Managing Affective-learning THrough Intelligent atoms and Smart InteractionS

D2.4 Full System Architecture

Workpackage	WP2 - User, System and Ethics Requirements
Editor(s):	Alexandre AHMAD, DXT Andrew POMAZANSKYI, NG Carmen L. PADRÓN NÁPOLES, ATOS Christos ATHANASIADIS, UM Clément RODRIGUES-VIGUIER, DXT Dimitris KATSIKAS, CERTH Dorothea TSATSOU, CERTH Enrique HORTAL, UM Enrique QUIROS FERNANDEZ, ATOS Esam GHALEB, UM George AGAPIOU, OTE Isis PANZA-CARRIER, DXT Javier SERRANO, ATOS Miquel MILA PRAT, ATOS Nadia POLITOU, ATOS Nelly LELIGOU, OTE Nicholas VRETOS, CERTH Nick AGIOTIS, OTE Norbert BARICHARD, DXT

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

	Stelios ASTERIADIS, UM Thomas TECHENE, DXT
Responsible Partner:	DIGINEXT
Quality Reviewers	Nelly LELIGOU, OTE Vaggelis SPYROU, NCSR
Status-Version:	Final – v1
Date:	Project Start Date: 01/01/2016; Duration: 36 months Deliverable Due Date: 31/03/2017 Submission Date: 28/04/2017
EC Distribution:	PU
Abstract:	This document presents the final system architecture of the MaTHiSiS Platform. Each component of the architecture is described, using a top down approach. Implementation details of each component can be found in specific deliverable. This document describes the technical choices made in order to comply with the security constraints, the usability and the adaptability of the platform. This document is an updated version of the deliverable *D2.3 - Full system architecture* released in M6.
Keywords:	MaTHiSiS System, Architecture, Software, Hardware, definitions
Related Deliverable(s)	*D2.1 - Formation of stakeholder groups (M3)* *D2.2 - Full scenarios of all use cases (M9)* *D2.3 - Full system architecture (M6)* *D2.5 - Evaluation Strategy (M9)* *D2.6 - Framework for impact assessment of MaTHiSiS against LEPOSA requirements (M6)* *D2.7 - The impact assessment report (M12)* *D3.1 - The MaTHiSiS Smart Learning Atoms (M12)* *D3.3 - The MaTHiSiS Learning Graphs (M12)* *D3.5 - Experience Engine (M12)* *D3.7 - Learner's Profile Repository (M12)* *D4.1 - MaTHiSiS Sensorial Component (M12)* *D5.1 - Description of the robotic layer (M12)* *D5.4 - Description of the mobile layer (M12)* *D5.7 - Description of the interactive whiteboards layer (M12)* *D6.1 - Adaptation and Personalization principles based on MaTHiSiS findings (M12)*

	D7.1 - Integration Strategy and planning (M6) *D7.2 - MaTHiSiS platform, 1st release (M12)*

Document History

Version	Date	Change editors	Changes
0.1	24/02/2017	Clément RODRIGUES-VIGUIER, DXT Carmen L. PADRÓN NÁPOLES, ATOS	Initial version derived from the D2.3
0.2	01/03/2017	Clément RODRIGUES-VIGUIER, DXT	Updated some sections managed by DXT
0.3	22/03/2017	Clément RODRIGUES-VIGUIER, DXT Enrique HORTAL, UM	Integration of DXT contributions Integration of UM contributions
0.4	29/03/2017	Clément RODRIGUES-VIGUIER, DXT Thomas TECHENE, DXT	Integration of DXT contributions Integration of UM contributions Integration of CERTH contributions Small adjustments from OTE comments
0.5	31/03/2017	Clément RODRIGUES-VIGUIER, DXT Isis PANZA-CARRIER, DXT	Integration of DXT contributions Consistency pass Merge of some sections to simplify the document
0.6	07/04/2017	Clément RODRIGUES-VIGUIER, DXT Isis PANZA-CARRIER, DXT Thomas TECHENE, DXT Enrique HORTAL, UM Nadia POLITOU, ATOS Miquel MILA PRAT, ATOS Enrique QUIROS FERNANDEZ, ATOS Vilma FERRARI, IMOTEC Nelly LELIGOU, OTE	Integration of DXT contributions Integration of UM contributions Integration of ATOS contributions Integration of IMOTEC contributions Integration of OTE contributions
0.7	12/04/2017	Clément RODRIGUES-VIGUIER, DXT	Integration of OTE contributions Text of the Section 3.4.1 and 3.4.2

Version	Date	Change editors	Changes
		Nelly LELIGOU, OTE	Consistency pass
0.8	14/04/2017	Clément RODRIGUES-VIGUIER, DXT Enrique HORTAL, UM	Integration of DXT contributions Integration of UM contributions
0.9	14/04/2017	Thomas TECHENE, DXT Dimitris KATSIKAS, CERTH Dorothea TSATSOU, CERTH Nadia POLITOU, ATOS Nelly LELIGOU, OTE	Integration of CERTH contributions Integration of ATOS contributions Integration of OTE contributions
0.9.1	26/04/2017	Clément RODRIGUES-VIGUIER, DXT Vaggelis SPYROU, NCSR Nelly LELIGOU, OTE	Updates after peer-reviewing
0.9.2	28/04/2017	Nadia POLITOU, Ana Piñuela, ATOS	Final quality check
FINAL	28/04/2017	Clément RODRIGUES-VIGUIER, DXT	FINAL VERSION TO BE SUBMITTED

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
AIR	Affect and Intent Recognition
API	Application Programming Interface
ASC	Autism Spectrum
CGLDC	Career Guidance Distance Learning Case
CLS	Cloud-based Learner's Space
DMZ	Dematerialized Zone
DSS	Decision Support System
EE	Experience Engine
I/O	Input / Output
IaaS	Infrastructure-as-a-Service
ITC	Industrial Training Case
IWB	Interactive Whiteboard
KPI	Key Performance Indicator
LA	Learning Action
LAO	Learning Actions Ontology
LEPOSA	Personal data protection, privacy, ethics and social acceptance framework
LGE	Learning Graph Engine
LGI	Learning Graphs Implementation
LM	Learning Material
LPR	Learner's Profile Repository
LRS	Learning Record Store
MEC	Mainstream Education Case
MF	MaTHiSiS Front-end
NAT	Network Address Translation

Abbreviation / acronym	Description
OS	Operating System (e.g. Linux, Windows, MacOS, iOS, Android, etc.)
PA	Platform Agents
PMLDC	Profound and Multiple Learning Disabilities Case
SC	Sensorial Component
SLA	Smart Learning Atom
US	User Space
UAV	User Authentication / Validation
VLAN	Virtual Local Area Network
WAF	Web Application Firewall

Table 1 - Definitions, Acronyms and Abbreviations

Project Description

MATHISIS is a 36 month duration project co-funded by the European Commission Horizon 2020 Programme (H2020-ICT-2015) under Grant Agreement No. 687772. It started on 1st January 2016.

One of the core objectives of MaTHiSiS project is to enhance learning environments and make use of computing devices in learning in a more interactive way, which will provide a product-system to be used in formal, non-formal and informal education. An ecosystem for assisting learners/tutors/caregivers for both regular learners and learners with special needs will be introduced and validated in 5 use cases: Autism Spectrum Case (ASC), Profound and Multiple Learning Disabilities Case (PMLDC), Mainstream Education Case (MEC), Industrial Training Case (ITC) and Career Guidance Distance Learning Case (CGDLC).

MaTHiSiS product-system consists of an integrated platform, along with a set of re-usable learning components (educational material, digital educational artefacts, etc.), which will respond to the needs of a future educational framework, and provide capabilities for: i) adaptive learning, ii) automatic feedback, iii) automatic assessment of learner's progress and behavioural state, iv) affective learning and v) game-based learning.

Within MaTHiSiS, an innovative structural tool of learning graphs is going to be introduced to guide the learner through the process of learning in the given scenario. To reach a learning objective, learner will have to "follow the path" of the learning graphs, built up on Smart Learning Atoms, which are certain learning elements that carry defined learning materials.

To ensure barrier free integration in the market, MaTHiSiS makes use of a range of interaction devices, such as specialized robots, mobile devices and interactive whiteboards. The consortium ensures easy-to-use solution with e.g. specialized graphical editor-like tool, allowing to easily create educational materials as well as the reusability within both mainstream education and vocational training setups.

Objectives of the project

A Cloud-based Learner's Space (CLS) will be developed to provide a system for adaptation/personalization in learning, interaction, data acquisition and analysis as well as content creation on the fly. This is a core component of the MaTHiSiS system which includes 3 crucial subsystems which create an innovative smart learning ecosystem: i) the experience engine, a graph-based interactive storytelling engine, that manipulates interactive content that is later sent to a device of tutor's/learner's choice; ii) the learning graph engine, responsible for adaptation of the Learning Graph based on learner's behaviour and interaction; iii) the Decision Support System (DSS) providing and collecting learning analytics and controlling synchronous and asynchronous interaction between devices. To ensure constant educational flow and augmented learner engagement, the emotion recognition and context aware cognitive/behavioural status extraction tools are introduced within the system addressed by the Sensorial Component.

For the purpose of validating MaTHiSiS approaches in learning environment, a set of Smart Learning Atoms (SLA) is going to be created for defined use cases. Such SLAs will adapt to each learner in a different way based on her/his particular needs, profile, cognitive affective state, relevance to specific learning requirements and previous performance. Further, an editor-like tool is introduced to be able to transform educational material into MaTHiSiS Learning Materials usable by SLAs through Learning Actions. The learning graphs then are going to be deployed to interact with the Cloud-based Learner's Space (CLS) as well as some front-end tools for tutors and caregivers to enable creation, editing and authoring of the learning contents and Learning Experiences.

MaTHiSiS will support learning across a variety of learning contexts and, with the use of a variety of devices (robots, interactive whiteboards, mobile devices and desktop/laptop computers), with personalized and adaptable, time and location independent learning paths, being transferred between the agents, always taking into consideration best knowledge and practices learnt from the previous device.

By the end of the project, MaTHiSiS will introduce a marketable innovation, aimed at the re-usability of educational and training content and fostering the interactivity between technology and learners/tutors/caregivers.

Executive Summary

This document presents the full system architecture picture of the MaTHiSiS platform after nearly one year of development. The architecture is divided into 3 main layers, namely the Back-end layer (Section 3.4.4.2), the Front-end layer (Section 5) and the Platform Agent layer (Section 6). This document details their objectives and describes the technical solutions, ranging from the hierarchy of sub-components to the definition of the communication channels, protocols and interfaces, staying at a high-level of description. For details, we refer to dedicated technical deliverables.

This full system architecture report is the result of an iterative process. The content below is being led from a list of end-user requirements, gathered and described in ***D2.2 - Full scenarios of all use cases***[2]. The objective is to tailor the architecture and the vision of the system to the end-user and market's needs.

In the extend of iterative process, this document is in addition related to the ***D7.1 - Integration strategy and planning***[16] deliverable. The latter introduces in detail the iterative development and validation methodology used in the MaTHiSiS project, i.e. the Agile methodology. This methodology guides the overall project, for example end-user requirements are translated into *user-stories*, which are the main inputs to define the full system architecture described below.

In addition to the innovative features defining the heart of the MaTHiSiS project, such as the Learning Experience adaptation and personalization, learner's affect state extraction, re-usability of learning contents, etc., this document introduces technical solutions to identify security issues. Indeed, the security aspects of the MaTHiSiS platform are covered in details in the complementary ***D2.6 -Framework for impact assessment of MaTHiSiS against LEPOSA requirements***[5] document.

Finally, since this report defines the overall technical picture of the MaTHiSiS platform, including the architecture within each work package but also the technical integration scheme of the work package results, this report serves as a reference document for all technical partners.

This deliverable is an updated version of the deliverable *D2.3 - full system architecture* [3] delivered in M6. Lots of adjustments have been made nearly in all sections, but here are the major updates:

- **Business Processes have been updated and detailed;**
- **Authentication mechanism is described;**
- **Full description of the entities of the data models;**
- **Sections updated in order to take into account the improvements and work done so far on the architecture, mainly in the Platform Agent Layer;**
- **Simplifications to improve the readability and clarity.**

1. Introduction

One of the core objectives of MaTHiSiS project is to enhance learning environments and make use of computing devices in education and training in a more interactive way, which will provide a product-system to be used in formal, non-formal and informal education. This is possible thanks to the use of state-of-the-art technologies from various fields, from computer vision to artificial intelligence. The development of such a system, the MaTHiSiS platform, requires experts from different organisations collaborating together towards a shared objective.

This document presents the full system architecture of the MaTHiSiS platform, after one year of intense development. In this document the architecture defining the technical components within each identified work package is overviewed, while the means to integrate the research results to create a unique platform, which are the definition of the communication channels, protocols and interfaces, are detailed. In this scope, and staying at a high-level of description, a global picture of the system architecture is depicted.

The full system architecture is being led from end-user requirements and also technical requirements. For example, the challenging affect state extraction requires a good level of quality of signals, depending on algorithms. On the other hand, this quality may not be reachable on all devices, or, worse, sensors providing such signals may not be present. All these aspects are managed. Furthermore, the Learning Experience adaptation and personalization mechanism deals with legal and security aspects which are the spine of this proposed architecture.

A top-down approach is followed in order to simplify the introduction of all concepts and components of the MaTHiSiS platform. This is achieved by providing a high-level view of the full system architecture, dealing with the security aspects, defining and describing all sub-components recursively.

The full system architecture is divided into 3 main layers, namely the Front-End layer, the Back-end layer and the Platform agent layer. A definition of the methodology used and an overview of the architecture are presented in Sections 2 and 3. Section 3.4.4.2 describes the Back-end layer, while the Front-end layer and the Platform agent layers are detailed in Sections 5 and 6 respectively. Moreover, Section 7 describes the knowledge and data models as known in M15. Technical reports provide more details and update on all of these aspects of the architecture. Finally, Section 8 gives the list of standards that are used in M15 by the MaTHiSiS platform. A conclusion is given in section 9 and references and annexes are visible in Sections 10 and 11 respectively.

2. Definitions and Methodology

2.1 Definitions

The following definitions describe the basic concepts and artefacts within the MaTHiSiS learning eco-system. They are all coming from ***D3.1 - The MaTHiSiS Smart Learning***[7], ***D3.3 -The MaTHiSiS Learning Graph***[8] or ***D3.5 - Experience*** **Engine**[9] deliverables but we present them here for convenience.

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.

Smart Learning Atoms (SLAs) are atomic and complete pieces of knowledge, competence and/or skills, which can be learned and assessed in a single, short-term learning process iteration from a learner. 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 an 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.

Learning actions (LAs) are *precise learning activities* to be deployed in the real world, *which each platform agent (PA) interprets in different ways and based on the learning materials available in different learning settings*, which makes them PA-agnostic, LM-independent and context/setting-independent. Learning actions stimulate and convey the learning process for one or more specific pieces of simple knowledge/skills (SLAs) to the learner. In a nutshell, LAs pertain to **what-to-do** in order to learn specific things.

Learning Action Materialisations (LAMs) are exactly the *PA-specific and/or Learning Material-specific interpretations of Learning Actions for different learning settings* (e.g. classroom, factory, etc.). In a nutshell, LAMs pertain to **how-to-do** what needs to be done in order to learn specific things.

Learning materials (LMs) consist of *specific digital and/or physical resources and artefacts* that are involved in the materialisation of a learning action (LAM).

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"[1]. Within MaTHiSiS, *a learning scenario ensures the appropriation of the knowledge, competence and/or skills encapsulated in specific Learning Graph*.

Learning Experience "refers to any interaction, course, program, or other experience in which learning takes place, whether it occurs in traditional academic settings (schools, classrooms) or non-traditional settings (outside-of-school locations, outdoor environments), or whether it includes traditional educational interactions (students learning from teachers and professors) or non-traditional interactions (students

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

learning through games and interactive software applications)"[2]. In the context of MaTHiSiS, it *refers to learning/training that takes place based on a specific MaTHiSiS-induced learning scenario*.

A **Learning Session** consists of *one iteration of the Learning Experience during the actuation of a learning scenario*. It starts from the time a learner logs in and starts interacting with the MaTHiSiS ecosystem until the time she/he logs out of the system.

2.2 Methodology

MaTHiSiS is combining a mixture of methodologies, tailored to the project´s needs.

For the purposes of the elicitation of the complex user requirements for the specification of MaTHiSiS, the User-centred design[3] methodology has been selected. In MaTHiSiS, we are interested in the guidance that USERfit[4] gives in terms of specifying the user requirements arising from the five Use Cases, so that the MaTHiSiS architecture can be tuned to these requirements.

All participants in WP2 followed the guidance of a precursor to USERfit - the RESPECT Handbook that highlights a series of steps when first identifying user requirements for a new system. In MaTHiSiS this methodology was extended to include techniques derived from context of use analysis and requirement prioritisation (e.g. MoSCoW). More information on this methodology and the user requirements elicitation is included in ***D2.2 - Full scenarios of all use cases***[2].

For the development of the MaTHiSiS platform, the consortium has adopted a well-known innovation friendly approach of software development, namely Agile. The Agile Manifesto[5] which was first declared in 2001 has a set of principles consisting mainly of the following:

- **Individuals and interactions** over processes and tools.
- **Working software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation.
- **Responding to change** over following a plan.

The manifesto also outlines twelve principles that elaborate on the values and give some more specific guidelines to follow. They may be grouped into three categories:

1. **Delivering working software**
    a. Principle #1: Highest priority is to satisfy the customer through early and continuous delivery of valuable software – need to ask client what they consider as valuable
    b. Principle #2: Deliver working software from a couple of weeks to a couple of months with a preference to the shorter timescale – the more frequent software deliveries the more opportunities for the client to provide input.
    c. Principle #3: Working software is the primary measure of progress – since the team will be focused on delivering working software, progress should be measured by that.
2. **Flexible design and adapting to change**
    a. Principle #4: Welcome changing requirements even late in development. Agile processes harness change for the product´s competitive advantage.
    b. Principle #5: Continuous attention to technical excellence and good design enhances agility. Having readable, simple code, and more flexible designs, will allow changes to be easily implemented.

[2] http://edglossary.org/learning-experience/
[3] User-Centered Design: http://www.e-learning.co.il/home/pdf/4.pdf
[4] USERfit : http://www.education.edean.org/index.php?row=3&filters=f16&cardIndex=21
[5] The Agile Manifesto: http://www.agilemanifesto.org/

   c. Principle #6: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
   d. Principle #7: Simplicity, the art of maximizing the amount of work not done, is essential. Focus on delivering what is essential, and reducing unnecessary work by writing less code and documentation, and focusing on delivering a high impact product that is as simple as it can be.

3. **Collaborative communication and organisation**
   a. Principle #8: Build projects around motivated individuals. Give them the environment and support that they need, and trust them to get the job done.
   b. Principle #9: The best architectures, requirements, and designs emerge from self-organizing teams, including things such as assigning tasks and choosing tools to use.
   c. Principle #10: Business people and developers must work together daily throughout the project.
   d. Principle #11: The most efficient and effective method for conveying information to and within a development team is face-to-face conversation.
   e. Principle #12: At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Agile primarily helps into delivering valuable software, on time. This can be achieved by collaborating with the end users, focusing on delivering a functional product, interacting between the people involved and by adopting change positively.

The MaTHiSiS consortium has adopted Scrum[6] methodology, as a lightweight framework based on the Agile Manifesto values. The rationale behind this selection is the complexity of MaTHiSiS as a project which requirements might be changing according to the trials results and feedback. Thus, to ensure that those are adopted in the final MaTHiSiS product it is important to adopt an iterative and incremental development, with prioritized functionalities, continuous synchronization and easy communication between all needed parties. Scrum is characterized by the following three pillars:

- **Transparency** – everyone can see every part of the product. It is essential to agree on common standards and terminology from the beginning of the process and all interesting should have clear understanding of what functionality needs to be completed / supported in each task.
- **Inspection** – frequent inspections should be done in order to point out all flaws early in the development process. The inspections should be done frequently, but not that frequent as to get in the way of software development.
- **Adaptation** - if someone points out that the product development starts to stray from the vision, the team must adjust and adapt to prevent further deviation.

The three pillars of Scrum, as outlined above, are supported by the following four specific techniques:

- **Sprint planning** occurring at the beginning of each Sprint.
- **Regular Scrum meeting**, a short meeting for the development team to synchronize its activities and inspect the progress. During the meeting, the team members should outline:
  - What did I do since the previous meeting?
  - What will I do until the next meeting?
  - Is there any impediment that prevents me from completing my tasks?
- **Sprint Review** held at the end of each sprint to review what the team has done during the sprint and adjust the product backlog if necessary.

[6]Scrum: http://scrummethodology.com/

- **Sprint Retrospective** held at the end of each Sprint and before the next one. Its purpose is to inspect how the team performed and create a plan for improvements for the next sprint. Usually the team discusses on what went well, what went wrong and what could be done to improve it.

The Scrum approach and how MaTHiSiS has adopted it during development and integration was detailed in ***D7.1 - Integration strategy and planning***[16].

D2.1 - Formation of stakeholder groups[1] started with the definition of the Personae for the different types of learners of MaTHiSiS. Those Personae were expanded to cover all types of roles identified as potential end users of MaTHiSiS. The mapping between the MaTHiSiS Roles listed in Section 3.3 and the Personas is performed in ***D2.2 - Full scenarios of all use cases***[2].

Since User Stories are an important part of the Agile approach and serve as the bases for development with the Scrum methodology, the set of User Stores is listed in Section 9 and the full set of User Stories for MaTHiSiS compiled by the technical partners in ***D2.2 - Full Scenarios for all Use Cases***[2] in order to define the core functionality of the platform and its components. The user stories defined in above mentioned documents are base for a high level release planning that is carried out in the WP7. All user stories are assigned to the different platform releases ensuring that all features are developed up to the beta version of the platform. The user stories as well serves for a more fine-grained plan through the MaTHiSiS product backlog, with the identification of more detailed tasks, responsibilities, duration, dependencies, etc. through which the consortium is tracking progress throughout development and integration in the WP7.

The user stories defined in above mentioned documents are the base for a high level release planning that is carried out in the WP7. All user stories are assigned to the different platform releases ensuring that all features are developed up to the beta version of the platform. The user stories as well serve for a more fine-grained plan through the MaTHiSiS product backlog on Gitlab, with the identification of more detailed tasks, responsibilities, duration, dependencies, etc.

3. MaTHiSiS Architecture

3.1 General overview and description of the high-level architecture

The MaTHiSiS ecosystem consists of three (3) distinct architectural layers, the Back-end Layer, the Front-end Layer and the Platform Agent Layer, as illustrated in Figure 1. The MaTHiSiS architecture is *n*-tier, where each layer contains multiple tiers that are organized into working spaces.

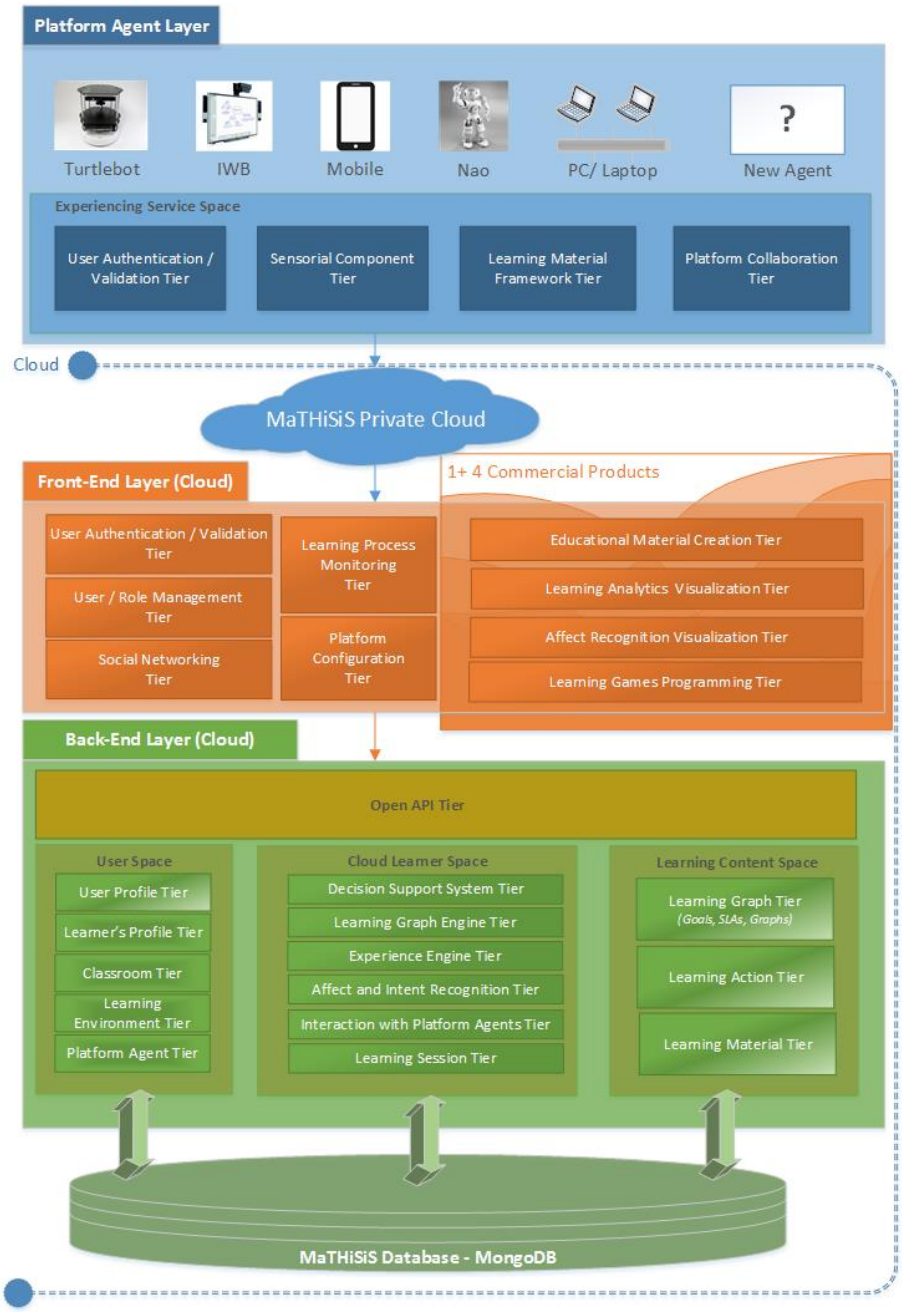


Figure 1 - MaTHiSiS high-level architecture

3.1.1 Platform Agent Layer

The **Platform Agent Layer** encapsulates all the tiers and underlying components residing in each supported platform agent (Nao robot, TurtleBot, interactive whiteboard, mobile, desktop/laptop). Given that the MaTHiSiS is platform agent agnostic by design and by default, the overall architecture takes under consideration the incorporation of new agents in a seamless way.

The tiers within the Platform Agent Layer are grouped into a single space, called Experiencing Service Space. This space is responsible for providing input related to the affect state of the learner, recording the interactions with the agents and materializing the learning actions on each agent, taking into account the unique characteristics of each agent (e.g. motion, display capabilities, interaction level, processing power, etc.).

The Platform Agent Layer is the only layer within the MaTHiSiS ecosystem that resides outside the MaTHiSiS cloud.

3.1.2 Front-End Layer

The **Front-End Layer** encapsulates all the tiers that are responsible for delivering the Human-Computer-Interaction User Interfaces of the MaTHiSiS ecosystem. The main objective of the Front-End Layer is to provide high quality User Experience (UX) interfaces for every single role within the platform (Learner, Tutor, Caregiver and Administrator).

The single entry point to the Front-End Layer is **the User Authentication and Validation Tier**, which will be responsible for performing Single-Sign-On operations and validate the identity of each user. The same Tier will act as a Reverse Proxy in terms of forwarding requests from the Platform Agent Layer to the Back-End Layer, while at the same time preserving the state of the requests to the RESTful[7] API.

The User / Role Management Tier encapsulates all necessary interfaces in order to provide management operations such as the New User Registration, the Role Assignment, the definition of access level and permissions, the definition of supervising users, etc.

The Social Networking Tier is responsible for materializing the integration with some of the most popular social networks in terms of authentication and interaction. The same Tier will carry out operations concerning the user account merging (local account and social networks accounts), in order to identify MaTHiSiS users uniquely and facilitate the interactions according to the specific learning goals.

The Learning Process Monitoring Tier provides a visual toolset for tracking the actual progress and performance of learners during Learning Sessions in real time. In addition, this tier supports a wide range of UI for supervision activities, giving the opportunity to the tutor, the caregiver or even the learner him/herself to adjust specific properties of the Learning Session and refine the total Learning Experience.

The Platform Configuration Tier provides a typical set of user interfaces for managing and configuring specific parameters of the system such as the integration of the available platform agents in a MaTHiSiS system instance, the email server properties for notification purposes, the selection of alternative data encryption techniques (i.e. field level, database level), etc.

There is a set of +4 Commercial products that encapsulates and shares the different MaTHiSiS functionality tiers and represents the final MaTHiSiS outcomes which can be eventually exported as standalone products.

The Educational Material Creation Tier is part of the +4 Commercial Products and responsible for delivering high-end user interfaces for managing the Learning Content within MaTHiSiS ecosystem. This

[7] RESTful definition: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

User Interface is at the core of the platform and encapsulates all the tools that a Tutor / Stakeholder needs in order to create rich and efficient Learning Sessions, incorporating self-explaining interfaces for Learning Graph (goals and SLAs) and Learning Material authoring.

The Learning Analytics Visualization Tier is part of the +4 Commercial Products and provides sophisticated reporting on the usage and learning performance using the MaTHiSiS ecosystem for all the different user roles and the diverse user cases, exploring Learning Experience metadata that will be stored by the system and visualized via specific metrics and dimensions.

The Affect Recognition Visualization Tier is part of the +4 Commercial Products and provides an autonomous end-to-end set of User Interfaces, from the collection of cognitive and/or physical state information (face, gaze, skeleton, speech and inertial sensor data from mobiles) to the decision making on the affective state of the learner.

The Learning Games Programming Tier is part of the +4 Commercial Products and responsible for the provision of a user interface toolset for authoring rich and interactive Learning Materials in form of learning games, following the specifications of the MaTHiSiS ecosystem.

3.1.3 Back-End Layer

The Back-End Layer is the backbone of the MaTHiSiS ecosystem and incorporates all necessary components and mechanisms for providing support to the deployment of different learning scenarios according to the MaTHiSiS overall concept[8].

The single point of entry to this layer is the Open API, consisting of a rich and powerful set of RESTful API calls to almost all major back-end activities. In addition, some efficient channels of communication exists between the Back-End layer and the Platform Agent layer, using the WebSocket protocol[9], in order to reach the ambitious real-time remote monitoring and operating of the platform agents by the MaTHiSiS platform.

The Back-End Layer encapsulates three distinct spaces which are:

- The User Space containing the User Profile Tier and the Learner Profile Tier. The main concept behind this is that the MaTHiSiS ecosystem is a multi-role system which means that a user can play different roles when using the system: he/she can be potentially a learner for a specific Learning Session (i.e. industrial case) and a tutor in another learning scenario (i.e. mainstream education). For this case, the system is designed to retain only the minimum set of data in accordance to the potentially different roles of a user.
- The Cloud Learner Space is the core component of the MaTHiSiS ecosystem, implementing and customizing mature research outputs to provide innovative modules to support the implementation of adaptive learning, non-linear, ubiquitous, affective, game-based and micro learning.
- The Learning Content Space contains all necessary structures and mechanisms for creating, serializing and storing non-personalized Learning Graphs, Learning Actions and Learning Materials.

3.2 MaTHiSiS Topology

The main objective of the following MaTHiSiS topology diagram is to provide a self-explaining visualization of the distinct networks and basic network elements as well as the infrastructure-as-a-service that will

[8] MaTHiSiS overall concept and Learning approach, [18] MaTHiSiS DoA part B pages 12 and 14
[9] Websocket protocol: https://tools.ietf.org/html/rfc6455

actually host and serve the MaTHiSiS ecosystem. In addition, the topology diagram demonstrates the implementation of measures which adhere to the principles of data protection by design and by default.

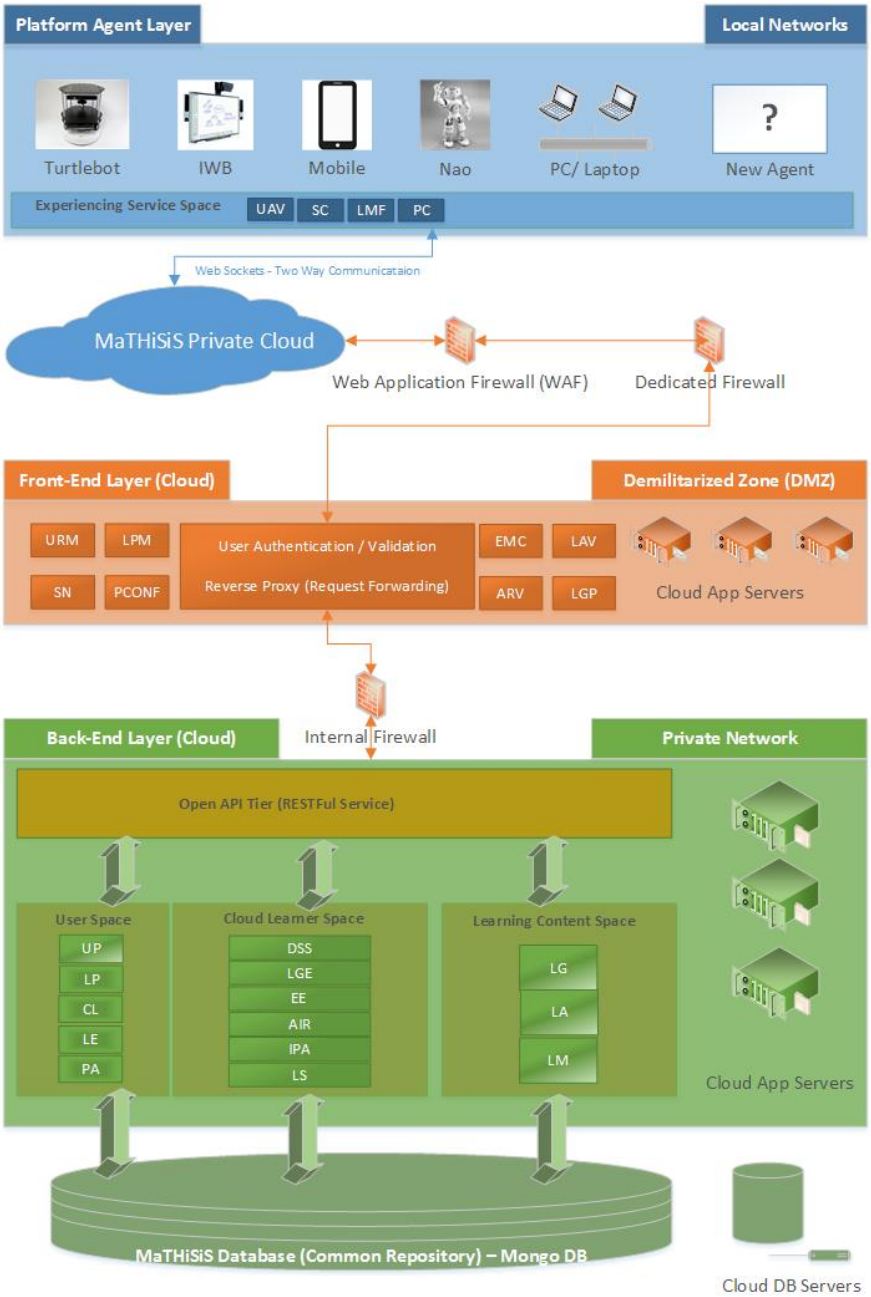


Figure 2 - MaTHiSiS Topology

The MaTHiSiS Ecosystem involves three interacting networks described in sections below.

3.2.1 Local Networks

These are the networks were the platform agents live and are physically located on a school or a training centre for example. The security of the local networks is out of the project's scope. Despite of this, the software components capturing and transmitting personal data (i.e. Sensorial Component) will address the following security considerations:

1. Personal and Sensitive Data encryption
2. Personal and Sensitive Data secure transmission (i.e. TLS[10] 1.2)

3.2.2 Demilitarized Zone (DMZ)

This network is physically located on the cloud infrastructure (IaaS[11]) and its purpose is to host the front-end interfaces. This type of network acts as a gateway to the public internet and conforms with the following security considerations:

1. Incoming traffic is controlled through the CloudStack´s Firewall in order to protect against attacks such as SQL Injections, Cross Site Scripting, Denial of Service, Distributed Denial of Service, Spoofing, etc. Incoming traffic is filtered by the Firewall dealing with port access and NAT[12] mappings:
    a. Port 80 for general open public access to the frontend
    b. Port 443 for SSL communication
    c. Port 8888 for the WebSocket server connection
2. Personal and Sensitive Data encryption
3. Personal and Sensitive Data secure transmission (i.e. TLS 1.2)
4. A centralized User Authentication / Validation component controls the access to the Front-End. This mechanism will authorize all users of MaTHiSiS, via a Single-Sign-On interface taking advantage of both local accounts as well as social network accounts.
5. DMZ communicates with the MaTHiSiS Back-End layer via the Open API, which is a RESTful API (a set of RESTful web services). The communication restricts access to port 443 (SSL connection).

3.2.3 Private Network

This network is physically located on the cloud infrastructure (IaaS) on a distinct Virtual Machine and its purpose is to host the back-end modules and the system database. This network should conform to the following security restrictions:

1. Incoming traffic is filtered by a Firewall dealing with port access (i.e. 443) and NAT mappings.
2. This network is invisible to the external users and the only way to access it is the use of the Open API call-backs (web services). In this way, an external user cannot gain access to the core platform layers nor to the database.
3. Specific database fields concerning personal and sensitive data should be encrypted (i.e. DES algorithms).
4. The MaTHiSiS back-end encapsulates the three major Spaces (subsystems) with different level of security:
    a. The User Space (US) which stores data for all users. These data are personal information which is common for all users, such as identification data, demographics, etc. For Learners, the dataset is extended to store and manage information related to the Learning history of the user within MaTHiSiS as well as learning preferences and other sensitive data regarding his/her special needs like medical data (autism, PMLD, etc.) that will be used to ensure the personalization and adaptation of the learning process.

[10] TLS definition: http://www.cisco.com/c/en/us/support/docs/security/email-security-appliance/200038-TLS-Configuration-Frequently-Asked-Quest.html#anc2
[11] IaaS definition: http://www.gartner.com/it-glossary/infrastructure-as-a-service-iaas/
[12] NAT definition: http://www.cisco.com/c/en/us/support/docs/ip/network-address-translation-nat/4606-8.html

b. The Cloud Learner Space (CLS) in which the whole MaTHiSiS learning process is being materialized. In terms of data management, CLS retrieves all types of data (personal and sensitive) for decision making purposes. For this reason, the platform will enforce encryption techniques.
c. The Learning Content Space (LCS) which handles the educational material management processes (graphs, goals, SLAs), with no connection to actual learners and thus no personal or sensitive data is managed.

3.3 System Roles

There is a set of roles for the users of MaTHiSiS platform namely administrator, tutor, learner and parent or caregiver. Next each of these roles is explained as follows:

Administrator (Super-admin - Pilot site admin): Users with this role will be able to:

- Configure MaTHiSiS ecosystem using the Platform Configuration and User Management User Interfaces (Users/Learners management, Platform Agents management, Learning Environment management, Local Network management, etc.);
- Manage Social Networking aspects of the learning environment (e.g. set Facebook, YouTube or other social networks used to support the communities working with the platform);
- Manage learning processes in case of need for initial testing. User with this role will be able to conduct all actions described for teacher role and learner roles.

Tutor: The Tutor role is for those that have pedagogical knowledge, which allows them to:

- Create Learning Graphs (create/edit learning goals, Smart Learning Atoms, Learning Actions with their materialization);
- Integrate Learning Materials;
- Manage learners profile information;
- Manage a learning process through the assessment of the attainment of specified learning goals, visualize the performance of different learners during the experience and make modifications to such experience considering the recommendations proposed by DSS.

