the **data providers,** which are other GRAPPLE modules that represents source data for GVIS (section 2). The **source data** available for visualization is described in section 3. Then we will provide a detailed list of the **information** we want to provide to the learners and instructors with GVIS (section 4). Details on the software implementation are provided in section 5. Finally, some mock-ups are described in section 6.

2 Data providers

The data providers are the places in the GRAPPLE infrastructure where the input data for GVIS are located. We identified the following data providers:

Table 1: Data providers for GIVIS

Data provider	Description	Interface to GVIS
GAT	The GAT (GRAPPLE Authoring Tool-set) is the authoring component of a GRAPPLE course. It is composed by three elements, DM (Domain Model), CRT (Concept Relationship Type) and CAM (Conceptual Adaptation Model). Some data from these elements, defined by authors/instructors by means of these components, are needed by visualizations (such as the list of concepts of a course, or the expected knowledge level for concepts). The GAT has an internal repository that contains all the data structures. Data from this repository can be extracted by means of specific webservice interface.	GVIS fetch GAT data through a webservice interface provided by the CAM component.
GUMF	Contains all user-related data. It is the main source of data for GVIS. In particular, GUMF will contain the data generated by • GALE, such as whether a concept is currently suitable for a user or not, the level of knowledge of learner over the concepts, • LMSs, such as the role of the user in a course, the list of students, the list of activities GALE and LMS store their user data into GUMF through the Grapple Event Bus (GEB). GVIS connects to GUMF	GRAPPLE Event Bus.

GVIS is connected to GAT and GUMF by mean of specific interfaces and Event Bus. In Figure 1 is depicted the schema of communication interfaces between GVIS and other data providers.

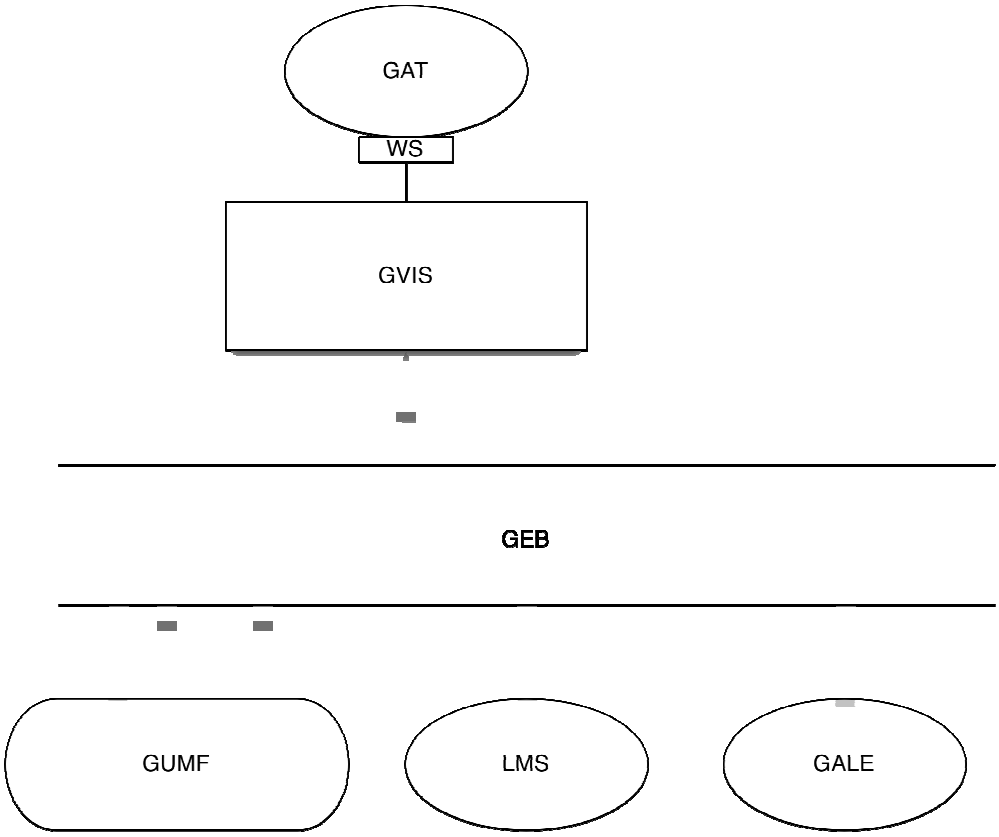


Figure 1: Schema of communication interfaces between GVIS and data providers

3 Source Data

Visualizations require well-defined information about the data type of the data to visualize. This is for three reasons: firstly, the data type can influence what visualization can be used; secondly, the data type determines what data can get combined in one visualization; and finally, the information types partially reflect the meaning of the provided information.

We identified the following data types:

- **Text** - textual information represented as a sequence of characters
- **Number** - numerical information. Can be integer or float
- **List** - ordered sequence of data of type Text or Number. List elements are identified by an integer. It could also be a list of records. A record is a sequence of Text or Numbers.

This table reports the list of data needed by the visualizations. All these data is provided to GVIS by one of the data providers identified in the previous table.

Table 2: list of source data for GVIS

Name	***Description***	***Data Type***	***Data source[1]***	***Comments.***
ConList	List of all concepts of a course.	List	GAT (CAM)	
ConSuit	Whether a particular concept is suitable for a user or not	Boolean	GUMF (GALE)	This information is inferred by GALE, and then stored into GUMF.
ConRel	List of existing domain relationships between two concepts	List	GAT	For future use. This source data is currently unused in information descriptors
StuList	List of student enrolled in a course	List	GUMF (LMS)	
StuAccess	The number of times a user access a course	Integer	GUMF (LMS)	For future use. This source data is currently unused in information descriptors
ConVis	The number of times that a user visited a concept in GALE	Integer	GUMF (GALE)	
UsrRole	Provides information about the role of the user in a course. (teacher, student, tutor, …)	Text	GUMF (LMS)	
KnoLevel	Knowledge level of the student in a particular concept.	Number	GUMF (GALE)	
KnoHist	Knowledge level of the student in a particular concept at a particular time in the past	Number	GUMF	
KnoLevExp	The expected knowledge level of the student in a particular concept.	Number	GAT (CAM)	The expected knowledge level is defined by the instructor and corresponds to the minimum knowledge level that the learner has to acquire to have a sufficient knowledge of a concept for finishing a course. A default will be assumed and the author will change the defaults with CRTs if he so wishes.
KnoRange	Range of values (maximum and minimum values) for knowledge level.	Numbers	GUMF	We expect that the range of values for knowledge level is [0.0, 1.0]. However, if the range is different, the GUMF have to provide us with these values.

[1] The GRAPPLE module where this data is generated is indicated between parentheses.

