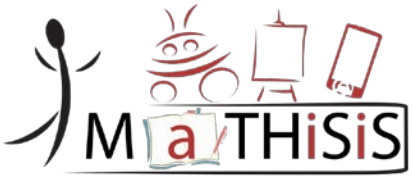

Managing Affective-learning THrough Intelligent atoms and Smart InteractionS

D3.3 The MaTHiSiS Learning Graphs

Workpackage	WP3 – Smart Learning Atoms and Graph Tools
Editor(s):	Dorothea TSATSOU, CERTH Nicholas VRETOS, CERTH Norbert BARICHARD, DXT Clément RODRIGUES-VIGUIER, DXT
Responsible Partner:	CERTH
Quality Reviewers	Robert HOUGHTON, UoN Ilona LASICA, EOPPEP
Status-Version:	Final-v1.0
Date:	Project Start Date: 01/01/2016; Duration: 36 months Deliverable Due Date: 31/12/2016 Submission Date: 09/01/2017
EC Distribution:	Report, Public
Abstract:	This deliverable defines and exemplifies the structure for representing the learning scenario to the MaTHiSiS learners, namely the MaTHiSiS Learning Graphs (LGs). It further describes the library implementation that handles the creation, access and modification of the Learning Graphs, namely the LG lib, as well as the corresponding Open API

This document is issued within the frame and for the purpose of the *MATHISIS* project. This project has received funding from the European Union's Horizon 2020 Programme (H2020-ICT-2015) under Grant Agreement No. 687772

	implementation used to seamlessly access the features of LG lib throughout the MaTHiSiS platform. In conclusion, the deliverable presents the early front-end interface implementation on the MaTHiSiS platform, used to create, view, access and modify Learning Graphs.
Keywords:	Learning Graphs, Learning Content Editor, LG lib, LG lib Open API
Related Deliverable(s)	D2.2 *Full scenarios of all use cases* D2.3 *Full System Architecture* D3.1 *The MaTHiSiS Smart Learning Atoms* D7.2 *MaTHiSiS Platform, 1st release*

Document History

Version	Date	Change editors	Changes
0.1	20/10/2016	Dorothea TSATSOU, CERTH Nicholas Vretos, CERTH	Table of Contents
0.2	07/12/2016	Norbert BARICHARD, DXT Clément RODRIGUES-VIGUIER, DXT	Input Section 3.4
0.3	13/12/2016	Dorothea TSATSOU, CERTH Nicholas Vretos, CERTH	Input Sections 2.1, 2.2, 3.2
0.4	19/12/2016	Dorothea TSATSOU, CERTH Nicholas Vretos, CERTH	Input Sections 3.1, 3.3
0.5.1	22/12/2016	Dorothea TSATSOU, CERTH Nicholas Vretos, CERTH	Input Sections Executive Summary, 1, 2.3, 6
0.5.2	22/12/2016	Norbert BARICHARD, DXT Clément RODRIGUES-VIGUIER, DXT	Additions Section 3.4
0.6	27/12/2016	Dorothea TSATSOU, CERTH	Homogenisation, Internal review version
0.7	03/01/2017	Dorothea TSATSOU, CERTH	First peer reviewer's feedback incorporated
1.0	09/01/2017	Dorothea TSATSOU, CERTH	Second peer reviewer's feedback incorporated FINAL VERSION TO BE SUBMITTED

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
ASC	Autism Spectrum Case
CGDLC	Career Guidance and Distance Learning Case
ITC	Industrial Training Case
LA	Learning Action
LAM	Learning Action Materialization
LCM	Learning Content Manager
LCS	Learning Content Space
LES	Learning Experience Supervisor
LG	Learning Graph
LGE	Learning Graph Engine
MEC	Mainstream Education Case
PA	Platform Agent
PMLDC	Profound and Multiple Learning Disabilities Case
SLA	Smart Learning Atom
US	User Space

Table 1: Definitions, Acronyms and Abbreviations

Project Description

The MaTHiSiS learning vision is to provide a novel advanced digital ecosystem for vocational training, and special needs and mainstream education for individuals with an intellectual disability (ID), autism and neuro-typical learners in school-based and adult education learning contexts. This ecosystem consists of an integrated platform, along with a set of re-usable learning components with capabilities for: i) adaptive learning, ii) automatic feedback, iii) automatic assessment of learners' progress and behavioural state, iv) affective learning, and v) game-based learning.

In addition to a learning ecosystem capable of responding to a learner's affective state, the MaTHiSiS project will introduce a novel approach to structuring the learning goals for each learner. Learning graphs act as a novel educational structural tool. The building materials of these graphs are drawn from a set of Smart Learning Atoms (SLAs) and a set of specific learning goals which will constitute the vertices of these graphs, while relations between SLAs and learning goals constitute the edges of the graphs. SLAs are atomic and complete pieces of knowledge which can be learned and assessed in a single, short-term iteration, targeting certain problems. More than one SLA, working together on the same graph, will enable individuals to reach their learning and training goals. Learning goals and SLAs will be scoped in collaboration with learners themselves, teachers and trainers in formal and non-formal education contexts (general education, vocational training, lifelong training and specific skills learning).

MaTHiSiS is a 36-month long project co-funded by the European Commission Horizon 2020 Programme (H2020-ICT-2015), under Grant Agreement No. 687772.

Executive Summary

The scope of this document is to describe the novel educational structural tool of MaTHiSiS, which enables non-linear execution of a learning scenario while fostering personalised and adaptive learning, i.e. the Learning Graph (LG), and how it is integrated and accessed within the MaTHiSiS ecosystem. This is a public document, aimed to introduce this educational tool to relevant stakeholders, while it will constitute the main reference document for the integration of LGs in the MaTHiSiS ecosystem.

The MaTHiSiS learning approach will rely on the organisation of learning scenaria in Learning Graphs, which will encapsulate the knowledge, skills and/or competences to acquire during the learning process in order to achieve specific learning goals, as interconnected vertices in a graph structure. Furthermore, Learning Graphs will capture the progress/uptake over the learning scenario for each specific learner taking part in the learning process, based on both personalised metrics, captured historically during the learner's experience with the MaTHiSiS ecosystem, as well as through the learner's affective response and performance over targeted learning activities. This progress will be represented as a personalised competence weight on the graphs' vertices.

The deliverable will firstly delve into the definition of the concept of LGs and relevant concepts within the MaTHiSiS ecosystem, and will further detail the corresponding LG representation structures (i.e. graph representation format, nodes and edges attributes) and functionalities implemented (i.e. create and manipulate graphs and individual elements in them) in order to create, access and manipulate LGs. The first pre-alpha prototype of the front-end LG editing interface is also presented, which will be used during the MaTHiSiS driver pilots. These structures, methods and interface will evolve over time depending on the technical requirements that might surface during the progress of the development and the first validation of the MaTHiSiS system, as well as based on the feedback of end users during the driver pilots.

1. Introduction

This document comprises Deliverable D3.3 - The MaTHiSiS Learning Graphs and describes one of the outcomes of work package WP3 - Smart Learning Atoms and Graph Tools, more particularly the outcome of Task 3.2 – Learning Graphs Implementation. It is closely related to Deliverable D3.1 – The MaTHiSiS Smart Learning Atoms [4] and its outcomes will be core to the implementation of the MaTHiSiS Learning Graph Engine (LGE) [1].

It is a public technical report, intended to describe the structure underneath the core MaTHiSiS learning approach, namely Learning Graphs (LGs). It will be used as a reference during the development of relevant components for the MaTHiSiS platform and as the documentation to guide the integration of LGs into the MaTHiSiS ecosystem.

Task 3.2 partners, namely CERTH and DXT, have been involved in the definition of this concept, with CERTH being responsible for the implementation of the LG library and corresponding Open API used to represent, access and modify learning scenarios through unpersonalised LG structures as well as to represent and personalised learner-specific instances of these structures, adaptable according to the learners' particular learning styles and reaction to the learning process. DXT is responsible for developing the interface, to be employed by the end users of the platform, in order to compose, retrieve and modify unpersonalised LGs and to access and review personalised LG instances.

[1] To be reported in a subsequent deliverable, due on M24 of the project, namely D6.2 – The MaTHiSiS Learning Graph Engine

2. Learning Graphs and their use in Learning Scenarios

2.1 Objectives and definitions

A **Learning Scenario** is "an a priori description of a learning situation, independently of the underlying pedagogical approach. It describes its organization with the goal of ensuring the appropriation of a precise set of knowledge, competences or skills"[2]. Within MaTHiSiS, *a learning scenario ensures the appropriation of the knowledge, competence and/or skills encapsulated in specific Learning Graph*.

Smart Learning Atoms (SLAs) are atomic and complete pieces of learner knowledge, competencies and/or skills, which can be learned and assessed in a single, short-term learning process iteration. SLAs essentially comprise *primordial learning goals, constituents of more advanced learning goals, which cannot be further reduced to more primitive notions*. In a nutshell, they consist of the simplest of concepts pertaining to **what-to-learn** during the educational process (partaking in but not restricted to a particular learning scenario, cf. Deliverable 3.1 *The MaTHiSiS Smart Learning Atoms*.

Learning goals describe learners' skills or knowledge over a comprehensive learning objective. In essence, learning goals consist of the *particular competences the learners need to acquire in order to achieve a specific learning objective*. In a nutshell, they consist of complex/composite concepts pertaining to **what-to-learn** per learning scenario during the educational process.

A **Learning Graph (LG)** consists of *learning content components* (i.e. learning goals and SLAs) *and weighted relations between them*. The LG will guide the procedure of organising a learning scenario and will lead to the achievement of an educator's teaching/training objectives. In a nutshell, LGs consist of all the components/concepts pertaining to **what-to-learn** per learning scenario during the educational process.

The hierarchical organisation of the learning content components into interconnected atomic and composite units enables the learning experience to focus, in a non-linear manner, upon mastering specific learning content components in order to master the overall learning objective of each LG. This entails training for the *optimal* atomic learning content constituents (i.e. SLAs), which in turn implies optimally training for the composite learning goals. Determining the optimal SLAs to train for in this hierarchical graph structure is dependent on a) the personal competence score of each learner over the SLAs, in each iteration of the learning process and b) their importance over the ultimate target (learning goals). The correlation of these two variables will determine the veritable 'weakest link(s)' – or weakest SLA(s) – that the learning experience should focus on training, in each iteration of the learning process.

2.2 Methodology and dependencies

In order to facilitate the adaptability of the learning scenario to different learner specifications, LGs will take up two forms in the MaTHiSiS learning setting.

Unpersonalised, core LGs will comprise of the universal representation of the MaTHiSiS learning scenaria into a graph $G = (V, E)$ of interconnected learning content components (SLAs and learning goals). The interconnections (directed edges $\{E\}$) of these components (vertices $\{V\}$) form a hierarchy between the simpler components to the more composite ones, thus denoting that the composite goals are comprised of the simpler components (i.e. learning goals are comprised by SLAs or even other, simpler learning goals, which are in turn, ultimately, composed of some SLAs). The

[2] http://www.tel-thesaurus.net/wiki/index.php/Learning_scenario

interconnections also bear weights[3] which denote the participation of a constituent (source) learning content component to the target learning content component.

It is important to notice that given the atomicity and standalone nature of SLAs (cf. also Deliverable 3.1 *The MaTHiSiS Smart Learning Atoms* [4]), SLAs partaking to a LG can never consist of target vertices, rather only of source vertices (direction-wise with respect to graph edges), thus they cannot be interconnected to each other, as this would mean that an SLA is comprised of another SLA. In the simplest of manifestations, LGs comprise of *at least* one learning goal and one or more SLAs linked to it, as illustrated in Figure 1.

In this example graph structure there are two learning goals: *LGoal1* and *LGoal2*. *LGoal1* is comprised collectively by SLAs *SLA1*, *SLA2*, *SLA3* and *SLA4*, while *LGoal2* is comprised by SLAs *SLA4* and *SLA5*. Each SLA participates to their respective goals with different gravity (weight). We can observe also the fact that an SLA can participate to more than one goals, as in the case of *SLA4*.