Parent / Caregiver: Users with this role are able to:

- Be able to manage one or more learners;
- Start a learning experience for the learner;
- Select complementary resources from the list of resources provided in the LGs;
- Visualize the learner performance and profile information.

Learner: This role is the most important. There will be two different types of learners:

1. Supervised/dependent learner for those learners who will need some type of supervision either because they have special learning needs or they are minors without special needs.
2. Independent learner for those who are advanced learners even when they are or adult learners who are pursuing to improve certain set of competences/skills.

Both types of learners will be able to follow a specified Learning Experience interacting with any of the PAs (mobiles, robot, IWB and desktop/laptop). The independent learner can start a Learning Experience selecting Learning Graphs according to his/her learning needs. She/he will be able to visualize his/her performance and modify certain part of his/her profile (demographics, preferences, learning history).

In the case of supervised learner all previous actions will be conducted by his/her teacher/parent/caregiver.

3.4 Business Processes

MaTHiSiS ecosystem as learning supporting framework encapsulates three main business processes but also requires a forth one to ensure the adaptation of the framework to different learning contexts beyond the scheduled pilots:

1. The Learning Experience Process
2. The Learning Content Management Process
3. The Configuration of the MaTHiSiS platform
4. The Users Management Process

Each business process is subdivided in steps used to demonstrate the sequence of actions. When needed, a flow chart is used to show graphically how this takes place on each tier of the overall architecture.

Table 2 presents the legend of all boxes used in the description of the business processes.

LG	Learning Graph
SLA	Smart Learning Atom
LA	Learning Action
LM	Learning Material
LP	Learner's Profile
UP	User Profile
LE	Learning Environment
Classroom	Classroom
PA	Platform Agent
LS	Learning Session
AIR	Affect and Intent Recognition
IPA	Interaction with Platform Agent
USR	User Space Repositories
LCSR	Learning Content Space Repositories
CLSR	Cloud Learner Space Repositories
DSS	Decision Support System
LGE	Learning Graph Engine
EE	Experience Engine
UAV	User Authentication / Validation
LPM	Learning Process Monitoring
URM	User and Role Management
PC	Platform Configuration
SN	Social Networking
LAV	Learning Analytics Visualization
LGP	Learning Games Programming
EMC	Educational Material Creation
ARV	Affect Recognition Visualization
UAV	User Authentication / Validation
SC	Sensorial Component
PC	Platform Collaboration
LMF	Learning Material Framework

Table 2 - Business Processes legend

3.4.1 BP1 – The Learning Experience Process

The Learning Experience Process is divided into three discrete steps:

- Learning Experience Initialization;
- Learning Experience Loop;
- Learning Experience Monitoring.

3.4.1.1 Learning Experience Initialization

Roles involved: Tutor, Learner

This step describes the initialization of the Learning experience and a Learning Session.

1. To create a Learning Experience, the user has to select the corresponding Learning Graph on which the learner will have to work on.
2. A Learning Session is dedicated to the fulfilment of a Learning Experience, so as to make the learner work and progress through the Learning Graph attached to a Learning Experience. To setup a Learning Session, the user will have to define:
    - Who will participate, i.e. a learner in the case of a Tutor or him/her-self in the case of an Independent Learner.
    - What is the environment, i.e. which Platform Agent to use, in which learning environment.
3. After that, the user will be able to start the Learning Session, then creating it in the system.

It is worth mentioning that working on a specific Learning Graph, thus working on specific SLAs, will, by default in the MaTHiSiS system, cause the learner to progress on theses SLAs across many Learning Experiences, i.e. in the cases of experiences that share the same SLAs.

Table 3 describes the components involved in this step.

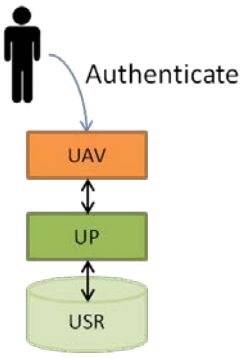
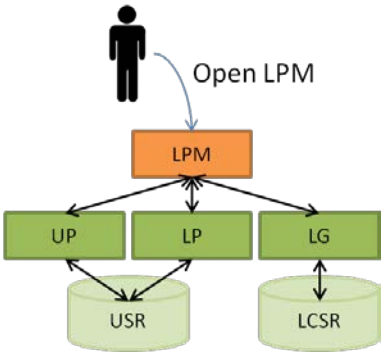
Diagram	Description
	The user authenticates him/her-self through the MaTHiSiS front-end web application. The front-end uses the back-end User Profile lib to check the identity of the user. If the identity is validated, the front-end lets the user continue. Else, it displays an error to the user.
	The user moves to the Learning Process Monitoring part of the front-end. This tool loads the specific content for the user and adapts itself depending on his/her role. To that purpose, the tool gets information from the User Profile lib and the Learner Profile lib. The Learning Graphs available for the user, either personalized or unpersonalized, are queried from the Learning Graph library in order to present themselves through the front-end.

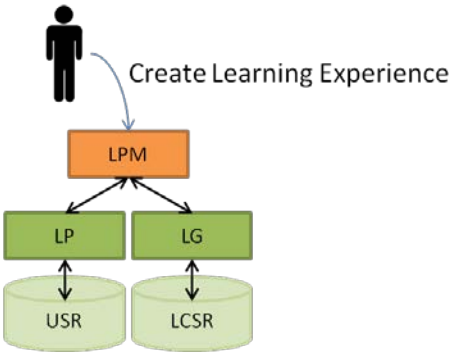
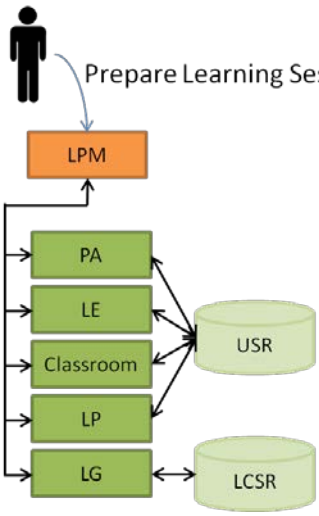
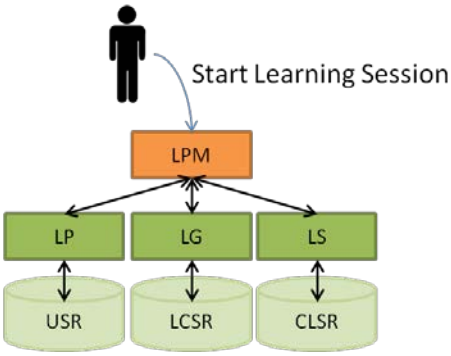
Diagram	Description
	The user selects an unpersonalized Learning Graph, and creates a new Learning Experience from it for him/her-self in the case of an Independent Learner or for a specific learner, or group of learners, in the case of a Tutor. The front-end tool will ask the Learner Profile library to remember this new Learning Experience. A personalized Learning Graph is created for the learner(s) involved.
	To progress through an existing Learning Experience, a learner needs to work through Learning Sessions. The user wants to prepare a new Learning Session for that purpose, for him/her-self in the case of an Independent Learner, or for a specific learner, or group of learners, in the case of a Tutor. The front-end tool then gets the information needed, i.e. the list of usable Platform Agents, the list of usable learning environments and optionally the list of attending learners in the case of a Tutor. Related libraries of the back-end are used to get this information, and everything is presented to the user to let him/her choose among the possibilities.
	When the session is prepared, the user can click the start button in the front-end. This action implies the creation of a new Learning Session for each attending learner using the related library of the back-end.

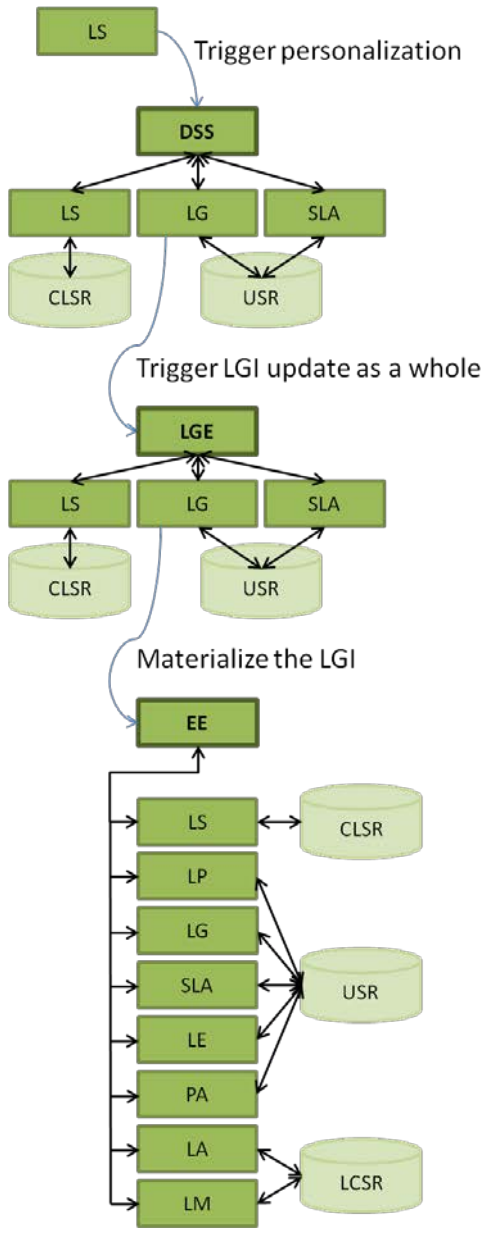
Diagram	Description
	The Learning Session library, after having created the new session for each learner, triggers the personalization, i.e. the first loop of the process. The personalization starts with the Decision Support System (DSS) that will get information about the personalized Learning Graph involved in the session of each learner, and the related personalized SLAs. Here, the DSS will update independently the weight of the personalized SLAs for this session. The update of the personalized LG triggers the Learning Graph Engine (LGE). This component updates the personalized LG as a whole. That means it will update the weight of personalized SLAs taking into consideration the weights of the others. When this update is finished, the Learning Graph library triggers the materialization of the personalized LG instance by calling the Experience Engine (EE). The EE has to choose the best materialization for the learner in the current context of the session. To that purpose, this component gets information about the learner's profile, the learning environment and the Platform Agent from the related libraries. This allows to know the context. Then, the EE gets the best Learning Action based on the weights computed by the LGE and apply the context to get the best LA materialization for this loop. In the end, the EE has chosen the next Learning Action and Learning Material for the learner in this Learning Session. The EE asks the Platform Agent library to start this LM.

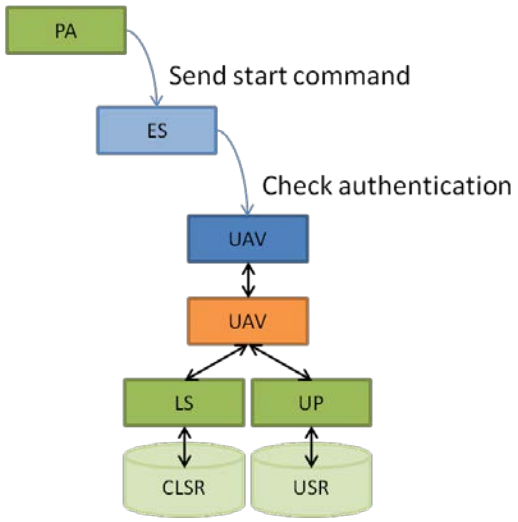
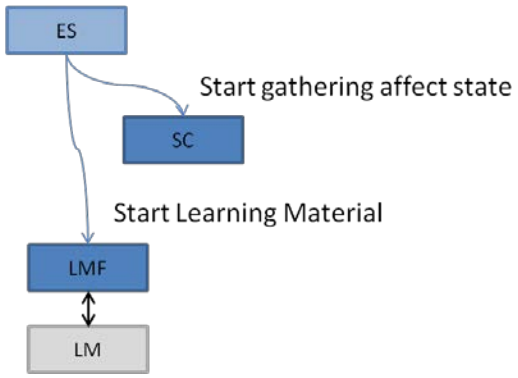
Diagram	Description
	When the PA library receives the command from the EE, the library forwards it to the device representing the Platform Agent. The Experiencing Service (ES) receives this command and first tries to know if it's a valid query. To that purpose, it asks the local User Authentication / Validation (UAV) component to validate the query. The UAV component residing on the PA will ask the UAV component of the front-end to validate the authenticity of the query for this Learning Session.
	If the command is valid, the ES starts the Sensorial Components (SC) in order to gather the affective state clues of the learner, and then asks the Learning Material Framework (LMF) to start the concrete application associated to the LM.

Table 3 - BP1 - Step 1 - Learning Experience Initialization - Components involved

3.4.1.2 Learning Experience Loop

Roles involved: Tutor, Learner

This step describes what happen in the platform during a Learning Session, which is a turn in the loop dedicated to the achievement of the Learning Experience associated.

As MaTHiSiS system manages different contexts for the Learning Sessions, mainly different kinds of Platform Agents used by the learner and the different places where the session occurs, the full platform is involved in this step.

Table 4 describes the components involved in this step.

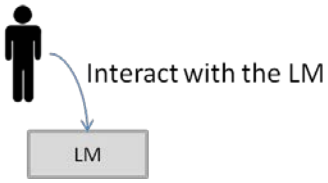
Diagram	Description
	The learners interacts with the Learning Materials.

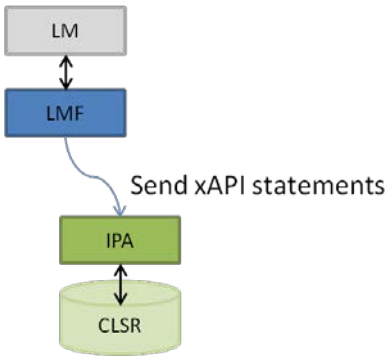
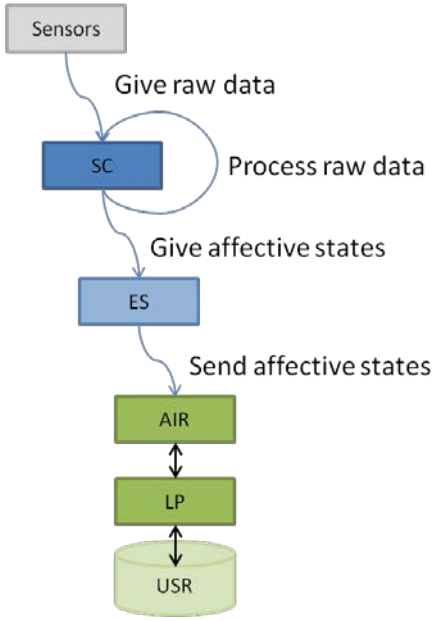
Diagram	Description
	All relevant interactions of the learners with the Learning Material are gathered by the platform thanks to the Learning Material Framework (LMF). These interactions are sent to the MaTHiSiS cloud, as xAPI statements, to the Interaction with Platform Agent (IPA) component. The IPA component treats this information and stores the xAPI statements in the Learning Record Store (LRS), part of the Cloud Learner Space Repositories (CLSR).
	While the learner is interacting with the Learning Material, the Sensorial Components (SC) gather raw data from sensors in parallel. The SC processes this raw data locally to extract a meaningful information for the MaTHiSiS platform, i.e. affect-related data per modality. When the information has been extracted, the SC informs the ES of the "current" affective states of the learner, per modality. The ES then sends this information to the Affect and Intent Recognition (AIR) library. The AIR library is in charge to initiate the multi-modal fusion of the affective states per modality. The result is the extracted and consolidated affective state of the learner. This information is stored in the history of the learner.

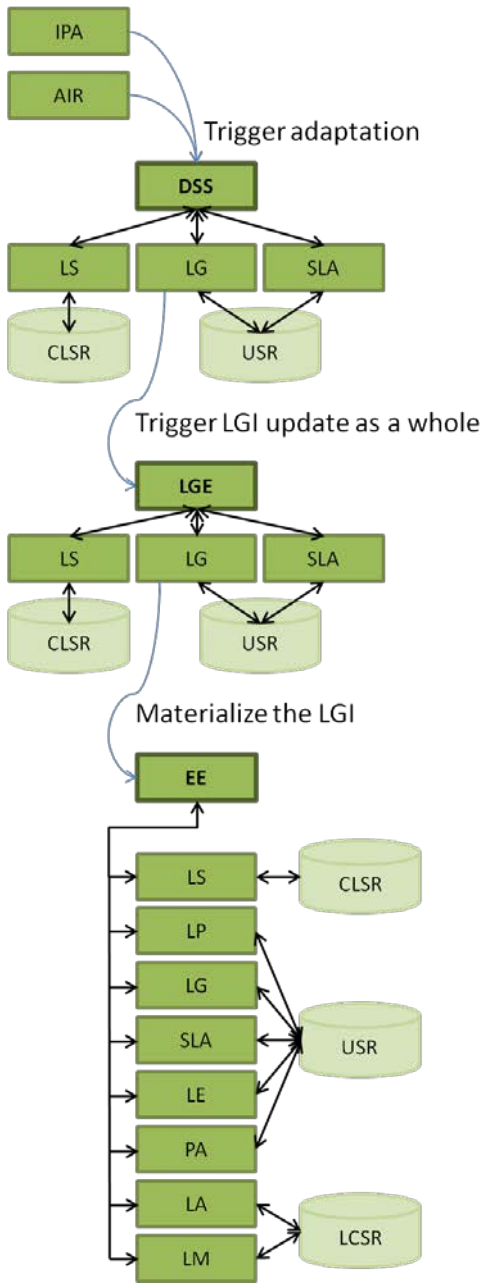
Diagram	Description
	When both the IPA component and the AIR library have received information from the ES residing on the PA used by the learner, that would merit a change in the learning experience, a new adaptation is triggered. The adaptation starts with the Decision Support System (DSS) that will get information about the personalized Learning Graph involved in the session, and the related personalized SLAs, in addition to the affective states extracted from the interactions and the sensors. Here, the DSS will update independently the weight of the personalized SLAs for this session. The update of the personalized LG triggers the Learning Graph Engine (LGE). This component updates the personalized LG as a whole. That means it will update the weight of personalized SLAs taking into consideration the weights of the others. When this update is finished, the LG lib triggers the materialization of the personalized LG instance by calling the Experience Engine (EE). The EE has to choose the best materialization for the learner in the current context of the session. To that purpose, this component gets information about the learner's profile, the learning environment and the Platform Agent from the related libraries. This allows the EE to know the learning context of the current session. Then, the EE gets the best Learning Action based on the weights computed by the LGE and applies the context to get the best LA materialization for this loop. In the end, the EE has chosen the next Learning Action and Learning Material for the learner in this Learning Session. If the learner needs to now work with another LM, the EE asks the PA lib to stop the currently running LM and asks to start the next LM.

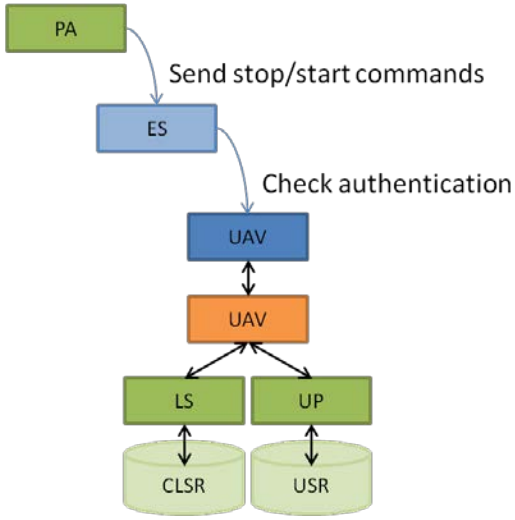
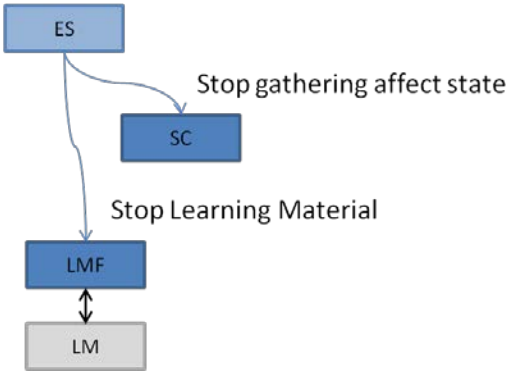
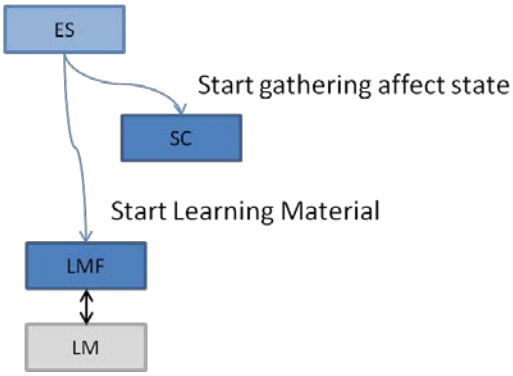
Diagram	Description
	When the PA library receives the commands from the EE, the library forwards them to the device representing the Platform Agent. The Experiencing Service (ES) receives these commands and checks if they are valid queries. To that purpose, it asks the local User Authentication / Validation (UAV) component to validate the queries. The UAV component residing on the PA will ask the UAV component of the front-end to validate the authenticity of the queries for this Learning Session.
	If the optional stop command is valid, the ES stops the Sensorial Components (SC) and then asks the Learning Material Framework (LMF) to stop the concrete application associated to the LM.
	If the start command is valid, the ES starts the Sensorial Components (SC) in order to gather the affective state clues of the learner, and then asks the Learning Material Framework (LMF) to start the concrete application associated to the LM.

Table 4 - BP1 - Step 2 - Learning Experience Loop - Components involved

3.4.1.3 Learning Experience Monitoring

Roles involved: Tutor, Independent Learner, Caregiver

This step describes how the users can monitor the progression of learners within MaTHiSiS.

The user can monitor the progression of learners through Learning Experiences by using the front-end, and more specifically these two parts:

- The Learning Analytics Visualization;

- The Learning Process Monitoring.

Using the former, the user will have the progression of learners in their Learning Experiences, taking into account all related learning sessions that occurred.

Using the later, the user will have the progression of learners during running learning sessions only. It thus offers specific features to monitor the running sessions.

Table 5 describes the components involved in this step. We consider that the user is already authenticated and connected to the MaTHiSiS front-end.

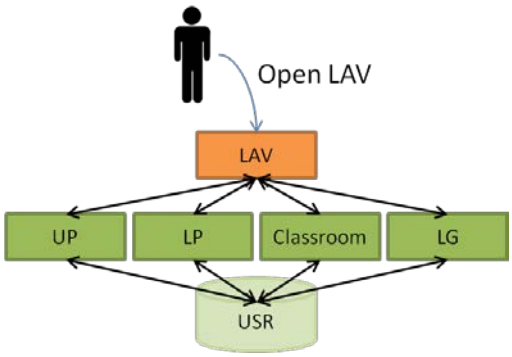
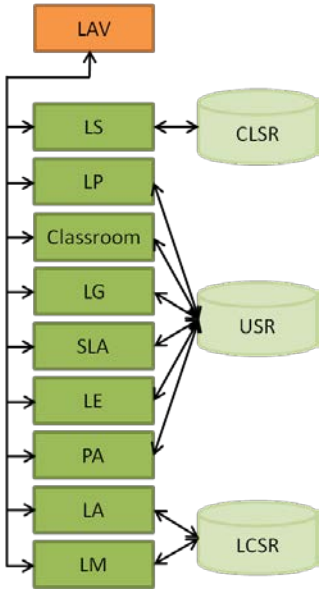
Diagram	Description
	From the main MaTHiSiS web application, the Tutor moves to the Learning Analytics Visualization (LAV) part of the front-end in order to monitor the Learning Experiences. The LAV will filter learners and optionally classrooms depending on the user that opens it.
	The LAV offers the possibility to access the list of Learning Experiences for a specific learner, or optionally for a full classroom in the case of a Tutor. Given a Learning Experience, the user can access the history of Learning Sessions already finished. For each of them, more information is accessible. As the LAV knows if they are running Learning Sessions, the user can directly access the Learning Process Monitoring from there.

Table 5 - BP1 - Step 3 - Learning Experience Monitoring - Components involved when using the LAV

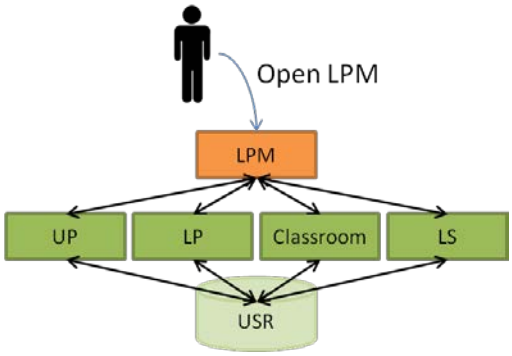
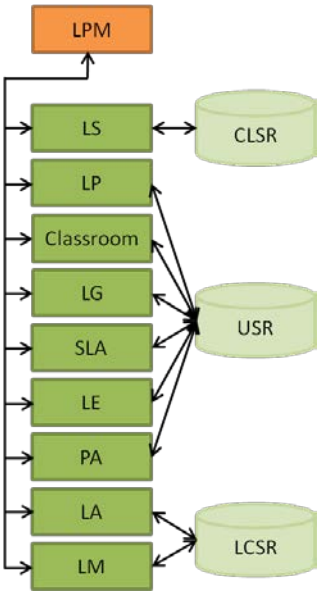
Diagram	Description
	From the main MaTHiSiS web application, the Tutor goes to the Learning Process Monitoring (LPM) part of the front-end in order to monitor the currently running Learning Sessions. The LPM will filter learners and optionally classrooms depending on the user that opens it.
	The LPM offers the possibility to follow and see what is happening during a running Learning Session for a specific learner, or optionally for a full classroom in the case of a Tutor. Given a Learning Session, the user will have access to some options. These will give the user the possibility to provide hints based on his/her expertise or even to adjust the Learning Session. The LPM knows the list of learners and optionally the list of classrooms. This allows the user to directly access the Learning Analytics Visualization from there.

Table 6 - BP1 - Step 3 - Learning Experience Monitoring - Components involved when using the LPM

3.4.2 BP2 – The Learning Content Management Process

This process is responsible for the management of the main concepts of MaTHiSiS representing the content of any Learning Experience, i.e. Learning Graphs, Smart Learning Atoms, Learning Actions with their materializations and Learning Materials.

To that purpose, MaTHiSiS offers, as a Front-end component, a specific and dedicated standalone application called the Learning Content Editor (LCE). It allows the edition of all kind of learning content, except the Learning Materials. When installed, the LCE can be automatically launched from the main MaTHiSiS web application through the Learning Content Manager tool, allowing a smooth transition for the user.

Therefore, the Learning Content Management Process is divided into five discrete steps:

- Learning Graph Management;
- Smart Learning Atom Management;
- Learning Action Management;
- Learning Action Materialization Management;
- Learning Material Management.

3.4.2.1 Learning Graph Management

Roles involved: Tutor

This step describes how to create and edit the Learning Graphs using the Educational Material Creation part of the front-end. The latter provides a specific tool dedicated to this kind of learning content.

Table 7 describes the components involved in this step. We consider that the Tutor is already authenticated and connected to the MaTHiSiS front-end.

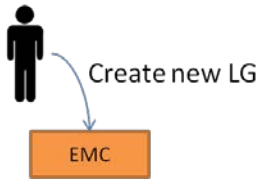
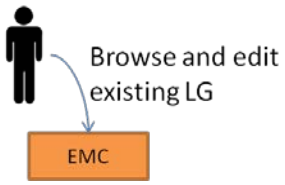
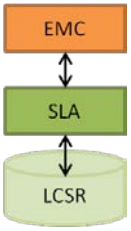
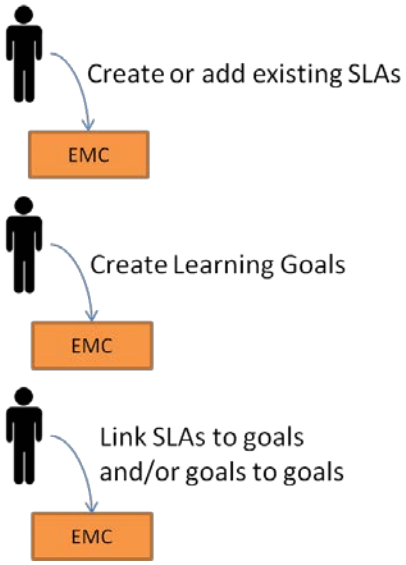
Diagram	Description
	From the main MaTHiSiS web application, the Tutor goes to the Educational Material Creation (EMC) part of the front-end in order to create a new LG.
	Alternatively, from the main MaTHiSiS web application, the Tutor goes to the Browser of the EMC part of the front-end in order to edit an existing LG.
	When the Learning Graph editor, a tool of the EMC opens, it asks the MaTHiSiS server to provide it with all existing SLAs usable by the Tutor.
	The Tutor can add existing SLAs to the Learning Graph, create new (empty) SLAs and use them too. The Tutor can also add new Learning Goals to the LG, and he/she can link SLAs with goals, or goals with goals, providing each time a relevance weight to that link. There is no communication with the MaTHiSiS server during the edition.

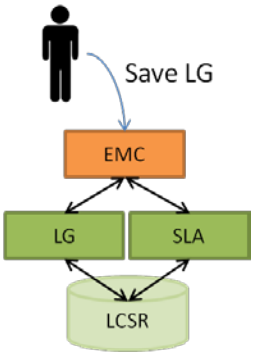
Diagram	Description
	When the Tutor saved the Learning Graph, its current state is stored in the cloud. This implies the creation or the update of the Learning Graph and the creation of new Smart Learning Atoms if any. Related back-end components are called to that purpose.

Table 7 - BP2 - Step 1 - Learning Graph Management - Components involved

3.4.2.2 Smart Learning Atom Management

Roles involved: Tutor

This step describes how to create and edit the Smart Learning Atoms using the Educational Material Creation part of the front-end. The latter provides a specific tool dedicated to this kind of learning content.

Table 8 describes the components involved in this step. We consider that the Tutor is already authenticated and connected to the MaTHiSiS front-end.

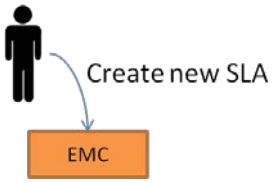
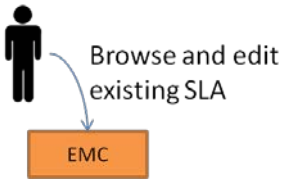
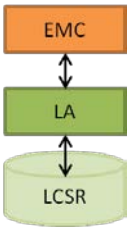
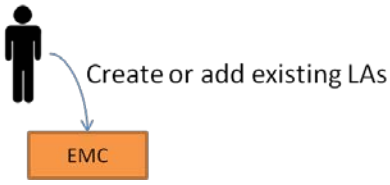
Diagram	Description
	From the main MaTHiSiS web application, the Tutor goes to the Educational Material Creation (EMC) part of the front-end in order to create a new SLA.
	Alternatively, from the main MaTHiSiS web application, the Tutor moves to the Browser of the EMC part of the front-end in order to edit an existing SLA.
	When the Smart Learning Atom editor, a tool of the EMC opens, it asks the MaTHiSiS server to provide it with all existing Learning Actions usable by the Tutor.
	The Tutor can add existing LAs to the Smart Learning Atom, create new (empty) LAs and use them too. There is no communication with the MaTHiSiS server during the edition.

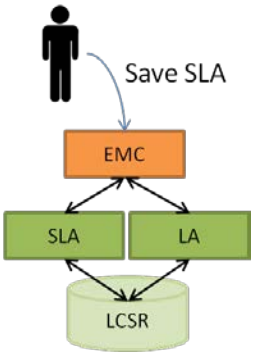
Diagram	Description
	When the Tutor saves the SLA, its current state is stored in the cloud. This implies the creation or the update of the SLA and the creation of new Learning Actions if any. Related back-end components are called to that purpose.

Table 8 - BP2 - Step 2 - Smart Learning Atom Management - Components involved

3.4.2.3 Learning Action Management

Roles involved: Tutor

This step describes how to create and edit the Learning Actions using the Educational Material Creation part of the front-end. The latter provides a specific tool dedicated to this kind of learning content.

Table 9 describes the components involved in this step. We consider that the Tutor is already authenticated and connected to the MaTHiSiS front-end.

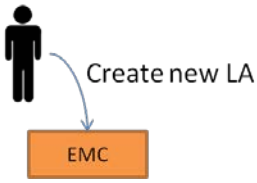
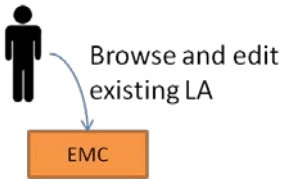
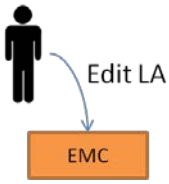
Diagram	Description
	From the main MaTHiSiS web application, the Tutor goes to the Educational Material Creation (EMC) part of the front-end in order to create a new SLA.
	Alternatively, from the main MaTHiSiS web application, the Tutor goes to the Browser of the EMC part of the front-end in order to edit an existing SLA.
	The Tutor can edit the few properties of the Learning Action. There is no communication with the MaTHiSiS server during the edition.

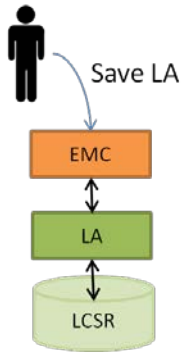
Diagram	Description
	When the Tutor saves the LA, its current state is stored in the cloud. This implies the creation or the update of the LA. Related back-end components are called to that purpose.

Table 9 - BP2 - Step 3 - Learning Action Management - Components involved

3.4.2.4 Learning Action Materialization Management

Roles involved: Tutor

This step describes how to edit the materialization of Learning Actions using the Educational Material Creation part of the front-end. The latter provides a specific tool dedicated to the materializations edition.

Table 10 describes the components involved in this step. We consider that the Tutor is already in the Learning Action editor.

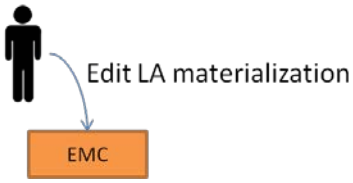
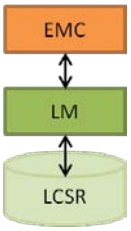
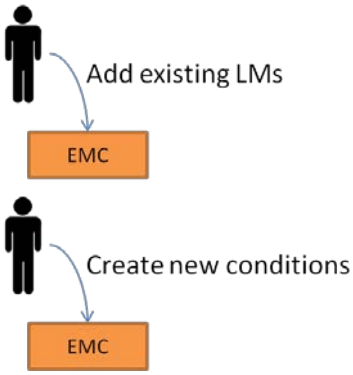
Diagram	Description
	From the Learning Action editor, the Tutor can access the materialization editor.
	When the Learning Action materialization editor, a tool of the EMC opens, it asks the MaTHiSiS server to provide it with all existing Learning Materials usable by the Tutor.
	The Tutor can add existing LMs to the materialization, create conditions on the context and links the former and the later. This way, the Tutor can choose which LM will be used in which context. There is no communication with the MaTHiSiS server during the edition.

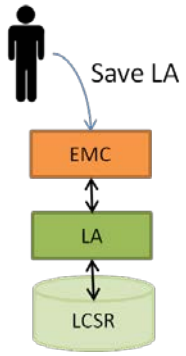
Diagram	Description
	Saving the materialization goes through the saving of the LA. Therefore, when the Tutor save the LA, its current state is stored in the cloud. This implies the creation or the update of the LA. Related back-end components are called to that purpose.

Table 10 - BP2 - Step 4 - Learning Action Materialization Management - Components involved

3.4.2.5 Learning Material Management

Roles involved: Tutor

This step describes how to create and edit the Learning Materials using the Educational Material Creation part of the front-end. The latter provides a specific tool dedicated to this kind of learning content.

Table 11 describes the components involved in this step. We consider that the Tutor is already authenticated and connected to the MaTHiSiS front-end.

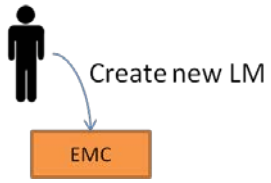
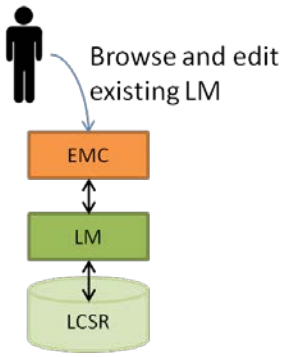
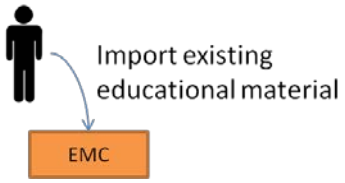
Diagram	Description
	From the main MaTHiSiS web application, the Tutor goes to the Educational Material Creation (EMC) part of the front-end in order to create a new LM.
	Alternatively, from the main MaTHiSiS web application, the Tutor moves to the Browser of the EMC part of the front-end in order to edit an existing and accessible LM.
	The Tutor can import existing educational material (e.g. a web-based game or a document) to create the associated LM. Obviously, the Tutor is able to change all the properties needed to make this external material understandable and usable by the MaTHiSiS platform. There is no communication with the MaTHiSiS server during the edition.

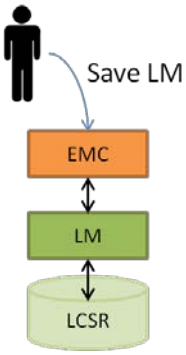
Diagram	Description
	When the Tutor save the LM, its current state is stored in the cloud. This implies the creation or the update of the LM. Related back-end components are called to that purpose.

Table 11 - BP2 - Step 5 - Learning Material Management - Components involved

3.4.3 BP3 – The Configuration of MaTHiSiS platform

The configuration of the MaTHiSiS platform is divided into the following steps:

- Create a new learning environment or browse and modify existing ones.
- Define a new PA and assign it to learning environments or browse and modify existing PAs technical specifications and requirements.
- Other specific settings related to the platform.

3.4.3.1 Learning Environment Management

Roles involved: Administrator

This step describes how to create and edit the Learning Environments using the Platform Configurator part of the front-end.

Table 12 describes the components involved in this step. We consider that the Administrator is already authenticated and connected to the MaTHiSiS front-end.

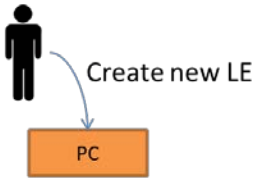
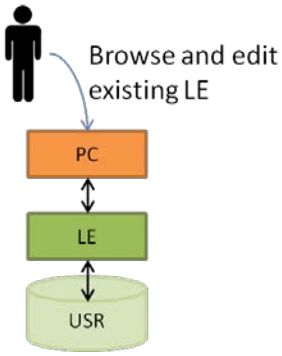
Diagram	Description
	From the main MaTHiSiS web application, the Administrator moves to the Platform Configurator (PC) part of the front-end in order to add a new Learning Environment (LE).
	Alternatively, from the main MaTHiSiS web application, the Administrator moves to the Platform Configurator part of the front-end in order to find and edit an existing and accessible LE. The Learning Environment Configurator asks the MaTHiSiS server to provide the list with all existing and accessible Learning Environments.

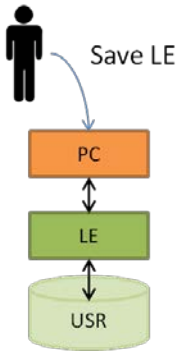
Diagram	Description
	When the Administrator save the LE, its current state is stored in the cloud. This implies the creation or the update of the LE. Related back-end components are called to that purpose.

Table 12 - BP3 - Step 1 - Learning Environment Management - Components involved

3.4.3.2 Platform Agent Management

Roles involved: Administrator

This step describes how to create and edit the Platform Agents using the Platform Configurator part of the front-end.