GoalList	List of all goals defined in a course.	List	GAT (CAM)	A goal represents a particular state reached by the learner during the learning process. For instance, it could indicate that a learner reached a milestone, or acquired enough knowledge of the topics of the course. Each goal corresponds to a particular CRT. We can have several goals. Goals are set by course authors in the CAM editor. This goal list is by default the first thing presented to the student when entering the course. If he does not want to have it displayed anymore, he ticks the option. Default implicit goal: reach all expected knowledge levels for all concepts. Additional goals can be set via CRTs.
GoalRea	Whether a user reached one particular goal	Boolean	GUMF (GALE)	A goal is defined with a particular User Model variable. This variable defined in the CAM and calculated by GALE.
ActList	List of learning activities of a course. A learning activity is *an action in a system or an environment that are related to learning. Currently, only LMSs actions are tracked into the GUMF.*	List	GUMF (LMS)	
ActVis	Number of times that a user visited a particular activity (from ActList)	Integer	GUMF (LMS)	

4 Information descriptors

This section contains some details about the information that GVIS will provide to learners and instructors. The following information has been compiled taking into consideration the source data identified in the previous section, and the user requirements and the analysis defined in the deliverable D 4.5a. The overall goal of the following information is to support learners with the acquisition of meta-cognitive skills, and the instructors to monitor the status of their students.

For each information item the following is indicated:

- A short name for the data (column title: Name)
- The name and a synopsis of the meaning of the data (column title: Information)
- Target users (column title: Main target users)
- Description of the required source data (column title: Required source data)

- The aggregations (or calculus) needed to derive this information (column title: Aggregators)
- Possible interaction of the users with the visualizations that represent that information (column title: User interactions).

Two or more information items described in the following table can be combined into a single graphical representation.

Moreover, we use the following colour coding for describing whether the information represents facts about the learner or about the class:

- Violet: information about learner and their peers
- Orange: information about the class
- Neutral: if the information as about both the learner and the class.

Two types of visualizations will be generated by GVIS: a **compact indicator** that can be placed as side block on the main page of the course or a **detailed visualization**, placed on a special **dashboard** that can be accessed by users of the system by clicking on the compact visualizations.

4.1 Descriptors for compact indicators

Information described in the following table will be implemented as "compact indicators" that the users will see as side block on the main page of the course in the Learning Management System. A compact indicator can implement one or more of the items described in the table (see section 6 for details).

Table 3: Descriptors for compact indicators.

Name	Information	Main target users (in order of relevance)	Required source data	Description of the aggregations
ConVisStud	Number of concepts visited by a learner	Learner	ConList, ConVis	Counts the number of concepts of the course (ConList) visited at least one time by the learner (ConVis)
ConVisClass	Number of concepts visited by the class	Instructor Learner	StuList, ConList, ConVis	The average value of concept visited (ConList, ConVis) per learner (StuList).
KnowStud	Overall knowledge acquired by a learner	Learner	ConList, KnoLev, KnoRange	The average of knowledge level (KnoLev) of the learner per concept of the course (ConList).
KnowClass	Overall knowledge acquired by the class	Instructor Learner	StuList, ConList, KnoLev, KnoRange	The average of the overall knowledge level (KnoLev, ConList) per student of the course (StuList)

ActStud	Number of activities performed by a learner	Learner	ActList, ActVis	Counts the number of activities of the course (ActList) performed at least one time by the learner (ActVis)
ActClass	Number of activities performed by the class	Instructor Learner	StuList, ActList, ActVis	The average of number of activities performed (ActList, ActVis) per learner (StuList).

4.2 Descriptors for Dashboard's views

Information described in the following table will be implemented with a special dashboard that can be accessed by users of the system by clicking on one of the compact visualizations described in the previous section. The dashboard will contain the widgets that will implement the items described in the table (see section 6 for details).

Table 4: Descriptors for dashboard's views.

Name	Information	Main target users (in order of relevance)	Required source data	Description of the aggregations	User interactions
GoalStudNum	Number of learners that have achieved a particular learning goal	Instructor	StuList, GoalRea	Counts the number of learners (StuList) that have achieved a learning goal (GoalRea)	
KnowConcStud	Knowledge level of the learner for every concept of the course	Learner Instructor Tutor	ConList, KnowLev, KnoRange	Given a learner, it provides, for each concept of the course (ConList), the current level of knowledge acquired by the learner (KnoLev)	Filtering on concepts Sorting by knowledge

KnowConcClass	Knowledge level of the class for every concept of the course	Instructor Tutor Learner	StuList, ConList, KnoLev, KnoRange	Provides the average of knowledge level (KnoLev) per student (StuList) for any concept of the course (ConList),	Filtering on learners, goals Sorting by knowledge
KnowConcExp	Expected knowledge level for every concept of the course	Learner Tutor Instructor	ConList, KnoLevExp	Provides information about the expected knowledge level (KnoLevExp) for every concept of the course (ConList).	Filtering on concepts
KnowConcStudHist	Knowledge level of the learner in the past for every concept of the course	Learner Tutor Instructor	ConList, KnoHist, KnoRange	Provides, for each concept of the domain (ConList), the level of knowledge acquired by the learner at a particular time in the past (KnoHist)	Manipulation of date (eg. through a slider)
KnowConcClassHist	Knowledge level of the class in the past for every concept of the course	Instructor Tutor Learner	StuList, ConList, KnoHist, KnoRange	Provides, for each concept of the domain (ConList), the average of the level of knowledge per student (StuList) at a particular time in the past (KnoHist)	Manipulation of date (eg. through a slider)
SuitConcStud	Indication about which concepts are currently suitable for the learner	Learner Tutor	ConList, ConSuit	Indicates which concepts (ConList) are currently suitable (ConSuit) for the learner	Filtering on concepts
SuitActStud	Indication about which activities are currently suitable for the learner	Learner Tutor	ActList, ActSuit	Indicates which activities (ActList) are currently suitable (ActSuit) for the learner	Filtering on activity (activity type?)
ActVisStud	Indicates which suitable activities have been visited by a student		ActList, ActVis.	Indicates which activities (ActList) have been visited (ActVis) by the learner	Filtering on activity (activity type?)

GoalStud	Indication of achievement for each learning goal.	Learner Tutor Instructor	GoalList, GoalRea	Indicates which learning goals (GoalList) are currently achieved (GoalRea) by the learner	Filtering on goal.