In correlation to real-world learning scenaria (cf. corresponding example in Deliverable 3.1), we can assume *SLA3* being a concept such as "Word recognition", *SLA4* being "Object recognition", *LGoal 1* being "Vocabulary improvement" and *LGoal 2* being "Subitizing[4]". In another learning scenario, *word recognition* and *object recognition* may be a required skill in order to master *vocabulary improvement* (cf. Deliverable 2.2 *Full Scenarios for all Use Cases* [1]), however their contribution to this goal is different, with *word recognition* being somewhat more important, thus the corresponding difference in participation weights to this goal. At the same time, *object recognition* might also be required in order to learn how to *subitize*, playing a more important role to this goal than to *vocabulary improvement*, as captured in its respective participation weights to the two goals.

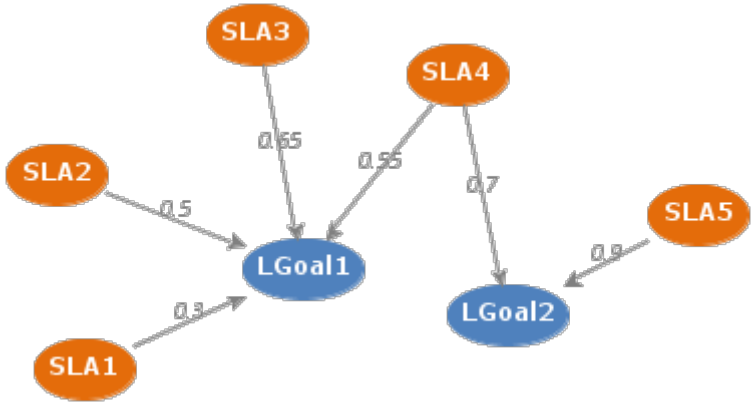

Figure 1: LG structure example, simple

It is also important to clarify that although LGs can maintain a hierarchy of nested learning goals, they do not represent a tree structure of cascading goals, rather they construct a non-linear network of interconnections between learning content components. Figure 2 presents an example of such a LG illustrating the notion of nested learning goals. In this structure, while *LGoal1* is comprised collectively by SLAs *SLA1*, *SLA2*, *SLA3* and *SLA4*, and *LGoal1* itself is a constituent of *LGoal3*, thus conveying the properties of SLAs *1-4* to *LGoal3*, we can see that *SLA1* is also re-connected to *LGoal3*, but this time with a higher participation degree. This indicates that the participation of *SLA1* to *LGoal1* was not sufficient enough to reflect the importance of *SLA1* to *LGoal3*, thus an additional weighted relation is needed between the two.

In a real-world scenario, if *SLA1* is correlated to the skill "Speech enunciation", *LGoal1* represents again "Vocabulary improvement" and *LGoal3* is "Communication/Socialisation skills" [5], it is easy to

[3] weight ∈ [0.0, 1.0]

[4] Subitizing is the ability to 'see' a small amount of objects and know how many there are without counting (http://teachmath.openschoolnetwork.ca/grade-1/number-sense/subitizing/)

[5] This scenario refers to cases that apply to learners with verbal expression capacities.

infer how clearly expressing oneself with *speech* is in fact to an extent a sub-skill required to improve one's *vocabulary*, and *vocabulary* is in turn an essential prerequisite for engaging in *communication*, however *speech enunciation* is fundamental to a much greater extent than in the previous goal towards practicing and enhancing *communication and socialisation skills*.

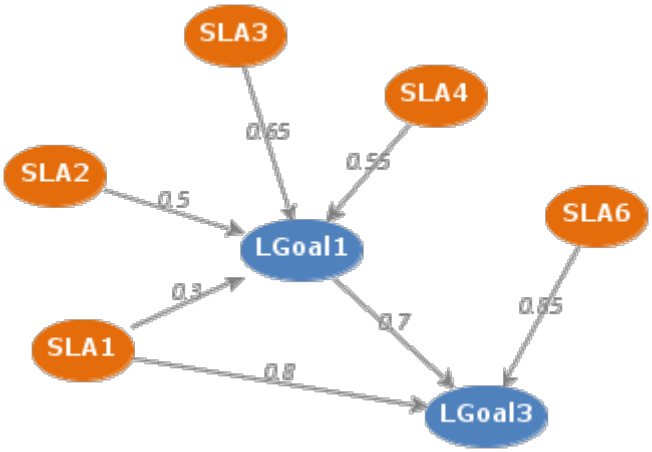

Figure 2: LG structure example with nested goals

Personalised LG instances (LGIs) will be created for each learner, upon the implementation of the learning scenario that involves these LGs. These instances (graphs $G' = (V', E')$) will incorporate a reference to their corresponding core unpersonalised counterparts. They differ in structure from their respective core graphs in that they allow assignment of a scalar weight to the vertices of the personalised graph per learner, indicating the uptake of the learner for the different learning content component.

During the personalisation and adaptation process, the LG instances' vertex weights will be adapted through the MaTHiSiS Learning Graph Engine (LGE). At the first step of the LGE implementation, all vertex weights, corresponding to SLA instances, will initially be assigned a value based on the learners' profile and historical record of transactions (personalisation) or based in their affect state and performance over the task/learning action at hand (adaptation). By employing the LG lib, through the process described in Formula (1), all subsequent learning goal vertices will also be assigned an initial weight. This state of weights in the graph will be hereafter referred to as the 'current state' of the LG instance. At a second step, the LGE will take into account the spectral analysis of the previous state of the LG instance (if existent) and the current state in order to finally produce new, predictive weights that will enable fastest converging the LG instance to its optimal state (all vertex weights = 1.0).

The edge weights remain unaltered through this process for each LG instance (therefore $\{E'\} = \{E\}$), since they represent the global participation weight of each constituent vertex to each target vertex and are not affected by learner performance. However, the personalised vertex weight of each (target) learning goal *does* take into account both the vertex and edge weights of each incoming neighbouring vertices. To this end, the weight *w* of each learning goal in a personalised LG instance is define by Formula (1), denoting that a learning goal weight is defined by the weighted average of all incoming source vertices' weights, subject to their participation (edge) weight to this goal.

$$\overline{w_{u_i}} = \frac{\sum_{i=1}^{n} w_{e_i} w_{v_i}}{\sum_{i=1}^{n} w_{e_i}} \qquad (1)$$

where $e \in E'$, $u, v \in V'$, u is the target learning goal and $u \sim v$.

From the aforementioned process, it is evident that, as in the case of standalone SLAs (cf. Deliverable D3.1 [4]), a history of **runtime LG instances** per learner needs to be stored, in order to enable access to previous states and historical records of these structures. To this end, a historical record of runtime (denoted as 'rtm') instances of LGs will be maintained, bearing a connection to the long-term (last state) personalised LG instance for each learner, along with the reference to the particular session that an adaptation of the LG instances' vertex weights has occurred (or at least examined).

The personalised LG instances (both last state and runtime) reside on the MaTHiSiS User Space (US), detailed in Deliverable D2.3 *Full System Architecture [2]*.

As far as dependencies with other MaTHiSiS structures and components go, it is easy to recognise that the LGs' most prominent dependency is the standalone Smart Learning Atoms (SLAs). Although SLAs can be maintained individually (cf. Deliverable 3.1 [4]), they cannot be trained unless they take part in at least one concrete learning scenario, fulfilled by a LG. LGs are represented following the same logic as SLAs, therefore each LG structure is related to their counterpart SLA structures that take part in the particular graph, i.e. core SLAs are related to unpersonalised LGs, SLA instances are related to corresponding instances of these LGs for the particular learner and SLA runtime instances are related to their respective runtime LG instance for the particular session.

Furthermore, unpersonalised LGs encapsulate the ID of the user (usually a tutor) that has initially created them, connecting them to the collection of users taking part in the MaTHiSiS 'universe'. LG instances are also connected to the users, in that they bear the ID of the learner for which each instance applies. As a reminder, a learner may be attached to only one unique instance of a LG (or a unique record of runtime LGs). Runtime LGs in particular also encode a reference to the particular learning session where they manifested.

Figure 3 graphically illustrates the interdependencies pertaining to LGs, based on the deployment of the MaTHiSiS database schema. The collections that the LGs are related to are portrayed as empty placeholders for visual simplification purposes.

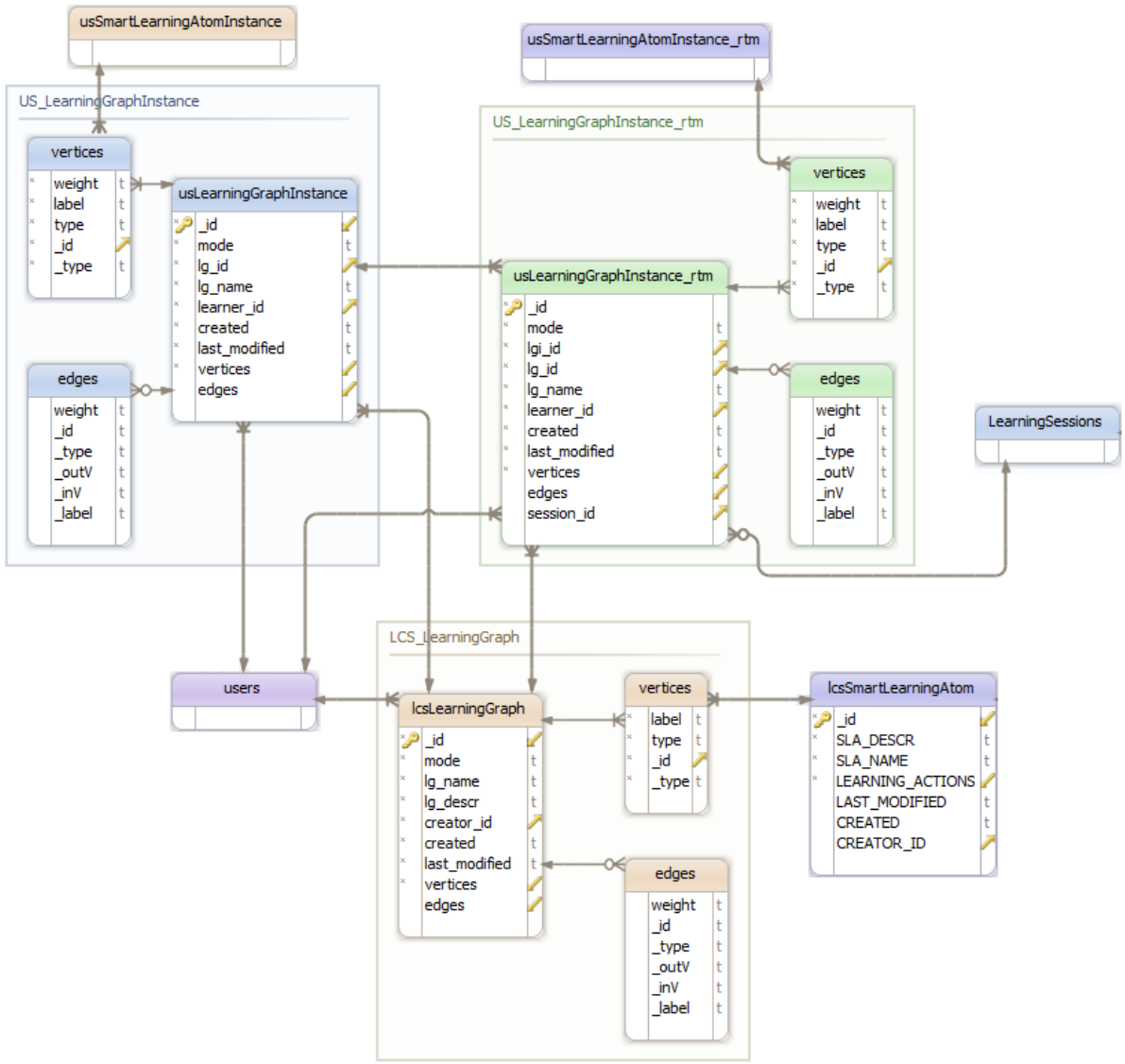

Figure 3: LGs and LG Instances and dependencies with other MaTHiSiS collections

2.3 Concrete learning scenaria and their LGs

The following example LGs are created based on the *second* 'Age 8-9' Learning Graph of the ASC scenario, described in Deliverable 2.2 *Full scenarios of all use cases* [1].