Table 13 describes the components involved in this step. We consider that the Administrator is already authenticated and connected to the MaTHiSiS front-end.

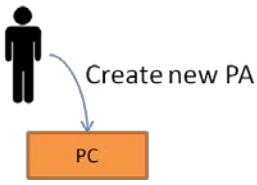
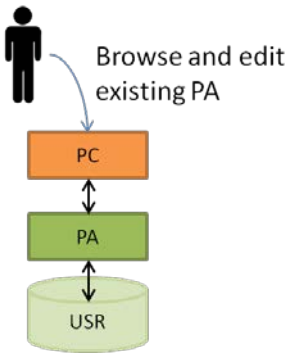
Diagram	Description
	From the main MaTHiSiS web application, the Administrator moves to the Platform Agents Configurator tool of the Platform Configurator part of the front-end in order to add a new Platform Agent (PA).
	Alternatively, from the main MaTHiSiS web application, the Administrator moves to the Platform Agents Configurator in order to find and edit an existing and accessible PA. The tool asks the MaTHiSiS server to provide the list and details of the existing PA that the user can access. Therefore, depending on the role of the user, the list of Platform Agents is adjusted.

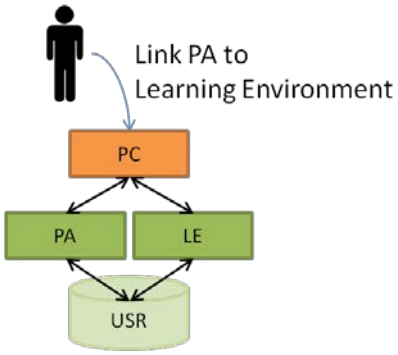
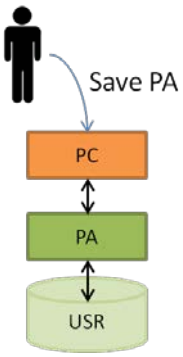
Diagram	Description
	The Administrator can also assign PA to Learning Environments through the same interface.
	When the Administrator save the PA, its current state is stored in the cloud. This implies the creation or the update of the PA. Related back-end components are called to that purpose.

Table 13 - BP3 - Step 2 - Platform Agent Management - Components involved

3.4.4 BP4 – The Users Management Process

The user management process acts in two different levels in MaTHiSiS platform. The first level concerns the centralize user management tool which a user can use to register and login himself, or an administrator to manage users and roles as detailed in section 3.3.

The second level of user management process is being used from each component which deals with users and is responsible for the appropriate handling of the user management.

The management of the users takes place in the main MaTHiSiS interface in three subsystems. These are the following:

- User Account Management;
- Classroom management;
- Learner Profile Management.

3.4.4.1 User Account Management

Roles involved: Administrator, Tutor, Caregiver, Learner

This step describes how to create and edit a User Account using the User Management part of the front-end.

Table 14 describes the components involved in this step. We consider that the user is already authenticated and connected to the MaTHiSiS front-end.

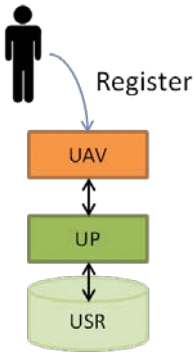
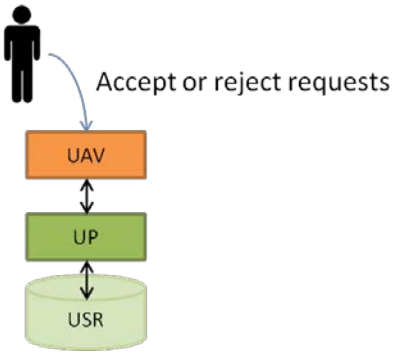
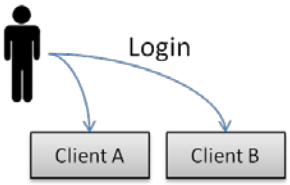
Diagram	Description
	The user enters the access management component which is responsible for the users' account management. The user completes the related form with the required fields about his/her account and he/she selects the preferred role(s).
	The Administrator can see all the pending requests for new users' account. Hence, the Administrator can accept or reject any request.
	After the Administrator's acceptance of the user's account, the user can enter the platform through any client which has been registered to MaTHiSiS. The client is responsible to handle its functionality relative to the user's role(s).

Table 14 - BP4 - Step 1 - User Account Management - Components involved

3.4.4.2 Classroom Management

Roles involved: Administrator, Tutor

This step describes how to create and edit Classrooms using the User Management part of the front-end.

Table 15 describes the components involved in this step. We consider that the user is already authenticated and connected to the MaTHiSiS front-end.

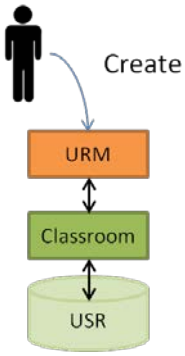
Diagram	Description
	From the main MaTHiSiS web application, the user moves to the Classroom Management tool of the User and Role Management part of the front-end in order to add a new Classroom.

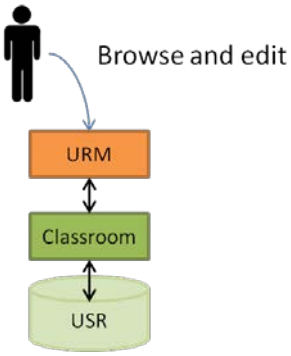
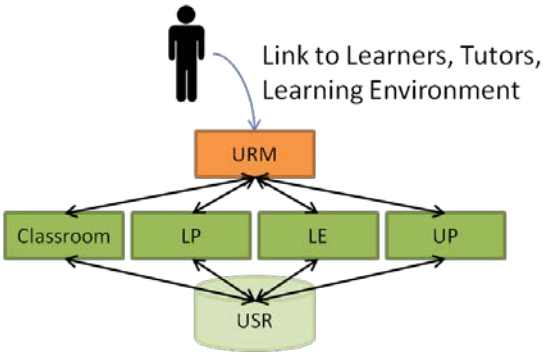
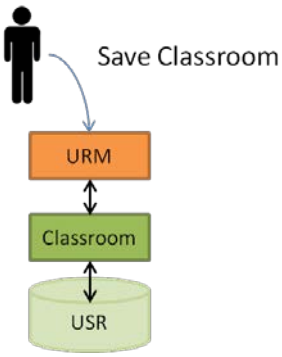
Diagram	Description
	Alternatively, from the main MaTHiSiS web application, the user moves to the Browser of the Classroom Management in order to find and edit an existing and accessible Classroom. The tool asks the MaTHiSiS server to provide the list and details of the existing Classrooms that the user can access. Therefore, depending on the role of the user, the list of Classrooms is adjusted.
	The user can also link the Classroom to a Learning Environment, a Tutor and/or assigns Learners through the same interface.
	When the user save the Classroom, its current state is stored in the cloud. This implies the creation or the update of the Classroom. Related back-end components are called to that purpose.

Table 15 - BP4 - Step 2 - Classroom Management - Components involved

3.4.4.3 Learner Profile Management

Roles involved: Administrator, Tutor, Caregiver, Learner

This step describes how an authorized user can manage a learner's profile. The learner's profile indicates information related to any disabilities, interaction preferences and learning style which the system takes into account in order to initiate, personalize and adapt the learning experience of the learner.

Table 16 describes the components involved in this step. We consider that the user is already authenticated and connected to the MaTHiSiS front-end.

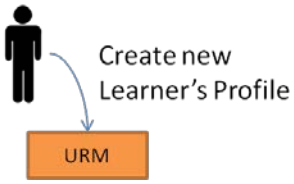
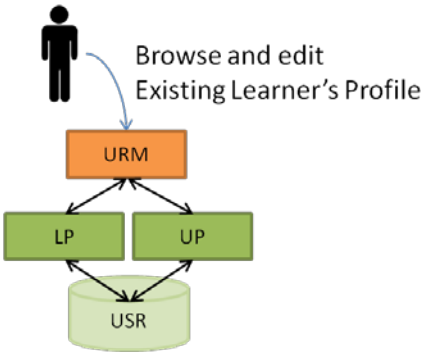
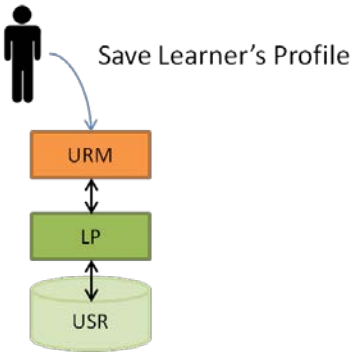
Diagram	Description
	From the main MaTHiSiS web application, the user moves to the Learners' Profile Management tool of the User and Role Management part of the front-end in order to add a new Learner. Alternatively, a learner's profile is created automatically when a new user is added as a learner.
	Alternatively, from the main MaTHiSiS web application, the user moves to the Browser of the Learners' Profile Management in order to find and edit an existing and accessible learner's profile. The tool asks the MaTHiSiS server to provide the list and details of the existing learners that the user can access. Therefore, depending on the role of the user, the list of learners' profile is adjusted.
	When the user save the Learner's Profile, its current state is stored in the cloud. This implies the creation or the update of the profile. Related back-end components are called to that purpose.

Table 16 - BP4 - Step 3 - Learner Profile Management - Components involved

4. Back-end layer

The MaTHiSiS Back-end layer is in charge of providing all internal components dedicated to the computation of the Learning Experiences personalization and adaptation, in addition to the management and storage of the information needed to that purpose.

The Open API Tier is the single point of entry to this layer. It gives access to the functionalities of their underlying components described in more details in Sections 4.2, 4.3 and 4.4. At this level, the APIs are responsible to validate the authentication of the callers, ensuring the security of data entering and leaving the MaTHiSiS Back-end layer.

Some Open APIs are RESTful APIs, thus using the well-known standardized HTTP protocol. Others use the standardized WebSocket protocol, allowing a two-way communication between the back-end and the Platform Agent layer.

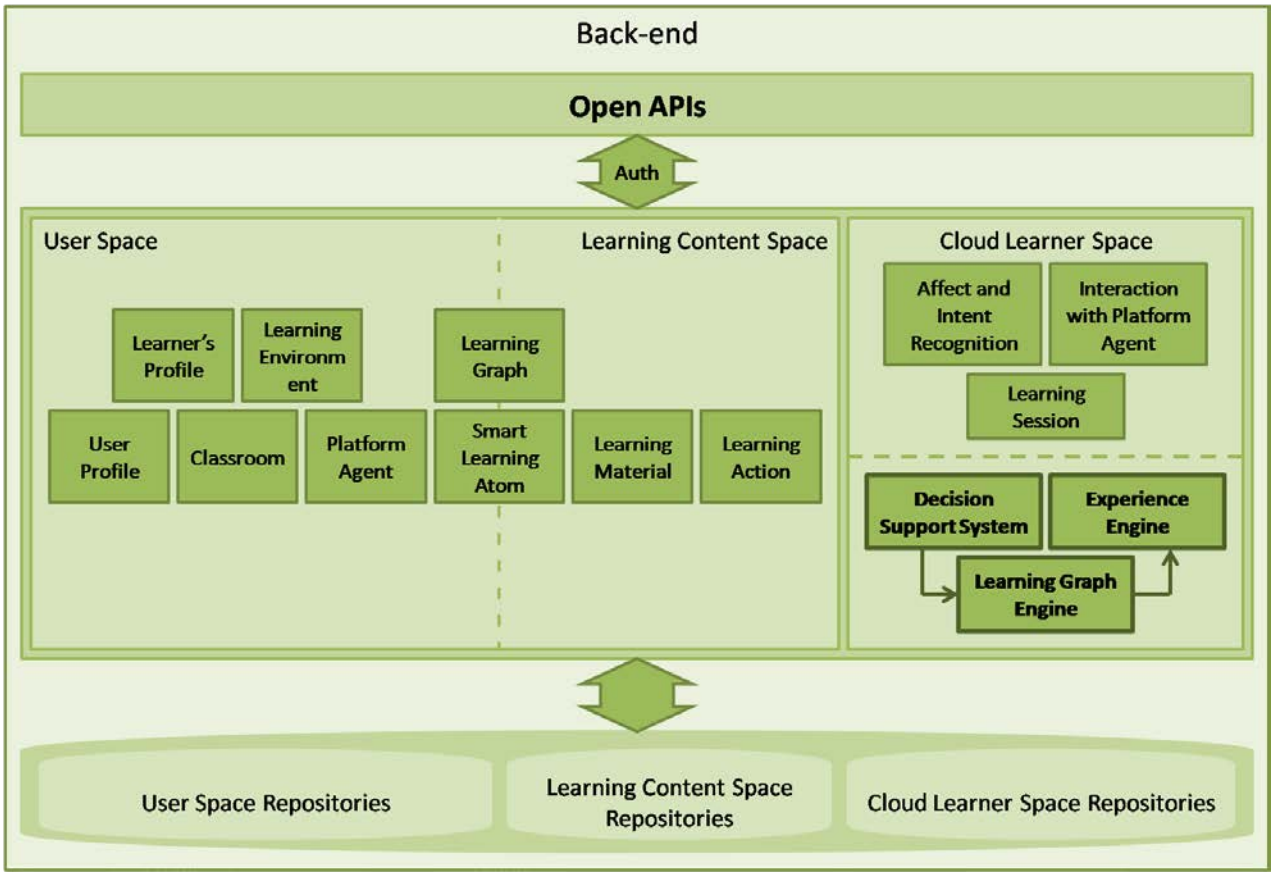


Figure 3 - MaTHiSiS Back-end layer

4.1 Authentication mechanism

MaTHiSiS ecosystem includes an authentication mechanism focusing on the identity management, authentication and authorization and resource registration. These procedures are performed from the component called User Authentication and Authorization Component (UAV). The core functionalities of this

component are based on the open source ForgeRock OpenAM[13] tool that uses some well-known protocols such as the OAuth2.0 protocol[14]included the OpenID connect[15] layer. It is worth to stress that the main mechanism for enforcing confidentiality in the communicated data is the usage of the readily available (to all involved applications) Secure Socket Layer / Transport Secure Layer (SSL/TSL) mechanisms.

According to the *RFC6749*[16], the OAuth2.0 protocol enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf. Figure 4 depicts an abstract view of the OAuth 2.0 flow. In the case of MaTHiSiS, we typically use the term **resource to refer to any application**. Furthermore, the role of resource owner can be assigned to the administrator or the developer of the application.

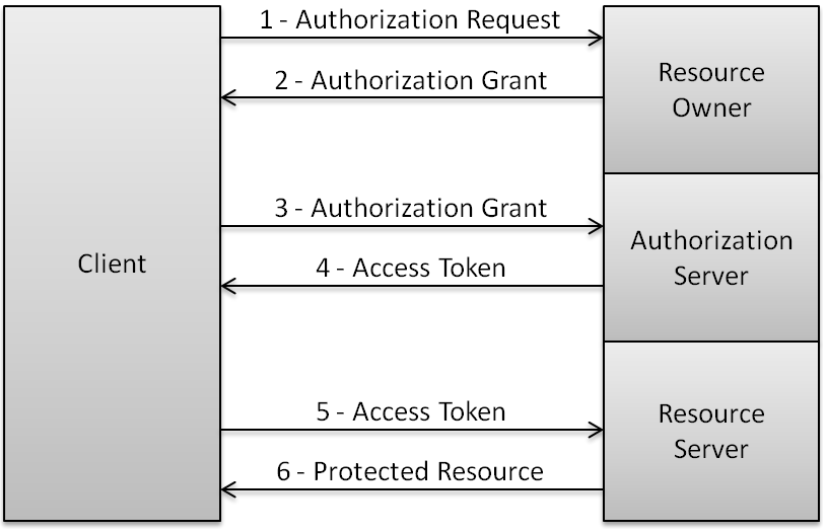


Figure 4 - Flow on OAuth2.0

According to RFC6749, the following steps take place:

- The client requests authorization from the resource owner. The authorization request can be made directly to the resource owner or indirectly via the authorization server as an intermediary.
- The client receives an authorization grant, which is a credential representing the resource owner's authorization, expressed using one of four grant types defined in RFC6749.
- The client requests an access token by authenticating with the authorization server and presenting the authorization grant.
- The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token.
- The client requests the protected resource from the resource server and authenticates by presenting the access token.
- The resource server validates the access token, and if valid, serves the request.

In MaTHiSiS ecosystem, the resource owner could be the developer or the administrator of a MaTHiSiS application or component.

[13]ForgeRock OpenAM: https://www.forgerock.com/platform/access-management/
[14] OAuth 2.0: https://tools.ietf.org/html/rfc6749
[15]OpenID: http://openid.net/
[16] https://tools.ietf.org/html/rfc6749

The back end functionalities of the UAV include the following ones:

- User registration;
- Account management;
- Role management;
- Resource registration (applications);
- Logging.

Any user in MaTHiSiS ecosystem is able to create his account via the UAV component entering attributes, such as the username, first name, last name, contact details (email account, phone), gender and address details. The user's username is unique. After the successful registration of the user and the activation of his account, each user can be associated with one or more available in that MaTHiSiS roles (see 3.3).

Both user account and related roles are kept in the RDBMS/LDAP of the UAV ensuring the data integrity. The registered users are able to manage their accounts and/or their roles and change their password on demand as well. The deactivation and deletion of their accounts are also a further capability.

Apart from the user management, each resource owner/developer can register the application in the UAV with the prerequisite that he has already been registered in UAV. In meanwhile, the resource owner will be notified about the CLIENT_ID and the CLIENT_SECRET that characterizes uniquely the registered application in UAV; it is something like the username and password of the application, so the CLIENT_SECRET must be secret and protected. Application management is also supported in UAV.

Therefore, UAV includes the list of registered MaTHiSiS users and the catalogue of MaTHiSiS applications. Each registered in UAV component user can access any registered application/resource providing his credentials. UAV component exposes a web service that authenticates the user based on his credentials. In case that the user credentials are valid and the user account is active, an access token is generated with limited time duration. A sample of this web service is depicted in Table 17.

Functionality	**Authorize user / retrieve the access token**
Endpoint	/openam/oauth2/access_token
Method	POST
Headers	Authorization Basic (client_id :client_secret), Accept : application/json, Content-type : application/x-www-form-urlencoded
Payload	grant_type=password&username=<username>&password=<password>&scope=<list_of scopes>
Response format	Structured (JSON)

Normal Response	Status: 200 OK
	{
	"scope": "string",
	"expires_in":"integer",,
	"token_type": "Bearer",
	"refresh_token": "string",
	"id_token": "string",
	"access_token": "string"
	}

Unauthorized response	Status: 401 UNAUTHORIZED
	{
	"reason": "Access token is not valid",
	"code": 401
	}

Table 17 - Retrieve access token of user

After successful user authentication, any application can invoke a set of web services to validate the access token, retrieve the account and role(s) of user based on access token and refresh the expired access token. The access token of the user has limited duration (see expires_in value in seconds).

Table 18, Table 19, Table 20 and Table 21 described some useful web services that UAV provides to registered applications.

Functionality	**Retrieve the user account based on access token**
Endpoint	/openam/oauth2/userinfo
Method	GET
Headers	Authorization Bearer access_token, Accept : application/json
Payload	-
Response	Structured (JSON)

Table 18 - Retrieval of user account

Functionality	**Retrieve the list of user's roles based on access token**
Endpoint	/api/oauth2/roles
Method	GET
Headers	Accept : application/json
Query parameters	access_token={access_token}
Response	Structured (JSON)

Table 19 - Retrieval of user's roles

Functionality	**Validate the access token**
Endpoint	/openam/oauth2/tokeninfo
Method	GET
Headers	Accept : application/json
Query parameters	access_token={access_token}

Response	Structured (JSON)

Table 20 - Validation of access token

Functionality	**Refresh the (expired) access token**
Endpoint	/openam/oauth2/access_token
Method	POST
Headers	Authorization Basic (client_id :client_secret), Accept : application/json, Content-type : application/x-www-form-urlencoded
Query parameters	Grant_type=refresh_token&access_token={access_token}
Response	Structured (JSON)

Table 21 - Refresh of access token

From the user perspective, UAV component is responsible to store the users account, authenticate/ authorize users and share in trusted/registered applications their roles and a part of their account (after their agreement with it; no password sharing) on demand. The trusted applications are responsible to consider the available roles of any user and adjust the capabilities of user on them. UAV does not perform any policy on the registered applications.

For instance, the Platform Configurator component may adjust the user capabilities based on the role of the logged in user. Therefore, only users with specific roles can create associations among a tutor, one or more classrooms and the list of learners.

4.2 Cloud Learner Space

CLS is the core of the MaTHiSiS system which will provide functionalities for personalization / adaptation in learning, profile storage, interaction and content provision on the fly. Next we describe in detail each of the components composing the CLS.

4.2.1 Decision Support System Tier

The DSS is responsible for driving multimodal learning analytics, personalization/adaptation mechanisms and knowledge transfer. It will be the brain of the system which will employ affect analysis, learner's profile and performance, as well as learning goals integration through a multimodal fusion scheme. Taking into account all these data, it will drive proper adaptations to SLAs, according to the key information of the learner's profile and his/her expectations in order to support the achievement of the defined learning goals.

In terms of knowledge transfer, the DSS controls the synchronous and asynchronous collaboration between different platform agents of the system thus orchestrating the different platform agents of the system ensuring the smooth transition of between them to facilitate the ubiquitous nature of the learning processes supported by MaTHiSiS.

Tier Name	**Decision Support System**
Component Name	Decision Support System (DSS)

Layer / Space	Back-end / Cloud Learner's Space
Inputs	
1	Source: AIR lib API Data: Affect-related features or/and labels based on both behaviour (face, gaze, skeleton motion, speech/sound, inertial sensors) and interaction with the materials Schema: JSON
2	Source: Learner's Profile lib API Data: User Profile Info, Goal scores history, SLA scores history (performance and affective states) Schema: JSON
3	Source: DSS tier Data: SLA scores history (performance for personalization and affective states for adaptation) Schema: JSON
4	Source: SLA lib API Data: Current personalized SLA competence weights Schema: JSON
5	Source: LS lib API Data: Current Learning Session information of the learner. Schema: JSON
6	Source: LM lib API Data: Learning Materials description associated to the current Learning Experience Schema: JSON
7	Source: PA lib API Data: Platform Agents description associated to the current Learning Experience Schema: JSON
8	Source: Platform Collaboration tier Data: Data from corresponding new types of PAs or LM (for asynchronous collaboration) Schema: To be defined in D6.4 (M24)
9	Name: Learning Record Store Data: Information about interaction between the learner and the PA Schema: JSON - xAPI statement
Outputs	

1	Destination: LGlib API Data: Updated personalized SLA competence weights associated to a personalized LG Schema: JSON
2	Destination: Learner's Profile lib API Data: Affective States updates (from multimodal fusion) and performance (from learning analytics) for history, SLA scores history updates (performance and affective states) Schema: JSON
3	Destination: DSS tier Data: Affective States updates (from multimodal fusion) and performance (from learning analytics) for history Schema: JSON
High – Level Operations	
1	Multimodal (early and/or late) fusion based on the (non-fixed number of) inputs received
2	Learning Analytics
3	Asynchronous Collaboration
4	Synchronous Collaboration
Constraints	
1	None

Table 22 - Decision Support System Tier - Decision Support System

This tier implements several operations. Firstly, this component will calculate the global affect state of the learners. In order to obtain this affective state, DSS will use the features provided by SC and IPA (which represent the emotional state of learners for different modalities), the information about the current learning interactions stored in the LRS as well as those data stored in LPR (which includes the personal preferences and needs; and qualifications of learner) and parameters related to the SLA involved in the learning process (mainly, vertex weights).

Secondly, the DSS will allow the synchronous collaboration among PAs using the information about the last learner's interaction stored in the LPR and the information related to each type of PA and learning material involved in the learning activity. More detailed information and technical aspects will be included in the deliverables concerning Task 6.3 activities (namely, ***D6.4/5 – Synchronous and asynchronous collaboration among platform agents*** due in M24 and M33, respectively).

Synchronous collaboration will make possible the smooth deployment of a learning process avoiding obstacles related to physical location of the learners. In the case of multi-learner scenario, the DSS will allow the real time collaboration between them, using PAs of a heterogeneous nature.

Asynchronous collaboration includes several operations. Firstly, this component will allow the learners to continue their learning process in different PAs and/or learning environments (i.e. they start the lesson in a computer at school but they want to continue working at home with their tablets) in individual scenarios. Finally, the asynchronous collaboration functionality include in the DSS will facilitate the integration and

accommodation of new kind of PAs using the knowledge related to interactions with learning materials through other PAs. The description of these interactions will be derived from xAPI specification, using the ADL's vocabulary[17], and the Learning Actions Ontology (section 7.1). In this way, asynchronous collaboration will facilitate this accommodation allowing the re-usability of an existing MaTHiSiS-actuated learning scenario upon introduction of an unknown PA to the Learning Experience, while avoiding (or, at least reducing) the training needed. Using asynchronous collaboration, a teacher could, for example, introduce a PA which is not available in the system and which includes unknown SCs to perform the same learning actions as in the learners' tablets, providing proper adaptation of the learning content based on previous interactions though this well-known PA.

4.2.2 Learning Graph Engine Tier

During the personalisation and adaptation process, the vertex weights of a personalised LG instance will be adapted through the MaTHiSiS Learning Graph Engine (LGE). Personalization in the context of MaTHiSiS pertains to previously acquired skills, learning preferences, level of expertise and historical data. Adaptation will be related to the learner's conditioned response to the Learning Experience during Learning Sessions.

The DSS will assign weight on personalised SLA instances, denoting the competence level of a learner 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).

Then, the LGE will initially update the vertex weights of a personalised LG instance, based on the SLA weights received from the DSS, for all the SLAs that take part in the LG instance. In addition, learning goal weights will be updated based on their related SLA weights.

On 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.

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 (edge weights), in order to predict vertex weights that will optimally (more rapidly) bring the LG instance to its completion. This will entail possible re-calculation of the SLA weight received from the DSS.

Methodology of LGE's dynamic LG vertex weights parameterisation through spectral analysis:

- Update vertex weights for a particular LG instance
    - Update learning goals' weights after SLA weight update (based on neighbouring vertices' weights)
- Spectral analysis of LG instance (combinatorial Laplacian matrix)
- Choose vertex weights on a graph, subject to converging the graph to the predefined optimal state, in order to minimize a convex function of the positive Eigen values of the associated Laplacian matrix

As a final step, the LGE will provide directives to the Experiencing Engine (EE) concerning recommended Learning Actions. This, in term, will affect the actuation of a learning scenario and the overall Learning Experience for a given learner through EE's decisions.

The LGE parameterization will be modularized to confront both the single learner scenario as well as the multi-learner (synchronous collaboration) scenario.

The LGE is further described in deliverable ***D6.2 - The MaTHiSiS Learning Graph Engine*** due in M24.

[17] For further information, see http://xapi.vocab.pub/datasets/adl/

Tier Name	**Learning Graph Engine**
Component Name	Learning Graph Engine (LGE)
Layer / Space	Back-end / Cloud Learner's Space
Inputs	
1	Source: Decision Support System Data: Trigger to update a personalised LG instance *(technically mediated via the LG lib API)* Schema: N/A
2	Source: Decision Support System Data: Updated personalised SLA competence weights *(technically mediated via the LG lib API)* Schema: JSON – detailed in D3.1 [7]
3	Source: LG lib API Data: In-session personalized Learning Graph Schema: GraphSON (graph-standardised JSON) – detailed in D3.3 [8]
Outputs	
1	Destination: LG lib API Data: Updated personalized Learning Graph instance Schema: GraphSON (graph-standardised JSON) – detailed in D3.3 [8]
2	Destination: Experience Engine Data: Learning Actions prioritised list, as recommended by the Learning Graph Engine *(technically mediated via the LG lib API)* Schema: JSON – detailed in D6.2 (to be released after D2.4)
High – Level Operations	
1	Personalisation and adaptation of the Learning Graph • Personalisation (offline): based on learner's profile and history (performance in previous interactions with the Platform) • Adaptation (online): based on affective state and interactions
2	Online adaptation in multi learner scenarios (synchronous collaboration)
Constraints	
1	None

Table 23 - Learning Graph Engine Tier - Learning Graph Engine

4.2.3 Experience Engine Tier

The Experience Engine is responsible to decide how a personalized Learning Graph will be materialized in the specific Learning Context, i.e. for a specific learner using a specific Platform Agent in a specific learning environment. During a Learning Session, the Experience Engine uses the most appropriate Learning Action given by the Learning Graph Engine. The Experience Engine then chooses the right materialization, starting by choosing the appropriated Learning Material. For this purpose, the availability of Platform Agents to be used by this learner is taken into account. When more than one Platform Agent is available, it also takes the most relevant in that context. At the end, the chosen Learning Material is then remotely controlled by the Experience Engine on the selected Platform Agent using the Experiencing Service (Section 6.2) as a proxy. There, the final step of the Learning Action materialization happens by the execution of the Learning Material and thus allowing the learner to interact with it in the real world.

In addition, the Experience Engine asks for validation of its choices. This is made by providing the tutor with this information through the Learning Experience Supervisor (see Section 5.5). The validation will be optional and, for each kind of possible choice (i.e. Learning Action, Learning Material and Platform Agent), if none are given, the Experience Engine will assume that the tutor has validated its proposal.

The EE is further detailed in the deliverable ***D3.5 - Experience Engine***[9].

Tier Name	**Experience Engine**
Component Name	Experience Engine (EE)
Layer / Space	Back-end / Cloud Learner's Space
Inputs	
1	Source: Learning Graph Engine Data: Personalized weighted learning graph with the best weighted LA Schema: JSON and GraphSON (graph-standardised JSON)
2	Source: LA lib API Data: Learning Actions description with their materialization Schema: JSON
3	Source: LM lib API Data: Learning Materials description with their identifiers, to be send to the right PA Schema: JSON
4	Source: PA lib API Data: Platform Agents description with their current status Schema: JSON
5	Source: Learner's Profile Repository, User Repository Data: Learner's profile Schema: JSON

6	Source: Learning Experience Supervisor Data: Tutor overridden choices (Learning Action, Learning Material, Platform Agent and some Context parameters) Schema: JSON
Outputs	
1	Destination: PA lib API Data: Learning Material commands, with the context, i.e. the Learning Session identifier Schema: JSON
2	Destination: Learning Experience Supervisor Data: Possible Learning Actions, Learning Materials and Platform Agents with default choices Schema: JSON
3	Destination: Learning Session lib API Data: Chosen LA, chosen LM and chosen PA Schema: JSON
4	Destination: Learner's Profile lib API Data: Choices made for history Schema: JSON
High – Level Operations	
1	Choose the appropriate Learning Material and Platform Agent based on the learner's personalized Learning Graph and the current learning context
2	Do the first step of the Learning Action materialization
3	Ask for the validation of its choices through a specific interface
Constraints	
1	The Experience Engine will consider that Learning Materials are already installed or available for the different Platform Agents. However, a specific Learning Material event will be thrown by the Experiencing Service if the targeted Learning Material is unavailable.

Table 24 - Experience Engine Tier - Experience Engine

4.2.4 Affect and Intent Recognition Tier

This tier is made of only one library implementing the Affect and Intent Recognition module.

4.2.4.1 AIR lib

The AIR lib implements all the necessary functions for the processing of affect and intent information received from the Platform Agent layer, namely the Sensorial Component tier (as described in

Section6.2.2), and the Interaction with Platform Agents (as described in Section 4.2.5). This component is also responsible of triggering the adaptation mechanism after the verification of the fulfilment of all the requirements (such as the occurrence of a key moment or the prediction of the affective state of the learner involved in the learning experience).

Library Name	**AIR lib**
Tier Name	Affect and Intent Recognition
Layer / Space	Back-end / Cloud Learner Space
Related repositories	
	N/A
High – Level Operations	
1	Synchronization of SC and IPA data
2	Key moment detector and adaptation trigger
High – Level Methods	
1	Receive affect state information based on SC component data
2	Receive affect state information based on interaction parameters
3	Send affect-related information to the DSS (multimodal fusion)
4	Trigger adaptation when a key moment is received (DSS)

Table 25 - Affect and Intent Recognition Tier - AIR lib

AIR lib API is the Affect and Intent Recognition exposing interface. It is a RESTful service upon the functionalities of the AIR lib.

API Name	**AIR lib API**
Tier Name	Affect and Intent Recognition
Layer / Space	Back-end / Cloud Learner Space
Inputs	
1	Source: Sensorial component Data: Affect-related features or/and labels based on facial expressions, gaze, skeleton motion, speech and sound-based emotion recognition, mobile inertial sensors Schema: JSON
2	Source: Interaction with Platform Agents (IPA) Data: Affect-related labels from learners' interaction with the Learning Materials

	Schema: JSON
Outputs	
1	Destination: Decision Support System (multimodal fusion) Data: Affect-related features or/and labels based on both behaviour (face, gaze, skeleton motion, speech/sound, inertial sensors) and interaction with the materials (IPA) Schema: JSON
2	Destination: Learning Graph Engine (adaptation mechanism) Data: Trigger based on key moments Schema: N/A. It's a direct call to the Learning Graph Engine
High – Level Operations	
1	Provide access to the Sensorial Component
2	Provide access to the Interaction with Platform Agent
3	Call multimodal fusion (affect recognition from sensorial components and/or interactions)

Table 26 - Affect and Intent Recognition Tier - AIR lib API

4.2.5 Interaction with Platform Agents Tier

This tier is made of only one library implementing the Interaction with Platform Agents module.

4.2.5.1 IPA lib

The IPA lib implements all the necessary functions for the analysis of features associated to affective and/or cognitive learner responses while interacting with Platform Agents and Learning Materials. This component is also in charge of the pre-analysis of the interaction parameters to detect key moments which will lead to the trigger of the adaptation mechanism.

Its output is a transformation/selection of the input features to improve the affective recognition of the learners. The features to be analysed could include learners' actions (e.g. learner interaction with NAOs or learning materials presented in a tablet), challenge levels imposed through different use cases, time needed to accomplish a task, complexity of the environment and any additional information related to the interactions of learners and the Platform. Inputs will also include parameters concerning the results (scores, events, etc.) of the learning actions performed. This data must be PA-independent, homogenous across PAs and measurable.

Library Name	**IPA lib**
Tier Name	Interaction with Platform Agents
Layer / Space	Back-end / Cloud Learner Space
Related repositories	
1	Name: Learning Record Store

	Data: Information about interaction between the learner and the PA Schema: JSON - xAPI statement
High – Level Operations	
1	Affect understanding through interactions
High – Level Methods	
1	Receive xAPI statements
2	Filter xAPI statements and store them in the LRS
3	Perform affect understanding based on interaction parameters
4	Inform the AIR lib when a key moment notification is received

Table 27 - Interaction with Platform Agents Tier - IPA lib

IPA lib API is the Interaction with Platform Agents exposing interface. It is a RESTful service upon the functionalities of the IPA lib.

API Name	**IPA lib API**
Tier Name	Interaction with Platform Agent
Layer / Space	Back-end / Cloud Learner Space
Inputs	
1	Source: Platform Agent / Learning Material Framework Data: Information about learner responses while interacting with learning materials Schema: JSON - xAPI statement (final structure to be defined in D6.4 (M24))
2	Source: Learner's Profile lib API Data: SLA scores history (performance for personalization and affective states for adaptation) Schema: JSON
Outputs	
1	Destination: AIR lib API Data: Affect-related labels from learners' interaction with the Learning Materials and key moments information Schema: JSON
High – Level Operations	
1	Provide access to the Learning Record Store (storage of xAPI statements)

Table 28 - Interaction with Platform Agents Tier - IPA lib API

4.2.6 Learning Session Tier

This tier is made of only one library implementing the concept of Learning Session.

4.2.6.1 Learning Session lib

The Learning Session lib provides all the necessary information for the processing of sessions by the MaTHiSiS components.

A Learning Session is associated to one and only one learner, giving also the learning context, i.e. the platform agent(s) that will be involved and the learning environment where the session will occur. Of course, a Learning Session is associated to a Learning Experience, and it's something the Learning Session is aware of.

In addition, a Learning Session store run-time information like the current affect state and performance of the learner and the current materialization of the Learning Graph.

Library Name	**Learning Session lib**
Tier Name	Learning Session
Layer / Space	Back-end / Cloud Learner Space
Related repositories	
1	Name: User Space Repositories Data: Learning Sessions information Schema: JSON – detailed in D7.2 [17]
High – Level Operations	
1	Declare Learning Sessions
2	Provide all information about a Learning Session
3	Give the current status of the Learner during the session
4	Give the current Learning Material executed
High – Level Methods	
1	Create a new Learning Session from the Front-End
2	Destroy an existing Learning Session
3	Update an existing Learning Session
4	Materialize a Learning Session
5	Finalize a specific Learning Session: interrupt or end it

Table 29 - Learning Session Tier - Learning Session lib

Learning Session lib API is the Learning Session exposing interface. It is a RESTful service upon the functionalities of the Learning Session lib.

API Name	**Learning Session lib API**
Tier Name	Learning Session
Layer / Space	Back-end / Cloud Learner Space
Inputs	
1	Source: Front-end / Learning Experience Supervisor Data: Learning Session definition Schema: JSON – detailed in D7.2 [17]
2	Source: Experience Engine Data: Update of the current Learning Action and Learning Material Schema: JSON – detailed in D7.2 [17]
3	Source: IPA lib API Data: Update of the last performance status of the learner coming from the new interactions of the learner with the current Learning Material Schema: JSON – detailed in D7.2 [17]
4	Source: Decision Support System Data: Update of the affect states probabilities coming from the multimodal fusion Schema: JSON – detailed in D7.2 [17]
5	Source: Learning Action Materialization Data: Learning Material execution information (e.g. started successfully, not found, etc.) Schema: JSON - will be detailed in D3.6 (M24)
Outputs	
1	Destination: Decision Support System Data: Information needed to do the personalization and adaptation, i.e. the learner associated to the Learning Session Schema: JSON – detailed in D7.2 [17]
2	Destination: Learning Graph Engine Data: the Learning Graph instance associated to the Learning Session, in order to do the adaptation of the LGI as a whole Schema: JSON – detailed in D7.2 [17]
3	Destination: Experience Engine Data: Information needed to do the materialization, i.e. the current Learning Action, the current Learning Material, the learner, the platform agent and the learning graph instance associated to the Learning Session

	Schema: JSON – detailed in D7.2 [17]
High – Level Operations	
1	Provide access to and from Learning Session lib

Table 30 - Learning Session Tier - Learning Session lib API

4.3 User Space

Use Space is another important element of the MaTHiSiS system by providing functionalities for the management of user information as well as the learner profile data that are used by those components in charge of the personalization / adaptation of the different learning processes supported by MaTHiSiS. In this space, we also manage concepts related to the users: Classrooms, Learning Environments and Platform Agents. Next we describe in detail each of the tier composing the User Space (US).

4.3.1 User Profile Tier

This tier is made of only one library implementing the concept of user profile to store information related to them and to be used by other components.

4.3.1.1 User Profile lib

The User Profile lib contains all the necessary information about the users' profile. The goal is to be able to identify them and know their properties. A user profile contains the user's credentials, his/her profile information (first name, last name, birth date, etc.), his/her role(s) and the social groups he/she participates. For each role, it is included all privileges associated with it (a list will be prepared containing e.g. creation of graphs, creation of material, creation of social network group, modification of graphs, access to contents).

Library Name	**User Profile lib**
Tier Name	User Profile
Layer / Space	Back-end / User Space
Related repositories	
1	Name: User Space Repositories Data: User Profile information Schema: JSON
High – Level Operations	
1	Declare User Profile
2	Provide access and update functions for the User Profile
3	Manage the credentials related to a User Profile
4	Give the profile information related to a User Profile
5	Give the role(s) and associated privileges related to a User Profile

6	Give the social groups participation information related to a User Profile
High – Level Methods	
1	Create a new User Profile
2	Destroy an existing User Profile
3	Update an existing User Profile

Table 31 - User Profile Tier - User Profile lib

User Profile lib API is the User Profile exposing interface. It is a RESTful service upon the functionalities of the User Profile lib. The deliverable ***D3.7 - Learner's Profile Repository***[10] gives more information on this concrete API.