5 Software implementation

The GVIS module is a three-tier software architecture that allows a great flexibility, customization, and independence on the source data that comes from other grapple components. It has been designed to provide the following features:

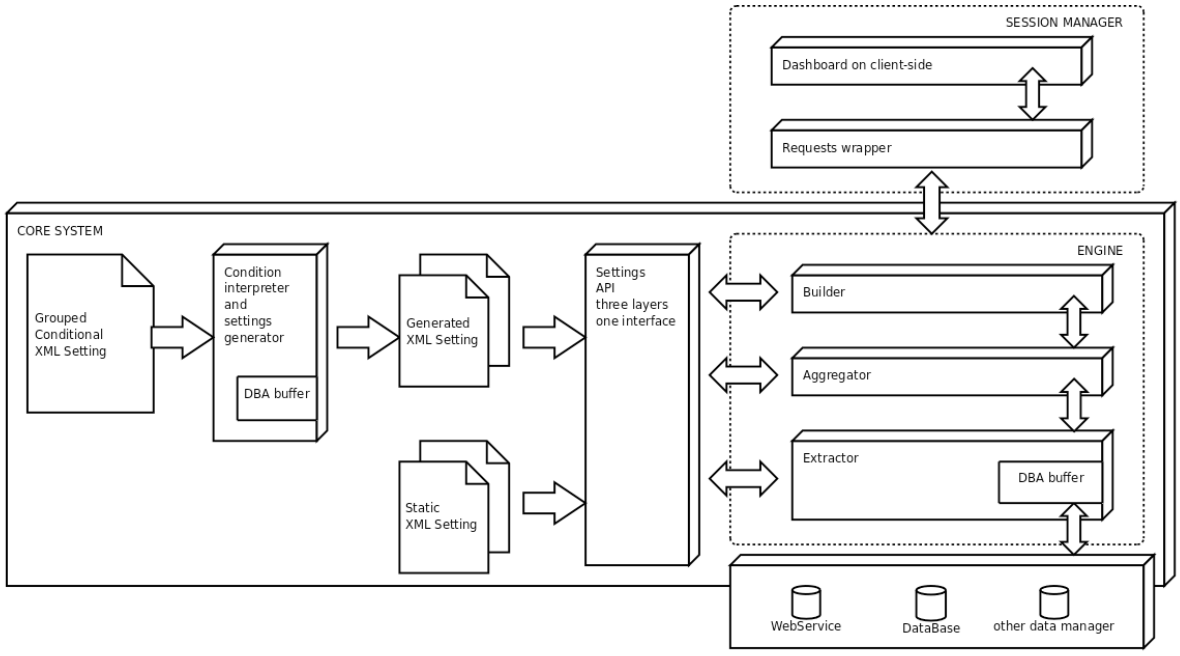
- Possibility to extract data from different data sources and with different methods (web service, database queries, others)
- Possibility to change or extend the set of source data without the need of changing the software code.
- Possibility to define, alter, or extend the data aggregations without the need of writing (or changing) the software
- Possibility to include adaptation on the graphical representations, by the inclusion of programmable conditions in the configuration files.

As future development, we plan to introduce also the following features:

- Possibility to filtering input data and rearrange data on widgets
- Possibility to dynamically create new widgets based on templates
- An editor for the configuration files.

We expect to report this progress in a next deliverable.

5.1 System Architecture



The architecture is composed by tree independent layers: the extractor, the aggregator and the visualizations:

5.1.1 Extractor

The **extractor** is in charge of retrieving source data from the data providers (CAM and GUMF). The extractor has other sub-modules, one for each data provider, in charge of fetching data from a particular provider and formatting it according to the internal format used.

The extractor loads its configurations from an XML file. In this file, the administrator of GVIS can define the list of source data (see table in section 3), where this data comes from (the data provider, described in section 2), the commands to retrieve this data (it could be a SQL/SPARQL statement, a GRAPPLE statement, etc), and the datatype (*numeric, string, list*).

5.1.2 Aggregator

The **aggregator** is in charge of applying some operations to create aggregated information. Aggregations are simple mathematical operations over the source data fetched by the extractor. By aggregations we create the information described is section 4.

At the moment, we have implemented the following aggregating functions:

1. **count**: counts the number of records in case of datatype *list*
2. **average:** calculates the average value of the arguments (they must be *list*). In case of more lists as arguments, it calculates the average of the elements of each list, and then the average of these results.
3. **sum:** sum of values of elements in a list(s)
4. **intersection:** receives two lists as arguments, and creates a new list that contains only the elements which exist in both input lists.
5. **union:** merges two lists in a new one.
6. **differ:** receives two lists as arguments, and purges from the first lists the elements that exist in the second list.
7. **each:** receives a list as argument, and creates a new list by extracting new data from data providers, using the values of the list as arguments for the new data extraction.

8 **OrderByField:** order a list of records in descending order by a means of one of the field

9 **ExtractCol**: extract a single column from a list of records, returning a list

10 **ExtractRow:** extract a single records from a list of records

11 **LimitByValue**: prune the list of records limiting it to entries that has the value of a field higher than a threshold

12 **LimitByNumber**: prune the list of records limiting it to the first "Number" entities

Aggregations are not limited to those listed is section 4: they can be extended with new data and operations, based on the model that the teacher or instructional designer want to provide to the learners. The aggregator loads its configurations from an XML file. In this file the administrator of GVIS can define new information data by the application of these aggregating functions on the source data. For instance, the information "Number of concepts visited by a learner" can be derived in the aggregator by the application of "count" function on the list of concepts visited by the learner. Aggregations are described in details in the column with title "Description of the aggregations" of tables in section 4.

5.1.3 Builder

The **builder** is in charge of creating the customized widgets for the dashboards, based on widgets templates. Both the dashboard and widgets are implemented in Flash technologies. Widgets are located in the dashboard, which runs in client computers and communicates data with the builder through JSON data interchange format.

The builder has a specific XML configuration file as well. In this file the GVIS administrator can decide how a specific data (derived by the aggregator, or fetched by the extractor) is mapped onto a particular graphical element of a widget.

5.2 Adaptive visual representation

All the three layers that compose the core system of GVIS have specific XML configuration files. These files may contain conditional parts, which allow the GVIS visualization to have a different behaviour with different source data values. This allows us to have visual representations adapted to the data that comes from the different information providers (GALE, GUMF, CAM).

For instance, we can decide that a particular widget may show a comparison of the knowledge level of a student with the class, ONLY if his current knowledge level is greater than 30%. Or we may want to show a particular widget only to the instructor of the course, not to the students.

This is implemented by including conditional instructions in the XML configurations files of the extractor, aggregator, and builder. To this end, the XML configuration files may contain variables, logical and arithmetical operators. We have implemented the following operators:

- == >(in the form of >) <(in the form of >) !=
- AND, OR, XOR, NOT

Here there is an example:

<op> contains the expression to be evaluated

<operands> contains the definition of variables contained in the expression

<true> branch executed in case the value of the expression is true

<false> branch executed in case the value of the expression is false

```
...
<cond>
        <op>(!(abcd) AND (var2 == 3) OR !(var3))</op>
        <operands>
                <val id="abcd">Concept.list</val>
                <val id="var2">Concept.knowledge</val>
                <val id="var3">Student.count</val>
        </operands>
        <true>
                ...
        </true>
        <false>
        ...

        </false>
</cond>
...
```

5.3 Filters

It will be possible to filter the data that comes from data sources. In particular, the dashboard will have a specific filtering area where the user can filter some data elements used in the widgets. This feature is still under development, and will be reported in next deliverable (D4.5c).

5.4 Configuration editor

It will be possible to edit the configuration files for the 3 levels of the GVIS infrastructure. This feature is still under development and will be reported in next deliverable (D4.5c).