2.3.1 Unpersonalised LGs

The LG represented in this section comprises the core LGs, developed for the *second* ASC 'age 8-9' Learning Graph of Deliverable 2.2. It was elected among the other LGs of D2.2 to exemplify a MaTHiSiS LG, because it most reliably portrays a connected LG with more than one learning goals and several SLAs. It should be noted that for the first release of the MaTHiSiS platform, in order to gradually introduce the complex of LGs to the MaTHiSiS users, LGs are opted to be as elementary as possible and therefore no edge weights are yet defined within the current MaTHiSiS scenarios, thus all the edges in the presented LG bear the default weight of 1.0.

Figure 1 presents a graphical illustration of the aforementioned LG, while Table 2 respectively provides the structure of this LG.

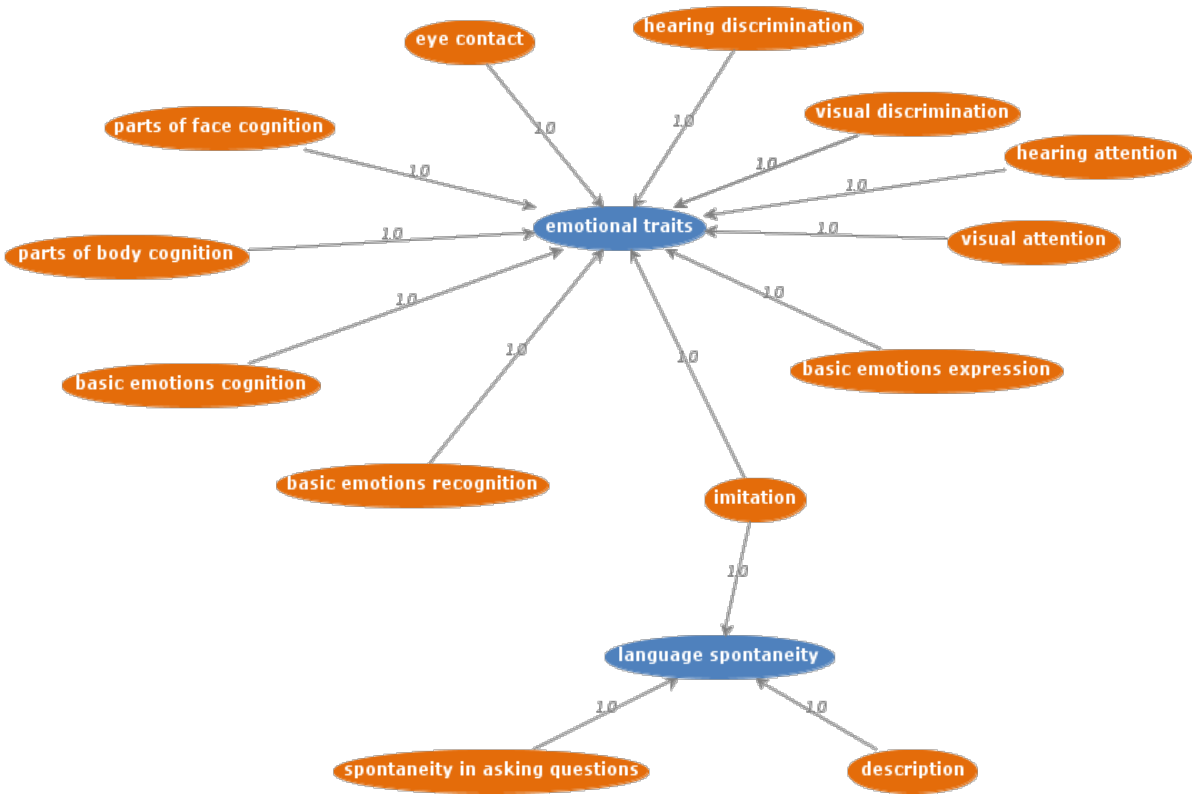

Figure 4: Graphical illustration of LG structure for the second case of 'Age 8-9' learners of the MaTHiSiS Autism Spectrum Case (ASC)

Table 2: The LG structure for the second case of 'Age 8-9' learners of the MaTHiSiS Autism Spectrum Case (ASC)

LCS_LearningGraph: asc_8-9_2


```
{
        "_id" : "58599f85657e8f335c2fa807",
        "mode": "NORMAL",
        "lg_name": "asc_8-9_2",
        "lg_descr": "2nd case of MaTHiSiS ASC for learners aged 8-9.",
        "creator_id": "582ee619f1d4655be2fbae3b",
        "created": "2016-12-13T18:53:16.320",
        "last_modified": "2016-12-13T18:53:16.320",
        "vertices": [{
                "label": "visual attention",
                "type": "SLA",
                "_id": " 58598eb9657e8f33883ab421",
                "_type": "vertex"
        }, {
                "label": "hearing attention",
                "type": "SLA",
                "_id": " 58598eb9657e8f33883ab422",
                "_type": "vertex"
        }, {
```

```
            "label": "visual discrimination",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab423",
            "_type": "vertex"
        }, {
            "label": "hearing discrimination",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab424",
            "_type": "vertex"
        }, {
            "label": "eye contact",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab425",
            "_type": "vertex"
        }, {
            "label": "imitation",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab426",
            "_type": "vertex"
        }, {
            "label": "parts of face cognition",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab427",
            "_type": "vertex"
        }, {
            "label": "parts of body cognition",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab428",
            "_type": "vertex"
        }, {
            "label": "basic emotions cognition",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab429",
            "_type": "vertex"
        }, {
            "label": "basic emotions recognition",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab42a",
            "_type": "vertex"
        }, {
            "label": "basic emotions expression",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab42b",
            "_type": "vertex"
        }, {
            "label": "spontaneity in asking questions",
```

```
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab42c",
            "_type": "vertex"
        }, {
            "label": "description",
            "type": "SLA",
            "_id": "58598eb9657e8f33883ab42d",
            "_type": "vertex"
        }, {
            "label": "emotional traits",
            "type": "LEARNING_GOAL",
            "_id": "58598eb9657e8f33883ab42e",
            "_type": "vertex"
        }, {
            "label": "language spontaneity",
            "type": "LEARNING_GOAL",
            "_id": "58598eb9657e8f33883ab42f",
            "_type": "vertex"
        }],
        "edges": [{
            "weight": "1.0",
            "_id": "0",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab421",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "1",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab422",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "2",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab423",
            "_inV": "58598eb9657e8f33883ab423",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "3",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab424",
            "_inV": "58598eb9657e8f33883ab42e",
```

```
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "4",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab425",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "5",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab426",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "6",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab427",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "7",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab428",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "8",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab429",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "9",
            "_type": "edge",
            "_outV": "58598eb9657e8f33883ab42a",
            "_inV": "58598eb9657e8f33883ab42e",
            "_label": "default"
        }, {
            "weight": "1.0",
            "_id": "10",
```

```
                "_type": "edge",
                "_outV": "58598eb9657e8f33883ab42b",
                "_inV": "58598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "11",
                "_type": "edge",
                "_outV": "58598eb9657e8f33883ab426",
                "_inV": "58598eb9657e8f33883ab42f",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "12",
                "_type": "edge",
                "_outV": "58598eb9657e8f33883ab42c",
                "_inV": "58598eb9657e8f33883ab42f",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "13",
                "_type": "edge",
                "_outV": "58598eb9657e8f33883ab42d",
                "_inV": "58598eb9657e8f33883ab42f",
                "_label": "default"
        }]
}
```


2.3.2 Personalised LGs

The following table presents a personalised instance of the LG portrayed in the previous section for a specific user.

Table 3: The personalised LG structure for the second case of 'Age 8-9' learners of the MaTHiSiS Autism Spectrum Case (ASC)

US_LearningGraphInstance: asc_8-9_2


```
{
        "_id" : "585aac3c657e8f2560239531",
        "mode": "NORMAL",
        "lg_id": "58599f85657e8f335c2fa807",
        "lg_name": "asc_8-9_2",
        "lg_descr": "2nd case of MaTHiSiS ASC for learners aged 8-9.",
        "learner_id": "5821fa0d1d56e4334ad48ed2",
        "created": "2016-12-14T12:37:20.560",
        "last_modified": "2016-12-14T12:37:20.560",
        "vertices": [{
                "weight": "0.7",
                "label": "visual attention",
                "type": "SLA",
                "_id": " 55598eb9657e8f33883ab421",
                "_type": "vertex"
        }, {
                "weight": "0.45",
                "label": "hearing attention",
                "type": "SLA",
                "_id": " 55598eb9657e8f33883ab422",
                "_type": "vertex"
        }, {
                "weight": "0.8",
                "label": "visual discrimination",
                "type": "SLA",
                "_id": "55598eb9657e8f33883ab423",
                "_type": "vertex"
        }, {
                "weight": "0.6",
                "label": "hearing discrimination",
                "type": "SLA",
                "_id": "55598eb9657e8f33883ab424",
                "_type": "vertex"
        }, {
                "weight": "0.35",
                "label": "eye contact",
                "type": "SLA",
                "_id": "55598eb9657e8f33883ab425",
                "_type": "vertex"
        }, {
                "weight": "0.75",
                "label": "imitation",
                "type": "SLA",
```

```
            "_id": "55598eb9657e8f33883ab426",
            "_type": "vertex"
        }, {
            "weight": "0.85",
            "label": "parts of face cognition",
            "type": "SLA",
            "_id": "55598eb9657e8f33883ab427",
            "_type": "vertex"
        }, {
            "weight": "0.6",
            "label": "parts of body cognition",
            "type": "SLA",
            "_id": "55598eb9657e8f33883ab428",
            "_type": "vertex"
        }, {
            "weight": "0.55",
            "label": "basic emotions cognition",
            "type": "SLA",
            "_id": "55598eb9657e8f33883ab429",
            "_type": "vertex"
        }, {
            "weight": "0.4",
            "label": "basic emotions recognition",
            "type": "SLA",
            "_id": "55598eb9657e8f33883ab42a",
            "_type": "vertex"
        }, {
            "weight": "0.6",
            "label": "basic emotions expression",
            "type": "SLA",
            "_id": "55598eb9657e8f33883ab42b",
            "_type": "vertex"
        }, {
            "weight": "0.2",
            "label": "spontaneity in asking questions",
            "type": "SLA",
            "_id": "55598eb9657e8f33883ab42c",
            "_type": "vertex"
        }, {
            "weight": "0.4",
            "label": "description",
            "type": "SLA",
            "_id": "55598eb9657e8f33883ab42d",
            "_type": "vertex"
        }, {
            "weight": "0.63",
```

```
                "label": "emotional traits",
                "type": "LEARNING_GOAL",
                "_id": "55598eb9657e8f33883ab42e",
                "_type": "vertex"
        }, {
                "weight": "0.38",
                "label": "language spontaneity",
                "type": "LEARNING_GOAL",
                "_id": "55598eb9657e8f33883ab42f",
                "_type": "vertex"
        }],
        "edges": [{
                "weight": "1.0",
                "_id": "0",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab421",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "1",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab422",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "2",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab423",
                "_inV": "55598eb9657e8f33883ab423",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "3",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab424",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "4",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab425",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
```

```
        }, {
                "weight": "1.0",
                "_id": "5",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab426",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "6",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab427",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "7",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab428",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "8",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab429",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "9",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab42a",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "10",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab42b",
                "_inV": "55598eb9657e8f33883ab42e",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "11",
                "_type": "edge",
```

```
                "_outV": "55598eb9657e8f33883ab426",
                "_inV": "55598eb9657e8f33883ab42f",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "12",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab42c",
                "_inV": "55598eb9657e8f33883ab42f",
                "_label": "default"
        }, {
                "weight": "1.0",
                "_id": "13",
                "_type": "edge",
                "_outV": "55598eb9657e8f33883ab42d",
                "_inV": "55598eb9657e8f33883ab42f",
                "_label": "default"
        }]
}
```

3. Learning Graph library (LGlib) implementation details

3.1 Data structures

The structures of the unpersonalised LGs, personalised LG instances and runtime instances are detailed in the tables below. As aforementioned, the unpersonalised LGs reside on the Learning Content Space (LCS) and the personalised instances (hereafter referred to as LGIs) on the User Space (US) – cf. Deliverable D2.3 *Full System Architecture* [2]. LGs must include at least two vertices (an SLA and a learning goal) and at least one connecting edge between two vertices. The serialisation model of LGs and LGIs follows the GraphSON format, which is a standard JSON-based format for representing individual graph elements (i.e. vertices and edges)[6]. Fields marked with (*) constitute obligatory components of the structures.

