.

beaconing

BREAKING EDUCATIONAL BARRIERS WITH CONTEXTUALISED PERVASIVE AND GAMEFUL LEARNING

Grant Agreement No. 687676

Innovation Action

ICT-20-2015

D3.6 System architecture

Due date	31.03.2017
Actual date	M15
Deliverable author(s)	Ioana Andreea Ștefan, Antoniu Ștefan, Massimiliano Cazzaniga, António Coelho, João Jacob, João Faria, Neil Judd, Michał Polak, Manuel Freire, Baltasar Fernández-Manjón, Oriol Janés, Ivan Martinez Ortiz.
Partner(s)	ATS, IMA, SUCCUBUS, INESC TEC, HFC, UCM, GEOMOTION, IMA, HWU
Contributors	Ancuta Florentina Gheorghe, Bogdan Drăgulin, Nicolae Oltei (ATS); Leonel Morgado, Hugo Paredes (INESC); Hossein Jamshidifarsani (HFC); Adam Jesionkiewicz (IFINITY); Laurent Auneau (SUCCUBUS); Theodore Lim, Aparajithan Sivanathan (HWU); Cristina Alonso, Ivan Perez, Victor Perez and Borja Manero (UCM); Pau Yanez (GEOMOTION).
Version	V.10
Status	Initial draft
Dissemination level	PU

Project Coordinator

Coventry University

Sylvester Arnab

Priory Street, Coventry CV1 5FB, UK

E-mail: s.arnab@coventry.ac.uk

Project website: http://www.beaconing.eu

This project has received funding under the Horizon 2020 Framework Programme of the European Union – Grant Agreement No. 687676

Version control				
Version	**Date**	**Author**	**Institution**	**Change and where applicable reason for change**
V1	02/11/2016	Ioana Andreea Ștefan	ATS	Set up the deliverable structure
V2	15/11/2016	Antoniu Ștefan	ATS	Added the initial proposal for the integrated Beaconing architecture
V3	17/02/2017	Massimiliano Cazzaniga	IMA	Mini games architecture
V4	17/02/2017	António Coelho, João Jacob, João Faria	INESC TEC	Contributions to the authoring tool and integration with mini-games and game plot.
V5	17/02/2017	Michał Polak	IFINITY	Contributions to the context-aware systems
V6	17/02/2017	Neil Judd	HFC	Contributions to the 'User Interfaces and Accessibility'
V7	21.02.2017	Manuel Freire, Baltasar Fernández-Manjón	UCM	Initial contribution regarding analytics; comments on other sections.
V8	24.02.2017	Oriol Janés	GEOMOTION	Contributions about: "context aware system" and "Context aware challenges authoring Tool"
V9	07/03/2017	Massimiliano Cazzaniga	IMA	Refinement of the description of the mini games configuration workflow.
V10	17/03/2017	Ioana Andreea Ștefan, Antoniu Ștefan	ATS	Contributions to the results and impact Refinement of the Beaconing Platform integrated architecture and of the usage scenarios

Quality control				
QA Version	**Date**	**QA Responsible**	**Institution**	**Change and where applicable reason for change**
V10	27/03/2017	Lawrence Howard	HFC	Internal Review
V10	28/03/2017	Laurent Auneau	SUCCUBUS	Internal Review
V10	29/03/2017	Jayne Beaufoy	COVUNI	Language Check

Release approval				
Version	**Date**	**Name**	**Institution**	**Role**
11	31/03/2017	Jannicke Baalsrud Hauge	BIBA	QM

Statement of originality
This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Contributors

Contributors				
Version of document contributed to	Chapter	Contribution description	Institution	Name of contributor
V3	2.4	Contributions to the Game Authoring Pipeline	SUCCUBUS	Laurent Auneau
V5	2.5	Contributions to the authoring tool and integration with mini-games and game plot.	INESC TEC	Leonel Morgado, Hugo Paredes
V5	2.7	Contributions to the ‘User Interfaces and Accessibility’	HFC	Hossein Jamshidifarsani
V5	2.3	Contributions to the context-aware systems	IFINITY	Adam Jesionkiewicz
V6	2.1	Contributions to the Beaconing architecture	HWU	Theodore Lim, Aparajithan Sivanathan
V6	2.8	Initial contribution regarding analytics; comments on other sections.	UCM	Cristina Alonso, Ivan Perez, Victor Perez and Borja Manero
V7	2.3	Contributions about: "context aware system" and "Context aware challenges authoring Tool"	GEOMOTION	Pau Yanez
V10	2.1, 2.2	Refinement of the Beaconing Platform integrated architecture and of the usage scenarios	ATS	Ancuta Florentina Gheorghe, Bogdan Drăgulin, Nicolae Oltei