6 Visualizations

GVIS is a dynamic, adaptive, and highly configurable infrastructure that aims at visually representing user data tracked and inferred by GALE and LMS, and stored in the GUMF. Currently, GALE, GUMF, and GEB are still in the development phase, and the GVIS module unfortunately cannot access its source data. We have created some mock-up of the visualizations that we have planned to implement for GVIS. Once the source data is available from data providers, we will implement the widgets based on the mock-ups defined in this section.

The visualization can be divided into 3 groups, according to the type of information they provide:

- Knowledge
- Activities
- Goals

The GVIS module supports two kinds of widgets: **compact indicators**, which are small widgets integrated into the LMS course web interface, and widgets to be placed into a **dashboard**, that opens a new window in the user interface.

6.1 Knowledge

The knowledge level on concepts of the course is the most important information that has to be presented by GVIS to the users. It can be presented to the learner, but also to teacher and to tutors. The overall knowledge acquired by a learner could be represented by the average of knowledge level of the learner per concept of the course.

6.1.1 Views

The first type of view is the overall knowledge level. The information that will be represented by the following views is referred in the information descriptors as:

- **KnowStud**: The overall knowledge acquired by a learner
- **KnowClass**: The overall knowledge acquired by the class

The following pictures present this value in two forms: a textual and a graphical form (both available for integration in the LMS interface):

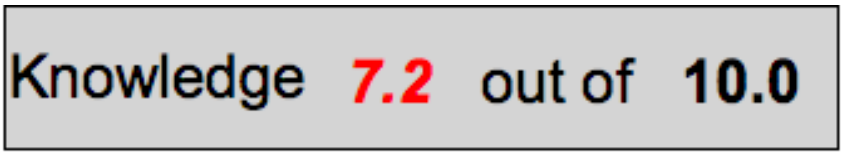


Figure 2: The overall knowledge level in textual view

In the graphical view we also represent (in yellow) the overall knowledge acquired by the class, where the green bar represents the learner's knowledge. On the teacher/tutor view, only the yellow bar is represented. The end point of the line represents the upper bound of the range of the knowledge level.

Figure 3: The overall knowledge level in graphical view (learners' version)

On click, a more detailed view will be provided.

More detailed views are in charge of representing the knowledge level for every concept in a course. To this end, a dashboard will be provided to users of GRAPPLE that will contain more complex visualizations. The dashboard is activated when the user clicks on one of the compact indicators.

In particular, we want to represent the following information descriptor:

- **KnowConcStud**: Knowledge level of the learner for every concept of the course
- **KnowConcClass**: Knowledge level of the class for every concept of the course
- **KnowConcExp**: Expected knowledge level (target) for every concept of the course
- **KnowStud**: The overall knowledge acquired by a learner

The first visualization is a bar chart (see Figure 4), each vertical bar represent every concept of the domain, the knowledge acquired by the student is encoded into the green bar, the knowledge acquired by the class is the yellow bar. If the expected knowledge is available then it is represented with a horizontal line. The dashed line represents the maximum knowledge level achievable. This view will be available to learners, tutor and teachers.

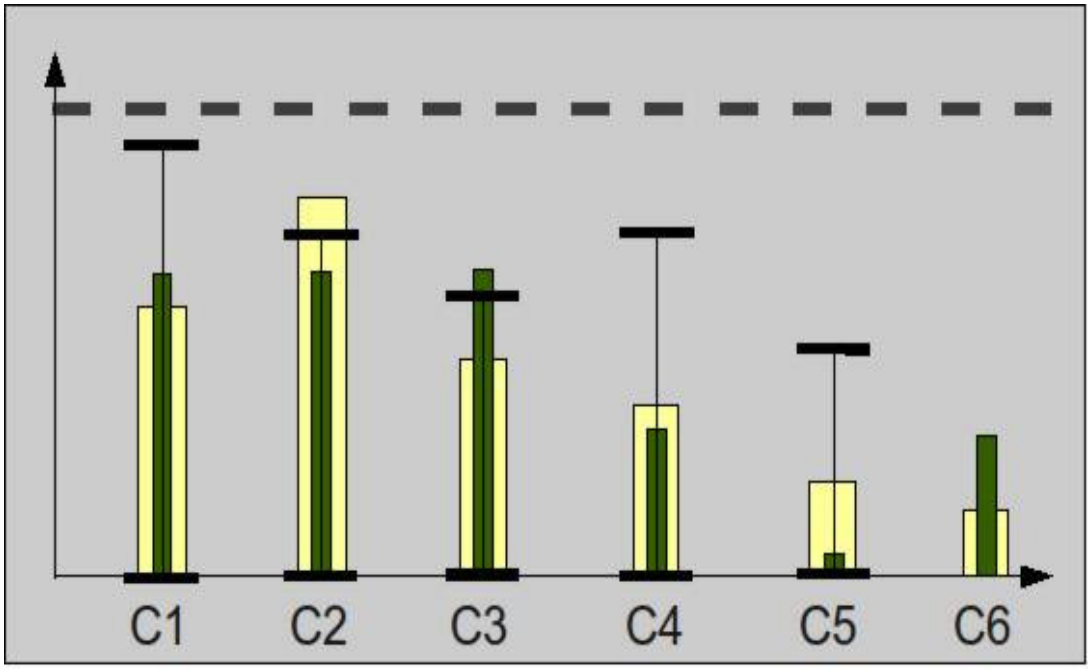


Figure 4: Knowledge level for a learner and class, and expected knowledge level, in expanded form

If historical data is available in the GUMF, it is interesting to compare the status of the knowledge of a student in the past with the current knowledge. The next visualization (Figure 5) is intended to compare the knowledge level of a student and the class at a particular time in the past (bar graph above) with the current status (bar graph below).