Table 4: Unpersonalised LG data structure (LCS_LearningGraph)

LCS_LearningGraph			
Component	Description	Value	Related to
_id*	The unique DB identifier of the LG structure	String (hex)	(US_LGI) lg_id
mode*	Internal GraphSON (i.e. standardised JSON for graphs, cf. Section 3.3) required field. Always set to “NORMAL”.	String (“NORMAL”)	-
lg_name*	The unique name of the LG	String	-
lg_descr	Details about what this LG is about	String	-
creator_id*	The unique DB identifier of the user (tutor) that created this LG	String (hex)	(Users) _id, for role: tutor
created	The date and time that this LG was first created	String (date/time)	-
last_ modified	The date and time that this LG was modified last, whether that might be modifications in the LG info fields, modification in its vertices names or edges (including edge weight modification)	String (date/time)	-
vertices*	The list of vertices that take part in the particular LG. Vertices encapsulate their own set of attributes, listed hereafter as vertices.*X.*	List (String/JSON object)	-

[6] https://github.com/tinkerpop/blueprints/wiki/GraphSON-Reader-and-Writer-Library

vertices.label*	The unique name of the vertex. If the vertex represents an SLA, then this name must be the same as the corresponding unique SLA_NAME.	String	-
vertices.type*	The type of the vertex in terms of learning content components. Can be only one of “SLA” or “LEARNING_GOAL”	String (“SLA” or “LEARNING_GOAL”)	
vertices._id*	GraphSON required field. The unique identifier for the vertex. If the vertex represents an SLA, then this name must be the same as the corresponding unique DB identifier of the SLA’s _id. If it represents a learning goal, then a random hex identifier is assigned to it.	String (hex)	(LCS_SLA) _id for vertices.type = SLA
vertices._type*	Internal GraphSON required field. Different from the ‘vertices.type’ field, this field defines the type of the graph component. For vertices, it is always set to “vertex”.	String (“vertex”)	-
edges*	The list of edges that take part in the particular LG. Edges encapsulate their own set of attributes, listed hereafter as edges.*X.*	List (String/JSON object)	-
edges.weight*	A weight in [0,1] that denotes the importance/participation of the source vertex to the target vertex. Default: 1.0.	Double (as String in structure)	-
edges._id*	Internal GraphSON required field. Represents a unique identifier for the particular edge. Edge identifiers are essentially a sequential enumeration of the graphs edges from 0 to n, n being the total number of edges in the graph.	Integer (as String in structure)	-
edges._type*	Internal GraphSON required field. Defines the type of the graph component. For edges, it is always set to “edge”.	String (“edge”)	-
edges._outV*	Internal GraphSON required field. The unique vertices._id that this edge is directed *from* (i.e. outwards direction). It can be either a vertex of vertices.type “SLA” or of vertices.type “LEARNING_GOAL”.	String (hex)	(LCS_LG) vertices._id

edges._inV*	Internal GraphSON required field. The unique vertices._id that this edge is directed *to* (i.e. inwards direction). It can ONLY be a vertex of vertices.type "LEARNING_GOAL".	String (hex)	(LCS_LG) vertices._id
edges._label*	Internal GraphSON required field. The label of the edge. LG edges are not labelled, therefore this field is always set to label "default".	String ("default")	-

Table 5: Personalised LGI data structure (US_LearningGraphInstance)

US_LearningGraphInstance			
Component	Description	Value	Related to
_id*	The unique DB identifier of the LG instance.	String (hex)	(US_LGI_rtm) lgi_id
mode*	Internal GraphSON (i.e. standardised JSON for graphs, cf. Section 3.3) required field. Always set to "NORMAL".	String ("NORMAL")	-
lg_id*	The unique DB identifier of the corresponding unpersonalised LG.	String (hex)	(LCS_LG) _id
lg_name*	The unique name of the LG.	String	-
learner_id*	The unique DB identifier of the user (learner) that this LGI applies to.	String (hex)	(Users) _id, for role: learner
created	The date and time that this LGI was first created	String (date/time)	-
last_ modified	The date and time that this LGI was modified last, whether that might be modifications in the LGI info fields, or modification in its vertices or edges (including vertex weight modification – not edge weight modification however. Edge weight modifications are only allowed on the LCS_LG level).	String (date/time)	-
vertices*	The list of vertices that take part in the particular LGI. Vertices encapsulate their own set of attributes, listed hereafter as vertices.*X*.	List (String/JSON object)	-
vertices.weight*	A weight in [0,1] that denotes the uptake of the learner for this particular vertex. Default: 0.3.	Double (as String in structure)	-

vertices.label*	The unique name of the vertex. If the vertex represents an SLA, then this name must be the same as the corresponding unique SLA_NAME.	String	-
vertices.type*	The type of the vertex in terms of learning content components. Can be only one of "SLA" or "LEARNING_GOAL"	String ("SLA" or "LEARNING_GOAL")	
vertices._id*	GraphSON required field. The unique identifier for the vertex. If the vertex represents an SLA, then this name must be the same as the corresponding unique DB identifier of the SLAI's _id. If it represents a learning goal, then a random hex identifier is assigned to it.	String (hex)	(US_SLAI) _id for vertices.type = SLA
vertices._type*	Internal GraphSON required field. Different from the 'vertices.type' field, this field defines the type of the graph component. For vertices, it is always set to "vertex".	String ("vertex")	-
edges*	The list of edges that take part in the particular LG. Edges encapsulate their own set of attributes, listed hereafter as edges.*X*.	List (String/JSON object)	-
edges.weight*	A weight in [0,1] that denotes the importance/participation of the source vertex to the target vertex. Must always be the same as the corresponding core unpersonalised LG's edges attributes and cannot be modified within the LGI.	Double (as String in structure)	-
edges._id*	Internal GraphSON required field. Represents a unique identifier for the particular edge. Edge identifiers are essentially a sequential enumeration of the graphs edges from 0 to n, n being the total number of edges in the graph.	Integer (as String in structure)	-
edges._type*	Internal GraphSON required field. Defines the type of the graph component. For edges, it is always set to "edge".	String ("edge")	-

edges._outV*	Internal GraphSON required field. The unique vertices._id that this edge is directed *from* (i.e. outwards direction). It can be either a vertex of vertices.type "SLA" or of vertices.type "LEARNING_GOAL".	String (hex)	(US_LGI) vertices._id
edges._inV*	Internal GraphSON required field. The unique vertices._id that this edge is directed *to* (i.e. inwards direction). It can ONLY be a vertex of vertices.type "LEARNING_GOAL".	String (hex)	(US_LGI) vertices._id
edges._label*	Internal GraphSON required field. The label of the edge. LG edges are not labelled, therefore this field is always set to label "default".	String ("default")	-

Table 6: Personalised LGI runtime record data structure (US_LearningGraphInstance_rtm)

US_LearningGraphInstance_rtm			
Component	Description	Value	Related to
_id*	The unique DB identifier of the LG runtime instance.	String (hex)	-
mode*	Internal GraphSON (i.e. standardised JSON for graphs, cf. Section 3.3) required field. Always set to "NORMAL".	String ("NORMAL")	-
lgi_id*	The unique DB identifier of the (long-term) last state of the corresponding personalised LGI	String (hex)	(US_LGI) _id
lg_id*	The unique DB identifier of the corresponding unpersonalised LG.	String (hex)	(LCS_LG) _id
lg_name*	The unique name of the LG.	String	-
learner_id*	The unique DB identifier of the user (learner) that this LGI applies to.	String (hex)	(Users) _id, for role: learner
session_id*	The unique DB identifier of the learner session that this LGI's vertex weights were modified during runtime of the learning process	String (hex)	(Sessions) _id
created	The date and time that this LGI was first created	String (date/time)	-

last_ modified	The date and time that this LGI_rtm was modified last. For the first release, since all runtime instances are stored separately in different structures for every change in the vertex weights, it is likely to always coincide with the 'created' date.	String (date/time)	-
vertices*	The list of vertices that take part in the particular LGI_rtm. Vertices encapsulate their own set of attributes, listed hereafter as vertices.*X.*	List (String/JSON object)	-
vertices.weight*	A weight in [0,1] that denotes the uptake of the learner for this particular vertex. Default: 0.3.	Double (as String in structure)	-
vertices.label*	The unique name of the vertex. If the vertex represents an SLA, then this name must be the same as the corresponding unique SLA_NAME.	String	-
vertices.type*	The type of the vertex in terms of learning content components. Can be only one of "SLA" or "LEARNING_GOAL"	String ("SLA" or "LEARNING_GOAL")	
vertices._id*	GraphSON required field. The unique identifier for the vertex. If the vertex represents an SLA, then this name must be the same as the corresponding unique DB identifier of the SLAI_rtm's _id. If it represents a learning goal, then a random hex identifier is assigned to it.	String (hex)	(US_SLAI_rtm) _id for vertices.type = SLA
vertices._type*	Internal GraphSON required field. Different from the 'vertices.type' field, this field defines the type of the graph component. For vertices, it is always set to "vertex".	String ("vertex")	-
edges*	The list of edges that take part in the particular LG. Edges encapsulate their own set of attributes, listed hereafter as edges.*X.*	List (String/JSON object)	-
edges.weight*	A weight in [0,1] that denotes the importance/participation of the source vertex to the target vertex. Must always be the same as the corresponding core unpersonalised LG's edges attributes and cannot be modified within the LGI_rtm.	Double (as String in structure)	-

edges._id*	Internal GraphSON required field. Represents a unique identifier for the particular edge. Edge identifiers are essentially a sequential enumeration of the graphs edges from 0 to n, n being the total number of edges in the graph.	Integer (as String in structure)	-
edges._type*	Internal GraphSON required field. Defines the type of the graph component. For edges, it is always set to "edge".	String ("edge")	-
edges._outV*	Internal GraphSON required field. The unique vertices._id that this edge is directed *from* (i.e. outwards direction). It can be either a vertex of vertices.type "SLA" or of vertices.type "LEARNING_GOAL".	String (hex)	(US_LGI_rtm) vertices._id
edges._inV*	Internal GraphSON required field. The unique vertices._id that this edge is directed *to* (i.e. inwards direction). It can ONLY be a vertex of vertices.type "LEARNING_GOAL".	String (hex)	(US_LGI_rtm) vertices._id
edges._label*	Internal GraphSON required field. The label of the edge. LG edges are not labelled, therefore this field is always set to label "default".	String ("default")	-

3.2 Functionalities

The LG library incorporates all the methods and functionalities required to create, access and modify the LG and LGI[7] data structures. It also handles the In this library, long-term LGI structures and runtime snapshots of LGs are treated as the same structure, applicable to different serialisation on the MaTHiSiS database through the LG lib Open API (detailed in Section 3.3). It is implemented as a Java library which is embedded to the Open API. More specifically, the library offers functionalities to:

- Create a new LG (Java data structure), based on input of the mandatory fields of the data structure.
- Recreate a (Java) data structure of an unpersonalised LG, based on a given GraphSON input of a serialised LG.
- Create a new LG instance (Java data structure), based on input of the mandatory fields of the data structure.