API Name	**User Profile lib API**
Tier Name	User Profile
Layer / Space	Back-end / User Space
Inputs	
1	Source: Front-end / User Management Data: User Profile description Schema: JSON
Outputs	
1	Destination: Front-end/ User Management Data: User Profile information Schema: JSON
High – Level Operations	
1	Provide access to and from User Profile lib
2	Provide User Profile information: credentials, profile information, roles and associated privileges and information about the participation to social groups

Table 32 - User Profile Tier - User Profile lib API

4.3.2 Learner's Profile Tier

This tier is made of only one library implementing the concept of Learner's profile to store information related to them and to be used by other components.

4.3.2.1 Learner's Profile lib

The Learner's Profile lib contains all the necessary information about the learners' profile. The goal is to be able to identify them and know their properties. A learner's profile contains the learner's demographics and preferences, his/her special needs (mainly learning difficulties) and his/her previous performances (final status of his/her on-going Learning Experiences).

The learner's profile is updated with the information provided by the DSS (affective state and performance), MaTHiSiS Front-end (information added manually by tutors/caregivers) and the Cloud Learner Space, such as references to the last Learning Experience performance.

Library Name	**Learner's Profile lib**
Tier Name	Learner's Profile
Layer / Space	Back-end / User Space
Related repositories	
1	Name: User Space Repositories Data: Learner's profile information (demographics, preferences and special needs) Schema: JSON
High – Level Operations	
1	Declare Learner's Profile
2	Provide access and update functions for the Learners' Profile
3	Give the demographics and preferences related to a Learner's Profile
3	Give the special needs related to a Learner's Profile
3	Give the use history of the platform related to a Learner's Profile
High – Level Methods	
1	Create a new Learner's Profile
2	Update an existing Learner's Profile
Constraints	
1	Security and legal aspects must be ensured for all countries

Table 33 - Learner's Profile Tier - Learner's Profile lib

Learner's Profile lib API is the Learner's Profile exposing interface. It is a RESTful service upon the functionalities of the Learner's Profile lib. The deliverable ***D3.7 - Learner's Profile Repository***[10] gives more information on this concrete API.

API Name	**Learner's Profile lib API**
Tier Name	Learner's Profile
Layer / Space	Back-end / User Space
Inputs	
1	Source: Front-end / Learner Profile Management Data: Learner's profile information (demographics, preferences and special needs) Schema: JSON
2	Source: Decision Support System Data: Affective States and performance updates for history Schema: JSON
3	Source: Learning Graph Data: Personalized Learning Graphs updates for history Schema: GraphSON (graph-standardised JSON)
4	Source: Experience Engine Data: Learning Actions, Learning Materials and Platform Agents choices for history Schema: JSON
Outputs	
1	Destination: Front-end / Learner Profile Management Data: Learner's profile information (demographics, preferences and special needs) and history Schema: JSON, GraphSON (graph-standardised JSON)
2	Destination: Decision Support System Data: Learner's profile information and history, used for the personalization and adaptation Schema: JSON
3	Destination: Learning Graph Engine Data: Learner's profile information Schema: JSON
4	Destination: Experience Engine Data: Learner's profile information used to determine the best materialization to use Schema: JSON
High – Level Operations	
1	Provide access to and from Learner's Profile lib

2	Provide Learner's Profile information: demographics, preferences, special needs and use history of the platform by the learner

Table 34 - Learner's Profile Tier - Learner's Profile lib API

4.3.3 Classroom Tier

This tier is made of only one library implementing the concept of Classroom to store information related to them and to be used by other components.

4.3.3.1 Classroom lib

The Classroom lib contains all the necessary information about the Classrooms. The goal is to be able to identify them and know their properties. A Classroom represents the link that exists between a set of Learners, a set of Tutors and a set of Learning Environments. Front-end components involved in the management of Learning Experiences of the learners need such information to propose a better usability.

For example, a Tutor will have the list of all Classrooms he is associated to. When he prepares a new Learning Session for a specific Classroom, the associated Learners and Learning Environments will be stressed out in the Front-end to help him. When a Tutor wants to see the progression through Learning Experiences, he will have the possibility to see it for a specific Learner or for a specific Classroom.

Library Name	**Classroom lib**
Tier Name	Classroom
Layer / Space	Back-end / User Space
Related repositories	
1	Name: User Space Repositories Data: Classroom information Schema: JSON
High – Level Operations	
1	Declare Classrooms
2	Provide access and update functions for the Classrooms
3	Give the set of Learners associated to the Classrooms
4	Give the set of Tutors associated to the Classrooms
5	Give the set of Learning Environments associated to the Classrooms
High – Level Methods	
1	Create a new Classroom
2	Destroy an existing Classroom
3	Update an existing Classroom

Table 35 - Classroom Tier - Classroom lib

Classroom lib API is the Classroom exposing interface. It is a RESTful service upon the functionalities of the Classroom lib. The deliverable ***D7.2 - MaTHiSiS platform, 1st release***[17] gives more information on this concrete API.

API Name	**Classroom lib API**
Tier Name	Classroom
Layer / Space	Back-end / User Space
Inputs	
1	Source: Front-end / Classroom management Data: Classroom description Schema: JSON
Outputs	
1	Destination: Front-end / Classroom management Data: Classrooms information Schema: JSON
High – Level Operations	
1	Provide access to and from Classroom lib
2	Provide Classrooms properties: set of Learners, set of Tutors and set of Learning Environments

Table 36 - Classroom Tier - Classroom lib API

4.3.4 Learning Environment Tier

This tier is made of only one library implementing the concept of Learning Environment to store information related to them and to be used by other components.

4.3.4.1 Learning Environment lib

The Learning Environment lib contains all the necessary information about the Learning Environments. The goal is to be able to identify them and know their properties. Back-end components involved in the personalization and adaptation of the Learning Experiences of the learners need such information to have better vision of the location where the sessions take place.

For example, the materialization need to know if we can use sound in a specific Learning Environment, and for that, the Experience Engine uses the Learning Environment lib to get the related information to take its decisions.

Library Name	**Learning Environment lib**
Tier Name	Learning Environment

Layer / Space	Back-end / User Space
Related repositories	
1	Name: User Space Repositories Data: Learning Environment information Schema: JSON
High – Level Operations	
1	Declare Learning Environments
2	Provide access and update functions for the Learning Environments
3	Give the properties of the Learning Environments through LAO Categories
High – Level Methods	
1	Create a new Learning Environment
2	Destroy an existing Learning Environment
3	Update an existing Learning Environment

Table 37 - Learning Environment Tier - Learning Environment lib

Learning Environment lib API is the Learning Environment exposing interface. It is a RESTful service upon the functionalities of the Learning Environment lib. The deliverable ***D7.2 - MaTHiSiS platform, 1st release***[17] gives more information on this concrete API.

API Name	**Learning Environment lib API**
Tier Name	Learning Environment
Layer / Space	Back-end / User Space
Inputs	
1	Source: Front-end / Platform Configurator Data: Learning Environments description Schema: JSON
2	Source: Learning Actions Ontology Data: Types of Learning Environment, types of properties of the Learning Environments Schema: OWL – detailed in D3.5 [9]
Outputs	
1	Destination: Front-end / Platform Configurator

	Data: Learning Environments information Schema: JSON
High – Level Operations	
1	Provide access to and from Learning Environment lib
2	Provide Learning Environments properties

Table 38 - Learning Environment Tier - Learning Environment lib API

4.3.5 Platform Agent Tier

This tier is made of only one library implementing the concept of Platform Agent to store information related to them and to be used by other components.

4.3.5.1 PA lib

The PA lib contains all the necessary information about the Platform Agents. The goal is to be able to identify them and know their capabilities. Back-end components involved in the personalization and adaptation of the Learning Experiences of the learners need such information to have better vision of the devices that are used.

For example, the materialization need to know the compatibilities between Learning Materials and Platform Agents, and for that, the Experience Engine uses the PA lib to get the related information to take its decisions.

In addition, the PA lib allow to send commands to the devices representing the Platform Agents, thus allowing to communicate with them through a simple and dedicated API.

Library Name	**PA lib**
Tier Name	Platform Agent
Layer / Space	Back-end / User Space
Related repositories	
1	Name: User Space Repositories Data: Platform Agent information Schema: JSON – detailed in D7.2 [17]
High – Level Operations	
1	Declare Platform Agents
2	Provide access and update functions for the PAs
3	Give the availability of the PAs for the Learning Materials through LAO Categories
4	Command the Platform Agent through Experiencing Service

High – Level Methods	
1	Create a new Platform Agent from the Front-End
2	Destroy an existing Platform Agent
3	Update an existing Platform Agent
4	Send the Start / Stop command for a running LM on an existing Platform Agent

Table 39 - Platform Agent Tier - PA lib

PA lib API is the Platform Agent exposing interface. It is a RESTful service upon the functionalities of the PA lib. The deliverable ***D7.2 - MaTHiSiS platform, 1st release***[17] gives more information on this concrete API.

API Name	**PA lib API**
Tier Name	Platform Agent
Layer / Space	Back-end / User Space
Inputs	
1	Source: Front-end / Platform Agent Configurator Data: Platform Agent definition. Device identification and capabilities Schema: JSON – detailed in D7.2 [17]
2	Source: Experience Engine Data: Commands to apply on the PA Schema: JSON – detailed in D7.2 [17]
3	Source: Front-end / Platform Configurator Data: Context information (e.g. device type, capabilities, location, roles) Schema: JSON – detailed in D7.2 [17]
4	Source: Platform Collaboration Data: Parameterised features of the PAs involved (for synchronous collaboration), parameterised features of a new PA (for asynchronous collaboration) Schema: *To be defined later in the project*
5	Source: Learning Actions Ontology Data: Types of Platform Agents, types of attributes associated to Platform Agents Schema: OWL – detailed in D3.5 [9]
Outputs	
1	Destination: Front-end

	Data: Information about PAs, to be used in different places of the front-end Schema: JSON – detailed in D7.2 [17]
2	Destination: Experiencing Service Data: Commands to apply on the PA Schema: JSON – detailed in D7.2 [17]
3	Destination: Decision Support System Data: Parameterised features of the PAs involved (for synchronous collaboration), parameterised features of a new PA (for asynchronous collaboration) Schema: *To be defined later in the project*
High – Level Operations	
1	Provide access to and from PA lib
2	Command the Experiencing Services residing on the PAs
3	Provide PA capabilities

Table 40 - Platform Agent Tier - PA lib API

4.4 Learning Content Space

The Learning Content Space includes all libraries related to the learning content management, as represented in MaTHiSiS.

4.4.1 Learning Graph Tier

The Learning Graph Tier provides the tools to create, access, manage and update the learning content, i.e. the knowledge/skills/competences to be acquired by a learner during the MaTHiSiS-enable Learning Experience. This includes SLAs, learning goals and learning graphs.

4.4.1.1 LGI lib

The LG lib contains the tools to create, access and modify a) Learning Graphs (LGs) per any given learning scenario and b) personalised LG instances per learner for any given scenario. LGs model the knowledge/skills to be learnt by the learners in the given scenario, in terms of SLAs and learning goals and the weighted relations between them. Personalised LG instances in addition model also the competence weight of a learner over these SLAs and learning goals.

The serialisation of LGs in the MaTHiSiS DB, access to and from existing LGs in the DB, deletion of LGs is possible through the LG lib exposing API. The LG data model is outlined in Section 7 and further analysed in the deliverable ***D3.3 - The MaTHiSiS Learning Graphs***[8].

Library Name	**LG lib**
Tier Name	Learning Graph
Layer / Space	Back-end / User Space, Back-end / Learning Content Space

Related repositories	
1	Name: Learning Content Space Repositories Data: Unpersonalized Learning Graph Schema: GraphSON (graph-standardised JSON) – detailed in D3.3 [8]
2	Name: User Space Repositories Data: Standing personalized Learning graph instance, runtime personalized learning graph instances Schema: GraphSON (graph-standardised JSON) – detailed in D3.3 [8]
High – Level Operations	
1	Model Learning Graphs
2	Provide access and update tools for the Learning Graphs (unpersonalized models and personalized instances per user)
High – Level Methods	
1	Create a new Learning Graph
2	Create a personalized instance of a Learning Graph (per user)
3	Get/set an existing Learning Graph or Learning Graph instance
4	Get/set vertices and edges of a Learning Graph (or LG instance) and their attributes

Table 41 - Learning Graph Tier - LGI lib

LG lib's exposing API provides RESTful services for creating, accessing and modifying Learning Graphs (LGs). The LG lib API is further detailed in deliverable ***D3.3 - The MaTHiSiS Learning Graphs***[8] and will be further refined in deliverable ***D3.4 - The MaTHiSiS Learning Graphs*** due in M24.

API Name	**LG lib API**
Tier Name	Learning Graph
Layer / Space	Back-end / User Space, Back-end / Learning Content Space
Inputs	
1	Source: Learning Content Editor Data: Unpersonalised Learning Graph (learning goals definition, relation to SLAs, significance weights of relations) Schema: GraphSON (graph-standardised JSON) – detailed in D3.3 [8]
2	Source: SLA lib Data: Unpersonalised SLAs definition information (label, ID), personalised SLA instance

	information (label, ID, competence score) Schema: JSON – detailed in D3.1 [7]
3	Source: Decision Support System Data: Trigger of the personalisation and adaptation process on the Learning Graph Engine Schema: N/A
4	Source: Learning Graph Engine Data: Updated personalized learning graph instance Schema: GraphSON (graph-standardised JSON) – detailed in D3.3 [8]
Outputs	
1	Destination: Learning Graph Engine Data: In-session personalized learning graph instance Schema: GraphSON (graph-standardised JSON) – detailed in D3.3 [8]
2	Destination: Experience Engine Data: Learning Actions prioritised list, as recommended by the Learning Graph Engine Schema: JSON – detailed in D6.2 (to be released after D2.4)
High – Level Operations	
1	Provide access to and from LG lib
2	Create, modify and delete LG data models in the MaTHiSiS DB
3	Mediate triggering the LGE and passing its results to the Experience Engine

Table 42 - Learning Graph Tier - LG lib API

4.4.1.2 SLA lib

The SLA lib contains the tools to create, access and modify a) Smart Learning Atoms (SLAs) and b) personalised SLA instances per learner. SLAs model the primordial, atomic pieces of knowledge/skills/competences to be learnt by the learners and can pertain to one or several learning scenarios. Unpersonalised SLAs in addition model also the relations of SLAs to specific learning actions, again per scenario or for several different scenarios.

The serialisation of SLAs in the MaTHiSiS DB, access to and from existing SLAs in the DB and the deletion of SLAs is possible through the SLA lib exposing API. The SLA data model is outlined in Section 7 and further analysed in the deliverable ***D3.1 - The MaTHiSiS Smart Learning Atoms***[7].

Library Name	**SLA lib**
Tier Name	Learning Graph
Layer / Space	Back-end / User Space, Back-end / Learning Content Space

Related repositories	
1	Name: Learning Content Space Repositories Data: Unpersonalized SLAs Schema: JSON – detailed in D3.1 [7]
2	Name: User Space Repositories Data: Standing personalized SLA instance, runtime personalized SLA instances Schema: JSON – detailed in D3.1 [7]
High – Level Operations	
1	Model SLAs
2	Provide access and update tools for the SLAs (unpersonalized models and personalized instances per user)
3	Provide tools to attach Learning Actions to unpersonalised SLA models
High – Level Methods	
1	Create a new SLA
2	Create a personalized instance of a SLA (per user)
3	Get/set an existing SLA or SLA instance
4	Get/set attributes of existing SLAs or SLA instances (including attached Learning Actions)

Table 43 - Learning Graph Tier - SLA lib

SLA lib's exposing API provides RESTful services for creating, accessing and modifying Smart Learning Atoms (SLAs). The SLA lib API is further detailed in deliverable ***D3.1 - The MaTHiSiS Smart Learning Atoms*** [7] and will be further refined in deliverable ***D3.2 - The MaTHiSiS Smart Learning Atoms*** due in M24.

API Name	**SLA lib API**
Tier Name	Learning Graph
Layer / Space	Back-end / User Space, Back-end / Learning Content Space
Inputs	
1	Source: Learning Content Editor Data: Unpersonalised SLAs (definition, including attachment to LAs) Schema: JSON – detailed in D3.1 [7]
2	Source: Decision Support System Data: Updated personalized SLA instances competence weights

	Schema: Single real value – detailed in D3.1 [7]
Outputs	
1	Destination: LG lib API Data: Unpersonalised SLAs definition information (label, ID), personalised SLA instance information (label, ID, competence score) Schema: JSON – detailed in D3.1 [7]
2	Destination: Decision Support System Data: In-session personalized SLA instance competence weights Schema: JSON – detailed in D3.1 [7]
High – Level Operations	
1	Provide access to and from SLA lib
2	Create, modify and delete SLA data models in the MaTHiSiS DB

Table 44 - Learning Graph Tier - SLA lib API

4.4.2 Learning Action Tier

The Learning Action Tier provides the tools to create, access, manage and update Learning Actions for a specific Learning Experience, as well as to access and attach information relevant to the materialisation of Learning Actions.

4.4.2.1 **LA lib**

LA lib contains the tools to model Learning Actions and their materializations. The LA data model allows reusability among SLAs.

From the LA lib exposing API, we are also able to build LA objects that use this data format. The LA data model is outlined in Section 7 and further analysed in deliverable ***D3.5 - Experience Engine***[9].

The materialization is created through the Learning Action Editor described in Section 5.1.1.3. The materialization is stored in the Learning Action data model as a binary content.

Library Name	**LA lib**
Tier Name	Learning Action
Layer / Space	Back-end / Learning Content Space
Related repositories	
1	Name: Learning Content Space Repositories Data: Learning Actions description Schema: JSON – detailed in D3.5 [9]
High – Level Operations	

1	Instantiate learning actions
2	Attach materializations to Learning Actions
High – Level Methods	
1	Create a new learning action by instantiating a type of LA from LAO
2	Destroy an existing learning action
3	Update an existing learning action
4	Get/set materialization of an existing learning action

Table 45 - Learning Action Tier - LA lib

LA lib's exposing API provides RESTful services upon the functionalities of the LA lib.

API Name	LA lib API
Tier Name	Learning Action
Layer / Space	Back-end / Learning Content Space
Inputs	
1	Source: Learning Content Editor Data: Learning Actions definition, associated with Learning Action materialisation attributes Schema: JSON – detailed in D3.5 [9]
2	Source: LM lib API Data: Available Learning Materials Schema: JSON – detailed in D3.5 [9]
3	Source: Learning Actions Ontology Data: Types of Learning Actions, types of attributes associated to Learning Actions Schema: OWL – detailed in D3.5 [9]
Outputs	
1	Destination: Learning Content Editor Data: Existing Learning Actions description Schema: JSON – detailed in D3.5 [9]
2	Destination: LG lib API Data: Learning Actions associated to the SLA instances in the Learning Graph instance associated to the current Learning Session

	Schema: JSON – detailed in D3.5 [9]
3	Destination: Experience Engine Data: Learning Actions description associated to the current Learning Session Schema: JSON – detailed in D3.5 [9]
High – Level Operations	
1	Provide access to and from LA lib

Table 46 - Learning Action Tier - LA lib API

4.4.3 Learning Material Tier

The Learning Material Tier provides the tools to declare, access, manage and update Learning Materials, as well as to create, access, manage and update identifiers allowing the link between existing external educational material (e.g. a learning game, a presentation, an e-learning course, etc.) and the Learning Materials as used within MaTHiSiS.

4.4.3.1 **LM lib**

LM lib contains the tools to model Learning Materials and their identifiers. The LM data model allow a reusability among LAs.

From the LM lib exposing API, we are also able to build LM objects that use this data format. The LM data model is outlined in Section 7 and further analysed in deliverable ***D3.5 - Experience Engine*** [9].

The LM identifiers can be of different kind, depending on the targeted PA. Indeed, how an existing education material can be used is dependent on the device. For example, a digital application is not packed the same way on mobile running iOS or Android, or more obviously on the NAO robot. LM identifiers help to link the abstract Learning Material to the concrete application or physical object that can be used in the real world by the learners.

Library Name	**LM lib**
Tier Name	Learning Material
Layer / Space	Back-end / Learning Content Space
Related repositories	
1	Name: Learning Content Space Repositories Data: Learning Materials description Schema: JSON – detailed in D3.5 [9]
High – Level Operations	
1	Declare Learning Materials
2	Link external educational materials to Learning Materials through identifiers
High – Level Methods	
1	Declare a new learning material by instantiating a type of LM from LAO

2	Destroy an existing learning material
3	Update an existing learning material
4	Create a new learning material identifier by instantiating a type of LM identifier from LAO
5	Destroy an existing learning material identifier
6	Update an existing learning material identifier

Table 47 - Learning Material Tier - LM lib

LM lib's exposing API provides RESTful services upon the functionalities of the LM lib.

API Name	**LM lib API**
Tier Name	Learning Material
Layer / Space	Back-end / Learning Content Space
Inputs	
1	Source: Front-end / Learning Material Configurator Data: Learning Materials definition, the Learning Material identifiers such as the location and name of the associated package. Schema: JSON – detailed in D3.5 [9]
2	Source: Learning Actions Ontology Data: Types of Learning Materials and types of Learning Material identifiers Schema: OWL – detailed in D3.5 [9]
Outputs	
1	Destination: Learning Content Editor Data: Existing Learning Materials description to be used in Learning Actions materializations Schema: JSON – detailed in D3.5 [9]
2	Destination: Experience Engine Data: Learning Materials and its information, especially the Learning Material Identifier involved in the selected Learning Action materialization associated to the current Learning Session. Schema: JSON – detailed in D3.5 [9]
3	Destination: Front-end Data: Information about LMs, to be used in different places of the front-end Schema: JSON – detailed in D3.5 [9]
High – Level Operations	

1	Provide access to and from LM lib

Table 48 - Learning Material Tier - LM lib API

5. Front-end layer

The MaTHiSiS Front-end layer is in charge of providing all software user interfaces needed to allow all kind of users to work with the MaTHiSiS Platform. The front-end will provide a set of tools for each kind of functionalities: management of users' account and learners' profile, link with social networks, creation of learning content, monitor and control of Learning Experience with their sessions, visualization of analytics related to learners, declaration of Platform Agents and learning environments.

Above all these functionalities, the front-end will secure the system by providing an authentication mechanism.

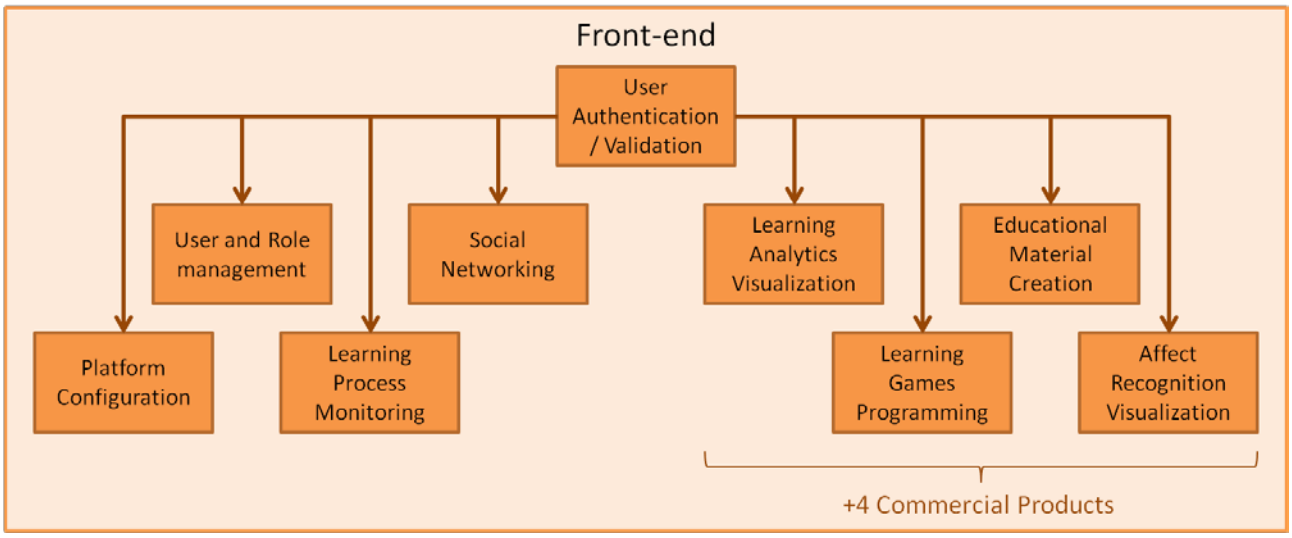


Figure 5 - MaTHiSiS Front-end layer

5.1 + 4 Commercial Products

MaTHiSiS Platform provides a central application product for delivering innovative ICT-enhanced Learning Experiences at educational institutions, vocational training facilities and informal settings. Within the MaTHiSiS context, the platform is designed as a complete infrastructure that enables the delivery of hypermedia adaptive content from different sources (open- or closed-content solutions) to a number of Platform Agents to support personalised Learning Experience. This deliverable sets an initial look at the MaTHiSiS Software Platform and its related four commercial products, its dependencies to different components (including hardware components) so that the functionality and the architecture of the system is realized and will be used to define the value proposition of the platform that will facilitate the commercialization and further exploitation of the system after the project end (these actions will be conducted as part of WP1 work plan).

The following sub-Sections describe a number of subcomponents introduced within the MaTHiSiS platform formed within number of Tiers. As per initial plan, four commercial out of the box software tools are envisioned for future commercial exploitation, namely:

- Educational Material Creation Tool
- Learning Analytics and Visualisation Tool
- Affect Recognition Software
- Learning Games Programming Tool

5.1.1 Educational Material Creation Tier

This tier has the responsibility to provide ways of creating the learning content that will be used during Learning Experiences. In order to achieve that goal, the *Learning Content Editor* has been designed to be a user-friendly tool that will effectively allow the creation of learning contents according to MaTHiSiS main concepts: Learning Graphs, SLAs, Learning Goals and Learning Actions with their materializations. This way, the Learning Content Editor offers an easy and straight forward solution for describing a learning scenario in the MaTHiSiS platform. However, the creation and configuration of the Learning Materials, more precisely the link between a LM and concrete applications or physical objects, is done through another tool name the *Learning Material Configurator*, available in the main MaTHiSiS Front-end web-application.

This application will be connected to the MaTHiSiS Learning Content Space services in order to retrieve the lists of already existing Learning Graphs, SLAs, Learning Actions and Learning Materials. Whenever the user will create a piece of learning content, he/she will have the possibility to upload it to this space to make it available and usable during a Learning Session.

Figure 7 and Figure 8 show initial versions of user interface of the Learning Content Editor. The first one is the log in screen and the second one the home page of a logged in user.

A first description of this tool is provided in ***D3.1 - MaTHiSiS Smart Learning Atoms***[7], ***D3.3 - MaTHiSiS Learning Graphs***[8] and ***D3.5 - Experience Engine***[9]. Next versions of this tool will be presented in ***D3.9 - MaTHiSiS Frontend Component*** and in ***D3.10 - MaTHiSiS Frontend Component*** due respectively in M24 and M33.

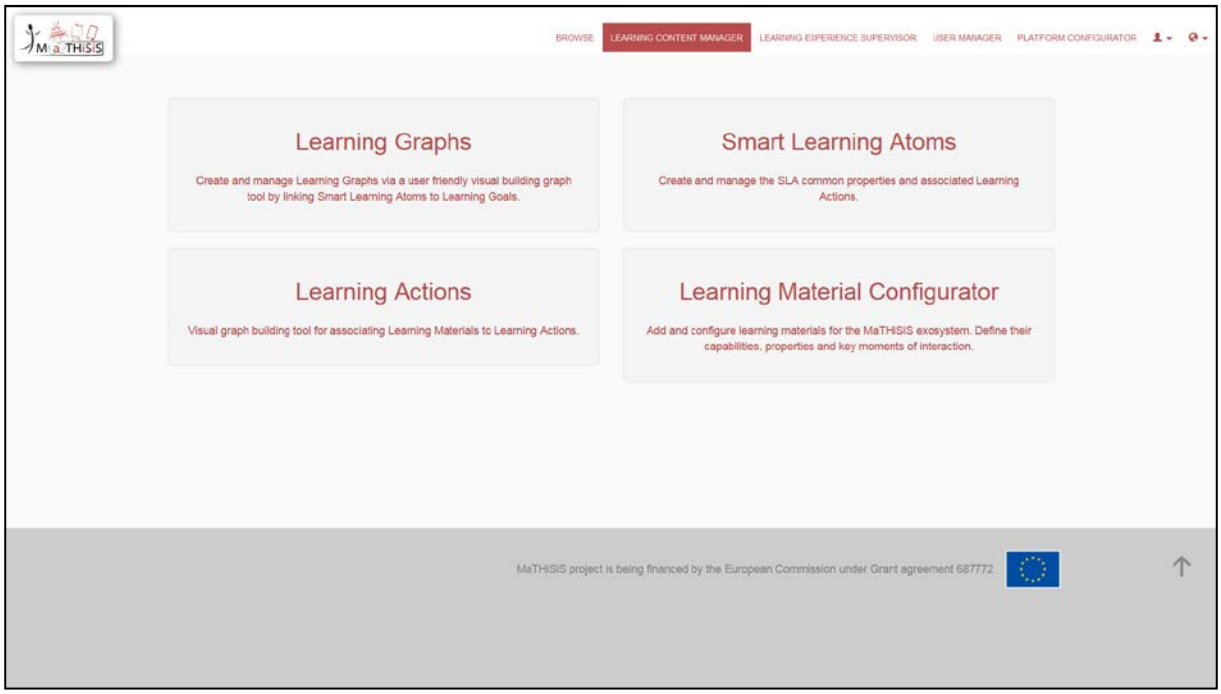


Figure 6 - MaTHiSiS Front-end initial version - Learning Content Manager

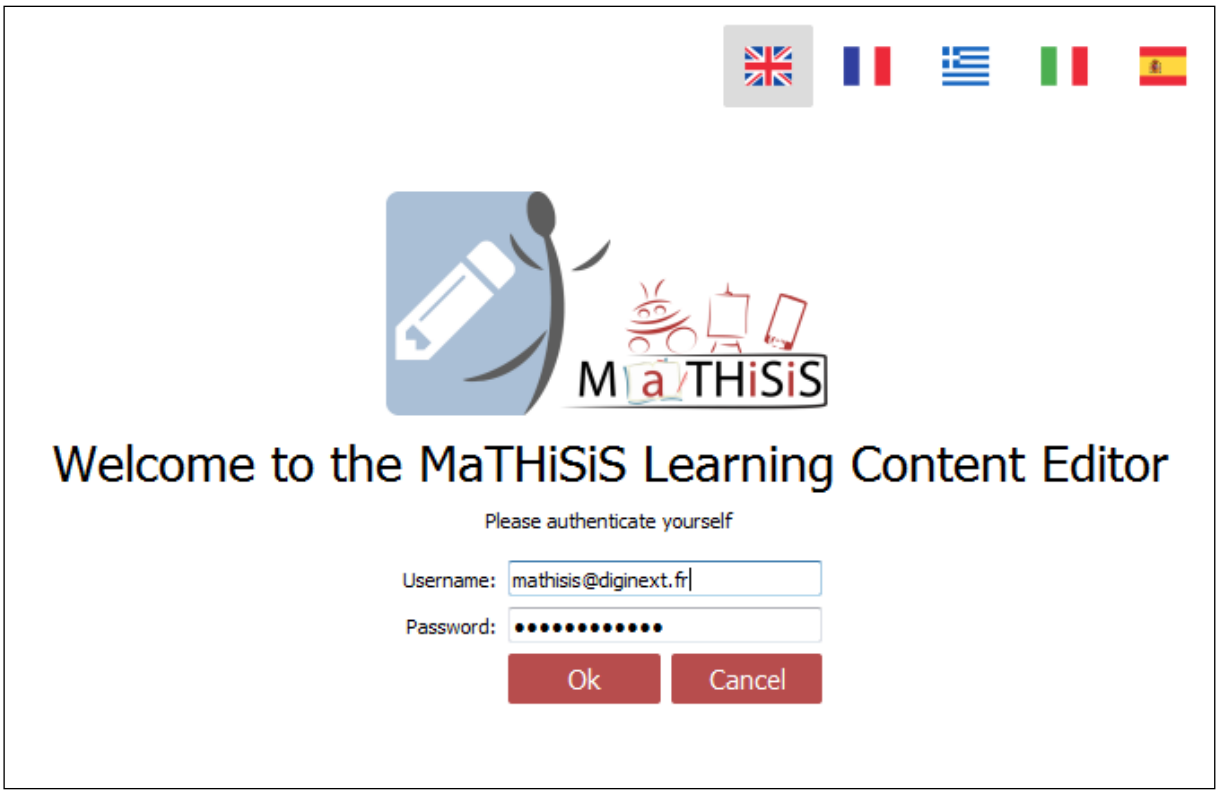


Figure 7 - LCE initial version - Learning Content Editor - Log in

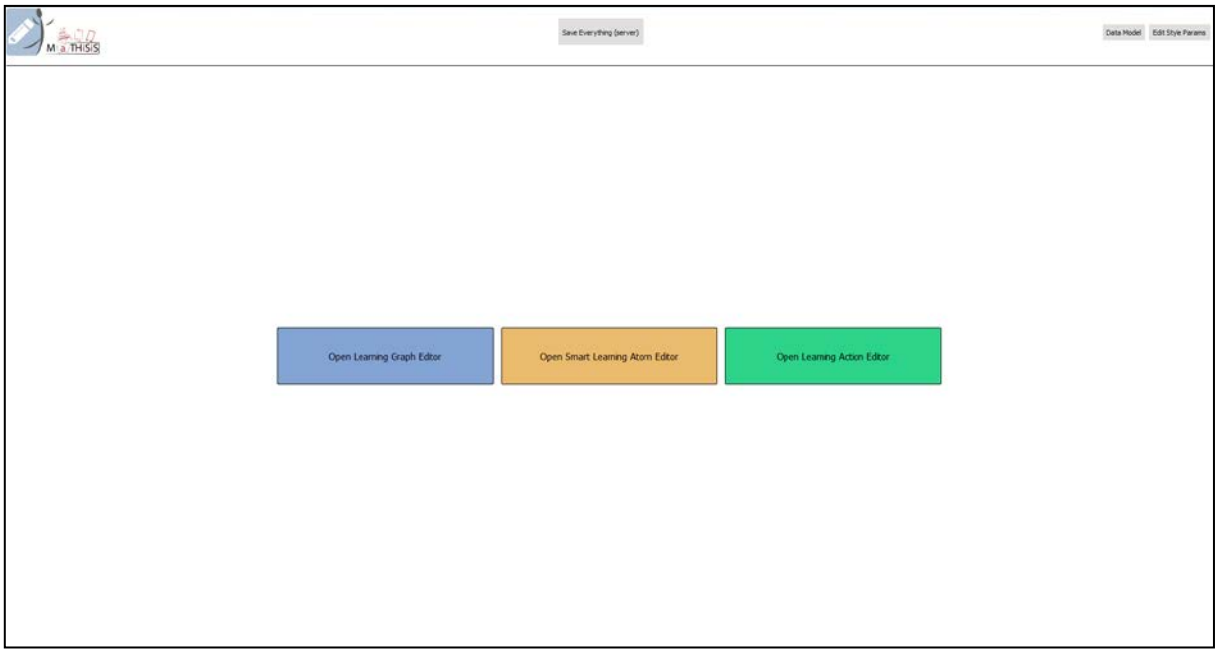


Figure 8 - LCE initial version - Learning Content Editor - Home

This application asks the user to connect to his/her MaTHiSiS user account in order to get his/her role and rights. Depending on them, functionalities will be enabled / disabled.

As shown in Figure 6, and if the LCE is installed on the same device, the MaTHiSiS front-end web application allows to start the LCE directly in the right editor, possibly editing the chosen entity. It is foreseen to also avoid the log-in step of the LCE by retrieving the authentication information from the web

application. More information about the link between the MaTHiSiS web application and the Learning Content Editor will be given in the ***D3.9 - MaTHiSiS Frontend component*** due in M24.

A help mechanism provides assistance to users, in order to explain how to use the LCE and how to build Learning Graphs, Smart Learning Atoms, Learning Actions and their materialization coming from existing educational materials. In addition, the application proposes tutorials and examples, and allows to build new LG/SLA/LA starting from them.

Tool Name	**Learning Content Editor**
Layer / Tier	Front-end / Educational Material Creation
Inputs	
1	Source: Open API - User Space Data: Authentication validation Schema: JSON
2	Source: Open API - Learning Content Space Data: Learning Graphs, SLAs, Learning Actions and Learning Materials Schema: JSON
Outputs	
1	Destination: Open API - User Space Data: User credentials for authentication Schema: JSON
2	Destination: Open API - Learning Content Space Data: Learning Graphs, SLAs, Learning Actions and Learning Materials Schema: JSON
Roles	
Administrator	To manage the set of existing learning content.
Tutor	To create new learning content or to manage his/her existing learning content.
High – Level Features	
1	Visual and user-friendly graph edition for Learning Graphs and Learning Actions materialization.
2	Connected to MaTHiSiS back-end to access the lists of existing pieces of learning content.
3	Connected to MaTHiSiS back-end to store and share the learning content owned by the user.
Constraints	

1	Performances need to fulfil specified KPIs[18].
2	User role and rights must be taken into account to restrict or to limit the functionalities

Table 49 - Tier - Educational Material Creation - Learning Content Editor

5.1.1.1 Learning Graph Editor component

This editor is dedicated to the creation and edition of Learning Graphs. It's a very simple and a user-friendly visual graph building tool that allows the linkage of SLAs and Learning Goals. This editor allows using existing SLAs or creating new empty ones, editable later through the SLA Editor. It is also possible to manage the visibility/rights of the Learning Graph for other users. Different kind of rights are applicable there: public in read only mode, so only usable as is for new Learning Experiences, public in read/write mode, so editable and usable for new Learning Experiences or private, so invisible to other users.

Component Name	**Learning Graph Editor**
Tier / Tool	Educational Material Creation / Learning Content Editor
Edited properties	
1	Learning Graph name and description
2	Set of SLAs used in the edited Learning Graph
3	Set of Learning Goals used in the edited Learning Graph
4	Learning Graph structure, i.e. weighted connections between SLAs and Learning Goals
Inputs	
1	Optional: existing Learning Graph for edition
2	Set of existing SLAs usable in the edited Learning Graph
Outputs	
1	Updated Learning Graph (either valid or under construction)
High – Level Features	
1	Visual and user-friendly graph editor
2	Connected to MaTHiSiS back-end to access the set of existing SLAs
3	Connected to MaTHiSiS back-end to optionally get an existing Learning Graph, to share newly created or modified Learning Graphs
Functionalities	
1	Common actions 'New', 'Open', 'Save' and 'Save As' for Learning Graph

[18] MaTHiSiS KPIs, MaTHiSiS DoA part B pages 8 and 9

2	Create and Edit properties and structure of a new Learning Graph
3	View and Edit properties and structure of an existing Learning Graph
4	View the set of already existing SLAs
5	Use or Duplicate an existing SLA
6	Creation of a new empty SLA
7	Launch the edition of a SLA, through the SLA Editor
8	View the summary of an existing SLA
9	Create a new Learning Goal
10	Duplicate an existing Learning Goal
11	View and Edit properties of a Learning Goal
12	Link an SLA and a Learning Goal, adding a weight to this link
13	Link a Learning Goal with another Learning Goal, adding a weight to this link
Constraints	
1	The definition of Learning Graph concept provides the validity domain that must be taken into account in this tool
2	Performances need to fulfil specified KPIs[19].