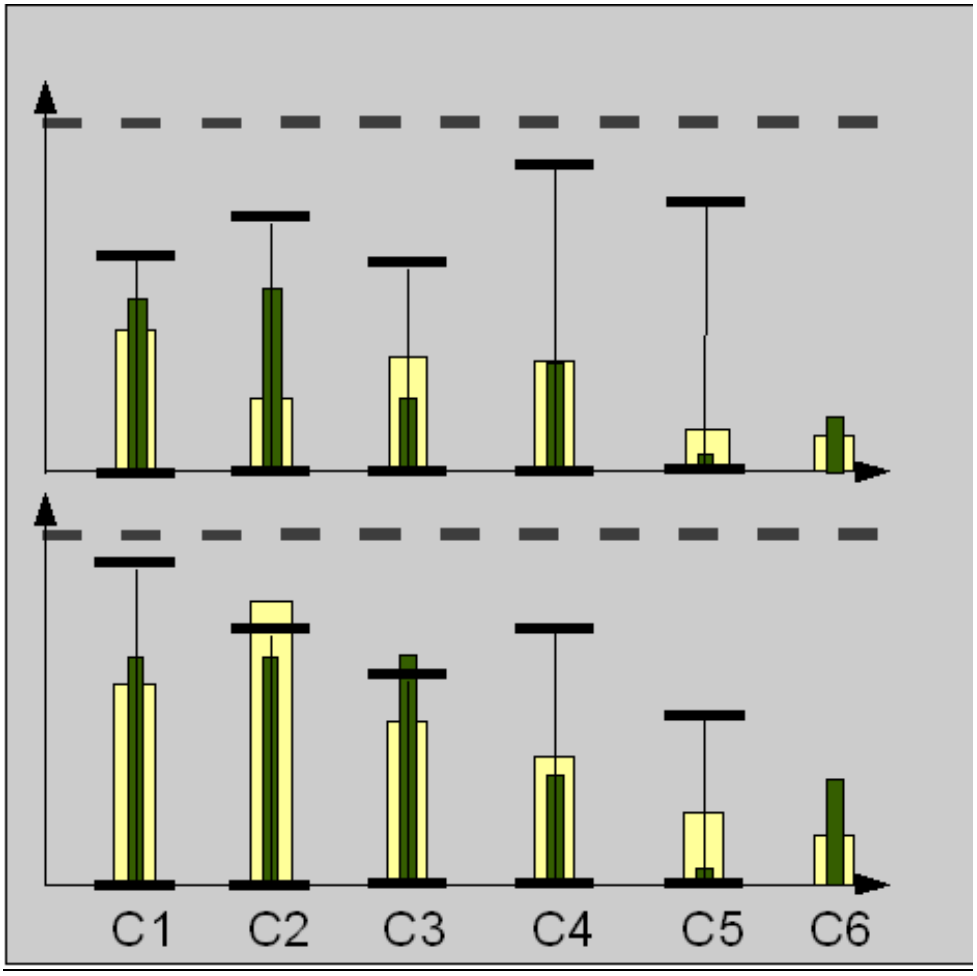


Figure 5: Knowledge level for a learner and class (with expected knowledge level) with history: the situation at a previous moment and, in the lower part, the current one

A different view will be provided to tutors, in order to allow a more specific comparison of concepts and learners. The visualization is based on a matrix, where learners are mapped into columns, and concepts into rows (see Figure 6). The level of knowledge of a student in a concept is mapped into the dimension (in particular, the radius) of each cell of the matrix. The colour of the cell represents whether the student reached (in green) or not (in red) the target knowledge level.

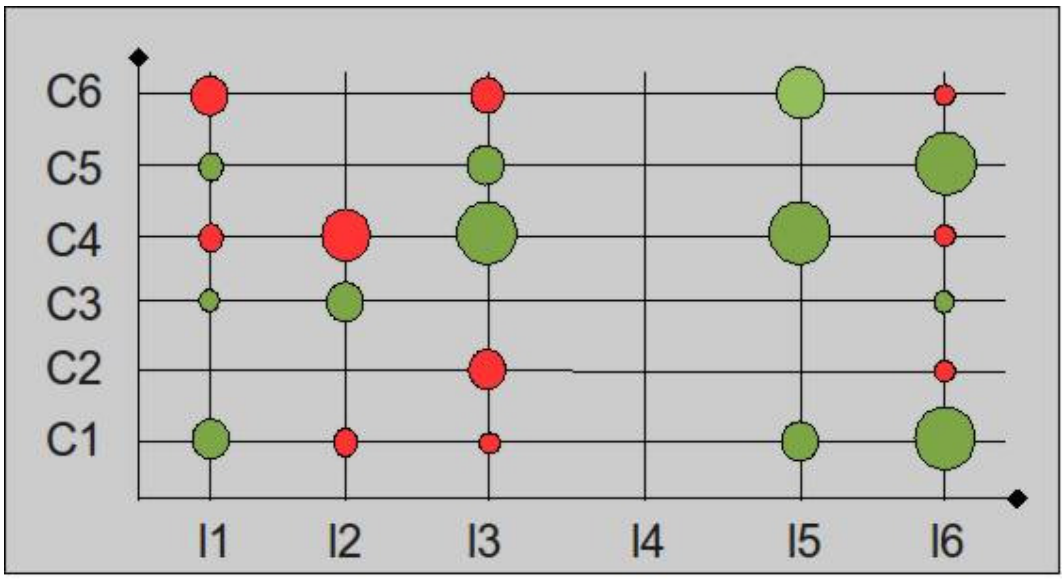


Figure 6: Knowledge level representation as matrix (learners - concepts)

In another visualization (see Figure 7), the bar graph near the concepts name gives an indication of overall knowledge of the class on that concept (red bar). The shape of the cell represents whether the student reached (square) or not (circle) the target knowledge level. The green bar next to the learner names indicates the overall knowledge acquired by the learner (KnowStud). The arrows can be used to interact with the view to sort the columns or the rows in ascendant or descendent order, according to the knowledge level of student for columns, or the overall knowledge of the class for rows.

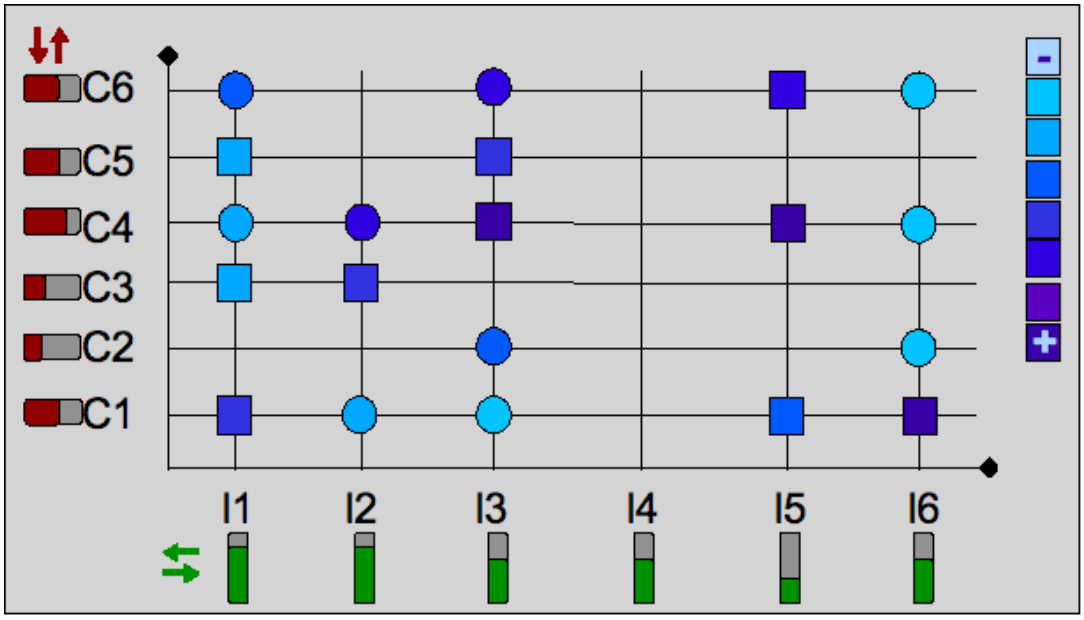


Figure 7: Knowledge level representation as matrix (learners - concepts)

6.2 Activities

6.2.1 Learner view

Learners will be provided with the information about the activities visited. A look-and-feel similar to the previous visualizations will be adopted. The information that will be represented by the following views is:

- **ActStud**: Number of activities performed by a learner
- **ActClass**: Number of activities performed by the class
- **ActList**: Number of activities available in the course

Figure 8: Number of activities visited in textual form

The following representation includes also the average value of activities visited by the students of the class (in yellow). These last two widgets will be available for integration in the LMS interface.