[7] LGI runtime structures are internally treated as LGI structures in the LG lib, with the additional lgi_id and session_id runtime fields applicable only through the LG lib Open API (cf. Section 3.3) for reading and serialising LGI_rtm structures.

- Recreate a (Java) data structure of a personalised LGI, based on a given GraphSON input of a serialised LGI.
- For each non-mandatory field missing from the input (parameters or GraphSON structure), provide default values to produce a complete data structure.
    - In the case of date/time fields, the current system date and time are set, unless explicitly stated otherwise.
    - In the case of vertices of LGIs (corresponding to SLA instances), initial default weight (0.3) is set in the very first instantiation of an LG to a personal LGI, unless explicitly stated otherwise.
- Retrieve and set (update) different fields of the structures.
    - For set operations, the 'last modified' field is automatically set to the current system date and time, unless this field is explicitly declared in the input (parameters or JSON).
- Retrieve all SLA-type vertices in a given LG, along with subsequent information, i.e. pointers to respective standalone SLA structures.
- Retrieve all SLA-type vertices in a given LGI for a given learner, along with subsequent information, i.e. (SLA) vertex weights for the learner and pointers to respective standalone SLAI structures.
- For facilitating the personalisation and adaptation process, a direct functionality to update SLAI weights is exposed, which allows to set new SLAI weights without having to explicitly retrieve corresponding SLAIs to access this property.
- Similarly, for facilitating the SLA editing process, a direct functionality to insert and remove learning actions from SLAs is exposed.
- Create GraphSON serialisations for each of the supported data structures (LG, LGI) to be inserted to the MaTHiSiS DB through the LG lib Open API.

3.3 Open API

The *JAX-RS*[8] Java API for RESTful Web Services was used to create web services according to the Representational State Transfer (REST) architectural pattern for the LG lib Open API. Resources are connected to the Java LG library. The library is in charge of the processing (access, creation, update) of LG and LGI structures and the Open API is responsible of receiving and transmitting the data to the callers. The Open API is also responsible for serialising the LG and LGI (and LGI runtime) structures on the MaTHiSiS database.

Access to the LG lib Open API is available through the central *<MaTHiSiS base URL>/api/lg/* URL pattern. MaTHiSiS components that will consume LG lib functionalities through the Open API will be able to get data from appropriate HTTP connections (bound to specific URLs), detailed in the tables below. Parameters marked with (*) are mandatory.

URL pattern	**GET api/lg/getLGs**
Method	GET
Content type	Application/JSON
Description	Return the list of all LGs in the MaTHiSiS repository (Learning Content Space). Future releases will consider retrieving a list of LGs subject to some restriction(s) (i.e. per creator id)
Responses	If Success return a list of pointers to corresponding LG structures in JSON format

[8] https://jax-rs-spec.java.net/

	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**
	-	-	{ "lgs": [{ "lg_id":"{lg _id}", "lg _name":"{lg _name}", }, ...] }

URL pattern	**GET api/lg/getLG**		
Method	GET		
Content type	Application/JSON		
Description	Return an unpersonalised LG for a given id (and optionally, name as cross-reference)		
Responses	If Success return a LCS_LearningGraph model in GraphSON (JSON) format		
	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**

	id* label	?id={lg_id } & label={lg_name}	{ "_id" : "{lg_id}" "mode" : "NORMAL", "lg_name" : "{lg_name}", "lg_descr" : "{lg description}", "creator_id" : "{tutor_id}", "created" : "{date/time}", "last_modified" : "{date/time}", "vertices" : [{ "label" : "{vertex_name}", "type" : "SLA", "_id" : "{vertex_id}"), "_type" : "vertex" }, "label" : "{vertex_name}", "type" : "LEARNING_GOAL", "_id" :"{vertex_id}"), "_type" : "vertex" }, ...], "edges" : [{ "weight" : "{edge+weight}", "_id" : "{enumeration}", "_type" : "edge", "_outV" : "{outward_vertex_id}", "_inV" : "{inward_vertex_id}", "_label" : "default" }, ...] }

URL pattern	**GET api/lg/getSLAs**		
Method	GET		
Content type	Application/JSON		
Description	Return the list of all vertices of type “SLA” in the given LG, which include pointers to corresponding LCS_SmartLearningAtom structures		
Responses	If Success return a list of pointers to corresponding SLA structures in JSON format		
	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**

	-	-	{ "lg_id": {lg_id}, "lg_name": {lg_name}, "slas": [{ "sla_id":"{sla_id}", "sla_name":"{sla_name}", }, ...] }

URL pattern	**GET api/lg/getLGIs**		
Method	GET		
Content type	Application/JSON		
Description	Return the list of all LG instances in the MaTHiSiS repository for the given learner		
Responses	If Success return a list of pointers to LGIs in JSON format		
	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**
	uid*	?uid={learner_id}	{ "lgis": [{ "lgi_id": "{lgi_id}", "lg_id": "{lg_id}", "lg_name": "{lg_name}", "learner_id": "{lg_name}", }, ...] }

URL pattern	**GET api/lg/getLGI**
Method	GET
Content type	Application/JSON
Description	Return a personalised LGI for a particular learner, given the LGI id (and optionally, name as cross-reference). This call also cross-checks the current state of US_SmartLearningAtomInstance structures for all SLAIs in the LG and updates either the vertex weights in the LGI or the weights of the corresponding SLAIs to the most recent state (according to the most recent 'last_modified' date/time in each corresponding structure).
Responses	If Success return a US_LearningGraphInstance model in GraphSON (JSON) format

	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**
	id* label	?id={lgi_id } & label={lg_name}	{ "_id" : "{lgi_id}" "mode" : "NORMAL", "lg_id" : "{lg_id}", "lg_name" : "{lg_name}", "learner_id" : "{learner_id}", "created" : "{date/time}", "last_modified" : "{date/time}", "vertices" : [{ "weight" : "{vertex_weight}", "label" : "{vertex_name}", "type" : "SLA", "_id" : "{vertex_id}"), "_type" : "vertex" }, { "weight" : "{vertex_weight}", "label" : "{vertex_name}", "type" : "LEARNING_GOAL", "_id" :"{vertex_id}"), "_type" : "vertex" }, ….], "edges" : [{ "weight" : "{edge_weight}", "_id" : "{enumeration}", "_type" : "edge", "_outV" : "{outward_vertex_id}", "_inV" : "{inward_vertex_id}", "_label" : "default" }, ….] }

URL pattern	**GET api/lg/getSLAIs**
Method	GET
Content type	Application/JSON
Description	Return the list of all vertices of type “SLA” in the given (last state) LGI, which include pointers to corresponding US_SmartLearningAtomInstance structures.

	This call also cross-checks the current state of US_SmartLearningAtomInstance structures for all SLAIs in the LG and updates either the vertex weights in the LGI or the weights of the corresponding SLAIs to the most recent state (according to the most recent 'last_modified' date/time in each corresponding structure).		
Responses	If Success return a list of pointers to corresponding LGI vertices of type 'SLA' in JSON format		
	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**
	id* label	?id={lgi_id } & label={lg_name}	{ "learner_id": {learner_id}, "lgi_id": {lgi_id}, "lg_id": {lg_id}, "lg_name": {lg_name}, "slais": [{ "slai_id":"{slai_id}", "sla_id":"{sla_id}", "sla_name":"{sla_name}", }, ...] }

URL pattern	**GET api/lg/getLGIs/rtm**		
Method	GET		
Content type	Application/JSON		
Description	Return the list of all LG runtime instances in the MaTHiSiS repository for the given learner and given session. Optional restriction per session.		
Responses	If Success return a list of pointers to LGI_rtms in JSON format		
	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**

	uid* sid	?uid={learner_id} &sid={session_id}	{ "session_id": "{session_id}", "learner_id": "{learner_id}", "lgis_rtm": [{ "lgi_rtm_id": "{lgi_rtm_id}", "lgi_id": "{lgi_id}", "lg_id": "{lg_id}", "lg_name": "{lg_name}", }, ...] }

URL pattern	**GET api/lg/getLGI/rtm**		
Method	GET		
Content type	Application/JSON		
Description	Return a personalised LGI_rtm for a particular learner and particular session, given the LGI_rtm id (and optionally, LG name as cross-reference). Optional restriction per session.		
Responses	If Success return a US_LearningGraphInstance_rtm model in GraphSON (JSON) format		
	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**

	id* label sid	?id={lgi_rtm_id} & label={lg_name} &sid={session_id}	<pre>{ "_id" : "{lgi_rtm_id}" "mode" : "NORMAL", "lgi_id" : "{lgi_id}", "lg_id" : "{lg_id}", "lg_name" : "{lg_name}", "learner_id" : "{learner_id}", "created" : "{date/time}", "last_modified" : "{date/time}", "session_id" : "{session_id}", "vertices" : [{ "weight" : "{vertex_weight}", "label" : "{vertex_name}", "type" : "SLA", "_id" : "{vertex_id}"), "_type" : "vertex" }, { "weight" : "{vertex_weight}", "label" : "{vertex_name}", "type" : "LEARNING_GOAL", "_id" :"{vertex_id}"), "_type" : "vertex" }, ...], "edges" : [{ "weight" : "{edge_weight}", "_id" : "{enumeration}", "_type" : "edge", "_outV" : "{outward_vertex_id}", "_inV" : "{inward_vertex_id}", "_label" : "default" }, ...] }</pre>

URL pattern	**GET api/lg/getSLAIs/rtm**
Method	GET
Content type	Application/JSON
Description	Return the list of all vertices of type “SLA” in the given runtime LGI, which include pointers to corresponding US_SmartLearningAtomInstance_rtm structures. Optional restriction per session.

Responses	If Success return a list of pointers to corresponding LGI vertices of type 'SLA' in JSON format		
	If Error HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Success Response Model**
	id* label sid	?id={lgi_id } & label={lg_name} &sid={session_id}	{ "learner_id": {learner_id}, "session_id": {session_id}, "lgi_rtm_id": {lgi_rtm_id}, "lgi_id": {lgi_id}, "lg_id": {lg_id}, "lg_name": {lg_name}, "slais": [{ "slai_rtm_id":"{slai_rtm_id}", "slai_id":"{slai_id}", "sla_id":"{sla_id}", "sla_name":"{sla_name}", }, ...] }
	sid		

URL pattern	**POST api/lg/postLG**		
Method	POST		
Content type	Application/JSON		
Description	Create or update an unpersonalised LG on the MaTHiSiS DB under the LCS_LearningGraph collection. Automatically detects LG id from input structure. If the LG is new, thus bears no id, it automatically creates a new structure and assigns a new unique id. At least two vertices (one of type 'SLA' and one of type 'LEARNING_GOAL') and one edge connecting them are mandatory. Declaring vertex ids for vertices of type 'SLA' is mandatory and must reflect an existing SLAI id. Declaring vertex ids for vertices of type 'LEARNING_GOAL' is not mandatory. Missing non-mandatory fields of the input model are filled in with default and/or newly generated values.		
Responses	If Success HTTP 200 status code (OK)		
	If Error HTTP 304 status code (Not Modified) If input empty HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Input Model** **(*)Marks mandatory fields for the input to be accepted**

	-	-	{ "_id" : "{lg_id}" "mode" : "NORMAL", *"lg_name" : "{lg_name}", "lg_descr" : "{lg description}", *"creator_id" : "{tutor_id}", "created" : "{date/time}", "last_modified" : "{date/time}", *"vertices" : [{ * "label" : "{vertex_name}", *"type" : "SLA", *"_id" : "{vertex_id}"), "_type" : "vertex" }, *"label" : "{vertex_name}", *"type" : "LEARNING_GOAL", "_id" :"{vertex_id}"), "_type" : "vertex" }, ….], *"edges" : [{ *"weight" : "{edge+weight}", "_id" : "{enumeration}", "_type" : "edge", *"_outV" : "{outward_vertex_id}", *"_inV" : "{inward_vertex_id}", "_label" : "default" }, ….] }