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

APPENDIX: DEFINITIONS OF TERMS

ADL	Advanced Distributed Learning
API	Application Programming Interface
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
J2EE	Java Enterprise Edition
JSON	JavaScript Object Notation
PCG	Procedural Content Generation
POI	Point of Interest
QR Code	Quick Response Code
SSO	Single Sign On
TCO	Total Cost of Ownership
URL	Uniform Resource Locator
xAPI	Experience API

EXECUTIVE SUMMARY

Deliverable 3.6 System architecture builds upon the outcomes of D3.3 Learning environment system specification, D3.4 Interface design specification, D3.5 Game design document, and D4.6 Game analytics and adaptation component design. The document details the work carried out by Beaconing partners in Task 3.5 Infrastructure, architecture and systems specification, as well as in undergoing WP4 tasks, informing and supporting the development processes. The document describes the core Beaconing components and their architectures: the Core services; the Personal data store; the Beacon repository; the location context; the Game plot Editor; the Gamified lesson path runtime: the Mini game runtime: the Authoring tool interface; the Procedural content generation; the Game deployment; the User interface; the Learning analytics data store; the Learning analytics services; and the Learning analytics reporting. The document presents the Beaconing Platform overview and how these components are integrated and interact on the Beaconing Platform. The usage scenarios presented herein concern the Game Plot authoring; the Gamified Lesson Path authoring; the User Interface; the Procedural Content Generation; the Game deployment and runtime; Analytics collection. The Beaconing back-end services are detailed, with special attention being given to personal data management.

1 INTRODUCTION

1.1 BACKGROUND

The Beaconing Project aims to enable the transition from disparate technologies that expose students and teachers to episodic, disconnected learning and teaching activities to continuous, collaborative, and informed experiences, supported by in-depth analytics. Moreover, moving towards ecosystems, and increasing end-user knowledge enables a new business models with extensive capabilities and relationships.

This deliverable presents the architecture of the Beaconing Platform, providing insights on its core components and workflows. It also articulates key usage scenarios, underlying the rationale for creating a dynamic and flexible solution, where existing and new technologies can be integrated to enhance learning processes.

1.2 ROLE OF THIS DELIVERABLE IN THE PROJECT

Building upon the outcomes of WP3 Requirements, design and specifications D3.1 Requirement analysis; D3.2 Technology and Learning environment inventory; D3.3 Learning environment system specification; D3.4 Interface design specification; and D3.5 Game design document), of WP4 Platform Development and Ecosystem Integration (D4.6 Game analytics and adaptation component design), as well as on the outcomes of the workshops organized in Nantes (November 1-3) and Madrid (February 1-3), this deliverable provides a unified view of the Beaconing approach, guiding the development processes in WP4 for the individual Beaconing components and the integrated ecosystem due in M18. The deliverable reflects the work carried out to create the Beaconing mock-ups for the core components: plot editor; authoring system; procedural content generator; context-aware system; user interfaces; learning analytics.

An updated version of the Beaconing Platform and Ecosystem architecture will be provided in M28, based on the feedback collected from testing and from the pilots.

1.3 APPROACH

The Beaconing Platform is based on an architecture of connected services that work together to provide teachers with integrated tools to create pervasive gamified learning experiences for students. The architectural approach is partially based on principles of loose coupling, in which individual components abstract their implementation details and expose only specific inputs and outputs based on data workflows and clear interactions with other components.

The design has focused on creating components that are self-contained, with minimal software dependencies, and with automated installation and testing procedures that allow reproducible results.

Automated installation provides a single execution point (script or installer) that can perform all the tasks needed to go from uninstalled (barebones hardware + operating system) to a fully-installed state. For unattended installations, users can provide configuration parameters as an answer file.