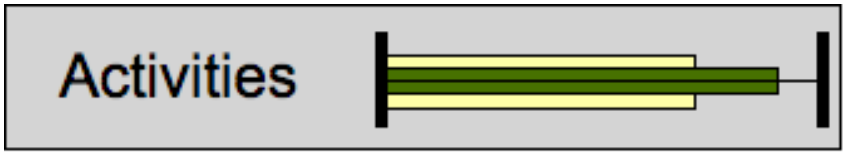


Figure 9: Number of activities visited in graphic form, green bar represents the student data, yellow bar represents the class data.

The learner, through the following representation, can explore details about each single resource.

The horizontal bar next to the activity name represent (in red) the percentage of learners that have accessed that activity, the vertical bars represent two types of information: in blue the percentage of the activities accessed by the learner, in green percentage of the activities accessed by the class. The learner can compare his status with that of the class. The bullet indicates the fact that the current learner has visited the resource.

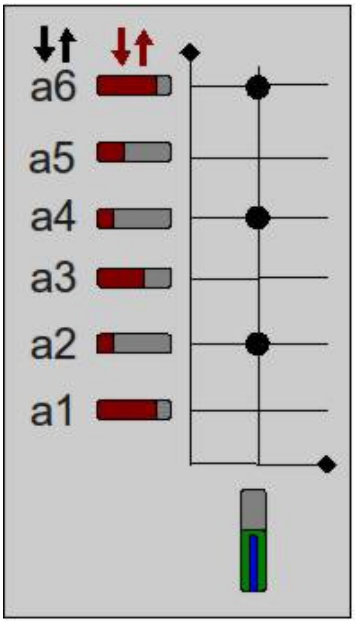


Figure 10: Analytic view of resource viewed by a learner

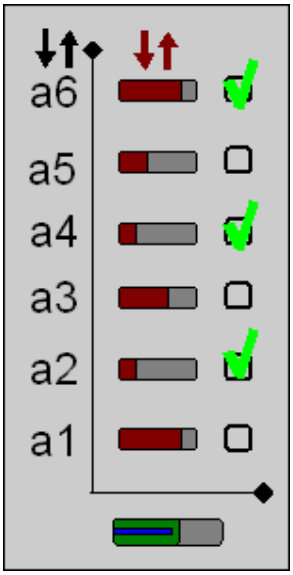


Figure 11: Analytic view of resource viewed by a learner – other possibility

Another way to encode the number of views for every activity is by the radius of the bullets or adding a new textual description.

6.2.2 Teacher/Tutor view

Teachers are supported by a view (Figure 12), based on a matrix, which represents the learners (or a selected subset) in columns and the resources in rows: the filled circles indicate which resources a learner accessed. Another possible representation could be the use of numbers to describe the times a learner accessed a particular resource, instead of using a binary variable like the circle presence/absence. The horizontal bars near the activities name encode the percentage of activities accessed by any student, while the vertical bars below the student name indicates the percentage of activities accessed by the student. The graph can be reorganized in ascendant or descendent order on the student names, activities, horizontal bars and vertical bars.

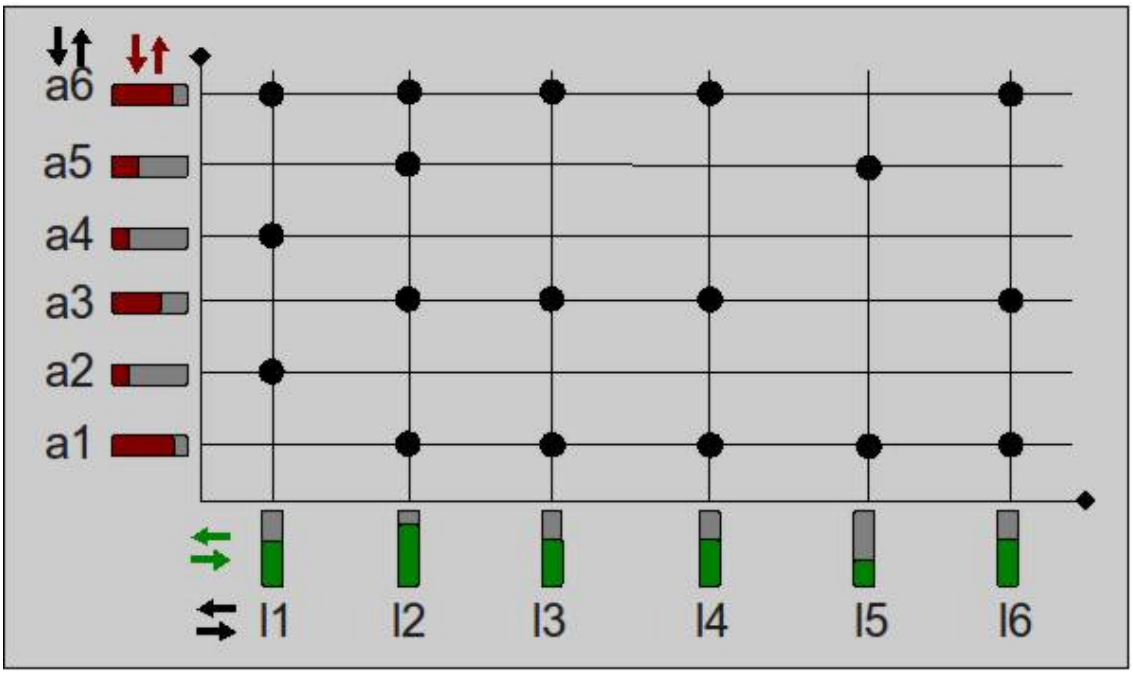


Figure 12: Details of resource Resources visited by students

6.3 Goals

Learning goals are important indicators of the advancement of the learning process. They could be represented by a combination of learner characteristics (like the knowledge achieved, the number of activities done, ecc). The list of goals will be defined by the instructional designer in the CAM. When the user reaches a particular goal, this information will be detected and stored in the user model.

The information that will be represented by the following views is:

- **GoalStudNum**: The number of learners that have achieved a particular learning goal
- **GoalStud**: Indication whether a learner achieved the learning goals

6.3.1 Learner view

The learner view presents only a list of all the goals settled in the current course, and which of them are achieved. This widget is one of the candidates to be embedded in the LMS interface. The detailed view that will allow learner to explore the analytic situation is presented in the left part of the Figure 13.

Like in the previous views (see Figure 10), the bars represent aggregated data and work as references.