Table 50 - Educational Material Creation - Learning Graph Editor

[19] MaTHiSiS KPIs, ***MaTHiSiS Description Of Action***[18] part B pages 8 and 9

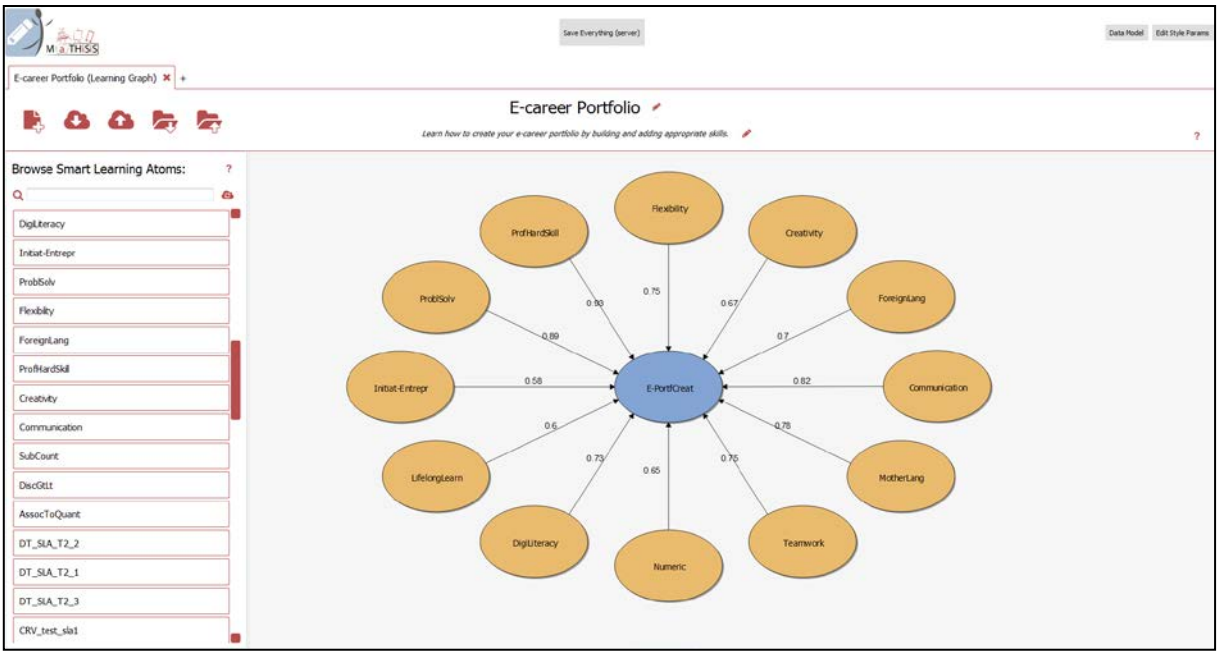


Figure 9 - LCE initial version - Learning Graph Editor

5.1.1.2 Smart Learning Atom Editor

This editor is dedicated to the creation of SLA. Through simple user interfaces, this tool allows the edition of common properties of an SLA and the list of associated Learning Actions. This editor allows using existing Learning Actions or the creation of new empty ones, editable later through the LA Editor. It is also possible to manage the visibility/rights of the SLA for other users. Different kind of rights are applicable there: public in read only mode, so only usable in Learning Graphs, public in read/write mode, so editable and usable in Learning Graphs or private, so invisible to other users.

Component Name	**Smart Learning Atom Editor**
Tier / Tool	Educational Material Creation / Learning Content Editor
Edited properties	
1	SLA name and description
2	Set of Learning Actions used by this SLA
Inputs	
1	Optional: existing SLA for edition
1	Set of existing Learning Actions usable in the edited SLA
Outputs	
1	Updated SLA (either valid or under construction)
High – Level Features	

1	Connected to MaTHiSiS back-end to access the set of existing Learning Actions
2	Connected to MaTHiSiS back-end to optionally get an existing SLA, share newly created or modified SLA
Functionalities	
1	Common actions 'New', 'Open', 'Save' and 'Save As' for SLA
2	Create and Edit properties and list of Learning Actions of a new SLA
3	View and Edit properties and list of Learning Actions of an existing SLA
4	View the set of already existing Learning Actions
5	Use or Duplicate an existing Learning Action
6	Creation of a new empty Learning Action
6	Launch the edition of a Learning Action, through the Learning Action Editor
7	View the summary of an existing Learning Action
Constraints	
1	None

Table 51 - Educational Material Creation - Smart Learning Atom Editor

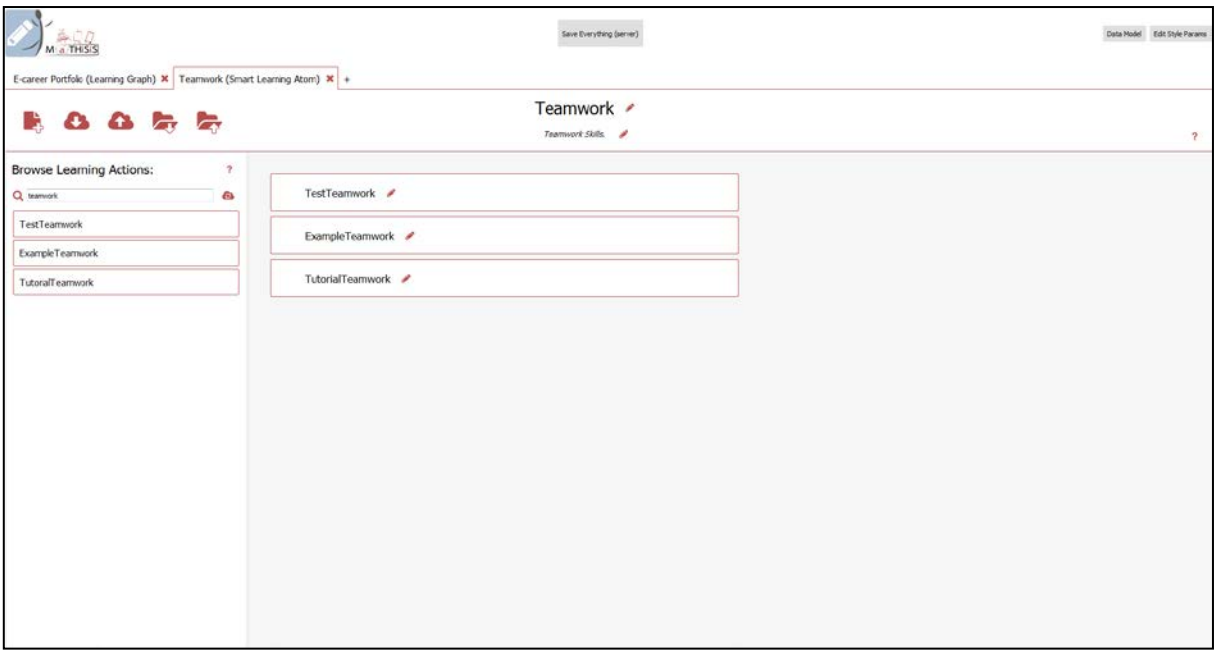


Figure 10 **- LCE initial version - Smart Learning Atom Editor**

5.1.1.3 Learning Action Editor

This editor is dedicated to the creation and edition of Learning Actions with their materialization. Mainly, it's a simple and a user-friendly visual graph building tool that allows associating Learning Materials to Learning Actions in order to build the materialization. In addition, it allows specifying conditions to dynamically change whether Learning Materials can be used or not in a specific Learning Session context (see Figure 11 for an example). This editor allows the usage of existing Learning Materials, that can be created and edited using the Learning Material Configurator, a tool available in the main MaTHiSiS Front-end web application. It is also possible to manage the visibility/rights of the LA for other users. Different kind of rights are applicable there: public in read only mode, so only usable in SLAs, public in read/write mode, so editable and usable in SLAs or private, so invisible to other users.

The graph, resulted of the edition process supported by this tool, will be executed by the Experience Engine in order to complete the first step of the Learning Action materialization process.

Component Name	**Learning Action Editor**
Tier / Tool	Educational Material Creation / Learning Content Editor
Edited properties	
1	Learning Action name and description
2	Learning Action Ontology category of the Learning Action
3	Set of possible Learning Materials to be used by this Learning Action, possibly conditioned on the Learning Session context (i.e. on the learner profile, the platform agent, the learning environment and the difficulty level computed by the platform)
Inputs	
1	Optional: existing Learning Action, for edition
2	Set of existing Learning Materials
3	Learning Action Ontology categories for Learning Actions, learners, learning environments, platform agents, difficulty and learning materials
Outputs	
1	Updated Learning Action (either valid or under construction)
High – Level Features	
1	Visual and user-friendly graph edition
2	Connected to MaTHiSiS back-end to access the set of existing and available Learning Materials
3	Connected to MaTHiSiS back-end to optionally get an existing Learning Action, share newly created or modified Learning Action
Functionalities	
1	Common actions 'New', 'Open', 'Save' and 'Save As' for Learning Action

2	Create and Edit properties and structure of a new Learning Action
3	View and Edit properties and structure of an existing Learning Action
4	View the set of already existing Learning Materials
5	Use or Duplicate an existing Learning Material
6	Creation of a new empty Learning Material
7	View the summary of an existing Learning Material
8	Create a new Condition
9	Duplicate an existing Condition
10	View and Edit properties of a Condition
11	Link a Condition and a Learning Material
12	Link a Condition and another Condition
Constraints	
1	More kinds of conditions must be foreseen for later version
2	The possibility to create a scenario across Learning Materials for a given materialization

Table 52 - Educational Material Creation - Learning Action Editor

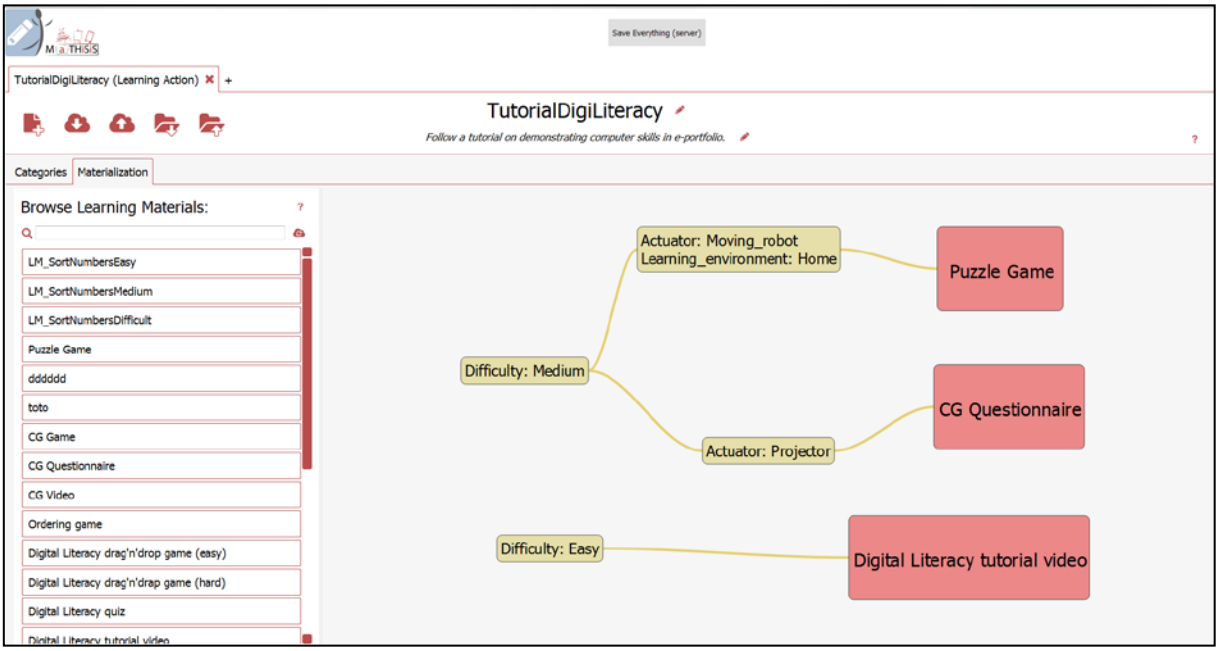


Figure 11 - LCE initial version - Learning Action Editor

5.1.1.4 Learning Material Configurator

This tool is used to manage the set of existing Learning Material accessible to the current user. From it, we are able to declare new Learning Materials or to remove or modify existing ones.

Learning Material Configurator is dedicated to the linkage of an existing and compatible educational material with MaTHiSiS. For that purpose, either digital or physical materials must be known, identified and well described in order to be usable during Learning Experiences supported by the MaTHiSiS approach.

This tool will be enhanced during time as it's here that any existing kind of compatible educational material can be imported into MaTHiSiS. That way, it will have to evolve to manage more cases.

During the project, we want to be able to import digital learning materials that can reside on all kind of Platform Agents. Here are the devices taken into consideration during the MaTHiSiS project:

- Desktop/laptop computers, that are under Windows 7, 8 and 10 and Linux;
- Mobile devices, that are under Android and iOS;
- Interactive Whiteboards, that are under Windows;
- TurtleBots, that are under ROS (usable through the Ubuntu distribution of Linux);
- NAO robots that are using a specific OS.

In parallel, we want to manage different kind of Learning Materials:

- Digital native applications;
- Digital web-based applications;
- Physical objects.

In order to make all of this possible, all digital applications have to use a specific interface called the Learning Material Framework (described in Section 6.2.4).

The case of physical object is more complex, but we have the same kind of behaviour. In order to be able to track what the learner is doing with such objects, we have to be able to identify them. That is why we are providing a way of creating specific LM identifiers for physical objects. This way, a related digital application can track the objects using these identifiers, allowing the use of the LM framework like pure digital Learning Materials.

The import of existing web-based application has been introduced to provide a way to quickly use existing e-learning courses within MaTHiSiS ecosystem. The possibility to use physical educational materials is also very important for some use cases. For example, some children prefer to work with concrete objects, or in an industrial case we want to learn how to use a specific machine in a factory.

As a summary, the Learning Material Configurator is able to import existing educational materials, either it's a LM specifically created for MaTHiSiS or not, even if in the latter case it is needed to make the material compatible with MaTHiSiS by integrating the LM Framework. In addition, LM identifiers can be created and associated to physical objects.

From this tool, the user is also able to define concretely how the system will be able to install, localize and launch on the device the Learning Material, depending on its nature and on its compatibility with the targeted Platform Agent. For example, the user will have to explain how to launch a native application on iOS, giving specific information that will allow finding and launching the application on the device. The same kind of configuration will be needed for Android as the information needed will be different between iOS and Android. In any case, this configuration will be done only once when the application will be imported as a Learning Material. Later, the later will be usable many times in different Learning Actions without taking into consideration this low-level information. That is why the Learning Material Configurator is more dedicated to technical users, even if it stays simple to use.

Component Name	**Learning Material Configurator**
Tier / Tool	Educational Material Creation / Learning Content Editor
Inputs	
1	Set of existing Learning Materials accessible to the current user
Outputs	
1	Updated set of Learning Material accessible to the current user
2	Updated Learning Materials
High – Level Features	
1	Connected to MaTHiSiS back-end to access the set of existing Learning Materials
2	Connected to MaTHiSiS back-end to share newly created or modified Learning Materials
Functionalities	
1	Common actions 'New', 'Save', 'Save As' and 'Delete' for Learning Material
2	Declare and Edit properties of a new Learning Material
3	View and Edit properties of an existing Learning Material
4	View the set of already existing Learning Materials
5	Duplicate an existing Learning Material
6	View the summary of an existing Learning Material
Constraints	
1	None

Table 53 - Educational Material Creation - Learning Material Manager

The full system allowing the creation of Learning Material Identifiers is in construction. More details will be given in the ***D3.9 - MaTHiSiS Frontend Component*** due in M24.

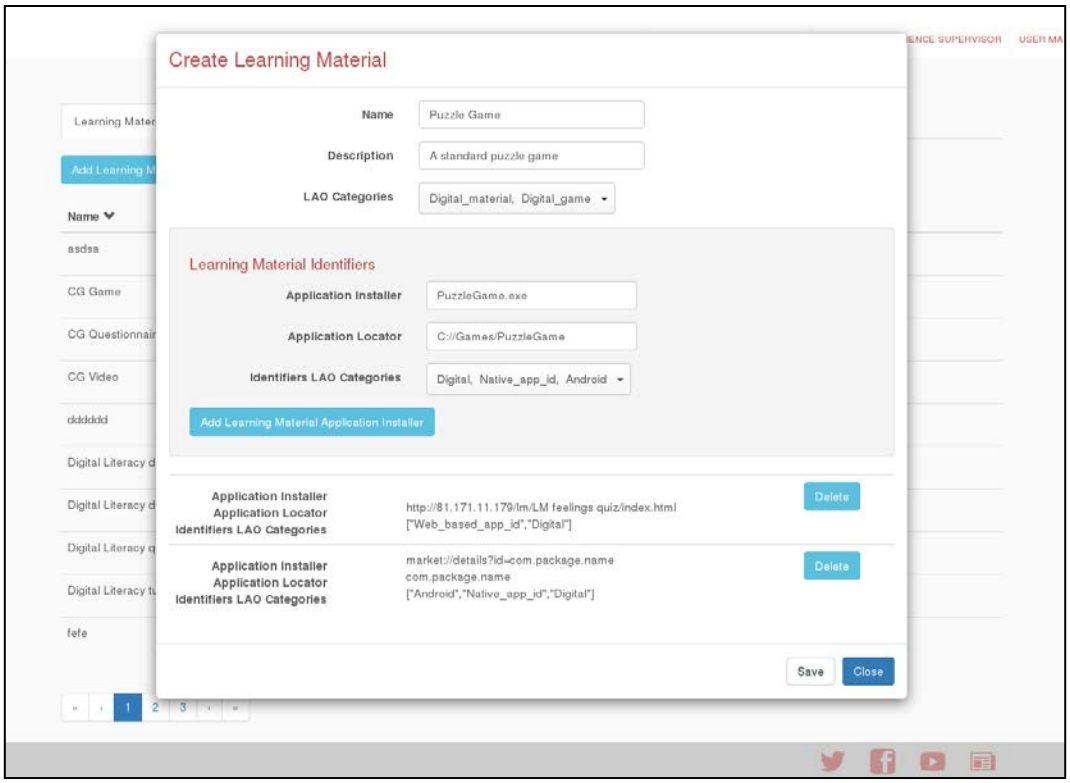


Figure 12 - MaTHiSiS Front-end initial version - Learning Material Configurator - Identifiers

5.1.2 Learning Analytics Visualization Tier

This tier includes all components related to the visualization of learning performance information. To this purpose, we introduce the Learning Analytics & Visualization Dashboard.

The Learning Analytics and Visualization Dashboard allows MaTHiSiS users to monitor the progress of a Learning Experience according to their specific needs. In the case of Tutor, he/she is able to check the progress of his/her learners on the different learning actions defined according to the lesson design and also gets some real-time information to be able to modify certain details of the LG to ensure that learners will be able to attain the specified learning goals.

Tier Name	**Learning Analytics Visualization Tier**
Layer	Front-end
Inputs	
1	Learner performance information (in terms of interaction status and progress) retrieved from the Learner Profile Repository and the Cloud Learner Space.
Outputs	
1	Visualization of a learners' or group of learner's performance during the different learning experience.
2	Visualization of the learning graphs evolution

High – Level Features	
1	Visual and user-friendly information
2	Connected to MaTHiSiS backend to access learners or group of learners performance (DSS, LS, LPR)
Constraints	
1	Each learner should have a valid user id in MaTHiSiS

Table 54 - Tier - Learning Analytics Visualization

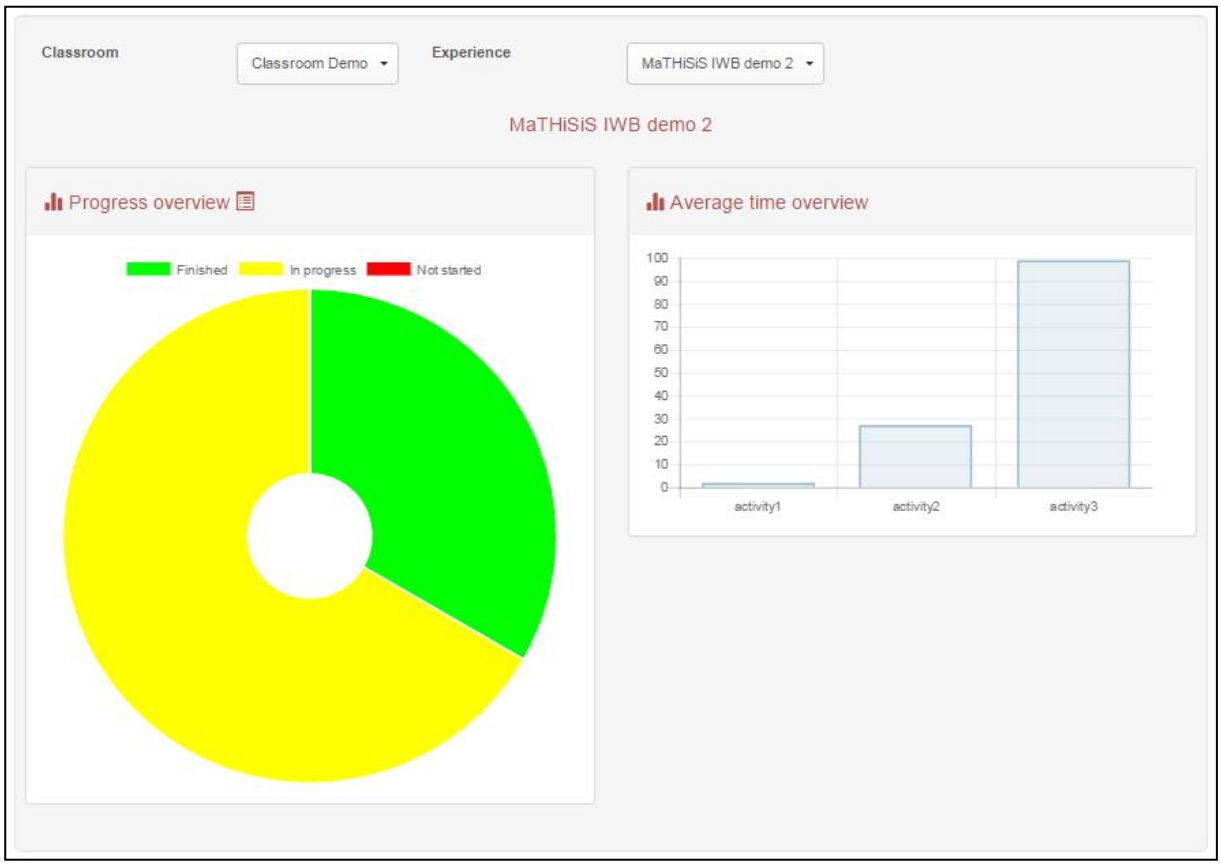


Figure 13 - MaTHiSiS Front-end initial version - Learning Analytics & Visualization Dashboard - By Classroom

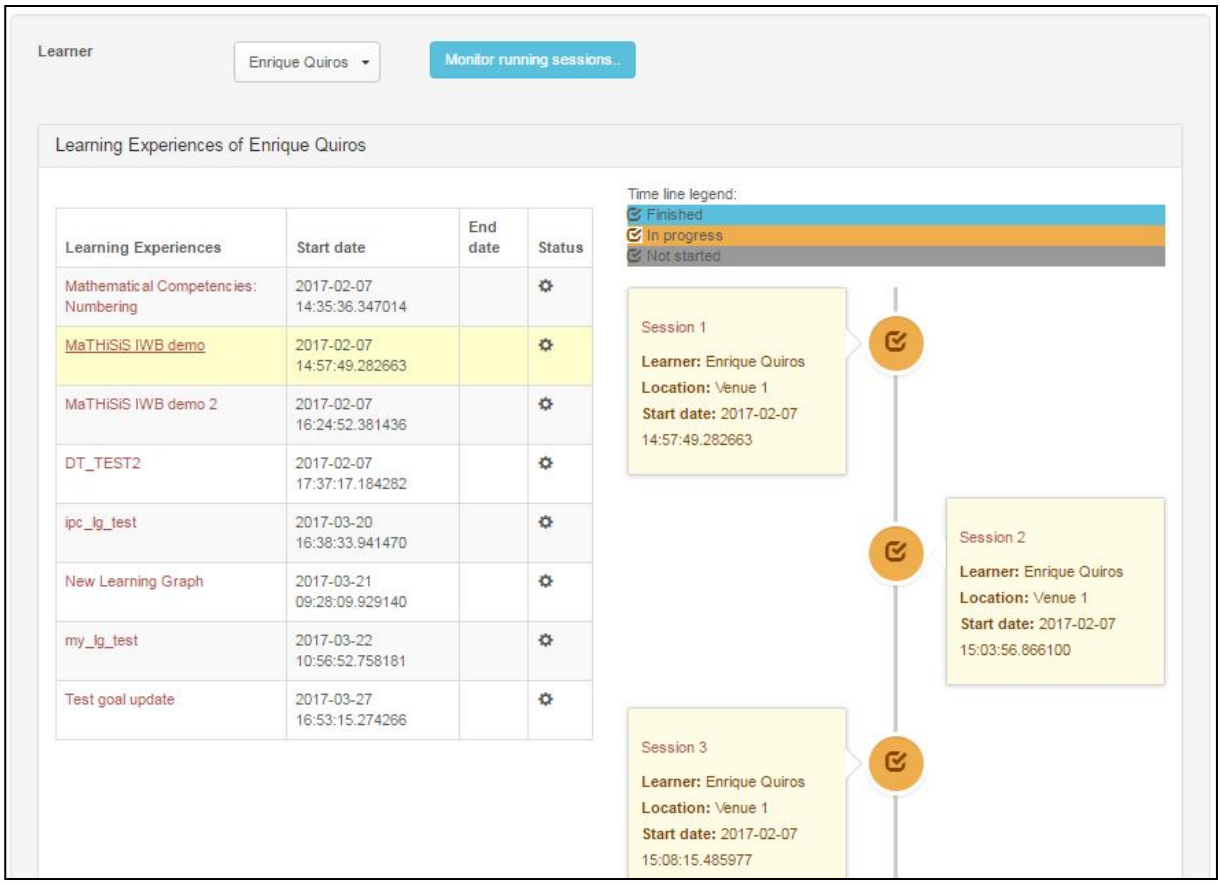


Figure 14 - MaTHiSiS Front-end initial version - Learning Analytics & Visualization Dashboard - By Learner

5.1.3 Affect Recognition Visualization Tier

The Affect Recognition Visualization Tier consists of an autonomous User Interface and a set of MaTHiSiS reusable components in terms of detecting and visualizing the Affective State of the user. The main objective of this stand-alone Tier is to provide an out-of-the-box solution corresponding to the challenge of identifying the Affective States of diverse learner groups (autism spectrum, PMLD, mainstream education, carrier guidance and industrial training), combing input from discrete modalities such as face, gaze, skeleton and speech.

The main components of the Affect Recognition Visualization Tier are:

1. Video and Sound capturer;
2. Video and Sound annotator;
3. Face Recognition;
4. Gaze Estimation;
5. Skeleton Analysis (when a depth sensor is available);
6. Speech Recognition;
7. Inertial sensor data collector (when using mobile devices);
8. Multimodal feature Analyzer;
9. Affect Identifier.

Tier Name	**Affect Recognition Visualization Tier**
Layer	Front-end
Inputs	
1	Learner affect information retrieved from the affect capturing and recognition tools
Outputs	
1	Visualization of a learners' affect state features
2	Visualization of the learners' affect state annotation
High – Level Features	
1	Visual and user-friendly information
2	Connected to MaTHiSiS platform agent layer and backend to access affect information (SC, IPA, DSS)
Constraints	
1	Each learner should have a valid user id in MaTHiSiS

Table 55 - Tier - Affect Recognition Visualization

5.1.4 Learning Games Programming Tier

This tier is in charge of the provision of software allowing the creation of learning games. We have chosen to provide the minimal set of libraries needed to interface any digital game with MaTHiSiS. In addition, during the project, we will integrate these libraries in existing software as plug-ins or any other relevant kind of integration depending on the technologies.

The libraries will take into consideration:

- The need, for the digital game, to send specific information to the MaTHiSiS platform regarding the interactions of the learner;
- The ability to work on all kind of Platform Agents and for all kind of Learning Materials, e.g. on Nao robots, web-based digital games, native digital games, etc.

The implication in terms of architecture is related to Sections 4.2.5, 4.3.5, 4.4.3 and 6.2.4. Moreover, this tier will have links with other components of the front-end, described in Sections 5.1.1.4 and 5.2.

5.2 Authentication / Validation Tier

As aforementioned in Section 4.1, UAV component is based on the open source ForgeRock OpenAM[20] tool and the protocols that are used are the OAuth2.0and the OpenID[21].

The MaTHiSiS user and role manager component undertakes the authentication and authorization of the user, i.e. it specifies whether a user is eligible and the resources that users are allowed to access. There are four different roles (Administrator, Tutor, Caregiver, Learner) which correspond in different access layers

[20]ForgeRock OpenAM: https://www.forgerock.com/platform/access-management/
[21]OpenID: http://openid.net/

for each subsystem of the MaTHiSiS platform. MaTHiSiS platform is a modular application which requires a dynamic and comprehensive way to authenticate users and components / plug-ins continually. Hence, not only users but also each separate component must be authenticated and authorized to communicate with others and with the core MaTHiSiS platform.

Prerequisite of the authentication and validation tier is that both users, including resource owners/developers and consumers, and the MaTHiSiS applications/resources should be registered in the UAV component. The resource owner/developer uses the UAV Graphical User Interface to create his account by entering the necessary information in the relevant form. With similar manner, any resource consumer creates an account in UAV. The resource owner/developer logs in the UAV application providing the credentials in the login form and registers his application/resource to be offered through MaTHiSiS.

Afterwards, any user can require access to any registered application. The flow is the following:

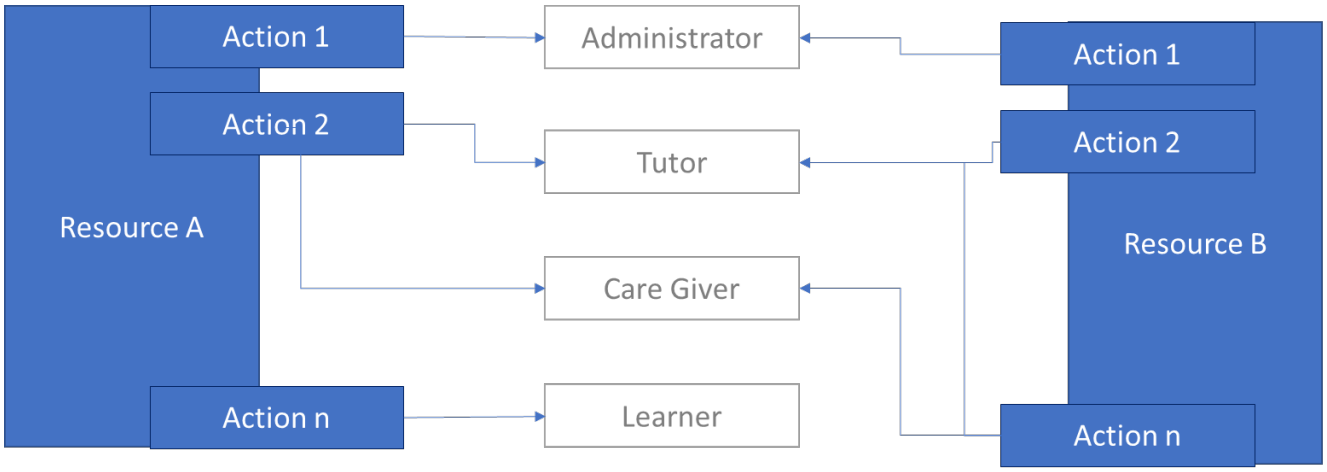
- The user accesses the login UI of the application and enters his personal credentials.
- The application authenticates the user using the authorization server of the UAV component:
    - Actually, the application invokes the web service of UAV described in Table 17 by providing the user's and itself credentials.
- The application retrieves the response of the web service:
    - If the user's credentials are invalid, a detailed message is sent to the application and the application informs the user
    - Otherwise, the user is authenticated and can access the application
        - The application retrieves a structured message including the access token of the user that expires in specific time.
        - In the next step, the application can exploits the web services described in Table 18, Table 19 to retrieve the user account and the roles of user, respectively.
        - In case that the application needs to store any part of the user account, should ask for permission from the user.
        - The application defines the actions of the user on it considering his role(s).
- The UAV component monitors and logs any action performed by the users such as an authentication attempt.

5.3 User / Role Management Tier

The User / Role Management Tier is in charge of the user information management and it is composed of the high level and internal organization functionalities as well as the User/Role editor. Next we further describe each of those components.

5.3.1 High level functionality and internal organization

As it is described in section 5.2, every resource owner is responsible to set the actions incorporated into the resource (component) and set the access right for each role to each of these actions. Hence, the architecture of the role management component offers the essential flexibility that the modular structure of the MaTHiSiS platform requires.



5.3.2 User / Role Editor

While the resource owner registers their resource to the MaTHiSiS main authentication/authorization subsystem, they are able to add actions to the resource and to set the corresponding permissions to each role. Therefore, the user and role editor consists of two levels of editing. The first concerns the usage of each resource and the responsible person is the resource owner – who is the responsible person for the right configuration of their own subsystem via the given functionality.

The second level concerns the accessibility to the entire MaTHiSiS platform which takes part outside of the main Platform Configurator user interface and is accessible only by the administrator of the platform. The administrator can edit the users and carry out necessary actions such as add, remove or edit user accounts, accept or reject registration requests with specific role, grant permissions to specific components etc.

This approach, thanks to oAuth2 protocol, offers flexibility and scalability to MaTHiSiS as it evolves.

5.4 Social Networking Tier

The users of MaTHiSiS will be offered the option to use social groups. In more detail, we can distinguish four types of groups organized around:

- Educational subjects (or graphs) aimed at tutors;
- Educational subjects aimed at learners (e.g. course-mates);
- Educational subjects aimed at all involved users, i.e. tutors, learners, caregivers;
- MaTHiSiS global social network including registered users.

Each time a user is registered in the platform, he is asked whether he desires to participates in the social networks associated with the subject and/or graphs he/she is interested / engaged in. The information of users per social network/group is kept in the users' repositories.

To this end, this component receives from the user interface the intentions of the user to appropriately modify the social network bindings.

5.5 Learning Process Monitoring Tier

The Learning Process Monitoring tier has the responsibility of providing ways for monitoring learners' Learning Experiences. To this purpose, a tool is introduced in order to manage Learning Experiences and Learning Sessions.

The Learning Experience Supervisor (LES) is dedicated to users that are allowed to manage a Learning Experience and therefore, Learning Sessions, from the beginning to the end. Obviously, this application must allow to start and to stop the Learning Experience and associated Learning Sessions.

To prepare, start and monitor a Learning Session, the Learning Experience Controller must be used. This tool is described in Section 5.5.1.

To monitor a full Learning Experience, outside a specific Learning Session, the Analytics Dashboard must be used. This tool is described in Section 5.1.2.

Tool Name	**Learning Experience Supervisor (LES)**
Layer / Tier	Front-end / Learning Process Monitoring
Inputs	
1	Source: Open API - Learning Content Space (SLA lib API, LGI lib API, LA lib API, LM lib API) Data: Learning Graphs, Learning Environments, Platform Agents Schema: JSON
2	Source: Open API - User Space (Learner's Profile lib API, User lib API) Data: Identifiers, Analytics Schema: JSON
3	Source: Experience Engine Data: Possible Learning Actions, Learning Materials and Platform Agents with default choices Schema: JSON
Outputs	
1	Destination: Experience Engine Data: Tutor overridden choices (Learning Action, Learning Material, Platform Agent and Context) Schema: JSON
2	Source: Open API - Clouse Learner Space (Learning Session lib API) Data: Request the creation of a new Learning Session Schema: JSON
Roles	
Tutor	To manage Learning Experiences of supervised learners under his/her responsibility To follow Learning Experiences of independent learners under his/her responsibility
Independent Learner	To manage his/her own Learning Experiences

High – Level Features	
1	Preparation of a Learning Session: available learners, available PAs, the learning environment and Learning Graphs per learner
2	During a Learning Session: validate or modify choices proposed by the back-end (i.e. Experience Engine)
3	View of learning analytics associated to a learner
Constraints	
1	Some foreseen functionalities will need performances

Table 56 - Learning Experience Supervisor

5.5.1 Learning Experience Controller component

The goal of this tool is to facilitate the set-up of Learning Sessions, to control them and to monitor the progression of the learners through them. It also allows validating the choices made by the platform, in interactive time. More generally, this controller serves as support to manage and follow learners' Learning Sessions associated to their Learning Experiences.

Before a Learning Session, as part of the initialization of the learning process, the user will be able to choose the learners that will participate, which Platform Agents will be used by the learners, the learning environment where will occur the session and of course, the Learning Experience the session belongs to. Figure 15 and Figure 16 show the initial version of the Learning Experience Controller dedicated to the preparation of a Learning Session.

During a Learning Session, it provides a way to monitor results and progression in interactive time and a way to let the user validate choices made by the platform: at least, choices made by the Experience Engine for the chosen Learning Action materialization, but other user requirements have to be taken into account properly. Other ways to configure a Learning Session could be addressed depending on the feedback gathered from end users during pilots.