Automated testing is similar in nature to automated installation. Components should bundle tests and/or demo data that allows an installer to validate that the installation works as intended. Running these tests or demos should be straightforward, ideally as simple as executing a single script and reading an output report.

Reproducible installation and testing refers to minimal dependencies on the specific hardware and software environment in which the component is run.

1.4 STRUCTURE OF THE DOCUMENT

Section 1 provides an overview of the role of this deliverable within the project workflow and the approach taken, highlighting the connections between the designs made in WP3 and linking with the development efforts in WP4.

Section 2 presents the Beaconing architecture, starting with the high-level platform architecture and workflows, and detailing the specifics of each of the core Beaconing components: context-aware-systems and communication; game authoring pipeline; authoring system; mini games (game client; game engine); user interfaces, including the accessibility approach; learning semantic system and learning analytics. It also describes the approach undertaken for the Beaconing Ecosystem, in an effort to support the integration of existing and emerging applications and systems that could strengthen the Beaconing solution and facilitate large-scale piloting.

Section 3 details the main conclusions on the architecture of the Beaconing components and workflows.

2 BEACONING ARCHITECTURE

2.1 PLATFORM ARCHITECTURE OVERVIEW

2.1.1 Key Components

The architecture of the BEACONING platform is built around a component model that splits core functionalities into multiple interdependent components. This provides an open architecture which allows integration of new components and provisioning of alternative implementations for each component depending on particular use cases.

Table 1. Key components BEACONING platform

Component	Task	Description
Core services	T4.1	This component acts as the main data store for the platform and provides services that other components can query to obtain information such as the student profile, available game assets, list of available game plots, mini-games and gamified lessons paths, etc.
Personal data store	T4.1	Stores personal identification data for students, separately from the main data stores.
Beacon repository	T4.2	Manages geo locations, beacon identifiers, QR codes and their associated purpose.
Location context	T4.2	Integrates with the gamified lesson path runtime and provides information about the student's current location/beacons in range.
Game plot editor	T3.4 T4.3	Allows the creation of game plots (the game perspective of the gamified lesson path that will be presented to students).
Gamified lesson path runtime	T4.3 T3.3	This component runs the gamified lesson path on the student's device.
Mini game runtime	T4.3 T3.4	Integrates into the gamified lesson path runtime to execute the mini-games included in the lesson path.
Authoring tool interface	T4.4	This tool allows learning designers and teachers to customize game plots from the learning perspective and turn them into gamified lesson paths.
Procedural content generation	T4.4	Generates multiple personalized gamified lesson paths for different student profiles from the standard gamified lesson path created using the authoring tool.
Game deployment	T4.4 T3.4	Matches the student profile with a personalized gamified lesson path and delivers it to the gamified lesson path runtime.

User interface	T4.5 T3.4	The main user interface of the platform where teachers can manage and track students, access platform tools and where students can see their own assignments and progress.
Learning analytics data store	T4.6	This component stores the experience data collected from the students.
Learning analytics services	T4.6	Interprets the collected analytics into meaningful reporting data.
Learning analytics reporting	T4.6	Provides a dashboard for learning analytics data that integrates into the main user interface for both teachers and students.