URL pattern	**POST api/lg/postLGI**
Method	POST
Content type	Application/JSON
Description	Create or update a personalised LG instance for a given learner on the MaTHiSiS DB under the US_LearningGraphInstance collection. Automatically detects LGI id from input structure. If the LGI is new, thus bears no id, it automatically creates a new structure and assigns a new unique id. Other missing non-mandatory fields of the input model are filled in with default and/or newly generated values. It also creates or updates corresponding US_SmartLearningAtomInstance structures in the DB, based on the corresponding information incorporated in the LGI attributes (learner id) and in the vertices of type 'SLA' (weight, label, id), by iteratively calling the

	SLA lib Open API call: *POST api/sla/postSLAI?id={vertex_id}&label={vertex_name}*		
Responses	If Success HTTP 200 status code (OK)		
	If Error HTTP 304 status code (Not Modified) If input empty HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Input Model** **(*)Marks mandatory fields for the input to be accepted**
	-	-	{ "_id" : "{lgi_id}" "mode" : "NORMAL", *"lg_id" : "{lg_id}", *"lg_name" : "{lg_name}", *"learner_id" : "{learner_id}", "created" : "{date/time}", "last_modified" : "{date/time}", *"vertices" : [{ "weight" : "{vertex_weight}", *"label" : "{vertex_name}", *"type" : "SLA", *"_id" : "{vertex_id}"), "_type" : "vertex" }, { "weight" : "{vertex_weight}", *"label" : "{vertex_name}", *"type" : "LEARNING_GOAL", "_id" :"{vertex_id}"), "_type" : "vertex" }, ….], *"edges" : [{ *"weight" : "{edge_weight}", "_id" : "{enumeration}", "_type" : "edge", *"_outV" : "{outward_vertex_id}", *"_inV" : "{inward_vertex_id}", "_label" : "default" }, ….] }

URL pattern	**POST api/lg/postLGI/rtm**

Method	POST		
Content type	Application/JSON		
Description	Create a personalised runtime LG instance record for a given learner and session on the MaTHiSiS DB under the US_LearningGraphInstance_rtm collection. Automatically detects LGI id from input structure. Automatically produces the LGI runtime id and inserts the session id to the serialisation of the LGI_rtm on the MaTHiSiS DB. Missing non-mandatory fields of the input model are filled in with default and/or newly generated values. It also creates corresponding US_SmartLearningAtomInstance_rtm structures in the DB if not already existing (runtime records are unique and not updated, SLAI_rtm structures will be created if and only if the functionality that created the LGI_rtm hasn't previously created the SLAI_rtms), based on the corresponding information incorporated in the LGI_rtm attributes (learner id, session id) and in the vertices of type 'SLA' (weight, label, id), by iteratively calling the SLA lib Open API call: *POST api/sla/postSLAI/rtm?id={vertex_id}&label={vertex_name}*		
Responses	If Success HTTP 200 status code (OK)		
	If Error HTTP 304 status code (Not Modified) If input empty HTTP 204 status code (No Content)		
Parameters	**Name**	**URL pattern**	**Input Model** **(*)Marks mandatory fields for the input to be accepted**

			{ "_id" : "{lgi_id}" "mode" : "NORMAL", *"lgi_id" : "{lgi_id}", *"lg_id" : "{lg_id}", *"lg_name" : "{lg_name}", *"learner_id" : "{learner_id}", *"session_id" : "{session_id}", "created" : "{date/time}", "last_modified" : "{date/time}", *"vertices" : [{ "weight" : "{vertex_weight}", *"label" : "{vertex_name}", *"type" : "SLA", *"_id" : "{vertex_id}"), "_type" : "vertex" }, { "weight" : "{vertex_weight}", *"label" : "{vertex_name}", *"type" : "LEARNING_GOAL", "_id" :"{vertex_id}"), "_type" : "vertex" }, ….], *"edges" : [{ *"weight" : "{edge_weight}", "_id" : "{enumeration}", "_type" : "edge", *"_outV" : "{outward_vertex_id}", *"_inV" : "{inward_vertex_id}", "_label" : "default" }, ….] }

URL pattern	**POST api/lg/updateLGI**
Method	POST
Content type	Application/JSON
Description	Update the weight of the vertices for a given personalised LG instance. Will be called after the first step of personalisation or adaptation (for each iteration in adaptation). Will call *GET api/lg/getLGI* in order to update vertex weights for all SLAIs taking part in

	the specific LGI, after those are iteratively changed in the first step of the personalisation or adaptation loop (through *POST api/sla/postSLAI* or *POST api/sla/updateSLAIweight*). Will call LG lib to update vertex weight for all vertices of type 'LEARNING_GOAL' based on the weighted average (cf. Formula (1)) updated SLAI weights. Will trigger the second step of the personalisation and adaptation loop in the LGE, in order to update LGI vertex weights based on spectral analysis of the graph. (Note: The LGE will finally call *POST api/lg/postLGI* to update the last state of the given LGI and *POST api/lg/postLGI/rtm* to create a new runtime record for this LGI – among potentially other calls.)		
Responses	If Success HTTP 200 status code (OK)		
	If Error HTTP 304 status code (Not Modified)		
Parameters	**Name**	**URL pattern**	**Input Model**
	id* label	?id={lgi_id} &label={lg_name}	{}

URL pattern	**DELETE api/lg/deleteLGs**	
Method	DELETE	
Description	Delete (drop) all LGs under the LCS_LearningGraph collection in the MaTHiSiS DB	
Responses	If Success HTTP 200 status code (OK)	
	If Error HTTP 304 status code (Not Modified)	
Parameters	**Name**	**URL pattern**
	-	-

URL pattern	**DELETE api/lg/deleteLGIs**	
Method	DELETE	
Description	Delete (drop) all LGIs under the US_LearningGraphInstance collection in the MaTHiSiS DB. If optional parameter *uid* is passed, delete only those occurrences that apply to this learner.	
Responses	If Success HTTP 200 status code (OK)	
	If Error HTTP 304 status code (Not Modified)	
Parameters	**Name**	**URL pattern**

	uid	?uid={learner_id}

URL pattern	**DELETE api/lg/deleteLGIs/rtm**	
Method	DELETE	
Description	Delete (drop) all runtime LGIs under the US_LearningGraphInstance_rtm collection in the MaTHiSiS DB. If optional parameter *uid* is passed, delete only those occurrences that apply to this learner. If optional parameter *sid* is passed, delete only those occurrences that apply to this session. Parameters *uid* and *sid* can be passed simultaneously for a corresponding combination of restrictions.	
Responses	If Success HTTP 200 status code (OK)	
	If Error HTTP 304 status code (Not Modified)	
Parameters	**Name**	**URL pattern**
	uid	?uid={learner_id}
	sid	&sid={session_id}

3.4 Interface with the Front-end

Learning Graphs are at the core of the pedagogical methodology introduced in MaTHiSiS and play a central role in the definition of the goals and activities of a Learning Experience. They appear in several parts of the MaTHiSiS Front-end:

- In the Learning Content Manager (LCM), all the LGs created by MaTHiSiS users, stored in the Learning Graphs Repository (LGR), can be browsed, viewed and edited.
- The LCM is where new LGs can be created then published, and existing ones can be edited by tutors.
- Both tutors and learners can view LGs involved in their Learning Experience in the Learning Experience Supervisor.

3.4.1 LGs in the Learning Content Manager

The core functionality of the LCM is to give tutors the tools to create and edit MaTHiSiS-related content. This has been the main focus of the development for the first prototype of the LCM. Regarding LGs, it can be declined in three main goals:

- Provide a tool for creating and editing LGs, with Learning Goals and Smart Learning Atoms (SLAs);

- Ensure the compatibility of these tools with the defined LG data model;
- Establish the communication (read/write) in the LCM using the LG lib Open API for LGs.

During the content creation process in MaTHiSiS, the different building blocks (LG, SLA, LA, LAM, LM) will not necessarily be created by the same user. Having a single, unified edition tool for all of these elements would potentially result in a tool that is too complex and not easy to use. It has hence been decided to create separate and independent tools for each, while still ensuring these edition tools could still be connected together to allow for a quick navigation between them when working on different elements at once.

This thought process led to the definition of two additional goals for the LCM:

- Edition tools must be simple and focused on a single concept;
- Navigation between the different edition tools must be quick and easy, but not required at all times to create content.

The first step towards reaching these goals was to create UI mock-ups for the different LCM tools in order to further refine the list of functionalities required. The approach taken for designing the UI has been to work on mock-ups for all the tools (LG Editor, SLA Editor, LA Editor, LAM Editor, LM Editor) simultaneously to ensure that they all have the same look and feel and that similar functionalities in different tools would be implemented the same way in all the tools.

3.4.1.1 LG Editor

Figure 5 presents a screenshot of the UI mock-up for the LG Editor tool:

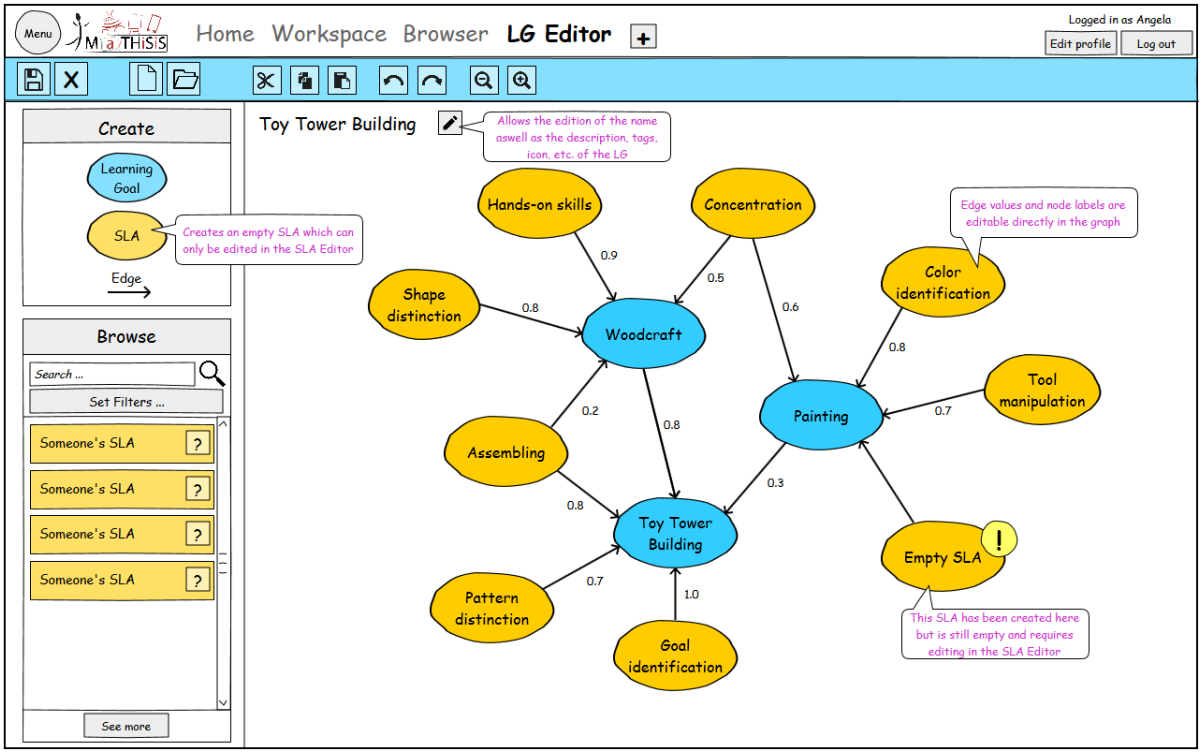

Figure 5: UI mock-up for the LG Editor

The Learning Graph is displayed as a graph with two different types of nodes:

- Learning Goals (here, in blue);
- Smart Learning Atoms (SLAs, here in orange).

The nodes are connected by oriented, weighted edges, representing the connections between two Learning Goals or between a Learning Goal and a SLA, the weight being the importance of a SLA or Learning Goal towards another Learning Goal. New edges can be drawn between two existing nodes, and their weight edited directly in the graph.