Component Name	**Learning Experience Controller**
Tier / Tool	Learning Process Monitoring / Learning Experience Supervisor
Edited properties	
1	Set of learning Experiences for a chosen learner or group of learners
2	Set of Learning Sessions for a chosen learner or group of learners
3	Properties of a new Learning Session for a chosen learner or group of learners
Inputs	
1	Set of manageable learners
1	Set of manageable group of learners

2	Set of available Learning Experiences
3	Set of Learning Experiences with their status
4	Set of Learning Sessions with their status
5	Set of pre-defined Platform Agents usable in this context
6	Set of pre-defined learning environments of the user
Outputs	
1	Updated set of Learning Experiences
2	Updated set of Learning Sessions
High – Level Features	
1	Preparation of a Learning Session: available learners, available PAs, the learning environment and Learning Experience per learner
2	During a Learning Session: validate or modify choices proposed by the system
3	Provision of hints to help the system to adjust its choices
Functionalities	
1	View all learners or group of learners under the responsibility of the user
2	View all Learning Experiences accessible by the user
3	View all Learning Experiences of learners under the responsibility of the user
4	View all running Learning Sessions of learners under the responsibility of the user
5	Actions to 'Setup', 'Start', 'Interrupt', 'Pause', 'Resume' and 'Monitor' a Learning Session
6	Actions to 'Setup', 'Start', 'Stop' and 'Monitor' a Learning Experience
Constraints	
1	none

Table 57 - Component - Learning Experience Controller

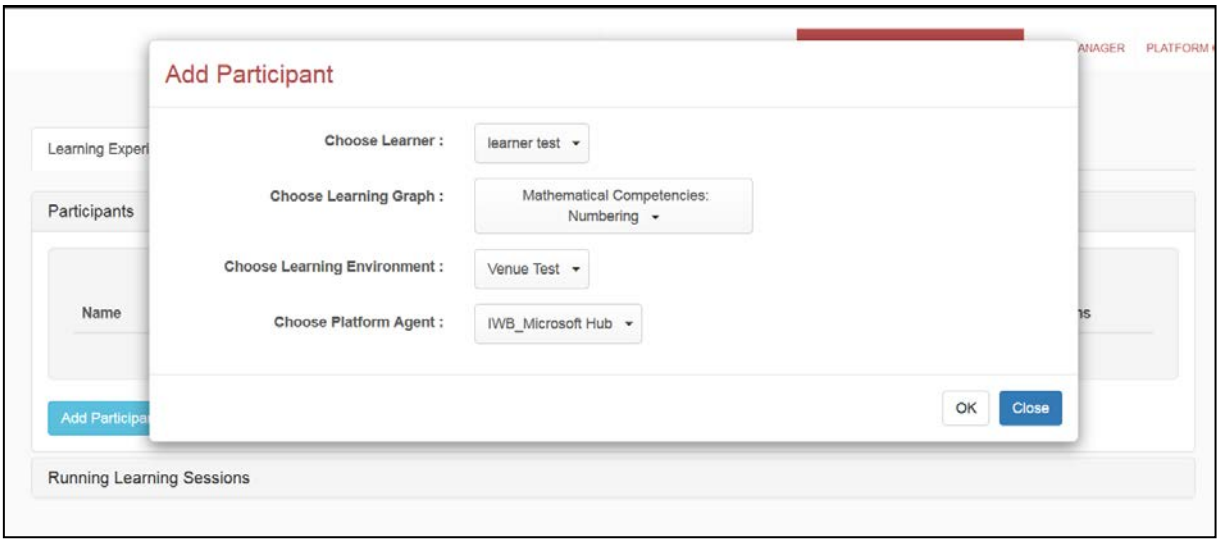


Figure 15 - MaTHiSiS Front-end initial version - Learning Experience Supervisor - Add Participant

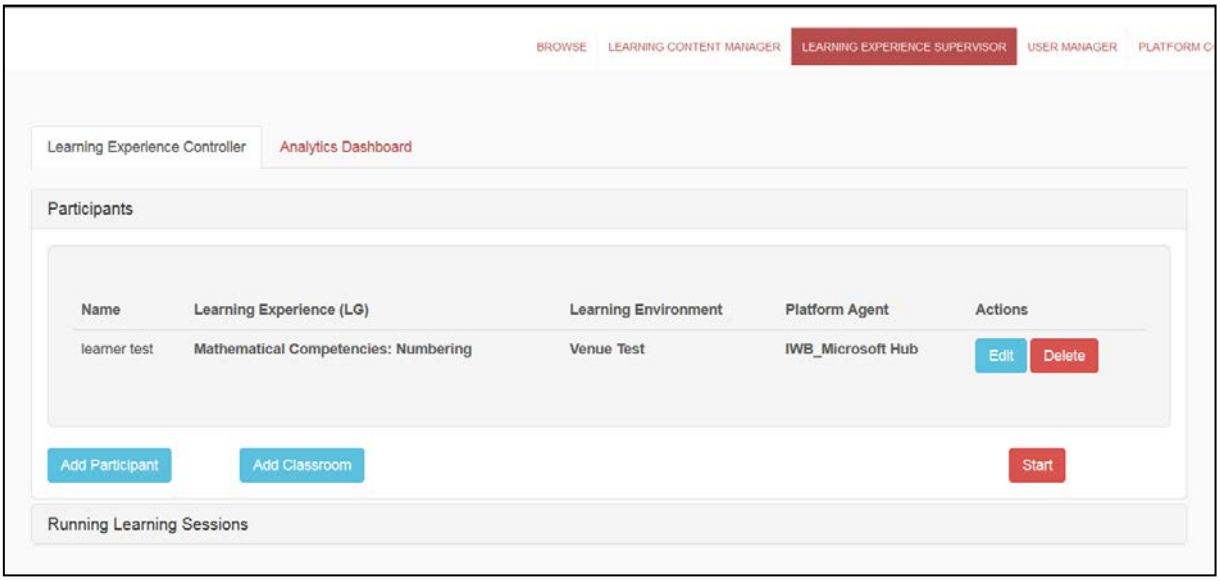


Figure 16 - MaTHiSiS Front-end initial version - Learning Experience Supervisor - Learning Session Preparation

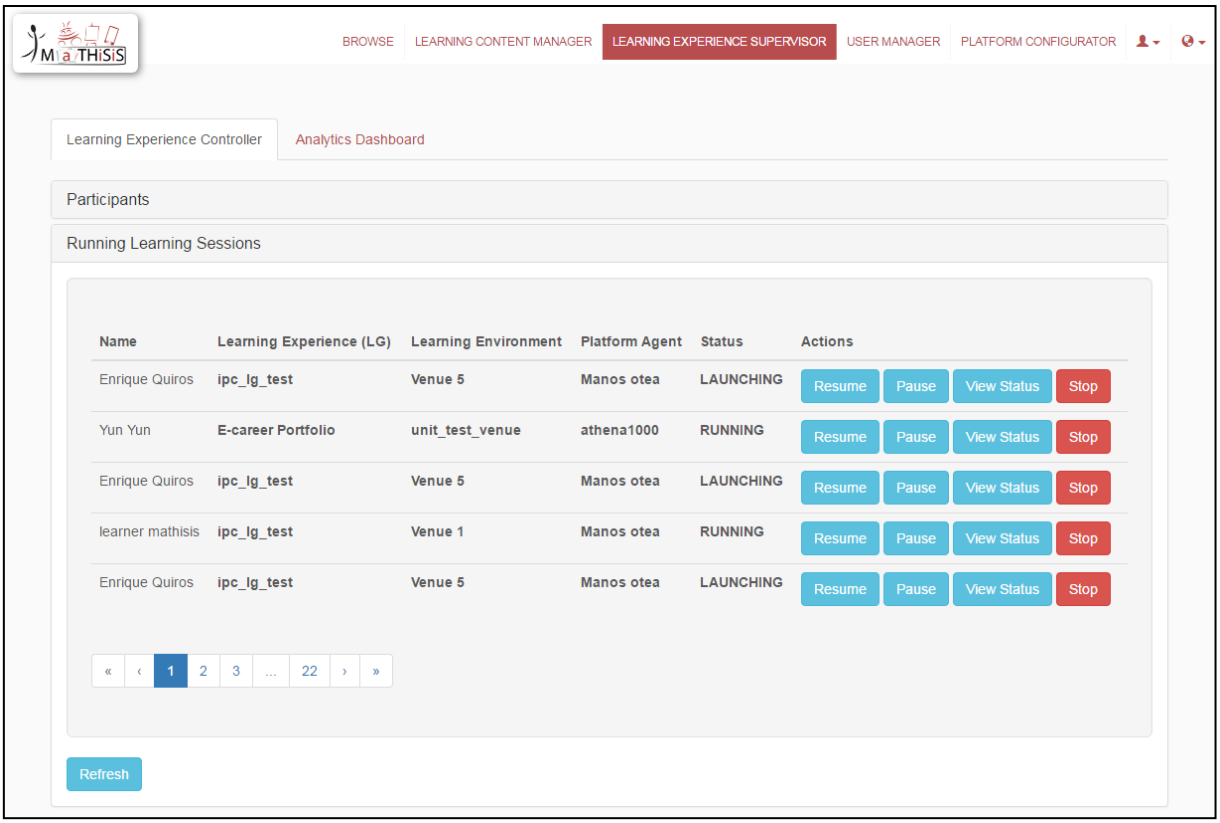


Figure 17 - MaTHiSiS Front-end initial version - Learning Experience Supervisor - Running Learning Sessions

5.6 Platform Configuration Tier

The MaTHiSiS Platform Configurator is in charge of the installation and customization of the MaTHiSiS platform according to the technical, contextual requirements of the adopter's settings. This area is only accessible by the Administrator. In detail, there is a group of options for platform management:

- Platform Agents Configurator: it allows Platform Agents creation and edition.
- Learning Environment Configurator: it permits the creation and administration of learning environments.

Tier Name	**Platform Configuration**
Layer / Tier	Front-end / Platform Configuration Tier
Inputs	
1	Set of manageable Platform Agents
2	Set of manageable Learning Environments
Outputs	
1	Updated set of Platform Agents
2	Updated set of Learning Environments

High – Level Features	
1	Declaration of the learning context of users: available platform agents, available learning environment.
Functionalities	
1	View all Platform Agents available
2	Actions to 'Add', 'Remove' and 'Configure' a Platform Agent
3	View all Learning Environments managed by the user
4	Actions to 'Add', 'Remove' and 'Configure' a Learning Environment
Constraints	
1	Only the Administrator will have access to the platform Configurator.

Table 58 - Tier - Platform Configuration

6. Platform Agent Layer

This layer is dedicated to Platform Agents, i.e. specific devices specifically selected for their technical capabilities, ubiquitous use in day-to-day life and their acceptance in some specific learning scenarios.

Five kind of devices as Platform Agents are currently managed: desktop/laptop computers, mobile devices (i.e. smart phones and tablets), interactive whiteboards, the NAO robot and TurtleBots. Figure 18 shows some examples of the existing PAs.

Platform Agents are used in order to provide learners with learning action materializations and to extract their affect state and track their interactions with the learning materials to be able to adapt and personalize the learner's Learning Experiences. One of the aims at defining this layer is to ensure the extensibility of the platform, facilitating the introduction of new kind of Platform Agents in the future, without the need of making big modifications in the MaTHiSiS platform.

After presenting the general architecture of this layer, we describe in the Experiencing Service Space section with the components and the specificities of each kind of Platform Agents.

Figure 18 - Examples of devices used as Platform Agents

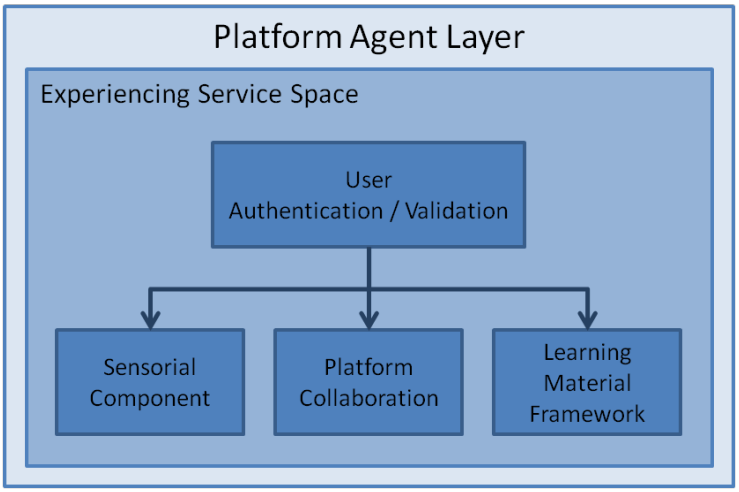


Figure 19 - MaTHiSiS Platform Agent layer

6.1 General architecture

This section presents the general architecture of the Platform Agent layer. This section presents the description and the entities that constitute this sub-system.

During the analysis of the architecture, the necessity of decoupling the Experiencing Service (ES) in two separate entities arose, a server and a client, called *Experiencing Service - Server* and *Experiencing Service - Client* correspondingly. This architecture is used, as many devices that implement either the Experiencing Service, either play the role of Platform Agent are not capable of handling the processing and the transmission of the Sensorial Data, as well as the communications with MaTHiSiS Cloud.

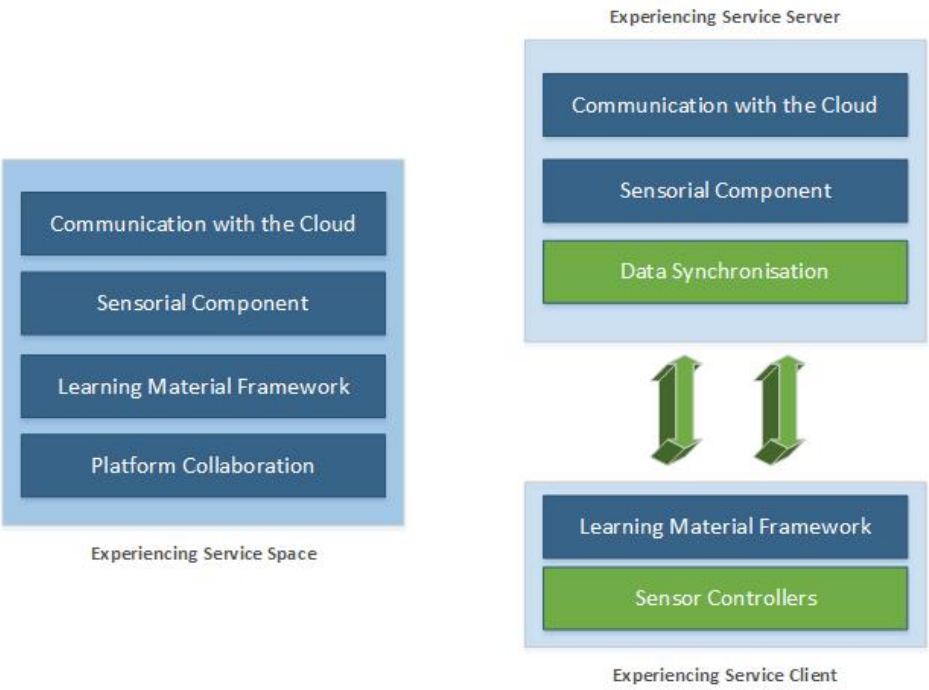


Figure 20 - Experiencing Service Server and Client

The **Experiencing Service - Server** consists of the following sub modules:

- Communication with MaTHiSiS cloud;
- Communication with ES-Client;
- Sensorial Component;
- Synchronization of the Sensorial Data:
    - Face;
    - Gaze;
    - Skeleton;
- Sound.

The **Experiencing Service - Client** consists of the following sub modules:

- Communication with ES-Server;
- Sensor Controllers;
- Learning Material Framework (LMF).

The proposed architecture is represented in Figure 21 below.

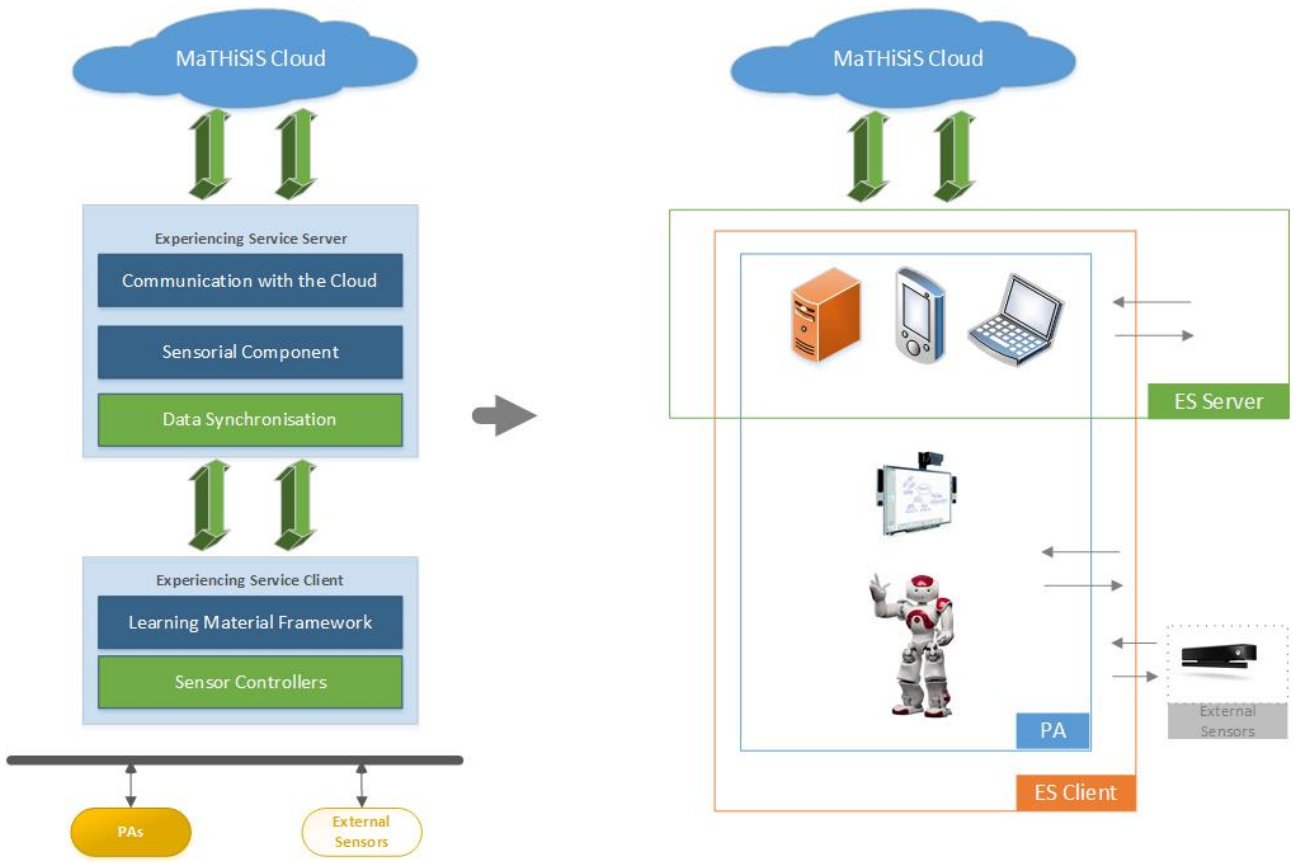


Figure 21 - Experiencing Service Architecture

It depicts which devices can act as PA's, ES-Servers or ES-Clients. During each session multiple clients and PAs may be present, but only one device should be the server. For setup purposes, an initial configuration, handled by the cloud, should be executed depending on the present devices, the desired acquired data and the desired LM. It has to be made clear that some devices can simultaneously represent the ES-Server, an ES-Client and the PA or a subset of these three.

For the next section, we do not make the difference between the server and the client as there are not differences regarding the functionalities provided by this layer.

6.2 Experiencing Service Space

This space is the entry point for MaTHiSiS on Platform Agents. Sub Sections describe the minimal set of components needed in order to be able to make a device compatible and usable with the MaTHiSiS platform.

To this purpose, the Experiencing Service is introduced in order to package all specific components of MaTHiSiS on Platform Agents. It's called a service because it will run on each Platform Agent without interfering with their common behaviour, in order to let learners use their devices simply, like they could have used them without MaTHiSiS.

This service aims at facilitating the last step of the Learning Action materialization process and to retrieve information regarding the learner's affect state and interactions. To theses purposes, external Learning Materials, i.e. already installed applications, will be managed, and specific sensors of each device will be used to extract the learner's affect state.

In addition, the Learning Materials will have to be associated with the Learning Material Framework described in Section 6.2.4, in order to be able to track the interactions of the learners.

6.2.1 User Authentication / Validation Tier

This component is responsible to validate the identity of the protagonists during the communication between the Platform Agent and the MaTHiSiS cloud. This will be achieved without the intervention of the learner thanks to the Learning Session mechanism in place within MaTHiSiS.

When a session is started, the Tutor gives the list of learners assigned to PAs In addition, the PAs are automatically connected to the MaTHiSiS platform, allowing the Experience Engine to inform them on which LM must be started for which learner.

To implement this component consistently, it is foreseen to make the LMF compatible with the new xAPI Launch algorithm[22] within MaTHiSiS. More details will be given in deliverables due in M21 or M24 as not only this layer will be involved in this task.

6.2.2 Sensorial Component Tier

Given the raw data from PAs sensors, the Sensorial Component (SC) will implement algorithms intended to capture information about the affective state of a learner from different input sensors (e.g. cameras, microphones etc.). More specifically, the role of the sensorial component is to effectively translate a physical behaviour into an affect state or a feature representation so as to understand each learner's uptake of knowledge during the learning process.

To this end, the Sensorial Component is composed by five different modalities, aiming to extract affective information by different types of sensors and different behavioural cues: facial expression analysis, gaze estimation, skeleton motion analysis, affect recognition based on inertial sensors in mobile devices and speech- and sound- based analysis.

The sensorial component may extract two types of information, i.e., affective states and feature vectors. The affective states or the extracted features from each modality, i.e. body skeletons, colour frames, etc., will convey the context of the physical behaviour and will be used by the AIR lib (cf. Section4.2.4) in order to fuse affective cues from the different SC modalities and ultimately extract a single multimodal affective state per learner in a given moment of the MaTHiSiS Learning Experience.

The Sensorial Component will run as soon as the learner starts to interact with the learning material. Moreover, the complexity of (some of) the sensorial component algorithms limits them from being executed directly on devices with limited processing and memory capacities. Below follows a brief description of each Sensorial Component modality and their algorithmic outline.

- **Skeleton motion analysis using depth sensors**: Given Kinect v2[23]-extracted skeletons, the specific sensorial component algorithm extracts features that convey the motion context of the learner based on the Principle of Slowness. These features are an enriched representation of the Kinect raw skeleton data that convey cues about the learner's affective state as recognised by their motion. In addition to the feature extraction step, an affective state label is obtained by classifying the aforementioned extracted features using a Support Vector Machine.
- **Gaze estimation:** In the MaTHiSiS context, a learner's gaze will be estimated in uncontrolled illumination conditions without expecting from the user any special head pose. Gaze estimation algorithm will predict the gaze direction from the input received by a typical RGB camera. In

[22] xAPI Launch algorithm: https://github.com/adlnet/xapi-launch#the-xapi-launch-algorithm
[23] https://en.wikipedia.org/wiki/Kinect

addition, a non-varying input is a 3D face shape, and the output is a gaze feature vector, represented by a 3D vector. This vector can be binary labelled as "engaged" (looking at the learning content or the PA) or "not engaged" (not looking at the learning content or the PA).

- **Facial expression analysis:** This component takes advantage of the ability to represent facial landmarks as a graph. Facial landmarks are located as points tracing specific areas of the face, which are then used to create a graph. Based on these landmarks, features denoting cues about facial expressions are extracted using spectral graph analysis. Ultimately, these features are classified into emotions which in term can be classified as affective state labels for a given camera sequence. Facial analysis module predicts the affect labels in real time.
- **Mobile device-based emotion recognition**: Inertial and touch screen sensors embedded in mobile devices can be used to recognize the current affective state of the learner. Using the data provided by these sensors, the Platform can detect abnormal behaviours that could lead to the identification of the learners' affective state such as abrupt movements of the devices or erratic movements on the touch screen, detecting if the user has left the device or is still using it, etc. Machine learning techniques will be used to establish the affective state of the learners based on inertial sensors data and users' touch. The output of this module is a set of descriptors of 3D motion and 2D surface gestures and/or the predicted affective state.
- **Speech recognition and speech-based affect recognition**: Focusing on detecting spoken words using a limited-vocabulary setup or using the pitch of the voice in a multi-lingual context. In addition, verbal cues are employed in order to extract estimates of the user's emotional state that supplements the rest of the affective analysis that is based on other modalities and devices. The output of this module is a feature vector on each audio segment.

The Sensorial Component is further detailed in deliverable ***D4.1 - MaTHiSiS Sensorial Component***[11] and will be further refined in deliverable ***D4.2 - MaTHiSiS Sensorial Component*** due in M18.

Tier Name	Sensorial Component (SC)
Layer / Space	Platform Agent / Experiencing Service Space
Inputs	
1	Source: Platform Agents Data: Raw signals from PA sensors (audio stream, video stream, speed, acceleration, etc.) Schema: N/A (data from PAs does not follow a specific structure; rather they can be characterised as different signals coming from each type of sensor)
Outputs	
1	Destination: AIR lib API Data: Labels and/or features related to Facial Expressions, Gaze Estimation, Skeleton Motion, Speech Recognition and speech- and sound-based emotion, inertial sensors Schema: JSON – detailed in D6.2 (to be released after D2.4)

2	Destination: Platform Collaboration tier Data: Affect-related features – Synchronous collaboration Schema: To be defined in D6.4 (in M24)
High – Level Operations	
1	Gaze estimation
2	Face detection/recognition
3	Facial expressions extraction
4	Skeleton extraction
5	Skeleton motion extraction
6	Speech and sound based emotion recognition
7	Speech recognition
8	Inertial and touch screen sensor-based affective state extraction
9	Affective state extraction per modality
Constraints	
1	Quality of signals (video, audio) will constrain the quality of extraction/recognition
2	Algorithmic performance (in terms of complexity, response time) may constrain the execution of one or more modalities on limited-resource devices
3	Operating system constraints imposed by specific sensors and/or libraries used in one or more SC modalities

Table 59 - Tier - Sensorial Component

6.2.3 Platform Collaboration Tier

This tier will be in charge of the on-PA synchronous and asynchronous collaboration, performing communication and knowledge sharing among platform agents of different nature.

The collaboration will help the user to perform several operations. First of all, this tier will facilitate the incorporation of new PAs to the system, reducing the training of this new agent using the information available from known PAs. Moreover, the synchronous collaboration will be in charge of the accommodation of SLAs to different PAs in order to avoid obstacles related to physical location of the learners. Also this synchronous collaboration will allow the collaboration between different PAs in multi-learner scenarios. The Decision Support System (DSS) will use the Platform Agent Collaborations Scheme lib in order to handle PAs both for communication between each other and the case that new PAs are imported to the learning process.

The main component in the Platform Collaboration Tier at the moment is the Platform Agents Collaboration Scheme Tool which implements the operations of the tier.

Tier Name	Platform Collaboration Tier
Layer	Platform Agent layer
Inputs	
1	Source: LA lib API Data: Available PAs involved in the Learning Experience and corresponding Learning materials. Schema: JSON
2	Source: IPA lib API Data: Interaction parameters related to collaborative scenarios – Synchronous collaboration Schema: To be defined in D6.4 (in M24)
3	Source: Learner's Profile lib API Data: SLA scores history (performance and affective states) – Asynchronous collaboration Schema: JSON
Outputs	
1	Destination: Decision Support System Data: Parameterised features of the PAs involved Schema: To be defined in D6.4 (in M24)
High – Level Features	
1	Synchronous collaboration among PAs
2	Transfer learning for the integration of new PAs
Constraints	
1	Semi-supervised transfer learning (at least a reduced among of data is required for the transfer learning techniques)

Table 60 - Tier - Platform Collaboration

6.2.4 Learning Material Framework Tier

The final step of the Learning Action materialization process occurs on Platform Agents. This tier is dedicated to components responsible for this final phase, but also to make the join between the Learning Materials and the MaTHiSiS platform.

The Learning Material Framework (LMF) is in charge of applying the commands received from the Experience Engine. It is then responsible to launch locally installed Learning Materials when receiving a *start* command. After, it is also responsible to apply the *pause*, *resume*, *interrupt* and *reconfigure* commands that can be sent by the Experience Engine. In addition, the LMF will gather, or receive, depending on the implementation, the interactions of the learner with the Learning Material. It will be responsible to transfer them to the Interaction with Platform Agent component as xAPI statements.

Concerning Learning Materials, they must be made available on Platform Agents before a Learning Session will start.

Whenever something goes wrong (e.g. the LM cannot be found, the LM crashed), the LMF will inform the MaTHiSiS cloud to let it take decision and react to adjust and adapt the learner's Learning Experience.

Tier Name	**Learning Material Framework**
Layer / Space	Platform Agent / Experiencing Service Space
Inputs	
1	Source: PA lib API Data: Learning Material commands, sent by the Experience Engine Schema: JSON
2	Source: Learning Material Data: Learner's interactions events Schema: LMF API for Learning Materials
Outputs	
1	Destination: PA lib API Data: Relevant system events that can imply changes in the materialization Schema: JSON
2	Destination: Learning Material Data: Commands received from the PA lib API Schema: LMF API for Learning Materials
3	Destination: Interactions with Platform Agents Data: Learner's interactions with the Learning Material Schema: xAPI statements (JSON)
High – Level Features	
1	Responsible to manage Learning Materials, already installed on the Platform Agent
Constraints	
1	Technologies used to manage external Learning Materials may vary depending on the Platform Agent
2	This module must ensure the compatibility with the xAPI, at least for the representation of the learner's interaction with the Learning Material

Table 61 - Tier - Learning Action Materialization

6.3 Platform Agents specificities

Each Platform Agent initially available on MaTHiSiS Platform (namely TurtleBot, NAO, Mobile devices, Interactive Whiteboards and computer) has its own specificities that allow them to use different functionalities and to deploy different Learning Materials to implement the same or similar learning actions. For instance, while most of the PAs include a screen that can be used for several purposes, NAO has to use sounds or robot movements to represent similar behaviours. This section highlights the main specificities of each PA to provide an overview of the possibilities that they offer for the correct deployment of learning scenarios.

6.3.1 TurtleBot

This robot is mainly composed of a moving platform with include sensors (such as a Kinect camera) and a computer which extends the amount of sensorial information available, offering the possibility to use microphone, webcam or touchpad among others. The screen of the laptop can be also used to deploy useful information or to guide the learners during the Learning Experience. Moreover, making use of the possibilities of the laptop, several functionalities can be implemented such as avatar-based communication, colour and object recognition, playing videos, showing digital documents or presentations, etc.

The complete description of this Platform Agent is included in Section 11.2.1 and deliverable ***D5.1 - Description of the robotic layer***[12].

6.3.2 NAO

Nao is an autonomous humanoid robot and is controlled by a Linux-based operating system called NAOqi OS. One of the most powerful features of NAOqi OS and framework is that it provides all the required interfaces and functionalities that make the implementation of distributed applications feasible.

Nao robot collects and processes visual and sound data that are needed by the Experiencing Services and materialises the Learning Materials. Most Experiencing Services, though, require computer vision, speech recognition, decision making, and communication with external hardware devices (camera, sensors, etc.), procedures that demand high computing power in order to achieve real-time execution. These processing and timing requirements cannot be met by NAO processor itself, thus a distributed architecture is adopted. This distributed system consists of the robot and a high-end computer that is responsible for the processes that need higher computing power, as well as the handling of the communication with MaTHiSiS cloud and external hardware devices.

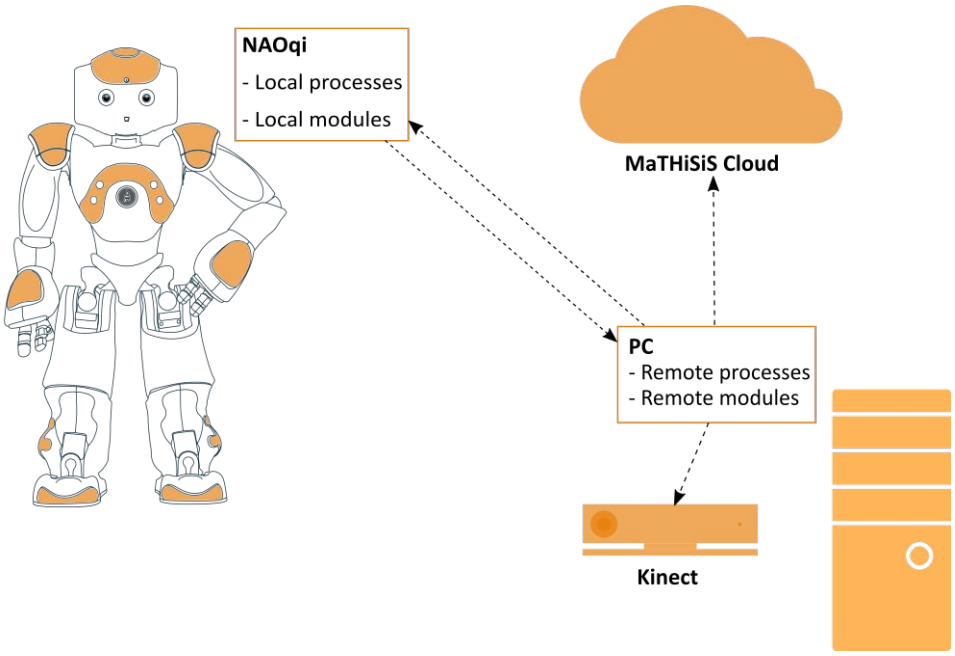


Figure 22 - NAO Platform Agent Architecture

The interfacing between the robot and the computer agent is achieved through the local area network with the use of NAOqi API for data and message sharing. The communication with the World Wide Web and the cloud, consequently, is implemented on the computer side due to the specifications set by the overall system architecture of MaTHiSiS system. Finally, another need of this Platform Agent is the calibration of the robot's camera with the external depth sensor of Kinect v2, as the spatial information that the latter provides is essential for some Learning Materials. The high performance of this device extends the set of system requirements, which the desktop computer an integral part of the agent.

The complete description of this Platform Agent is included in Section 11.2.2 and deliverable ***D5.1 - Description of the robotic layer*** [12].

6.3.3 Interactive Whiteboards

Interactive Whiteboards (IWBs) are a Platform Agent with a huge potential in educational scenarios. Touch screens, cameras, lots of sensors and a processor make them flexible and capable of working with different types of Learning Material.

IWBs are equipped with cameras, microphones, speakers, touch screen and are compatible with Kinect, so are compatible with most of the Sensorial component modalities managed in MaTHiSiS, except the ones that require sensors as accelerometer or gyroscope.

The complete description of this kind of Platform Agent is included in Section 11.2.3 and deliverable ***D5.7 - Description of the interactive whiteboards layer*** [14].

6.3.4 Mobile devices

The mobile device differs from the rest PAs in principle mainly due to the limited processing resources and different sensor capabilities. Mobile devices are equipped with sensors which can be used by MaTHiSiS in order to provide a fully interactive Learning Experience to the learners satisfying their varying needs. Mobile devices in MaTHiSiS capture the readings of the inertia sensors (gyroscope and accelerometer) as well as of the (2D) touch screen. The readings are captured and processed by the Sensorial Component to enable affect state detection and in turn, the adaptation of the Learning Experience. In MaTHiSiS, all the

components developed for the mobile devices are native mobile components (e.g. experiencing service) while the learning materials that run on them can be either native mobile apps or web based materials.

The complete description of this kind of Platform Agent is included in Section 11.2.4 and deliverable ***D5.4 - Description of the mobile layer*** [13].

6.3.5 Laptop/Desktop computers

The laptop and desktop computers can be used as is. Depending on the computation power of the device, it can be coupled with another more powerful one though. In any case, most of the computers are able to manage the sensors that will be useful within MaTHiSiS.

The Experiencing Service Client and Server are available on this kind of Platform Agent, thus implementing the general Platform Agent layer architecture described in Section 6.1.

The complete description of this kind of Platform Agent is included in Section 11.2.5.

7. Knowledge & Data models

7.1 Learning Actions Ontology

As mentioned before in deliverable ***D2.3 - Full System Architecture***[3] and elaborated in deliverable ***D3.5 - Experience Engine*** [9], the Learning Actions Ontology (LAO) will form the vocabulary and reference axiomatic knowledge behind Learning Actions and other relevant facets that will drive the materialisation of LAs within MaTHiSiS, considering also aspects of synchronous and asynchronous collaboration that may affect this materialisation.

More specifically, the ontology contains a taxonomy of **types of LAs**, and taxonomies concerning **learning contexts** (e.g. **learning environment types, learner types**), **types of PAs**, **types of learning materials** (LMs), **types of LM identifiers** and **types of sensors**. In addition, it will define global relations, restrictions and constructors among these facets and/or entities in their sub-hierarchies where conceptually appropriate.

LAO engineering will be iterative, taking into consideration pedagogical and technical requirements, which will be (re)evaluated during the driver and assisted pilots. The progress of this task will be reported in deliverable **D3.6 - *Experience Engine*** due in M24. Figure 23 illustrates the components in which categorizations from the LAO ontology will be used, in order to provide all the information needed to describe a Learning Experience. This information will be taken into consideration by the Experience Engine in the decision-making process towards recommending Learning Action Materialisations for a given Learning Experience.

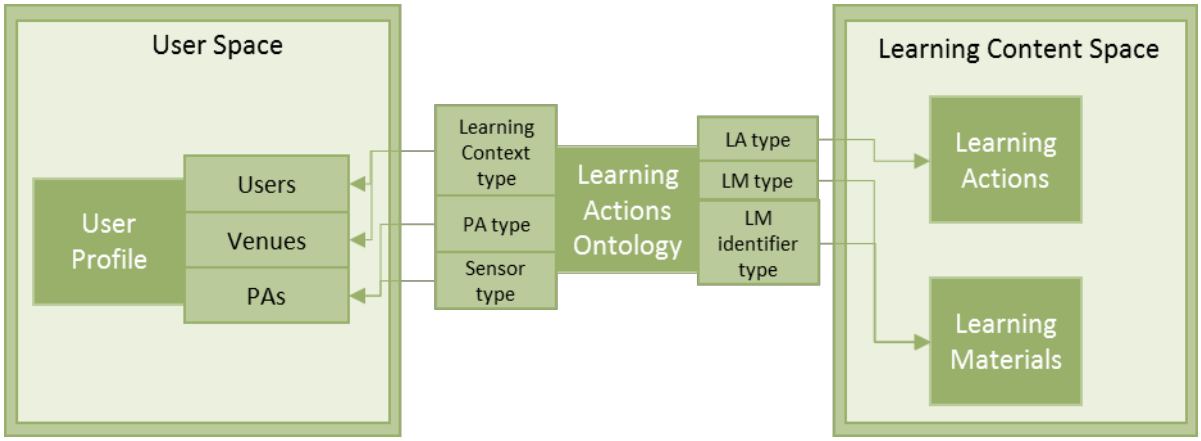


Figure 23 – Detailed illustration of LAO-back end components dependencies

7.2 Learning Content Entities

The Learning Content entities within the MaTHiSiS ecosystem comprise of two parts:

- The data models that carry information about *the aims of learning processes supported* within MaTHiSiS (i.e. the actual achievement of specific knowledge, skills, goals, etc.), consisting of Smart Learning Atoms and Learning Graphs.
- The data models that carry information about *how* the former *are taught* in MaTHiSiS (i.e. the physical and conceptual means employed to teach as well as parameters that influence them), consisting of Learning Actions, Learning Materials and other information related to the materialisation of the Learning Actions (e.g. the learning context).

The former is illustrated in Figure 24, while the latter is illustrated in Figure 25 and Figure 26.