Component interactions for key usage scenarios

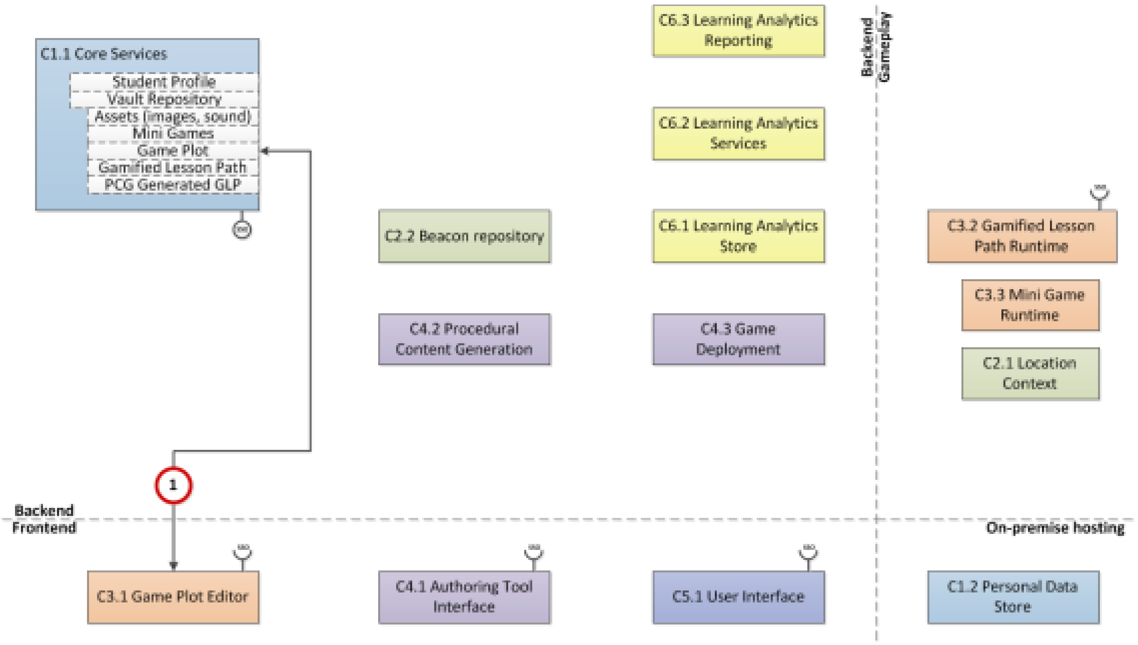


Figure 1. Game plot authoring

1. The game plot editor (C3.1) uses the core services component to save new game plots or to edit existing ones. The editor can also call the core services to retrieve a list of available assets and mini-games that can be included in the game plot.
2. During the authoring process, the authoring tool (C4.1) can create new gamified lesson paths or edit existing ones.
3. As described in the game design document (D3.5 Game design document), the game authoring pipeline is constructed so that learning designers and teachers that use the authoring tool start by adapting an existing game plot. For this purpose, the authoring tool can retrieve the list of game plots and their descriptive metadata from the core services component.
4. Retrieve the list of available mini-games (C3.3) and their descriptive metadata, for customization purposes.