From the Create panel on the left, the user can create new Learning Goals and new, empty SLAs. There is no specific editor attached to Learning Goals as they are simply a label, which can be edited at will in the graph. Newly created SLAs in the graph can also be renamed, and all of them can be edited, which opens up a SLA Editor in a new tab. A SLA created in the LG Editor tool will have to be edited before it can be published on the MaTHiSiS cloud, but it felt important to let the user be able to completely define his LG without having to constantly swap to a SLA Editor to create new SLAs as s/he needs them.

The Browse panel allows the user to view and access the SLAs stored on the MaTHiSiS cloud, and to add existing ones (made by him/her or by other MaTHiSiS users) to his/her LG.

Based on this mock-up, an initial prototype of the LG Editor has been developed for the first version of the platform in the LCM native application, focusing on the core functionality: creating and editing LGs. Existing SLAs cannot be browsed and used yet, but it is already possible to completely define all the nodes and edges of the graph, and to load / save LGs from and to the MaTHiSiS cloud, using the LG lib Open API.

Figure 6 presents a screenshot of the current state of the LG Editor in the LCM:

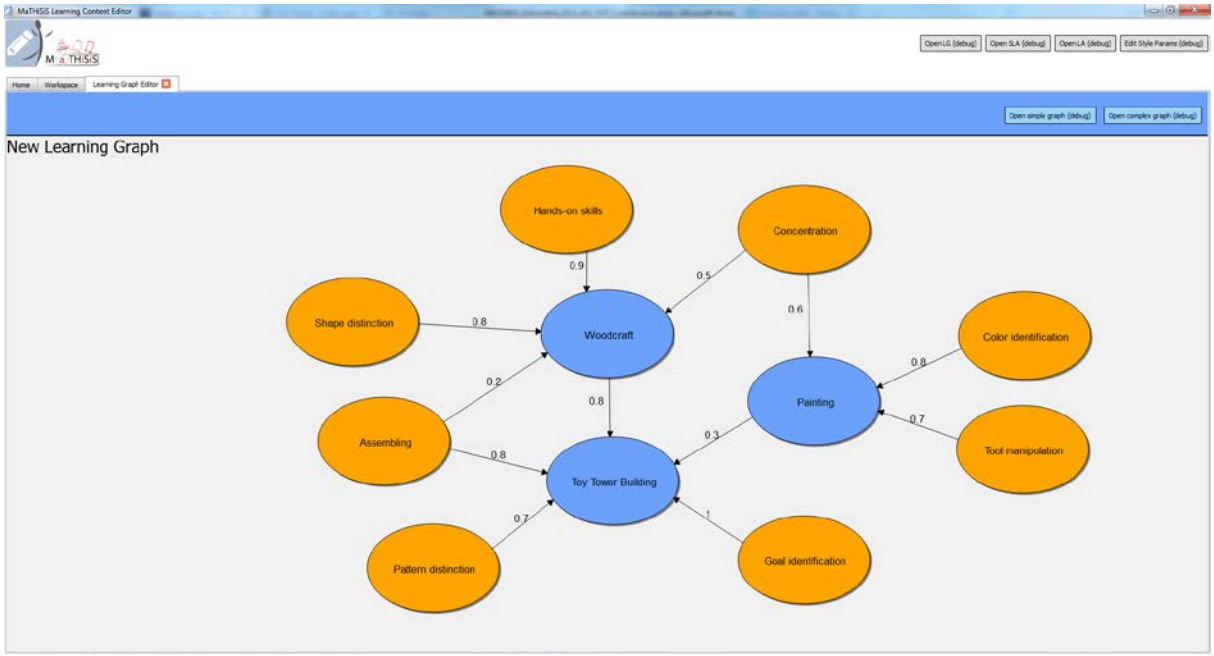

Figure 6: Current LG Editor

3.4.2 LGs in the Learning Experience Supervisor

In the Learning Experience Supervisor (LES), both tutors and learners can review their entire Learning Experiences, down to every single Learning Session. Tutors can also manage ongoing Learning Sessions for each of their learners. The Learning Graph is the central element of a Learning Experience, defining its main goals and how to achieve them. Once personalized for each learner, LGs also become an excellent way of tracking the learners' progress throughout the entire experience, both for the learner, the tutor and the MaTHiSiS decision support system.

As such, LGs must be clearly accessible during the entire Learning Experience, and the appropriate Learning Goals and SLAs of the LG must be mentioned whenever the user interacts with the MaTHiSiS platform. The following UI mock-up shows how the LES will display information about an ongoing Learning Experience. It provides a good example of Learning Goals and SLAs being highlighted for the user for each session of the experience:

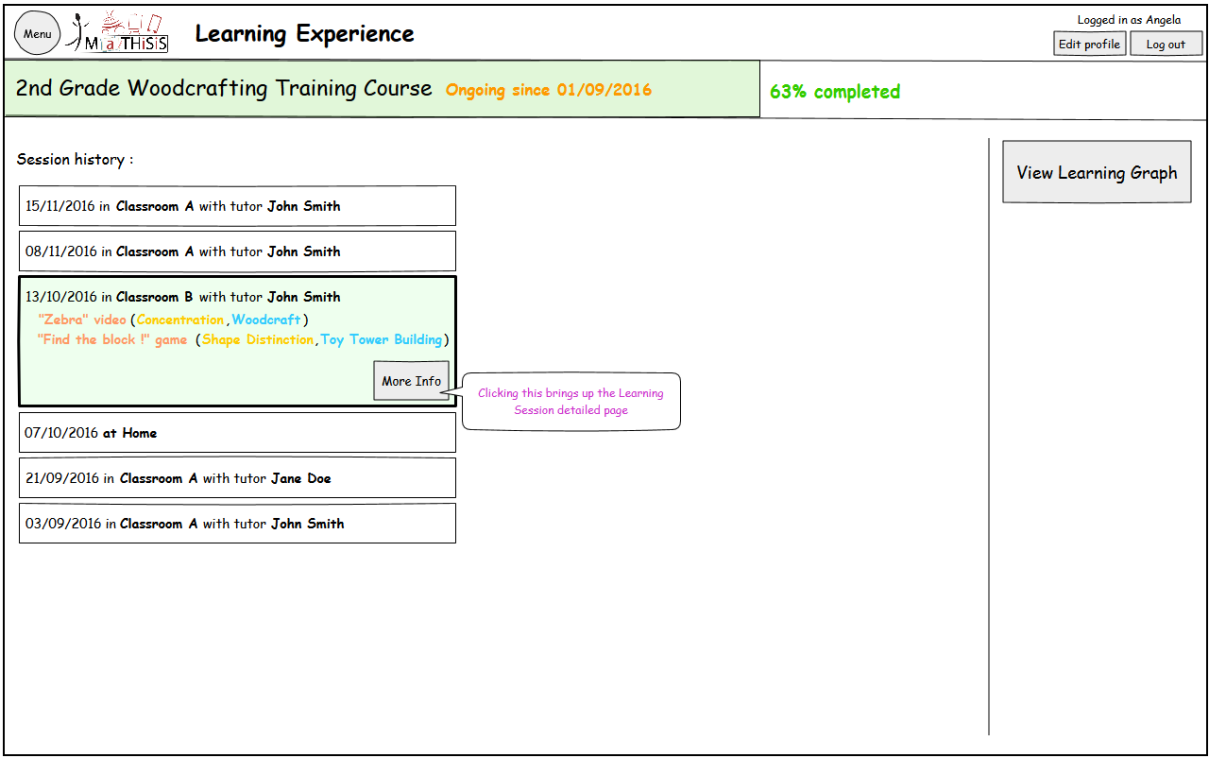

Figure 7: UI mock-up for the LES

3.4.2.1 Learning Experience Controller

In Figure 8 the UI mock-up for the Learning Experience Controller, part of the Learning Experience Supervisor.

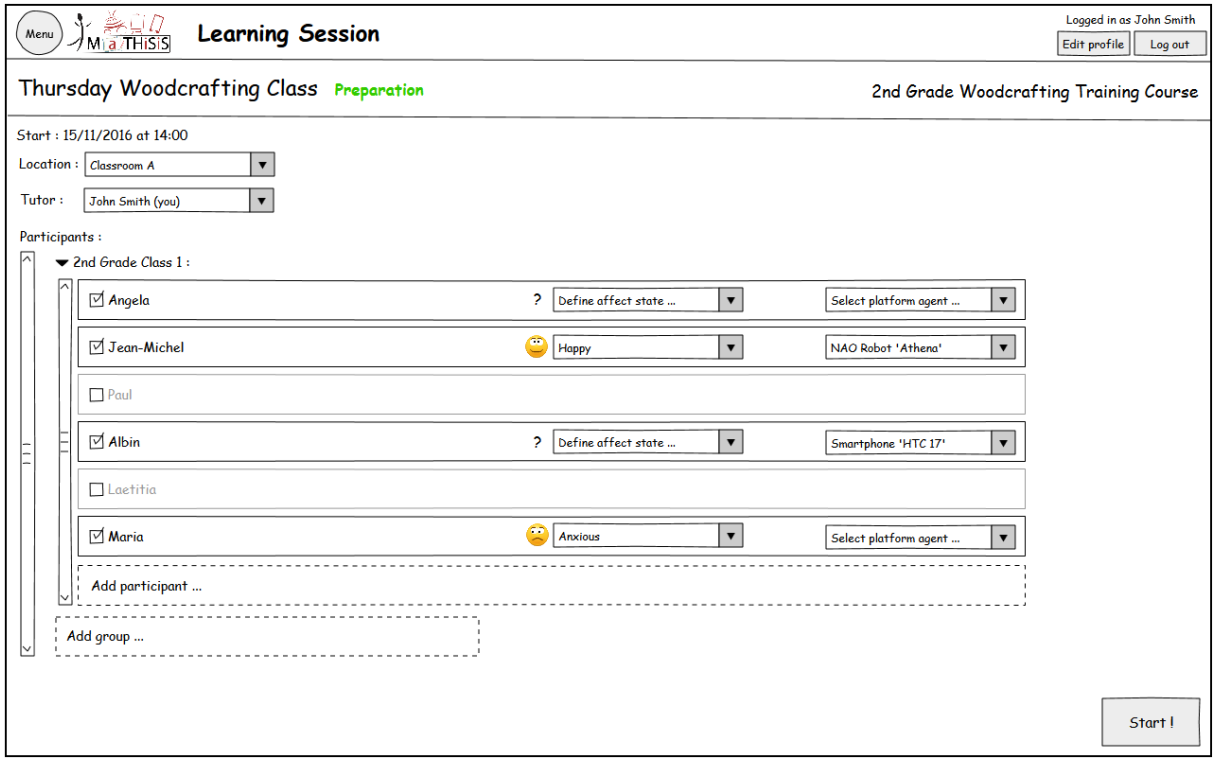

Figure 8: LEC UI mock-up - Set up of Learning Sessions

The Learning Experience Controller is displayed as a list of Learners, where a Tutor will manage the different Learning Session of each Learner. The Tutor must choose the location where the Learning Sessions will take place. He can also inform the initial affect state of the Learners so that they can start the Session with the most appropriate level of difficulty.

The Tutor can add to this list as many Learners as there are. He can add them one by one, picking in the available Learners, or add a group, which represent the population of a class for example.

For each Learner, a Platform Agent must be chosen. This is on those Platform Agent the materialization of the Learning Experience will take place.

On the bottom left-corner, the Start button is the trigger to launch the Learning Experiences of the Learners.

Figure 9 shows the UI mock-up for running Learning Sessions. There we will have the possibility to pause/resume/stop the Learning Session of a learner, and to provide a human feedback to the system to influence its decisions.