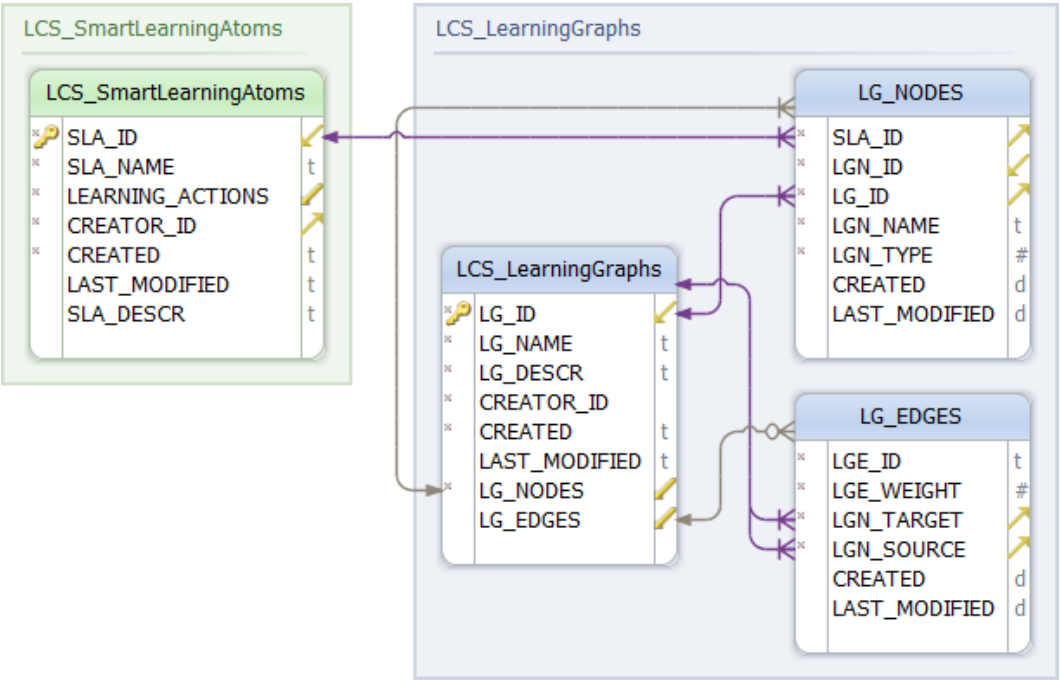


Figure 24 - The data models of the Learning Graph and Smart Learning Atom

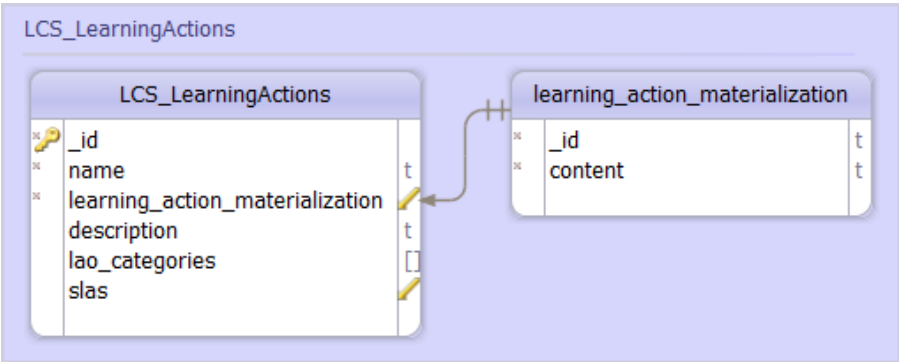


Figure 25 - The data models of the Learning Action and its materializations

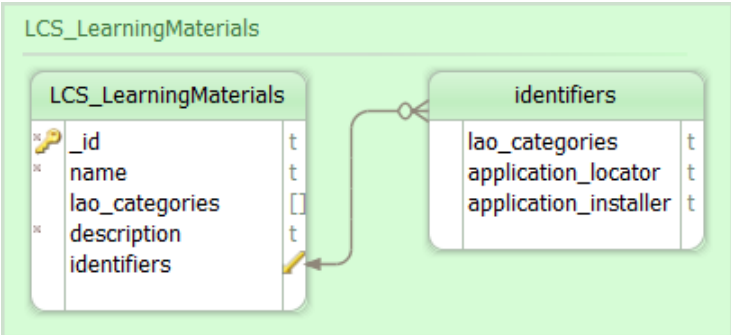


Figure 26 - The data models of the Learning Material with its identifier

7.2.1 Collection LCS_LearningGraphs

Columns			
The MaTHiSiS Learning Graphs comprising of multiple Learning Goals and associated SLAs			
*	LG_ID	integer	The unique identifier of the Learning Graph
*	LG_NAME	string	The label of the Learning Graph
	LG_DESCR	string	Optional. A short description of the contents of the LG for sharing.
*	CREATOR_ID	integer	The user (i.e. Tutor) that created the Learning Graph
*	CREATED	date	
	LAST_MODIFIED	date	
	LG_NODES	list	List of all nodes of the Learning Graph
	LG_EDGES	list	List of all edges of the Learning Graph

7.2.1.1 Collection LG_EDGES

Columns			
The edges within the learning graph			
*	LGE_ID	integer	The unique identifier of the Learning Edge within the Learning Graph
*	LGE_WEIGHT	double	The weight of the edge (default = 1.0)
*	LGN_TARGET	integer	The target node to be connected with the edge
*	LGN_SOURCE	integer	The source node to be connected with the edge
	CREATED	date	
	LAST_MODIFIED	date	

7.2.1.2 Collection LG_NODES

Columns			
The actual nodes of the Learning Graph			
*	LGN_ID	integer	The unique identifier of the node within the learning graph
*	SLA_ID	integer	The reference of the unique identifier of the Smart Learning Atom.
*	LG_ID	integer	The reference of the unique identifier of the Learning Graph that the Learning Graph Node belongs to.
*	LGN_NAME	string	The label of the learning node

*	LGN_TYPE	integer DEFO 1	0 for Goal, 1 for SLA, 2 for Other
	CREATED	date	
	LAST_MODIFIED	date	

7.2.2 Collection LCS_SmartLearningAtoms

Columns			
The MaTHiSiS Smart Learning Atoms.			
*	SLA_ID	integer	The unique identifier of the Smart Learning Atom.
*	SLA_NAME	string	The label of the SLA
*	LEARNING_ACTIONS	list	A list of the learning actions that are attached to the SLA.
*	CREATOR_ID	integer	The user (i.e. Tutor) that created the SLA.
*	CREATED	date	
	LAST_MODIFIED	date	
	SLA_DESCR	string	Optional. A short description of the SLA for sharing.

7.2.3 Collection LCS_LearningActions

Columns			
The MaTHiSiS Learning Actions			
*	_id	oid AUTOINCREMENT	The unique identifier of the Learning Action
*	name	string	The label of the LA
*	learning_action_materialization	map	The materialization information of the LA
	lao_categories	list	The LAO categories the LA falls in
	slas	list	The list of SLAs referencing this LA
	description	string	Optional. A short description of the LA for sharing.

7.2.3.1 Collection learning_action_materialization

Columns			
The MaTHiSiS Learning Actions Materializations			
	_id	string AUTOINCREMENT	The unique identifier of the Learning Action materialization
	content	string	The encoded content for the materialization

7.2.4 Collection LCS_LearningMaterials

Columns			
The MaTHiSiS Learning Materials			
*	_id	string AUTOINCREMENT	The unique identifier of the Learning Material
*	name	string	The label of the LM
*	lao_categories	array	The LAO categories the LM falls in
*	description	string	Optional. A short description of the LM for sharing.
	identifiers	list	The identifiers associated to the LM

7.2.4.1 Collection identifiers

Columns			
The MaTHiSiS Learning Materials Identifiers			
	lao_categories	array	The LAO categories the LM Identifiers falls in
	application_locator	string	The information needed to locate the LM on the PA
	application_installer	string	The information needed to install the LM on the PA

7.3 User Space Entities

The information to be used/managed by the system about Users and Learner's profile, is presented in the following Figure. There is a differentiation between the information of MaTHiSiS users (i.e. users with any of the roles defined in section 3.3) and the data related to the Learner's profile, which is only valid for those users with the role "Learner". The following tables provide more details about each of the fields included on each of the data entities defined to manage user information in MaTHiSiS. In some case like Accessibility information of the learners, a common and shared vocabulary has been defined.

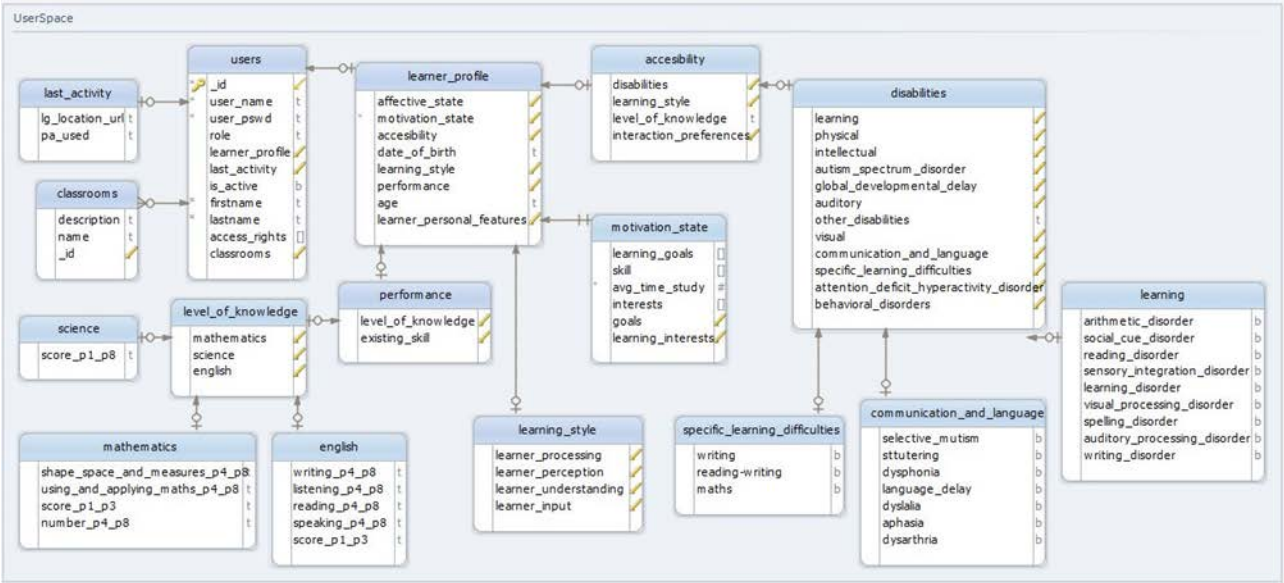


Figure 27 - MaTHiSiS User and Learner profile entities and relationships

In addition, the User Space contains three user-related collections as described below:

- *Learning Environment collection*, where possible locations for the Learning Sessions (e.g. a school or training centre) are stored. More details on the Learning Environments are provided in Section 4.3.4.
- *Classrooms collection*, where all information related to a classroom/cohort is stored, such as the Tutors, Learners and the Learning Environments. More details on the Classrooms are provided in Section 4.3.3.
- *Platform Agent collection*, where all information related to a Platform Agent is stored. It's mainly the minimal description that is useful for the platform to use it during Learning Session. More details on the Platform Agents are provided in Section 4.3.5.

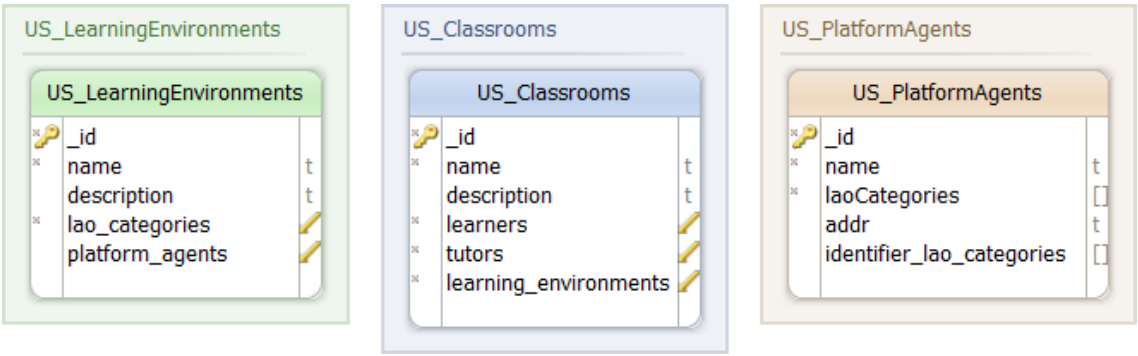


Figure 28 - The data models of the Learning Environment, Classroom and Platform Agent

Finally, the User Space contains the collections related to the personalized of Learning Graphs and Smart Learning Atoms for learners. Figure 29 show these collections and they can be compared to the unpersonalized collections shown in Figure 24.

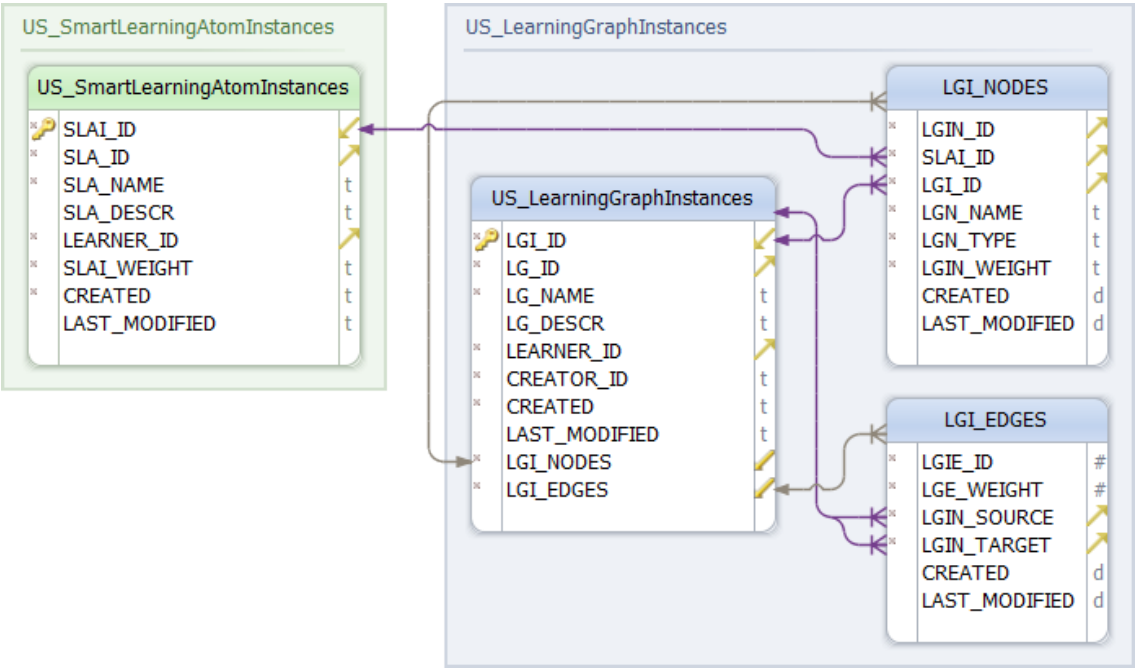


Figure 29 - The data models of the Learning Graph Instance and Smart Learning Atom Instance

7.3.1 Collection US_LearningGraphInstances

Columns			
The MaTHiSiS Learning Graph personalised instances			
*	LGI_ID	integer	The unique identifier of the Learning Graph Instance
*	LG_ID	integer	The reference to the unique identifier of the Learning Graph that the Learning Graph Instance belongs to.
*	LG_NAME	string	The label of the Learning Graph
	LG_DESCR	string	Optional. A short description of the contents of the LG for sharing.
*	LEARNER_ID	integer	The reference unique identifier of the learner that the Learning Graph Instance belongs to.
*	CREATOR_ID	integer	The user (i.e. Tutor) that created the Instance of the Learning Graph
*	CREATED	date	
	LAST_MODIFIED	date	
	LGI_NODES	list	List of all nodes of the Learning Graph instance
	LGI_EDGES	list	List of all edges of the Learning Graph instance

7.3.1.1 Collection LGI_NODES

Columns			
The actual nodes of the Learning Graph Instance.			
*	LGIN_ID	integer	The unique identifier of the node within the learning graph instance.
*	LGI_ID	integer	The unique identifier of the Learning Graph Instance
*	SLAI_ID	integer	The unique identifier of the Smart Learning Atom Instance.
*	LGN_NAME	string	The label of the learning node
*	LGN_TYPE	integer DEFO 1	0 for Goal, 1 for SLA, 2 for Other
*	LGIN_WEIGHT	double	The weight of the node within the Learning Graph Instance (default = 0.0)
	CREATED	date	
	LAST_MODIFIED	date	

7.3.1.2 Collection LGI_EDGES

Columns			
The edges within the learning graph instance.			
*	LGIE_ID	integer	The unique identifier of the Learning Edge within the Learning Graph Instance.
*	LGE_WEIGHT	double	The weight of the edge (default = 1.0)
*	LGIN_SOURCE	integer	The unique identifier of the source node within the learning graph instance.
*	LGIN_TARGET	integer	The unique identifier of the target node within the learning graph instance.
	CREATED	date	
	LAST_MODIFIED	date	

7.3.2 Collection US_SmartLearningAtomInstances

Columns			
The MaTHiSiS Smart Learning Atom Instances.			
*	SLAI_ID	integer	The unique identifier of the Smart Learning Atom Instance.

*	SLA_ID	integer	The unique identifier of the Smart Learning Atom.
*	SLA_NAME	string	The label of the SLA
	SLA_DESCR	string	Optional. A short description of the SLA for sharing.
*	LEARNING_ACTIONS	list	A list of the learning actions that are attached to the SLA.
*	SLAI_WEIGHT	double	The weight of the SLA Instance (default = 0.0)
*	CREATED	date	
	LAST_MODIFIED	date	

7.3.3 Collection US_UserAccount

Columns			
MaTHiSiS User information			
*	ID	integer	The unique identifier of the User information
	USER NAME	string	MaTHiSiS user name
	USER PSWD	string	MaTHiSiS user password
*	ROLE	string	One of the possible roles in MaTHiSiS (at least one)
	ID_LAST_ACTIVITY	integer	Unique Identifier of the last activity executed by this user

7.3.4 Collection US_User

Columns			
MaTHiSiS User information			
*	ID	integer	The unique identifier of the User information
	FIRST NAME	string	First name of the user
	LAST NAME	string	Last name of the user
	USER NAME	string	MaTHiSiS user name
	USER PSWD	string	MaTHiSiS user password
	IS ACTIVE	integer	Determines if the user exists in the system
*	ROLE	string	One of the possible roles in MaTHiSiS (at least one)
	ID_LAST_ACTIVITY	integer	Unique Identifier of the last activity executed by this user

7.3.5 Collection US_LastActivity

Columns			
Last Activity executed by a MaTHiSiS user			
*	ACTIVITY_ID	integer	The unique identifier of an activity executed in the platform
	LG_LOCATION_URL	string	Location of the Learning graph used for this activity
	PA_USED	string	Identifier and type of Platform agent used (vocabulary Types defined according to the LA Ontology)

7.3.6 Collection US_LastActivityPerformance

Columns			
Performance information of the last Activity executed by a MaTHiSiS user			
*	PERFORMANCE_ID	integer	The unique identifier of performance information. Such information is dependant of time.
	Timestamp	date	Timestamp of last action performed in this activity
	ACTIVITY_ID	date	A reference to the id of the Activity (see Last activity executed)
*	STATUS	string	Declares the current status of the activity (Finished or in-progress)
	AFFECTIVE STATUS	string	Includes information about the current affective status of the learner when conducting the activity

7.3.7 Collection US_LearnerProfile

Columns			
Learner profile of a MaTHiSiS user			
*	LP_ID	integer	The unique identifier of the Learner profile
	FULL NAME	string	Full name of the learner
	AGE	integer	Age of the learner
*	LEARNING NEEDS + ACCESIBILITY	integer	Reference to the Learning needs and accessibility records
	LEARNING PREFERENCES	string	Define which are the main student preferences for learning
	CURRENT LEARNING GOAL	integer	Reference to the last learning goal to be achieved.

*	LAST PERFORMANCE INFORMATION_ID (KEY)	integer	Reference to the unique identifier of the learner last performance information related to the current learning goal
	PORTFOLIO	integer	Reference to the existing Learner portfolio

7.3.8 Collection US_LearnerProfileAccessibility

Columns			
Accessibility information of Learner profile			
*	ACCESSIBILITY_ID	integer	The unique identifier of the Accessibility information for the instance of a Learner profile
	AUDITORY PROCESSING	string	A Value to define Learner special needs according to the vocabulary to be defined in the MaTHiSiS Learner information ontology
	VISUAL PROCESSING	string	A Value to define Learner special needs according to the vocabulary to be defined in the MaTHiSiS Learner information ontology
*	SENSORIAL	string	A Value to define Learner special needs according to the vocabulary to be defined in the MaTHiSiS Learner information ontology
	COGNITIVE	string	A Value to define Learner special needs according to the vocabulary to be defined in the MaTHiSiS Learner information ontology

7.3.9 Collection US_LearnerProfileDisabilities

Columns			
Disabilities section of the Learner Profile			
	PHYSICAL		determines if the user have physical disabilities
	- affectation_in_motor_functions	BOOL	(true/false)
	- no_affectation_in_motor_functions	BOOL	(true/false)
	LEARNING		determines if the user have learning disorders
	- arithmetic_disorder	BOOL	(true/false)
	- writing_disorder	BOOL	(true/false)
	- writing_disorder	BOOL	(true/false)

	- reading_disorder	BOOL	(true/false)
	- spelling_disorder	BOOL	(true/false)
	- Auditory_processing_disorder	BOOL	(true/false)
	- Visual_processing_disorder	BOOL	(true/false)
	- Sensory_integration_disorder	BOOL	(true/false)
	- Learning_disorder	BOOL	(true/false)
	- Social_cue_disorder	BOOL	(true/false)
	INTELLECTUAL		determines if the user have intellectual disorder
	- mild	BOOL	(true/false)
	- moderate	BOOL	(true/false)
	- severe	BOOL	(true/false)
	AUDITORY		determines if the user have auditory disabilities
	- visual_impairment	BOOL	(true/false)
	- blindness	BOOL	(true/false)
	AUTISM SPECTRUM DISORDER		determines if the user have any autism spectrum disorder
	- autism	BOOL	(true/false)
	- high_functioning_autism	BOOL	(true/false)
	OTHER DISABILITIES	String	free text to describe other possible disabilities of the user
	GLOBAL DEVELOPMENTAL DELAY	BOOL	determines if the user have global delay
	COMMUNICATION AND LANGUAGE		determines if there are communication disorders
	- Aphasia	BOOL	(true/false)
	- Selective_mutism	BOOL	(true/false)
	- Dysarthria	BOOL	(true/false)

	- Stuttering	BOOL	(true/false)
	- Language_delay	BOOL	(true/false)
	- Dyslalia	BOOL	(true/false)
	- Dysphonia	BOOL	(true/false)
	SPECIFIC LEARNING DIFFICULTIES		specifies other possible learning difficulties
	- Writing	BOOL	(true/false)
	- Maths	BOOL	(true/false)
	- Reading-writing	BOOL	(true/false)
	BEHAVIORAL DISORDERS	BOOL	determines if the user have behavioural disorders
	ATTENTION DEFICIT HYPERACTIVITY DISORDER	BOOL	determines if the user have ADHD

7.3.10 Collection US_LearnerProfileInteractionPreferencesDisplay

Columns			
Display section of the Learner profile			
	COLOR_SCHEMAS		preferred contrast level of the user
	- High_contrast	BOOL	(true/false)
	- Low_contrast	BOOL	(true/false)
	COLOR FILTER		Preferred color filter of the user
	- Normal	BOOL	(true/false)
	- Red_green	BOOL	(true/false)
	- Green_red	BOOL	(true/false)
	- Blue_yellow	BOOL	(true/false)
	- greyscale	BOOL	(true/false)
	FONT_OPTIONS		set of font, font size and space between lines preferred by the user

	MAIN MENU ON BOTTOM	BOOL	useful if users can't reach the menu on the IWB

7.3.11 Collection US_LearnerProfileInteractionPreferencesLanguage

Columns			
Language section of Learner Profile			
	LANGUAGE PREFERRED	String	determines the preferred language of the user
	LANGUAGE SPOKEN		list of languages spoken by the user
	- English	BOOL	(true/false)
	- Spanish	BOOL	(true/false)
	- French	BOOL	(true/false)
	- Greek	BOOL	(true/false)
	- Italian	BOOL	(true/false)

7.3.12 Collection US_LearnerProfileLevelOfKnowledge

Columns			
Performance/Level of knowledge section of Learner Profile			
	ENGLISH		different scores of the user in English
	- score_p1_p3	integer	Values from 1 to 3
	- reading_p4_p8	integer	Values from 4 to 8
	- writing_p4_p8	integer	Values from 4 to 8
	- speaking_p4_p8	integer	Values from 4 to 8
	- listening_p4_p8	integer	Values from 4 to 8
	MATHEMATICS		different scores of the user in maths
	- score_p1_p3	integer	Values from 1 to 3
	- number_p4_p8	integer	Values from 4 to 8
	- using_and_applying_maths_p4_p8	integer	Values from 4 to 8

	- shape_space_and_measures_p4_p8	integer	Values from 4 to 8
	SCIENCE		different scores of the user in science
	- score_p1_p8	integer	Values from 1 to 8

7.3.13 Collection US_LearnerProfileLearningStyle

Columns			
Learning style section of Learner Profile			
	LEARNER PROCESSING		determines the processing style of the learner
	- Active_learner	BOOL	(true/false)
	- Reflective_learner	BOOL	(true/false)
	LEARNER INPUT		determines the input style of the learner
	- Visual learner	BOOL	(true/false)
	- Verbal learner	BOOL	(true/false)
	LEARNER UNDERSTANDING		determines the understanding style of the learner
	- Sequential learner	BOOL	(true/false)
	- Global learner	BOOL	(true/false)
	LEARNER PERCEPTION		determines the perception style of the learner
	- Sensing_learner	BOOL	(true/false)
	- Intuitive_learner	BOOL	(true/false)
	LEARNER PERSONAL FEATURES		determines if the user is on a group of special needs
	- no_specific_needs	BOOL	(true/false)
	- specific_needs	BOOL	(true/false)
	- special_needs	BOOL	(true/false)
	- advanced_learners	BOOL	(true/false)

7.3.14 Collection US_LearnerProfileMotivationState

Columns			
Motivation state section of Learner Profile			
	GOALS	String	Every objective the user has, according to the LG used in the system
	LEARNING INTERESTS	String	List of other subjects where the user may have interests
	AVG TIME STUDY	integer	Average time of a study session of this user
	AFFECTIVE STATE		determines the user affective state
	- Frustration	BOOL	(true/false)
	- Arousal	BOOL	(true/false)
	- Flow	BOOL	(true/false)
	- Boredom	BOOL	(true/false)

7.3.15 Collection US_LearningEnvironments

Columns			
The MaTHiSiS Learning Environments			
*	_id	oid AUTOINCREMENT	The unique identifier of the Learning Environment
*	name	string	The label of the Learning Environment
	description	string	A description for the Learning Environment
	lao_categories	array	The LAO categories the Learning Environment falls in
	platform_agents	array	The Platform Agents that can be found in the Learning Environment

7.3.16 Collection US_Classrooms

Columns			
The MaTHiSiS Classrooms			
*	_id	oid AUTOINCREMENT	The unique identifier of the Classroom
*	name	string	The label of the classroom
	description	string	A description for the classroom
	learners	array	List of learners of the classroom
	tutors	array	List of tutors of the classroom
	learning_environments	array	List of related learning environments

7.3.17 Collection US_PlatformAgents

Columns			
The MaTHiSiS Platform Agents			
*	_id	oid AUTOINCREMENT	The unique identifier of the Platform Agent
*	name	string	The label of the PA
*	laoCategories	array	The LAO categories the PA falls in
	addr	string	The address of the PA
	identifier_lao_categories	array	The LAO categories of the LM identifier with which the PA is compatible

7.4 Cloud Learner's Space Entities

7.4.1 Collection CLS_LearningSessions

The Learning Session handles information for the proceedings and the adaptation of the Learner's experience. The information is several identifiers related to the session: the Learner, the tutor, the learning environment, Learner's status and all related information to the session.

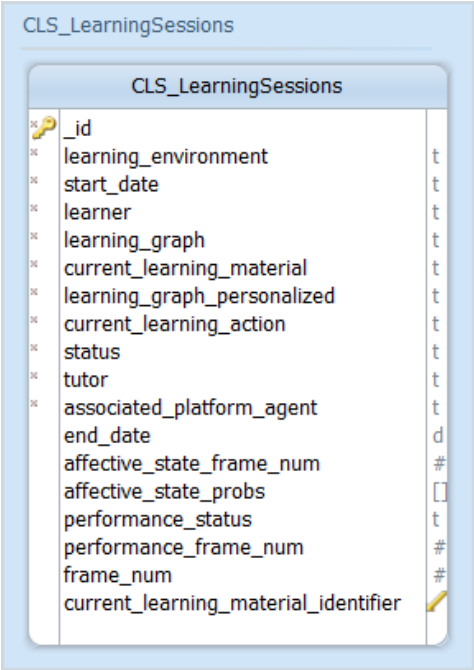


Figure 30 - The data model of the Learning Sessions

Columns			
The MaTHiSiS Learning Sessions			
*	_id	oid AUTOINCREMENT	The unique identifier of the Learning Session
*	learning_environment	string	The unique identifier of the learning environment associated to the LS
*	start_date	string	The start data and time of the LS
*	learner	string	The unique identifier of the learner associated to the LS
*	learning_graph	string	The unique identifier of the LG associated to the LS
*	current_learning_material	string	The unique identifier of the LM associated to the LS currently proposed to the learner
*	learning_graph_personalized	string	The unique identifier of the LG instance associated to the LS
*	current_learning_action	string	The unique identifier of the LA associated to the LS currently proposed to the learner
*	status	string	The current status of the LS
*	tutor	string	The unique identifier of the tutor that has started the LS

*	associated_platform_agent	string	The unique identifier of the PA associated to the LS
	end_date	string	The end date and time of the LS
	affective_state_frame_num	integer	The adaptation number at which we received the affective state of the learner
	affective_state_probs	list	The list of probabilities for each kind of affective state
	performance_status	string	Performance of the learner
	performance_frame_num	integer	The adaptation number at which we received the performance of the learner
	frame_num	integer	The number of adaptation triggered during the LS
	current_learning_material_identifier	map	The unique identifier of the LM identifier associated to the LS currently proposed to the learner on the associated PA

8. Standards

This is a list of already known standards that could be used in the platform. Some of them have been described by partners that have already used them.

IMS CP [http://www.imsglobal.org/content/packaging/]	
Description	The IMS Content Packaging Specification provides the functionality to describe and package learning materials, such as an individual course or a collection of courses, into interoperable, distributable packages. Content Packaging addresses the description, structure, and location of online learning materials and the definition of some particular content types. The Content Packaging Specification is aimed primarily at content producers, learning management system vendors, computing platform vendors, and learning service providers. Learning materials described and packaged using the IMS Content Packaging XML format should be interoperable with any tool that supports the Specification. Content creators can develop and distribute material knowing that it can be delivered on any compliant system, thereby protecting their investment in rich content development.
IMS LIP (IMS Learner Information Profile) [http://www.imsglobal.org/profiles/lipinfo01.html#1.1.2]	
Description	IMS Learner Information Package is based on a data model that describes those characteristics of a learner needed for the general purposes of: • Recording and managing learning-related history, goals, and accomplishments; • Engaging a learner in a Learning Experience; • Discovering learning opportunities for learners. The specification supports the exchange of learner information among learning management systems, human resource systems, student information systems, enterprise e-learning systems, knowledge management systems, resume repositories, and other systems used in the learning process. In this document such systems will be called learner information systems regardless of any other functionality they possess or roles they fulfil. The IMS Learner Information Package specification does not address requests for learner information or the exchange transaction mechanism. More details in annex Section 11.1.
IMS AccLIP (IMS Learner Information Package Accessibility for LIP Information Model) [http://www.imsglobal.org/accessibility/]	
Description	The IMS Accessibility project group focuses on adaptation or personalization of resources, interfaces and content to meet the needs of individuals. The group believes the best way to make a system or resource accessible to an individual is by meeting an individual's particular needs immediately within the learning context. Doing so decreases exclusion and increases usability. It is an effective way to meet legal accessibility requirements in many jurisdictions and has many business advantages for expanding the market of potential customers and users.
ISO IEC JTC1 SC36 (Information Technology for Learning, Education and Training)	
Description	To be described in a later version.
ISO IEC 24751:2008 (Information technology –Individualized adaptability and accessibility in e-	

learning, education and training)	
Description	ISO/IEC 24751 is intended to meet the needs of learners with disabilities and anyone in a disabling context. ISO/IEC 24751-1:2008 provides a common framework to describe and specify learner needs and preferences on the one hand and the corresponding description of the digital learning resources on the other hand, so that individual learner preferences and needs can be matched with the appropriate user interface tools and digital learning resources.
Emotion Markup Language (EmotionML) 1.0	
Description	To be described in a later version and according to the decisions made in modelling the affective status of learners in MaTHiSiS.
Experience API (xAPI) [http://www.adlnet.gov/tla/experienceapi/technical-specification]	
Description	The Experience API (xAPI) specification is flexible. It releases us from the constraints of only being able to track web-based formal learning; in addition, it is capable of tracking informal learning, social learning, and real world experiences. Example learning activities that can be tracked include reading an article, watching a training video, participating in a virtual world with augmented reality or simulation, using a mobile application, or having a conversation with a mentor. When a learner completes a learning activity, a simple human and machine readable activity statement is generated in a < actor >< verb >< object > format. Previous technologies require programming knowledge to understand the data, but with the activity statement format anyone in the learning process can interpret it. These statements are validated by and stored in a Learning Record Store (LRS). Third party reporting systems are able to access and query the data in an LRS to create analytics and visualizations. Example analytics that may be generated include how many times a learner attempted a question, how long a learner took to answer a question, or if a learner watched the introductory video and how that affected the overall score on the assessment. Analytics can be formulated for learners, instructors, groups, mentors, peers or any role in a learning environment. Learning designers and managers can use this information to make informed decisions, create more personalized learning, or modify their assessments, resulting in increased learner performance and an improvement in the overall Learning Experience.
WebSocket protocol [https://tools.ietf.org/html/rfc6455]	
Description	Official abstract of the WebSocket protocol: "*The WebSocket Protocol enables two-way communication between a client running untrusted code in a controlled environment to a remote host that has opted-in to communications from that code. The security model used for this is the origin-based security model commonly used by web browsers. The protocol consists of an opening handshake followed by basic message framing, layered over TCP. The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections (e.g., using XMLHttpRequest or <iframe>s and long polling).*"
OAuth 2.0 Authorization Framework [https://tools.ietf.org/html/rfc6749]	
Description	Official abstract of the OAuth 2.0 Authorization Framework: "*The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by*

	allowing the third-party application to obtain access on its own behalf."

Table 62 - Known standards

9. Conclusion

This document describes the full system architecture of the MaTHiSiS platform. After a first approach based on explorations and technological watch expertise of all technical partners involved in the project, provided in the deliverable ***D2.3 - Full System Architecture***[3], this document offers the updated vision that takes into consideration the outcome of the technical work of the first year of the project.

Through this document, we described the three layers of the high-level architecture. For the front-end layer, we provided the description of the tools used by end users, and the rationale behind each of them. For the back-end layer, we provided the description of the personalization and adaptation process, in addition of the description of the APIs, libraries and repositories involved in the data description, management and storage. Finally, for the platform agents' layer, we provided the description of the architecture needed to implement properly the final step of the learning materialization, taking into account the heterogeneous configurations of the installations.

We now have all technologies defined and even if the implementation could change, the architecture should not evolve significantly after this deliverable. In any case, the modifications, if any, and the implementation details are given through the technical reports of the project. In any case, this deliverable will serve as a reference for the future technical work.

10. References

[1] Nottingham Trent University (ed.): D2.1 *Formation of stakeholder groups*. Deliverable of the MATHISIS project, 2016.

[2] Nottingham Trent University (ed.): D2.2 *Full Scenarios for all Use Cases*. Deliverable of the MATHISIS project, 2016.

[3] DIGINEXT (ed.): D2.3 *Full system architecture*. Deliverable of the MATHISIS project, 2016.

[4] Nottingham Trent University (ed.): D2.5 *Evaluation Strategy*. Deliverable of the MATHISIS project, 2016.

[5] Vrije University of Brussels (ed.): D2.6 *Framework for impact assessment of MaTHiSiS against LEPOSA requirements M6*. Deliverable of the MATHISIS project, 2016.

[6] Vrije University of Brussels (ed.): D2.7 *The impact assessment report M12*. Deliverable of the MATHISIS project, 2016.

[7] DIGINEXT (ed.): D3.1 *The MaTHiSiS Smart Learning Atoms M12*. Deliverable of the MaTHiSiS project, 2017.

[8] Centre For Research and Technology Hellas (ed.): D3.3 *The MaTHiSiS Learning Graphs M12*. Deliverable of the MaTHiSiS project, 2017.

[9] DIGINEXT (ed.): D3.5 *Experience Engine M12*. Deliverable of the MaTHiSiS project, 2017.

[10] ATOS (ed.): D3.7 *Learner's Profile Repository M12*. Deliverable of the MaTHiSiS project, 2017.

[11] Centre For Research and Technology Hellas (ed.): D4.1 *MaTHiSiS sensorial component M12*. Deliverable of the MaTHiSiS project, 2017.

[12] University of Maastricht (ed.): D5.1 *Description of the robotic layer M12*. Deliverable of the MaTHiSiS project, 2017.

[13] OTE Academy (ed.): D5.4 *Description of the mobile layer M12*. Deliverable of the MaTHiSiS project, 2017.

[14] ATOS (ed.): D5.7 *Description of the interactive whiteboards layer M12*. Deliverable of the MaTHiSiS project, 2017.

[15] Nottingham Trent University (ed.): D6.1 *Adaptation and Personalization principles based on MaTHiSiS findings M12*. Deliverable of the MATHISIS project, 2016.

[16] ATOS (ed.): D7.1 *Integration Strategy and planning M6*. Deliverable of the MATHISIS project, 2016.

[17] DIGINEXT (ed.): D7.2 *MaTHiSiS platform, 1st release M12*. Deliverable of the MATHISIS project, 2017.

[18] MaTHiSiS Description of Action, 2016.

11. Annexes

11.1 IMS Learner Information Package

IMS Learner Information Package is based on a data model that describes those characteristics of a learner needed for the general purposes of:

- Recording and managing learning-related history, goals, and accomplishments;
- Engaging a learner in a Learning Experience;
- Discovering learning opportunities for learners.

The specification supports the exchange of learner information among learning management systems, **human** resource systems, student information systems, enterprise e-learning systems, knowledge management systems, resume repositories, and other systems used in the learning process. In this document such systems will be called learner information systems regardless of any other functionality they possess or roles they fulfil. The IMS Learner Information Package specification does not address requests for learner information or the exchange transaction mechanism.

IMS Learner Information Package is a structured information model. An XML binding is included but is not meant to exclude other bindings. The information model contains both data and meta-data about that data. The model defines fields into which the data can be placed and the type of data that may be put into these fields. Typical data might be the name of a learner, a course or training completed, a learning objective, a preference for a particular type of technology, and so on. Meta-data about each field can include:

- Time-related information;
- Identification and indexing information;
- Privacy and data protection information.

This meta-data is available for each and every field in the information model, either directly or via inheritance.

- Data Schema
- Identification: Biographic and demographic data relevant to learning;
- Goal: Learning, career and other objectives and aspirations;
- Qualifications, Certifications and Licenses (qcl): Qualifications, certifications and licenses granted by recognized authorities;
- Activity: Any learning-related activity in any state of completion. Could be self-reported. Includes formal and informal education, training, work experience, and military or civic service;
- Transcript: A record that is used to provide an institutionally-based summary of academic achievement. The structure of this record can take many forms;
- Interest: Information describing hobbies and recreational activities;
- Competency: Skills, knowledge, and abilities acquired in the cognitive, affective, and/or psychomotor domains;
- Affiliation: Membership of professional organizations, etc. Membership of groups is covered by the IMS Enterprise specification;

- Accessibility: General accessibility to the learner information as defined through language capabilities, disabilities, eligibilities and learning preferences including cognitive preferences (e.g. issues of learning style), physical preferences (e.g. a preference for large print), and technological preferences (e.g. a preference for a particular computer platform);
- Security key: The set of passwords and security keys assigned to the learner for transactions with learner information systems and services;º
- Relationship: The set of relationships between the core components. The core structures do not have within them identifiers that link to the core structures. Instead all of these relationships are captured in a single core structure thereby making the links simpler to identify and manage.

The learning information meta-data is broken into four categories:

- Time Information: Time of creation and time of expiration of a piece of data;
- Index and Source: Supports a pair consisting of a source and an ID assigned by that source, a local index that is used for cross-referencing, and a URI;
- Privacy and data protection information: Unstructured data to be determined by practice and implementation.
- All learning information data elements have meta-data sub-elements with the exception of atomic elements that can always inherit their meta-data. For example, in the Identification category, meta-data is associated with the Name element but not with its constituent elements since it is felt that the meta-data for the constituent elements cannot change independently of the meta-data for the Name element itself.

11.2 Platform Agents specifications

Platform Agents represent specific devices specifically selected for their technical capabilities, ubiquitous use in day-to-day life and their acceptance in some specific learning scenarios. All information described below comes from the ***D2.3 - Full system architecture*** [3] document and provide more technical details about each kind of Platform Agents.

11.2.1 TurtleBot

TurtleBot is a versatile robotic platform which can be used in many applications. TurtleBots are equipped with a Microsoft Kinect Sensor, assisting in navigation or activity recognition (depending on the usage), a small notebook with an embedded web-camera, serving mainly face-based interaction purposes. TurtleBots, thanks to the computer interface they can support, can stand between desktop and robot functionalities. This PA could interact directly with the learners or perform different activities while learners are interacting with a computer.