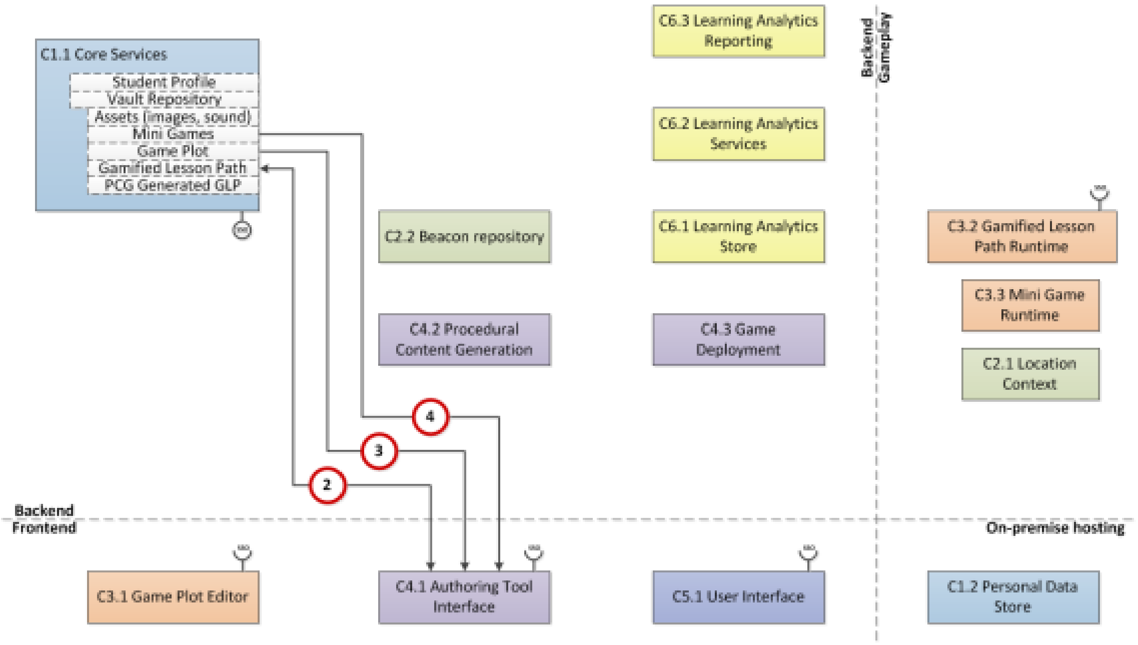


Figure 2. Gamified lesson path authoring

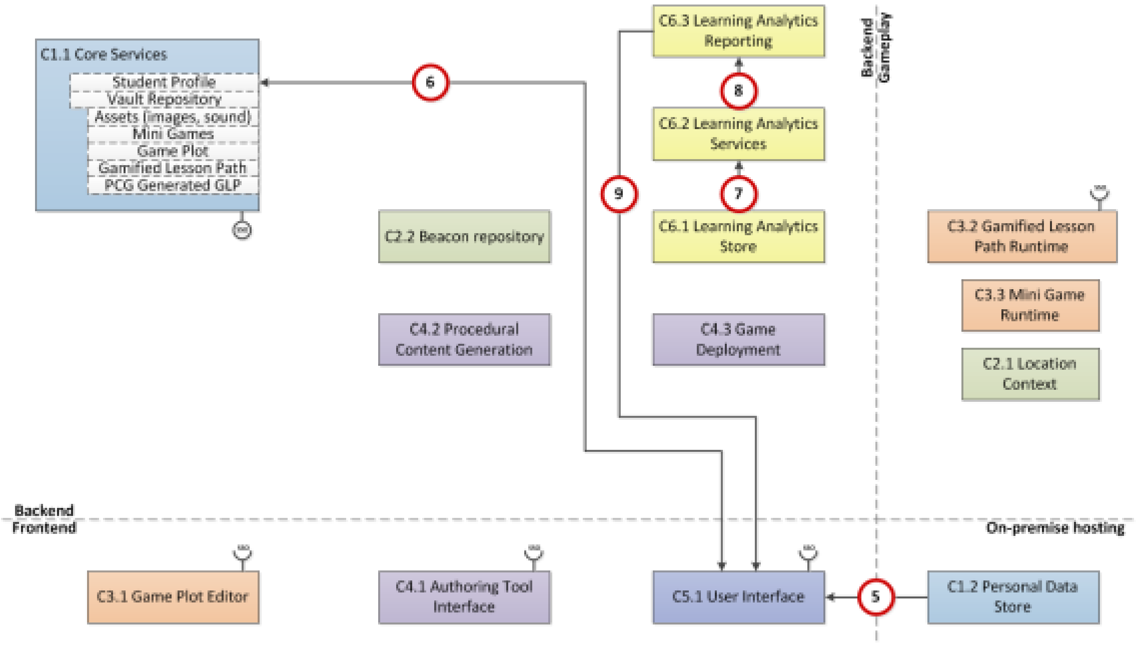


Figure 3. User interface

5. The user interface component (C5.1) will query and manage user data directly from the client’s browser, using a combination for CORS and JSONP. This will allow implementation scenarios where personal identification data is stored separately from the main components and is managed by the end-user organization.
6. The User Interface (C5.1) manages students and classes, using depersonalized internal system identifiers.
7. C6.2 retrieves xAPI statements from the LRS (C6.1).
8. C6.2 processes xAPI statements into meaningful reports and generates the dashboard (C6.3).
9. Display the learning analytics dashboard (C6.3) integrated into the main user interface (C5.1).