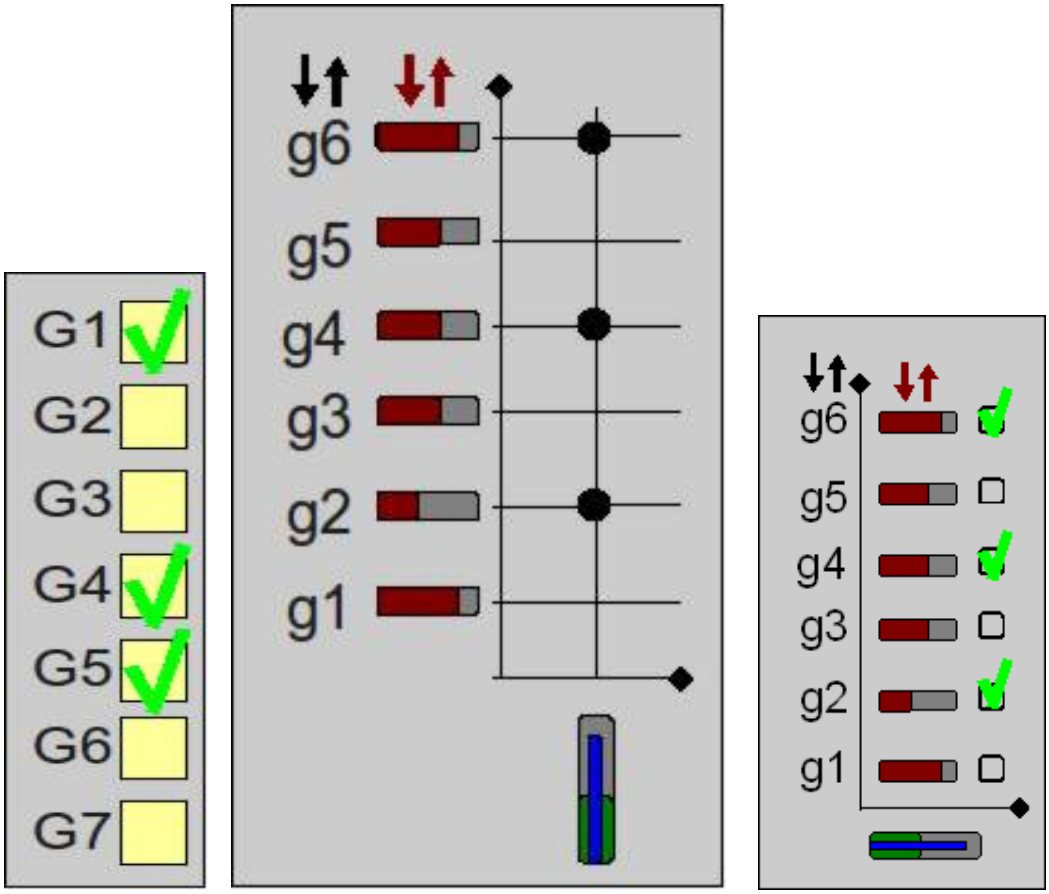


Figure 13: Goals achieved by the current learner. On the left: the representation of the compact indicator. On the centre and on the right: a representation of the detailed views for the dashboard.

6.3.2 Teacher/Tutor view

In the teacher/Tutor view a matrix of goals achieved by learners are presented as awareness information and as analysis tool, to identify problematic goals or problematic situations (like a student failing all the goals) or to show common patterns

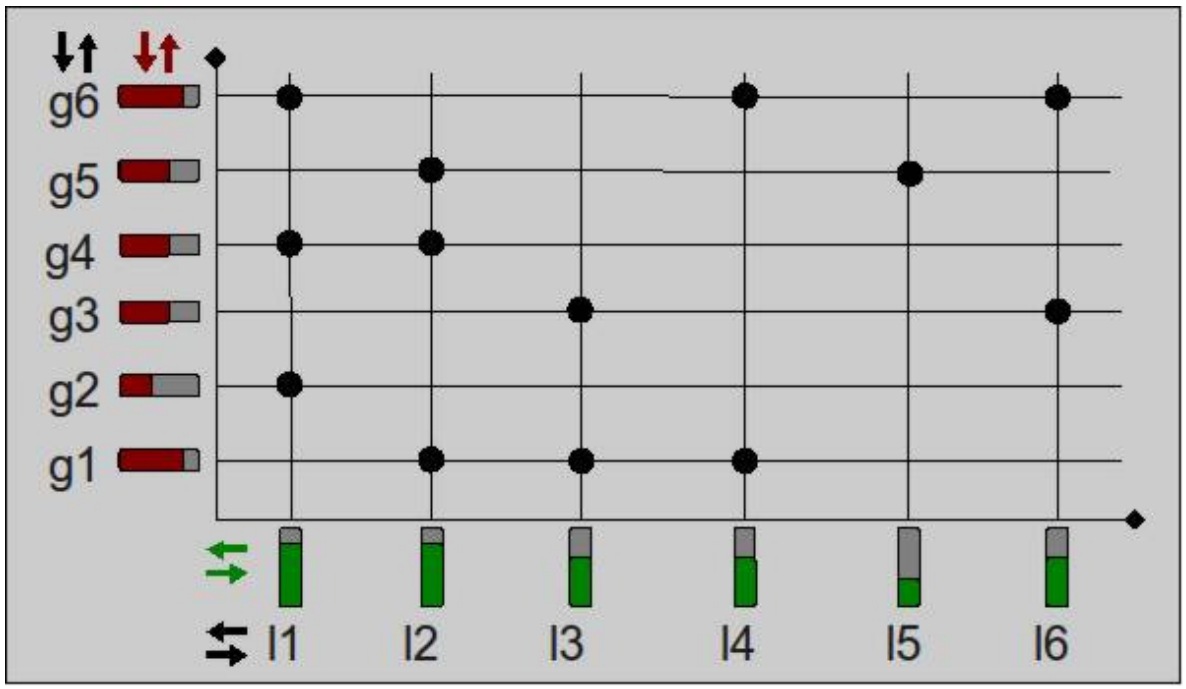


Figure 14: Goals achieved by learners

7 Conclusion and future work

In this document we have identified some key elements of the GVIS module. Source data and information descriptors give a description of the information we want to provide to our users, while mock-ups presented in section 6 illustrate some of the graphical representation that are being developed from partners involved in GVIS. A multi-layer software architecture (presented in section 5) is able to provide adaptive visualizations in form of compact indicators or set of widgets located in a user dashboard.

In the original planning of the GRAPPLE project, this deliverable should have described all the details of the GVIS software. Because of the delay on the implementation of other GRAPPE components upon which GVIS depends (in particular GUMF and GAT), the development of GVIS has been delayed and it is currently in the development phase. For these reason, some of the details of the software, and a complete description of the visualization produced by GVIS, has been postponed to the next deliverable of GVIS (D4.5c).

For the next year of the project, we expect to carry out the following activities, which will also be reported in D4.5c:

- Finalization of the software infrastructure. In particular, the modules that extract the data from GAT and GUMF, and the Builder component
- Evaluation of mock-ups, as illustrated in section 6 on this document, and final design of widgets
- Implementation of the widgets and compact indicators
- Evaluation of visualization with final users in real settings, and recommendations for further improvements.