The main hardware includes:

- Wheels (Kobuki moving platform)
- Kinect camera
- On-board laptop. The available hardware of the laptop must be define regarding our needs (e.g. touchpad, touchscreen, frontal camera, microphone, LAN or WAN connectivity)

The main technical abilities are:

- Storage of video and audio signals on the device
- Video and audio streaming over the network
- Multimedia content (audio, video)
- Local computation
- Obstacle detection

- Face detection analysis through video signal
- Colour and object recognition

Figure 31 - TurtleBot

Tier Name	**TurtleBot**
Layer	Platform Agent
Inputs	
1	Source: PA lib API Data: Learning Material commands, with the context, i.e. the Learning Session identifier Schema: JSON
2	Source: Learners Data: interactions with the device through sensors (e.g. camera, microphone)
Outputs	
1	Destination: Learners Data: Specific behaviour through the PA interface
2	Destination: AIR lib API Data: Pre-processed signals from sensors Schema: JSON
3	Destination: IPA lib API Data: Interactions of the learner with the Learning Materials Schema: JSON - xAPI statement (final structure to be defined in D6.4 (M24))
4	Destination: PA lib API Data: Device identification and capabilities Schema: JSON
Sensors / Hardware	
1	Wheels (moving platform)

2	Kinect camera or webcam	
3	On-board laptop (hardware to be defined regarding needs in later version - initially touchpad, touchscreen, frontal camera, microphone, LAN or WAN connectivity, etc.)	
Software		
1	Ubuntu 14.04 LTS	
2	ROS indigo	
Technical abilities		**Feasibility**
1	Video signal	
	• Streamed over the network	Medium
	• Stored locally on the device	High
2	Audio signal	
	• Streamed over the network	High
	• Stored locally on the device	High
3	Can do local computations	High
4	Can play sound through speakers	High
5	Can move	High
6	Can detect obstacles through Kinect camera	High
7	Bidirectional communication through the network	High
Constraints		
1	Need high availability of the device that communicate with the TurtleBot	
2	Need high communication bandwidth for the streaming ability	
High – Level Information		**Computed on**
1	Content restitution through the screen, speakers and movements	Device
2	Gaze extraction through video signal	Device & CLS
3	Gesture recognition through video signal	Device & CLS
4	Face detection/recognition through video signal	Device & CLS
5	Affect State through video and audio signals	Device & CLS
6	Interactions through touchpad/touchscreen	Device
7	Obstacles detection	Device

Table 63 - Tier - TurtleBot

11.2.1.1 Software specification

TurtleBot is running using a workstation PC (Ubuntu 14.04 LTS). This system combines two components (Yujin Robot's Kobuki and Microsoft's Kinect) into an integrated development platform for ROS (Indigo version) applications.

11.2.1.2 Hardware specifications

11.2.1.2.1 Kobuki specifications

The mobile platform which is part of this robotic system is the Kobuki. Below is a summary of its characteristics:

	Feature	Value	Description
Robot parameter	wheelbase(bias)	230 mm	Length between the centre of the wheels
	wheel radius	35 mm	
	wheel width	21 mm	
Magnetic Encoder	ticks per revolution	52 tick/rev	
	pulses per revolution	13 pulse/rev	
Gear Box	1st stage	1 : 10	Motor to wheel or encoder to wheel
	2nd stage	22 : 12	
	3rd stage	30 : 11	
	4th stage	35 : 12	
	5th stage	34 : 1	
	resultant ratio	6545/132 = 49.5833	6545 turns of motors (or encoders) will make 132 turns of wheels

Gyro details

- 3-Axis Digital Gyroscope
- Manufacturer: STMicroelectronics
- Part Name: L3G4200D
- Measurement Range: ±250 deg/s
- Yaw axis is factory calibrated within the range of ±20 deg/s to ±100 deg/s

Motor details

- Brushed DC Motor
- Motor Manufacturer: Standard Motor
- Part Name: RP385-ST-2060
- Rated Voltage: 12 V
- Rated Load: 5 mN·m
- No Load Current: 210 mA
- No Load Speed: 9960 rpm ± 15%
- Rated Load Current: 750 mA

- Rated Load Speed: 8800 rpm ± 15%
- Armature Resistance: 1.5506 Ω at 25°C
- Armature Inductance: 1.51 mH
- Torque Constant (Kt): 10.913 mN·m/A
- Velocity Constant (Kv): 830 rpm/V
- Stall Current: 6.1 A
- Stall Torque: 33 mN·m
- CONTROL METHOD
- Driven by voltage source (H-bridge)
- Controlled by Pulse-width modulation (PWM)

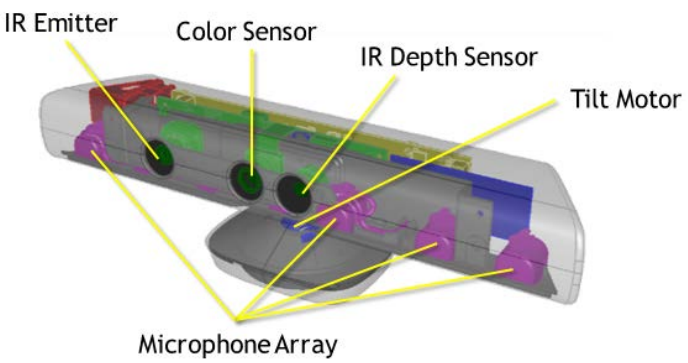
Battery details - Electronic characteristics

- Model: YCR-MO5
- Nominal capacity: 4S1P, 2200 mAh (±50 mAh)
- Battery name: Lithium-Ion Rechargeable name
- Battery model: HYB ICR18650NH 2200 mAh
- Nominal voltage: 14.4 V ~ 14.8 V
- Charging voltage: 16.8 V
- Discharge halt voltage: 9.6 V cut-off
- Nominal charging: 16.8, 1.1 A
- Max discharging current: 10 A (TBD)
- Internal resistance: 350 mΩ below
- Temperature: thermistor 103 F (25 °C)
- Weight: 210g below

11.2.1.2.2 Kinect camera specifications

TurtleBot is equipped with a Kinect v1 for Windows. Its main specifications are the following:

- An RGB camera that stores three channel data in a 640x480 resolution. This makes capturing a colour image possible.
- An infrared (IR) emitter and an IR depth sensor. The emitter emits infrared light beams and the depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. This makes capturing a depth image possible.
- A multi-array microphone, which contains four microphones for capturing sound. Because there are four microphones, it is possible to record audio as well as find the location of the sound source and the direction of the audio wave.
- A 3-axis accelerometer configured for a 2G range, where G is the acceleration due to gravity. It is possible to use the accelerometer to determine the current orientation of the Kinect.



Feature	Specifications
Viewing angle	43° vertical by 57° horizontal field of view
Vertical tilt range	±27°
Frame rate (depth and color stream)	30 frames per second (FPS)
Audio format	16-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing including acoustic echo cancellation and noise suppression
Accelerometer characteristics	A 2G/4G/8G accelerometer configured for the 2G range, with a 1° accuracy upper limit.
Depth camera	320x240
Max Depth Distance	~4.5m
Min Depth Distance	40 cm (in near mode)

11.2.2 NAO

NAO is an endearing, interactive and customizable humanoid robot companion.

Tier Name	NAO
Layer	Platform Agent
Inputs	
1	Source: PA lib API Data: Learning Material commands, with the context, i.e. the Learning Session identifier Schema: JSON
2	Source: Learners Data: interactions with the device through sensors (e.g. tactile sensors, camera, microphone)
Outputs	
1	Destination: Learners Data: Specific behaviour through the PA interface
2	Destination: Tutor or independent learners Data: MaTHiSiS frontend
3	Destination: AIR lib API Data: Pre-processed signals from sensors

	Schema: JSON
4	Destination: IPA Data: Interactions of the learner with the Learning Materials Schema: JSON - xAPI statement (final structure to be defined in D6.4 (M24))
5	Destination: PA lib API Data: Device identification and capabilities Schema: JSON
Sensors / Hardware	
1	Humanoid – weight: 5.4Kg – height: 574mm
2	Motherboard: • ATOM Z530 1.6 GHz CPU • 1 GB RAM • 2 GB Flash memory • 8 GB Micro SDHC
3	Battery: ~60 min (active use) v 90 min (Normal use) – Charging duration < 3h00
4	1 RJ45 - 10/100/1000 base T for Ethernet
5	IEEE 802.11 a/b/g/n for WiFi
6	USB (2.0 or less) (can plug Kinect, Asus 3D sensor, Arduino device, etc.)
7	Two 2D Cameras
8	Infra-red
9	Laser
10	Microphones
11	Loudspeakers
12	LEDs
13	Tactile sensors
14	Force Sensitive Resistors (FSRs)
Software	
1	NAOqi OS (GNU/Linux distribution based on Gentoo)
2	NAOqi (main software of NAO)
3	NAOqi supports OpenCV, OpenNI, OpenNI2, SQLite
4	Generally, values of sensors and events are accessible
Technical abilities	Feasibility

1	Video signal	
	• Streamed over the network	Medium
	• Stored locally on the device	High

Frames over network (uncompressed YUV422 images):

	local	Gb Eth.	100Mb Eth.	WiFi g
40x30 (QQQQVGA)	30fps	30fps	30fps	30fps
80x60 (QQQVGA)	30fps	30fps	30fps	30fps
160x120 (QQVGA)	30fps	30fps	30fps	30fps
320x240 (QVGA)	30fps	30fps	30fps	11fps
640x480 (VGA)	30fps	30fps	12fps	2.5fps
1280x960 (4VGA)	29fps	10fps	3fps	0.5fps

2	Audio signal	
	• Streamed over the network	High
	• Stored locally on the device	High
3	Can do local computations	High
4	Can play sound through speakers	High
5	Can move	High
6	Can make gestures	High
7	Can detect obstacles through sonar	High
8	Bidirectional communication through the network	High

Constraints

1	For face detection (Recognition is more demanding): • Face width: minimum 20 pixels in the image. For an adult, this corresponds to around 2 meters in a QVGA image and 4 meters in VGA. • Tilt/Pan: maximum +/- 15 deg (0 deg corresponding to a face facing the camera). • Rotation: in image plane: maximum +/- 45 deg.
2	For obstacle detection (NAO v5): • Frequency: 40kHz • Resolution: 1cm-4cm (depending on distance) • Detection range: 0.20 m - 0.80 m (< 0.20 no distance info, only existence) • Effective cone: 60◦ Resolution is the minimum change in distance that can be measured by the sensor when the target moves
3	Camera do not take clear image during walking

4	Community Aldebaran, suggests to use Asus Xtion because of no official drivers of Kinect in Linux
5	For movement detection camera has to be stable
6	When using Ethernet connection, tilting his head back could damage the head
7	Might confuse similar objectives
8	Be careful the CPU usage, there are cheap and expensive modules
9	SQL stores data in the disk, thus frequently storing effects high CPU usage
10	Laser sensor reduces the efficiency. Possible knocked out if NAO falls
11	Sonars (unlike laser) can detect glass and wall, however it is more expensive in CPU terms
12	Inaccurate walk and drift after abundant usage, thus odometry it's not a good idea
13	Need high communication bandwidth for the streaming ability

High – Level Information		Computed on
1	Content restitution through speakers, movements/gestures and LEDs	Device
3	Gaze extraction through video signal	Device & CLS
4	Skeleton motion recognition through video signal	Device a CLS
5	Facial expressions analysis through video signal	Device & CLS
6	Affect State through video and audio signals	Device & CLS
7	Interactions through tactile sensors	Device
8	Obstacles detection	Device

Table 64 - Tier - NAO

11.2.2.1 Additional software

NAO robot comes with software that will be useful during the development phase of the project.

Choregraphe:

- Create animations, behaviours and dialogs.
- Test them on a simulated robot, or directly on a real one.
- Monitor and control your robot.
- Enrich Choregraphe behaviours with your own Python code.
- You can retrieve software from website (as soon as NAO arrives)

Python and C++ SDKs:

- Python API allows:
  - use all of the C++ API from a remote machine, or
  - create Python modules that can run remotely or on the robot.
- C++ SDK is for:

- Develop with your favourite IDE: Visual Studio, Qt Creator, Eclipse or XCode.

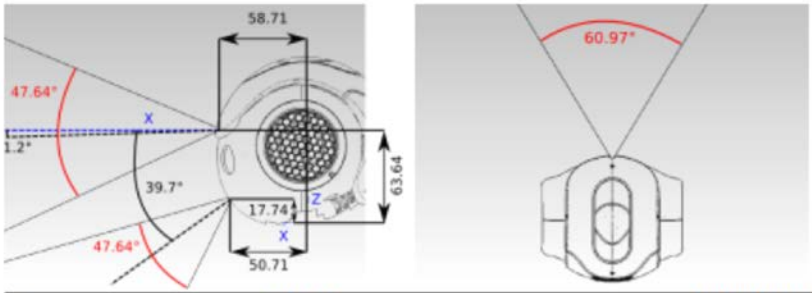
- Use the NAOqi framework on your PC. You can run the NAOqi executable on your platform, and use it with a simulator, or you can use the framework to write software which communicates with your robot.
- On Linux and Mac, you can cross-compile libraries to embed in Aldebaran robots.

Simulators:

- Choregraphe can connect with Webots
- As well as, C++/Python SDK are supported.

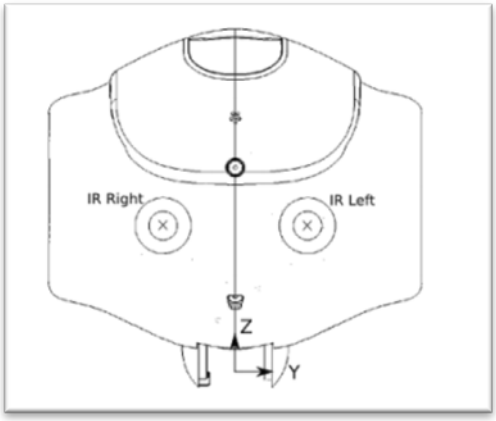
11.2.2.2 **NAO Hardware specifications**

11.2.2.2.1 2D Camera Specifications



Camera	Model	MT9M114
	Type	SOC Image Sensor
Imaging Array	Resolution	1.22 Mp
	Optical format	1/6 inch
	Active Pixels (HxV)	1288x968
Sensitivity	Pixel size	1.9μm*1.9μm
	Dynamic range	70 dB
	Signal/Noise ratio (max)	37dB
	Responsivity	2.24V/Lux-sec (550 nm)
Output	Camera output	1280*960@30fps
	Data Format	(YUV422 color space)
	Shutter type	Electronic Rolling shutter (ERS)
View	Field of view	72.6DFOV (60.9HFOV,47.6VFOV)
	Focus range	30cm infinity
	Focus type	Fixed focus

11.2.2.2.2 Infra-Red

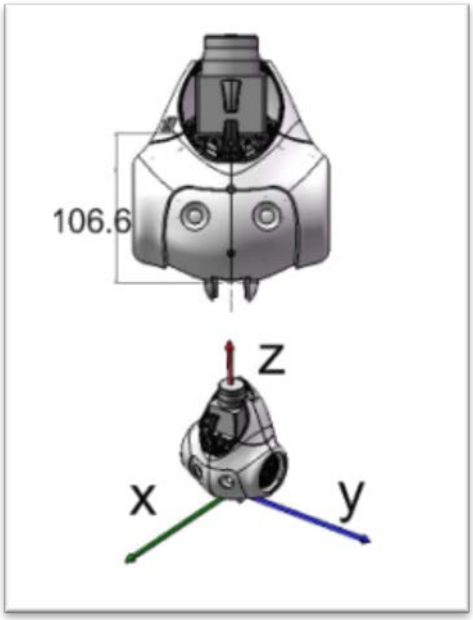


2 x I/R. (Academics only):

- Wavelength = 940 nm.
- Emission angle = +/- 60◦
- Power = 8 mW/sr

Remote control is supported. To control or to be controlled.

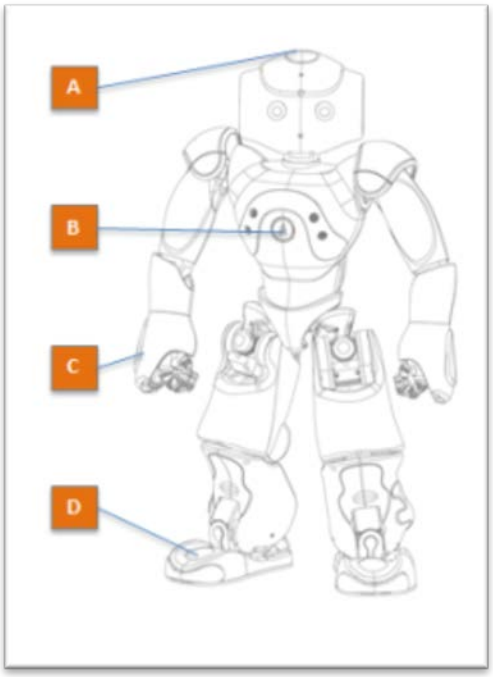
Laser Head (optional device)



- Wave length: 785nm.
- Resolution: 3 percent.
- Detection range: 0.2m - 5.6m.
- Angular range: 240◦

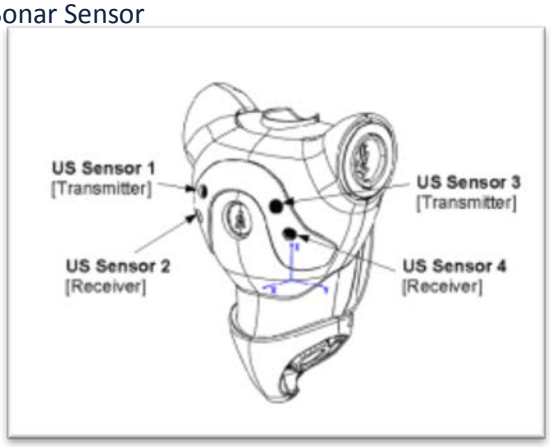
Polar and Cartesian coordinates.

Tactile Sensors

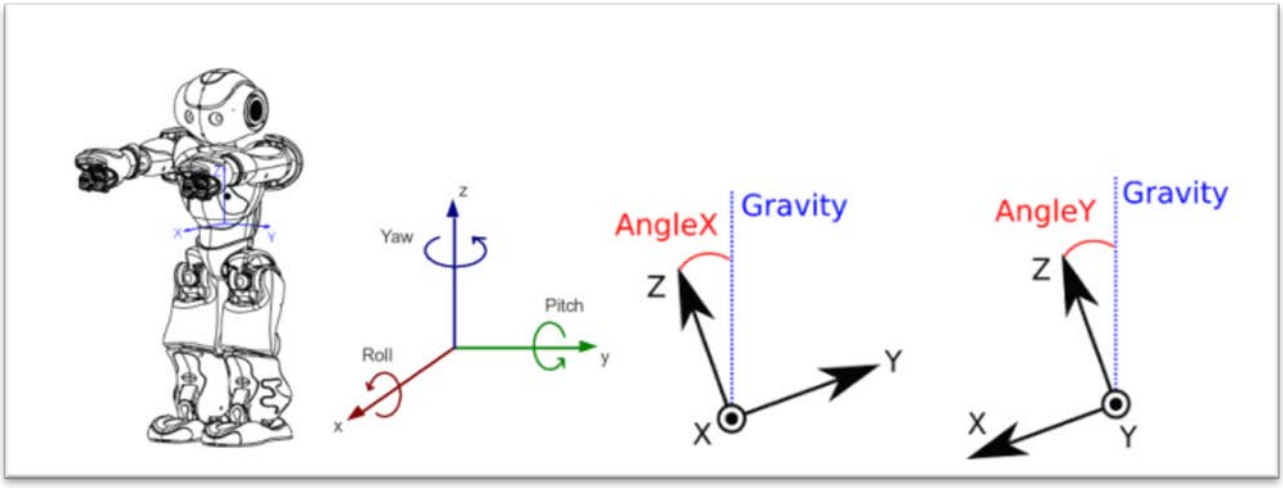


- A : 3 on head
- B : 1 on chest
- C : 3 on each arm
- D : 2 on each foot

11.2.2.2.3 Sonar Sensor



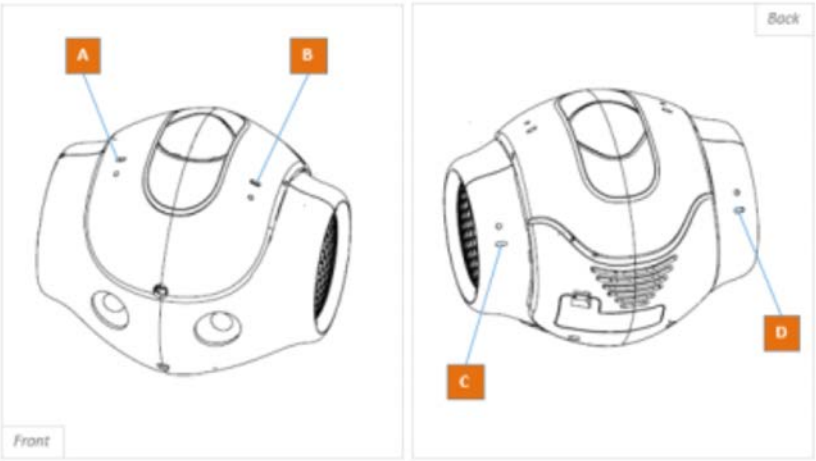
11.2.2.2.4 Inertial Unit



The output data enables an estimation of the torso speed and attitude (Yaw, Pitch, Roll).

- Gyroscope: rotation speed of the torso (rad/s), except z axis.
- Accelerometer: acceleration of the torso (m/s2).
- The above information feeds an Aldebaran Algorithm to compute angle (rad), except z angle.

11.2.2.2.5 Microphones



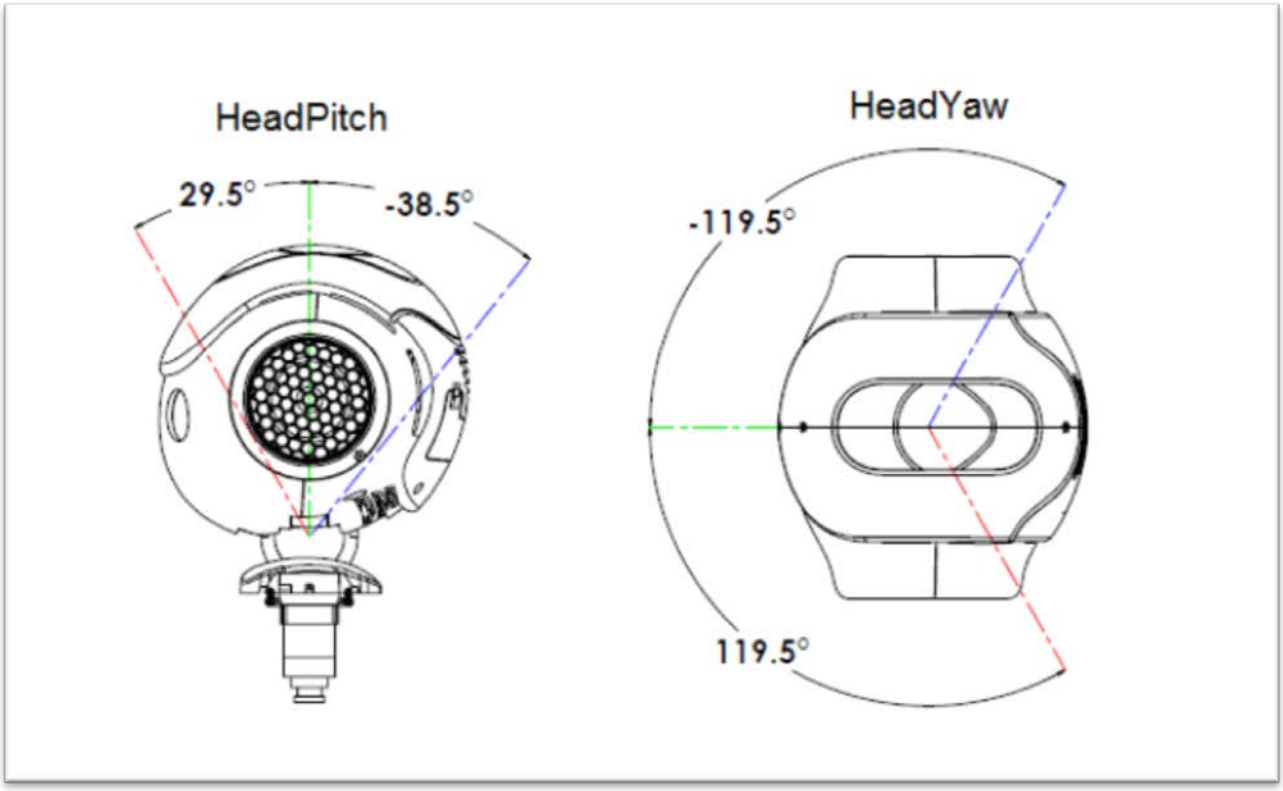
Microphones	x4 on the head
Sensitivity	20mV/Pa +/-3dB at 1KHz
Frequency range	150Hz to 12kHz

11.2.2.2.6 Joint Position Sensors

Specifications:

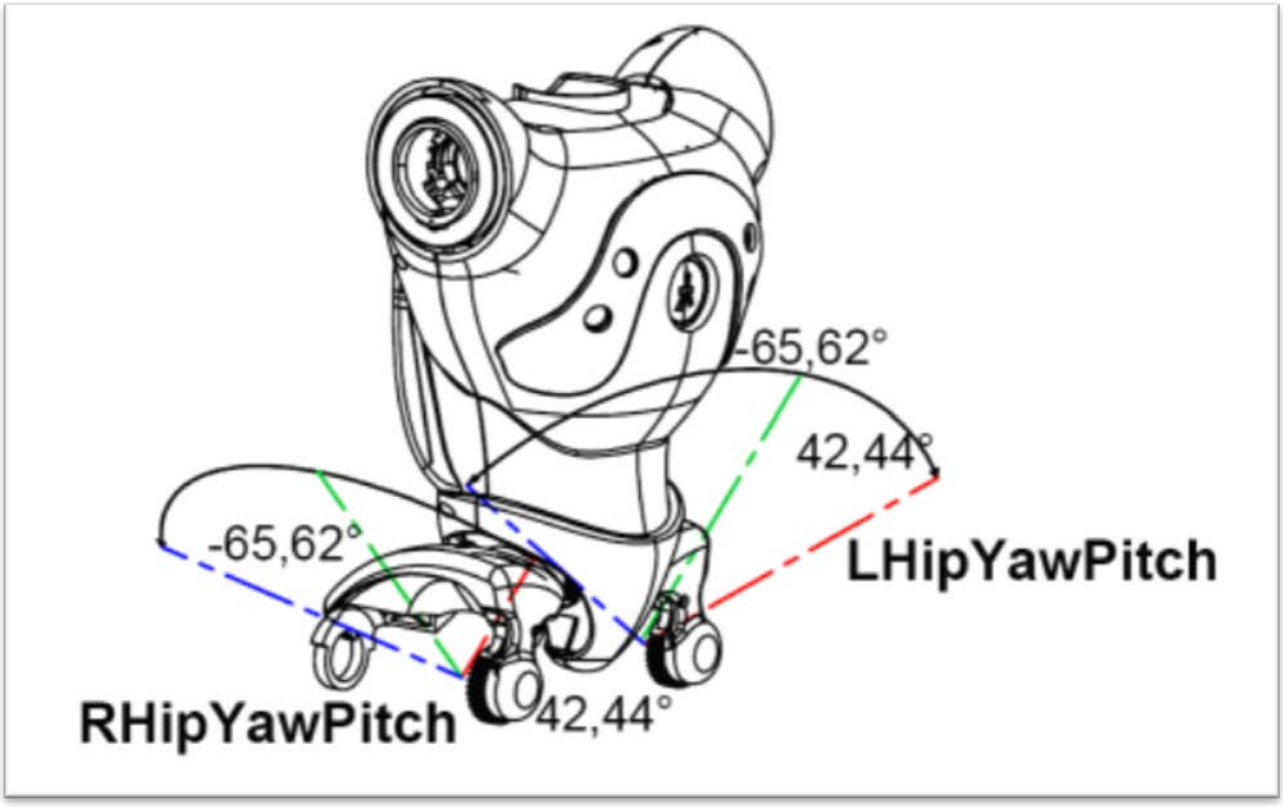
- 36 x MRE (Magnetic Rotary Encoders) using Hall-effect sensor technology.
- 12 bit precision, i.e. 4096 values per turn corresponding to about 0.1◦ precision

11.2.2.2.7 Head Joints



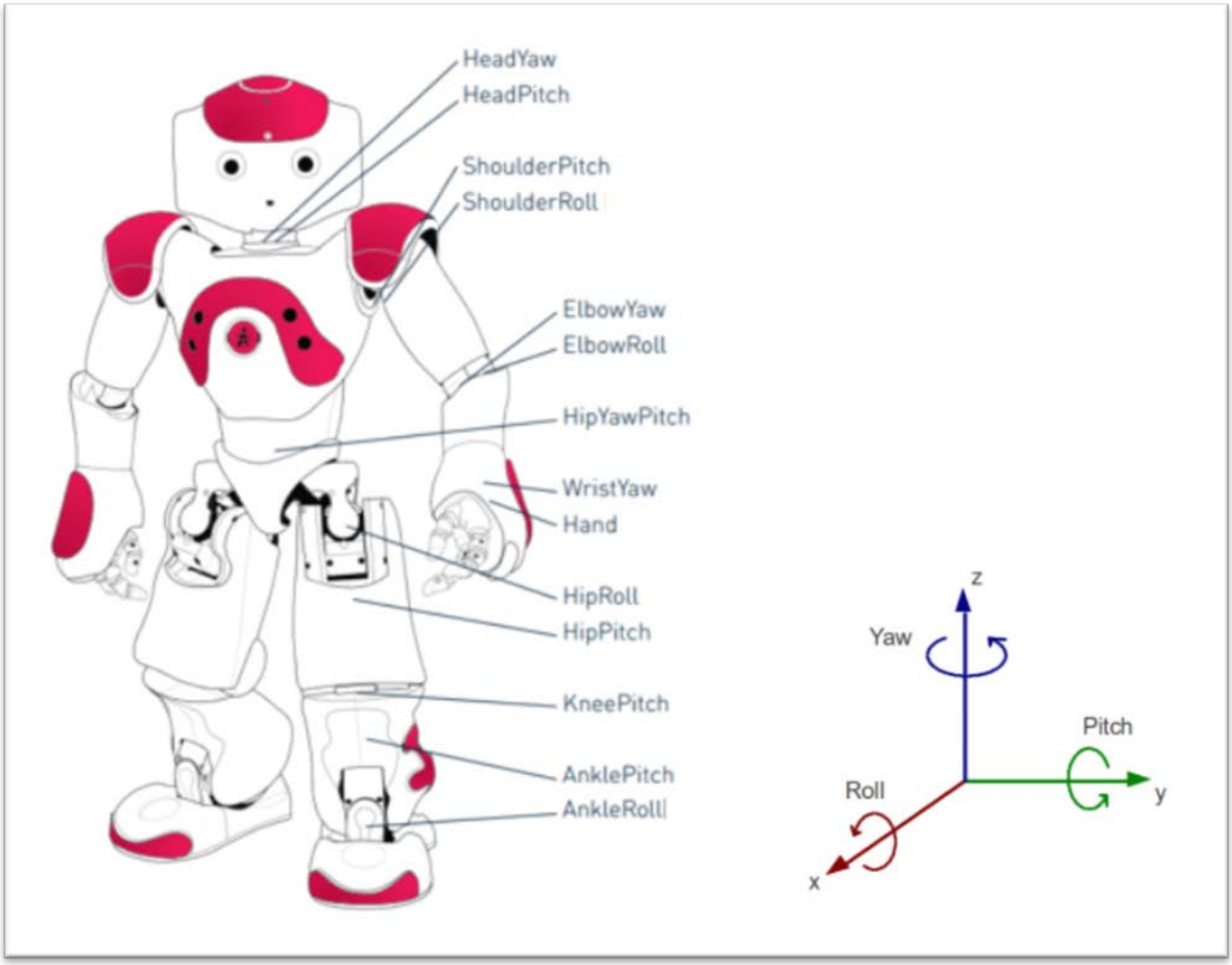
Due to potential shell collision at the head level, the Pitch motion range is limited according to the Yaw value.

11.2.2.2.8 Pelvis Joints



LHipYawPitch and RHipYawPitch are physically just one motor so they cannot be controlled independently. In case of conflicting orders, LHipYawPitch always takes the priority.

11.2.2.2.9 Motors



11.2.3 Interactive Whiteboard

Interactive whiteboards are whipeable boards used for teaching/learning or presentation purposes. At the time of writing this document, two different types of IWB are being assessed as MaTHiSiS Platform Agent. The types of sensors and interactions to be supported in the ecosystem will depend on the type of IWB chosen. Next, both types are described.

11.2.3.1 **IWB – Integrated projector**

Tier Name	**IWB - Integrated projector**
Layer	Platform Agent
Inputs	
1	Source: PA lib API Data: Learning Material commands, with the context, i.e. the Learning Session identifier Schema: JSON
2	Source: Learners Data: interactions with the device through sensors (e.g. movement sensors, microphone)

Outputs	
1	Destination: Learners Data: Specific behaviour through the PA interface
2	Destination: Tutor or independent learners Data: MaTHiSiS frontend
3	Destination: AIR lib API Data: Pre-processed signals from sensors Schema: JSON
4	Destination: IPA Data: Interactions of the learner with the Learning Materials Schema: JSON - xAPI statement (final structure to be defined in D6.4 (M24))
5	Destination: PA lib API Data: Device identification and capabilities Schema: JSON
Sensors / Hardware	
1	External camera (USB connection)
2	External microphone (USB connection)
3	External speakers
4	Screen
5	Integrated projector
6	Movement sensors on the edge of the screen
7	Network connection
Software	
1	Generic OS/specific distro with proprietary software

Technical abilities		Feasibility
1	Medium durability (sensors can be damaged)	N/A
2	No reflection	N/A
3	No light emitted by the screen	N/A

High – Level Information		Computed on
1	Classic interactions through the screen	Device

Table 65 - Tier - IWB - Integrated Projector

11.2.3.2 IWB – LED screen

Tier Name	IWB - LED screen
Layer	Platform Agent
Inputs	
1	Source: PA lib API Data: Learning Material commands, with the context, i.e. the Learning Session identifier Schema: JSON
2	Source: Learners Data: interactions with the device through sensors (e.g. touchscreen, camera)
Outputs	
1	Destination: Learners Data: Specific behaviour through the PA interface
2	Destination: Tutor or independent learners Data: MaTHiSiS frontend
3	Destination: AIR lib API Data: Pre-processed signals from sensors Schema: JSON
4	Destination: IPA Data: Interactions of the learner with the Learning Materials Schema: JSON - xAPI statement (final structure to be defined in D6.4 (M24))
5	Destination: PA lib API Data: Device identification and capabilities Schema: JSON
Sensors / Hardware	
1	Camera
2	Microphone
3	Speakers
4	Touchscreen
5	USB or another type of ports
6	Network connections
Software	
1	Generic OS
2	Android built-in

Technical abilities		Feasibility
1	Low durability (touchscreen can be damaged)	N/A
2	Heavy	
3	Can be used as a common computer (another PA)	
Constraints		
1	Reflection is a problem because the screen emits light by itself	
High – Level Information		**Computed on**
1	Classic interactions through the screen	Device
2	Affect State through motion-gesture detection	Sensorial Component

Table 66 - Tier - IWB - LED screen

11.2.4 Mobile devices

The mobile devices will be either smart phones or tablets. The ubiquitous aspect of this kind of Platform Agents will help to foster the acceptance of the MaTHiSiS platform.

Tier Name	**Smart phone & Tablet**
Layer	Platform Agent
Inputs	
1	Source: PA lib API Data: Learning Material commands, with the context, i.e. the Learning Session identifier Schema: JSON
2	Source: Learners Data: interactions with the device through sensors (e.g. touchscreen, camera, gyroscope)
Outputs	
1	Destination: Learners Data: Educational Material using capabilities of the device (e.g. screen, speakers) Schema: JSON
2	Destination: Tutor or independent learners Data: MaTHiSiS frontend
3	Destination: AIR lib API Data: Pre-processed signals from sensors Schema: JSON
4	Destination: IPA Data: Interactions of the learner with the Learning Materials

	Schema: JSON - xAPI statement (final structure to be defined in D6.4 (M24))
5	Destination: PA lib API Data: Device identification and capabilities Schema: JSON
Sensors / Hardware	
1	Touchscreen
2	At least one microphone
3	Speakers
4	Motion sensors (accelerometer, gravity sensors, gyroscopes, rotational vector sensors)
5	Compass (orientation magnetometers)
6	WLAN or cellular networks (e.g. 3G, LTE, 4G)
7	GPS
8	Ambient light / temperature sensor
9	Internal memory (1-4GB RAM, 2-32GB storage)
10	External memory slots (to achieve memory capacity up to 128GB) (not exploitable by the application)
11	Bluetooth
12	Vibrator
Software	
1	OS: consideration of Android, iOS
Technical abilities	
1	Video signal • Streamed over the network • Stored locally on the device
2	Audio signal • Streamed over the network • Stored locally on the device
3	Limited local computations
4	Can display through the screen
5	Can play sound through speakers
6	Can vibrate through the vibrator
7	Bidirectional communication through the network

8	Geolocation through GPS	
9	Interaction through the touchscreen	
Constraints		
1	Cellular network connection may be unavailable or intermittent	
2	Need high communication bandwidth for the streaming ability	
3	Battery autonomy using many sensors (in the order of few hours, to be in the safe side)	
High – Level Information		Computed on
1	Content restitution through the screen, speakers	Device
2	Geographic zone entering/leaving through GPS geolocation (e.g. at home, at school, etc.)	Device & CLS
3	Affect State through motion-gesture detection	Sensorial Component
4	Interactions through motions (e.g. tilt, shake, rotate, swing, etc.)	Device
5	Classic interactions through the touchscreen	Device

Table 67 - Tier - Mobile devices

The MaTHiSiS mobile application has to capture data that enable the computation/definition of a) the affect state of the learner, which implies the use of the inertia device's sensors, b) the performance of the learner with respect to successful completion of the exercises and c) the interaction of the learner with the material when real-time multi-party learning experience is offered. This mandates the implementation of a native mobile app. This allows for accessing the device's sensors. The supported operating systems are Android and iOS. The important thing to note here is that in order to monitor the performance of the learner, MaTHiSiS-compatible learning material has to be created. The MaTHiSiS compatible materials are any type of native mobile apps or web based materials which send the performance of the learner to the CLS formatted in xAPI statements.

Apart from the functionality offered through all types of MaTHiSiS PAs, the mobile devices differ in that:

1. They support three types of roles (administrators, parent/caregiver and learner).
2. They are equipped with sensors (gyroscope, accelerometer) and actuators (e.g. screen) that will be exploited primarily to derive the affect state of the learner and accordingly adapt the learning experience.
3. They are involved in synchronous and asynchronous collaboration scenarios (as will be elaborated in WP6 deliverables.

The mobile device differ from the rest PAs in principle. Mobile devices are equipped with sensors which can be used by MaTHiSiS in order to provide a fully interactive learning experience to the learners satisfying their varying needs. This capturing service runs in the background tasks of the device and sends continuously the relevant data to SC2 residing at the CLS. MaTHiSiS mobile application is able to provide to two or more pupils a simultaneously participation in one learning material to ensure an asynchronous communication between the devices exchanging information on the performance of users. In this manner, learning materials can be created based on the competitiveness of pupils.

11.2.5 Desktop and laptop computers

These devices allow the use of several sensors (i.e. webcam and other video capture devices, microphone, controllers such as mouse, keyboard, touchscreen or joystick). Moreover, computers provide important possibilities to include multimedia content (mainly screen and speakers).

This kind of devices has a wide range of possible functionalities through multiple components which can be added to provide different forms of interaction such as haptic devices, virtual controllers or bio-signal acquisition devices, increasing their possibilities.

The main technical abilities are:

- Storage of video and audio signals on the device;
- Video and audio streaming over the network;
- Multimedia content (audio, video);
- Local computation;
- Face detection analysis through video signal;
- Speech recognition through audio signal;
- Skeleton motion recognition through depth camera.

As Platform Agents, these devices are ideal to implement the Experiencing Service Server functionality, allowing the use of their full computational capabilities. Moreover, the compatibility of computers with all existing technologies makes them a very good choice to propose a simple way to integrate MaTHiSiS into existing environments.