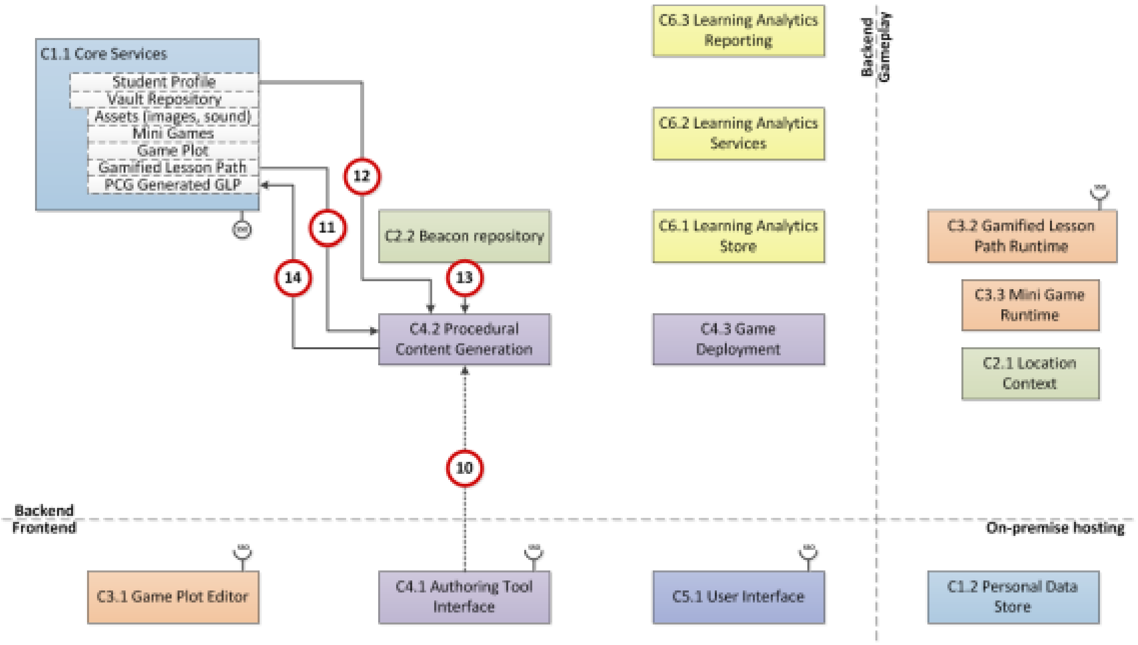


Figure 4. Procedural content generation

10. When the gamified lesson path authoring is complete, teachers can trigger the PCG based adaptation (C4.2) of the gamified lesson path into multiple versions, each adapted for specific learner profiles.
11. C4.2 retrieves the generic gamified lesson path from core services (C1.1).
12. C4.2 uses learner profile data from core services (C1.1) to determine how many versions to generate.
13. C4.2 configures location parameters (Beacon IDs, GPS coordinates, barcodes, etc.) using information from the Beacon repository (C2.2).
14. C4.2 save the gamified lesson path adaptations generated by the PCG into the core services (C1.1) repository.