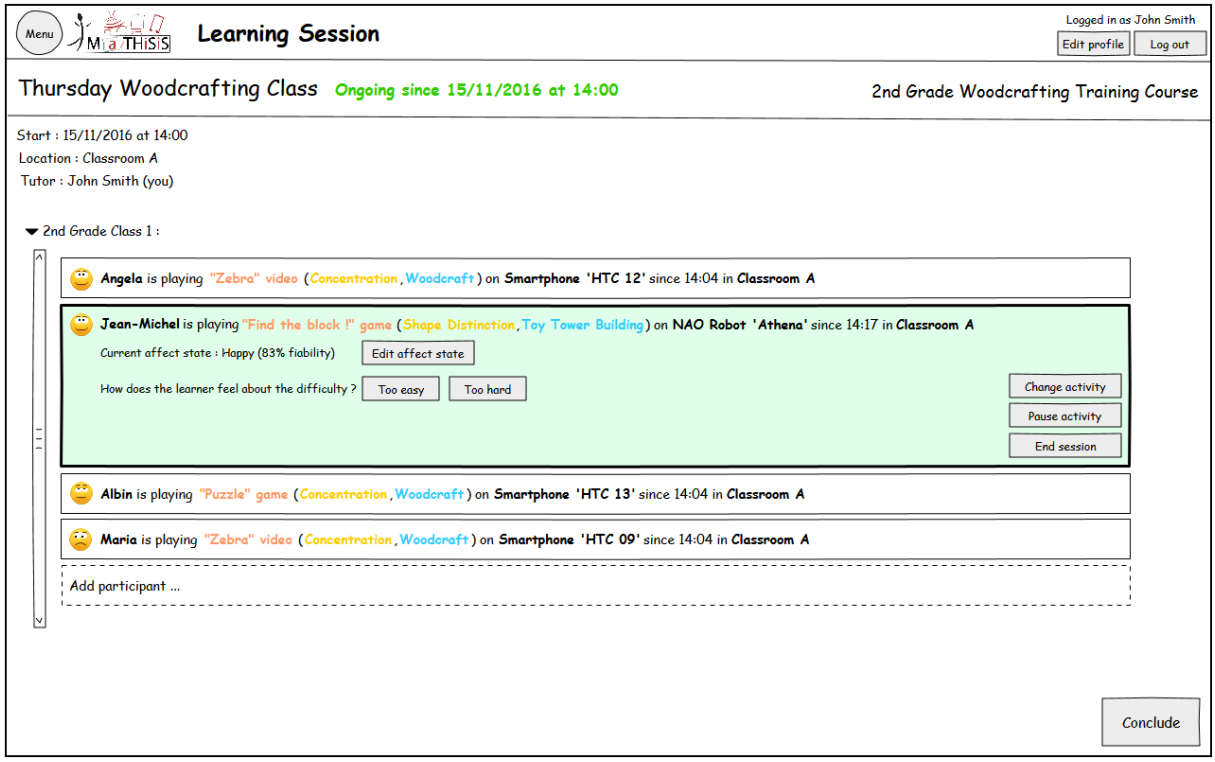

Figure 9: LEC UI mock-up - Running Learning Sessions

3.4.2.2 LGs in the Learning Experience Controller

Figure 10 presents the first version of the Learning Experience Controller in the LES, at M12:

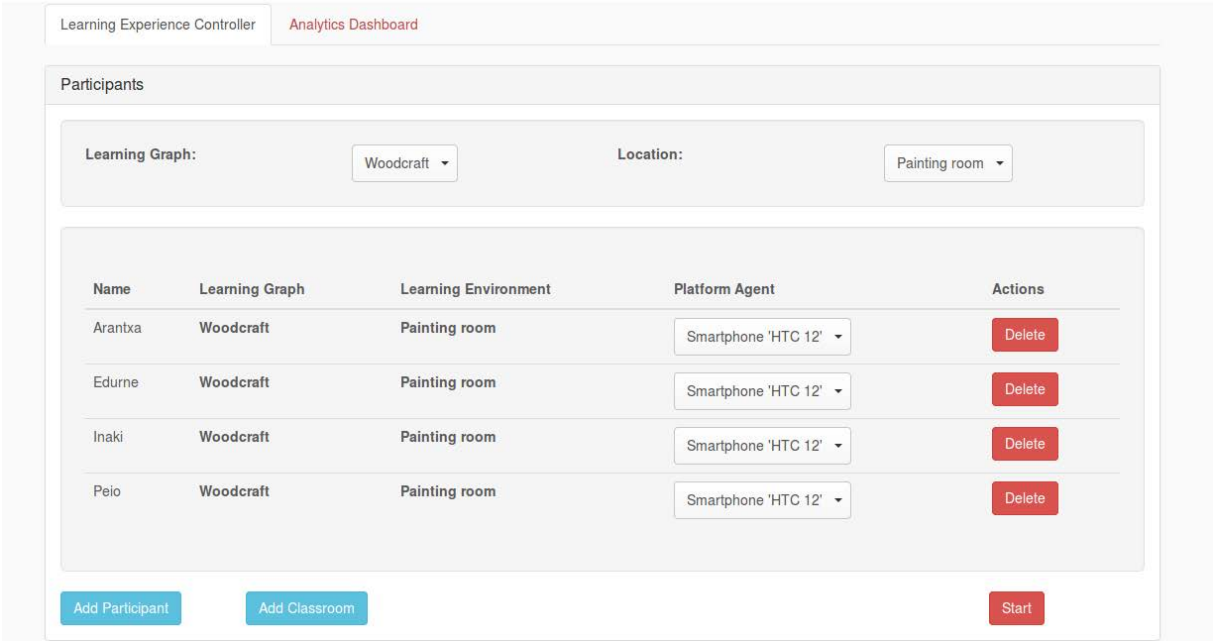

Figure 10: LEC first version - Set up of Learning Sessions

The main difference with the UI mock-up is the presence of the Learning Graph, which gives information about the Learning Experience the Learner is about to live. For the first version,

it has been decided to not include other options foreseen on the prototypes described above.

Figure 11 shows what the LEC presents to the users for running Learning Sessions. Like for the set up, we have only the Stop button available in the first version of this tool.
What is planned to do next, is to have another view that will be called the Running Learning Experience. The list of Learners with a Running Learning Experience will be displayed in this view. The difference with the Learning Experience Controller view is the presence of specific options to manage the Learning Experience (operations like start/pause/stop).

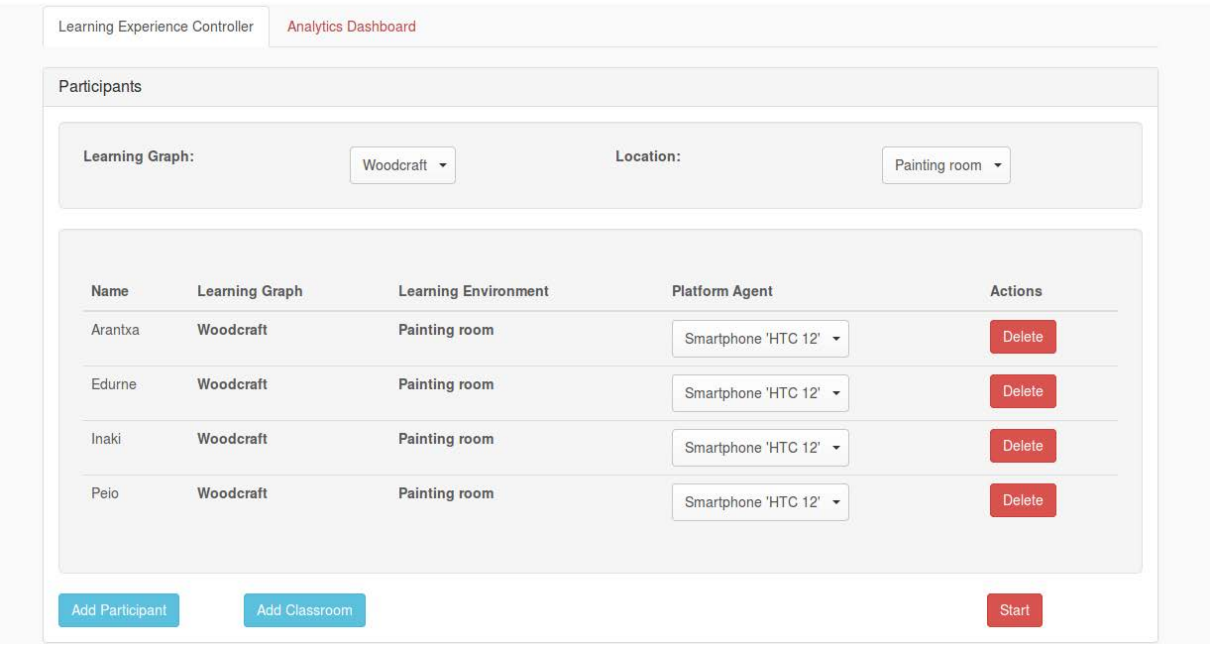

Figure 11: LEC first version - Running Learning Sessions

4. Conclusion

This document presented the novel, adaptable structure adopted within MaTHiSiS in order to organise the MaTHiSiS learning approach: the MaTHiSiS Learning Graphs (LGs). This structure ensures optimal, non-linear fulfilment of the MaTHiSiS learning scenaria.

The LGs comprise of two manifestations: a) a global, unpersonalised structure that represents the learning content (objectives) for all learners partaking in a specific learning scenario and the interrelations between them, which can be re-used by any learning process enabler (tutor/trainer or caregiver) using the MaTHiSiS platform and b) a learner-specific personalised structure, which instantiates the information captured in a corresponding unpersonalised LG, while at the same time enables the representation of particular learners' uptake over the learning content, for each iteration of the learning process.

The learner-specific representation of the learning content within personalised LGs will reflect both the long-term competence of the learner over the content during the entire learning process for each particular LG, as well as the short-term fluctuation of learner competences during each iteration of the learning process, according to her/his affective state and performance. This process will be handled by the MaTHiSiS Learning Graph Engine (LGE), which will adapt the personalised LG instance vertex weights per learner, according to the respective circumstances. The LGE will particularly exploit the graph structure of the LGs through spectral analysis of the LG instances. Spectral analysis will allow taking into account both the change in LG vertex weights as well as the importance of these vertices within the graph, in order to predict vertex weights that will optimally (more rapidly) bring the LG instance to its completion. This process will be further detailed in Deliverable D6.2 *The MaTHiSiS Learning Graph Engine* (due M24).

Furthermore, this document underlines the first set of functionalities developed for the first release of the MaTHiSiS platform, implemented within the LG library and integrated in MaTHiSiS platform through the corresponding Open API, which will enable the creation and use of the LG structures by dependent MaTHiSiS components, those being most prominently the Learning Content Editor (LCE), the Learning Experience Supervisor (LES), the Smart Learning Atoms (SLAs) and of course the Leaning Graph Engine (LGE). A detailed description of the relevant components of the LCE and LES that concern the use of LGs through the MaTHiSiS front-end interface is also presented in this dicument.

It is expected that the structures, implementation and integration details and relevant interface components will evolve in the future, depending on technical requirements and the feedback of end users. However the conceptual backbone, approach and outlined architecture will remain the same. The front-end will obviously evolve substantially following the MaTHiSiS driver pilots, since before them the platform lacks insight concerning concrete real-world experience of the use of this novel concept in actual learning environments and contexts.

The next version of this report, namely Deliverable D3.4 *The MaTHiSiS Learning Graphs* (due M24) will elaborate on the modifications and improvements that will pertain to this first implementation during the second year of the project, as new versions of the platform will be realised. In conclusion, the results of this document will be used as input in Deliverable D7.2 *MaTHiSiS platform, 1st release.*

5. References

[1] Nottingham Trent University (ed.): D.2.2 *Full Scenarios for all Use Cases*. Deliverable of the MATHISIS project, 2016.
[2] DIGINEXT (ed.): D2.3 *Full system architecture*. Deliverable of the MATHISIS project, 2016.
[3] Centre For Research and Technology Hellas (ed.): D3.3 *The MaTHiSiS Learning Graphs M12*. Deliverable of the MaTHiSiS project, 2017.
[4] DIGINEXT (ed.): D3.1 *The MaTHiSiS Smart Learning Atoms M12*. Deliverable of the MaTHiSiS project, 2017.
[5] DIGINEXT (ed.): D7.2 *MaTHiSiS platform, 1st release*. Deliverable of the MATHISIS project, 2017.
[6] MaTHiSiS Description of Action, 2016.