APPENDIX

NOTE:

Here we list the data collected and stored in the other GRAPPLE components. Data items with a gray background will be extracted and used in GVIS.

A. List of data available from the current version of GALE (Milky Way application)

PERSISTENT data means stored in the UM database

NON PERSISTENT data means reinitialized from the default value for the application every time the course is loaded for the current user

Name	datatype	Persistent	description	Comment
visited	integer	Yes	the number of times that the user visited a concept	Could be exported in GUMF
knowledge	integer	Yes	indicates the knowledge that the user is assumed to have about a concept	Could be exported in GUMF
Suitability	boolean	No	indicates whether a concept is currently suitable for the user or not	Could be exported in GUMF
Order		No	the order that the concept has in the treeview on the left of the window compared to its siblings	Not exported in GUMF
Resource	text	No	a URL to the file that is used for generating the page; this file is a template for the page in our example	Not exported in GUMF
Image	text	No	a URL to an image of the concept; this resource is used by the template file that is specified in the Resource attribute	Not exported in GUMF
Info	text	No	a URL to a file with information about the concept; this resource is also used by the template file Some concepts in our example have some additional attributes, but those are exceptions and not much different from the other concepts.	Not exported in GUMF

B. List of data available from the Domain Model

Following information is taken form Deliverable D7.2b.

In GRAPPLE, the Domain Model structures elaborated in GAT are stored in the Domain Model repository in IMS Vocabulary Definition Exchange (VDEX) format. The Domain Model encodes in IMS VDEX the data stored in the domain map. The attributes that describe a Domain Map are the following:

Domain Map

- **Name**: title of the Domain Map
- **CreationDate**: creation date of the Domain Map
- **Author** of the Domain Map
- **Shared/Local**: this flag indicates if this Domain Map has been made available to all the authors in the authoring community
- **Description**: it describes the scope of the Domain Maps
- **Keywords**: they are words used in a reference work to link to other Domain Maps
- **Identifier**: it identifies the single Domain Map. The identifier is extremely important when the Domain Map is shared among all the authoring community.
- **Concepts**
  - **Name**: title of the Concept
  - **Identifier**: it identifies the single concepts
  - **Description**: it describes the scope of the concepts
  - **Keywords**: they are words used in a reference work to link to other concepts
  - **Resources**: they are the contents linked to the single concepts
    - **Name**: name of single resource
    - **Location**: it indicates the path where the resource is available. If the Domain Map is available to all the community, the resources are stored in a Content Repository
    - **UsageType**: any resource can be used in a different context such as introduction, image, body text, conclusion, ..
    - **LOM metadata**: they are the standard LOM metadata used to manage resources
- **Relationships**
  - **Name**: name of the single relationship.

C. List of data available from the LMS

Following information is taken form Deliverable D7.2b.

LMSs generate user events that are relevant to GRAPPLE. These events are converted into IMS LIP 1.0.1 format, and stored in GUMF. A list of user parameters has been prepared for each event. The user parameters are available to the GRAPPLE system only if they are made public.

1. ***Access to a course***: it identifies the moment when the user access a course
2. ***Tests/quizzes***: the user has terminated a test
3. ***Registration***: the user logs the first time in the LMS integrated with GRAPPLE
4. ***Student enrolment***: the student is enrolled in a new course
5. ***User login***: the user logs in the LMS integrated with GRAPPLE
6. ***Role change***: the user has changed his role
7. ***Learning activity change***: a particular learning activity has been changed for a given user
8. ***Learning activity addition***: a particular learning activity has been added for a given user
9. ***Learning activity removal***: a particular learning activity has been removed for a given user.
10. ***Learning activity access:*** a user accessed a learning activity.

Details for parameters:

Event Name	Parameter	Datatype
Access to a course (it identifies the moment when the user access a course)	Course id	
	Date of access	
	User id	
	Course topic	
	Course title	
	Course date of creation	
Tests/quizzes (the user has terminated a test)	Course id	
	User id	
	Topic	
	Goal	
	Score	
	Scale description/Threshold	
	Timestamp – start (last)	
	Timestamp – end	
	Attempts	

Registration

(the user logs the first time in the LMS integrated with GRAPPLE)

User id	
Password	
Last Name	
First Name	
Organisation	
Email	
Language	
Gender	
Date of birth	
Street	
Town	
Postal code	
Region	
Country	
IP address	
Role	

Student enrolment

(the student is enrolled in a new course)

User id	
Role	
Course id	

User login

(the user logs in the LMS integrated with GRAPPLE)

User id	
Attended courses	
IP address	

Role change

(the user has changed his role)

User id	
Role	

Learning activity change

(a particular learning activity has been changed for a given user)

User id	
Course id	
Learning activity	

Learning activity addition

(a particular learning activity has been added for a given user)

User id	
Course id	
Learning activity	

Learning activity removal

(a particular learning activity has been removed for a given user)

User id	
Course id	
Learning activity	

D. List of data available from CAM

Following information is taken form Deliverable D3.3b.

- The **domainModels** and **crtModels** in use,
- A number of *crt* tags, containing the instantiation of crts from *the crtModel* with actual concepts from *the domainModel*.

The ***crt* tag** contains the following information:

- A unique identifier the uuid
- The shape and colour of the crt, for displaying the crt in the CAM tool, this is optional, defaults will be used if the tags are omitted.
- An optional position element. The position of a crt needs to be given if the crt is not binary (1 or >2 sockets). For binary crts the position is calculated, based on the position of the sockets.
- A number (≥1) of camSockets.

The ***camSocket*s** contain the instantiation with the actual **concepts**, they contain the following information:

- An optional caption, a name for the socket, for the authors' assistance.
- A unique identifier the uuid
- A socketId, the unique id of the specific socket in the crtModel
- A position, the position where the socket should be displayed in the CAM Tool.
- An optional shape and colour, the shape and colour a CAM socket should have, a default will be used if the tags are omitted.
- A number of entity tags.

The **entity** contains the instantiation with the actual concepts, they contain the following information:

- Zero or one dmID, the ID of the concept in the domain model that is assigned to the entityID, with
- Zero or more labels for the resource of a concept and an optional location for a resource

The following items have not been implemented in the CAM tool yet:

- A number of relationshipType elements, where a requirement for a concept to be involved in a certain relationship in the DM model can be expressed (e.g., the entity should have participated in an IS-A relation)
- The relative position in the socket instance, if it differs from the default.
- The size of the entity in the socket, if it differs from the default.
- The shape of the entity in the CAM instance, if it differs from the default.
- The image of the entity in the CAM instance, if it differs from the default.
- The colour of the entity in the CAM instance, if it differs from the default

NOTE: as a future development, from CAM will also be available:

- Expected knowledge levels
- List of goals

E. List of data available from GUMF

- Data provided by GALE
- Data provided by LMS
- Inferred internal data