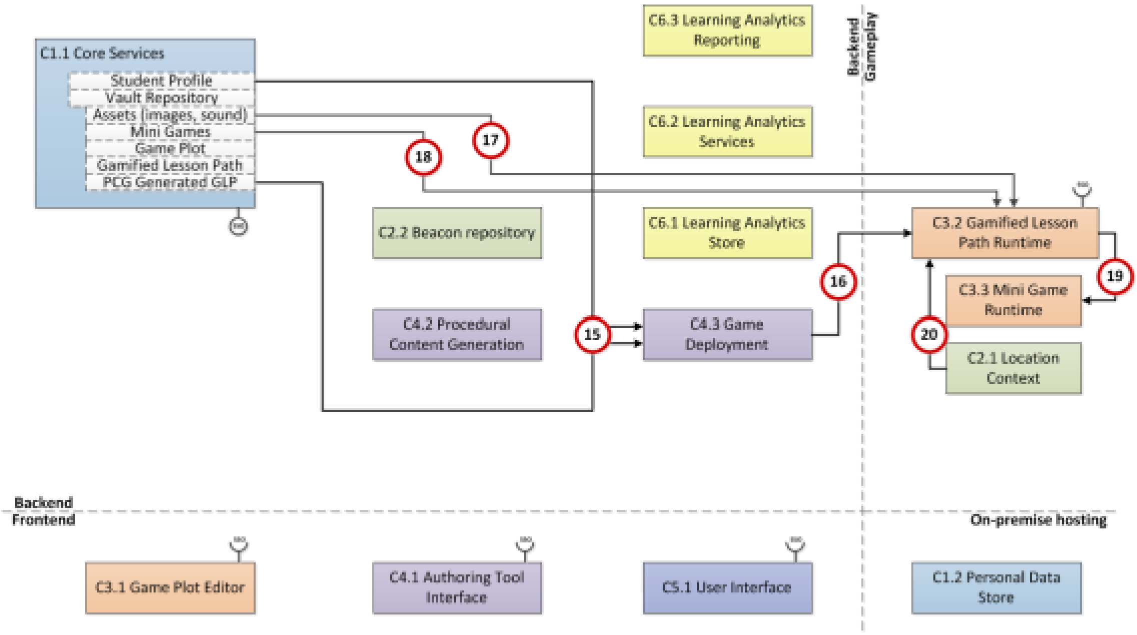


Figure 5. Game deployment and runtime

15. C4.3 matches the learner profile retrieved from core services (C1.1) with the most suitable gamified lesson path from available versions.
16. C4.3 provides the gamified lesson path to the runtime (C3.2), including player profile data such as the name of his chosen avatar and the student's progress on the main storyline.
17. C3.2 downloads the assets from core services (C1.1).
18. C3.2 download the mini-games references by the gamified lesson path.
19. C3.2 invokes the mini-game runtime (C3.3) to run a mini-game.
20. C2.1 provides context information (Beacon IDs, GPS coordinates, barcode information etc.) to the gamified lesson path runtime (C3.2).

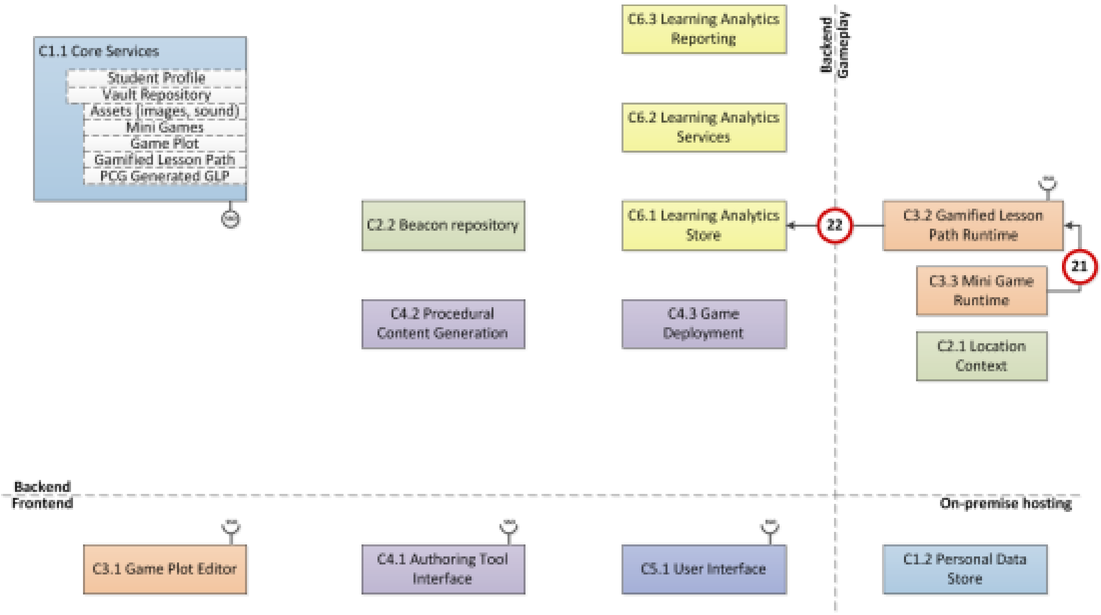


Figure 6. Analytics collection

21. Mini-games (C3.3) generate xAPI statements and pass them to the gamified lesson path runtime (C3.2).
22. The gamified lesson path runtime (C3.2) generates and submits xAPI statements, including statements generated by the mini-games (C3.3), to the learning analytics store (C6.1).

2.1.2 Beaconing Workflows

The diagram below details the interaction workflow between the core components and different user roles with regards to the game authoring pipeline.

Beaconing components provide a multi-layer approach to designing gamified lesson paths, with the purpose of separating most game authoring aspects from educational content authoring, so that it better fits the roles of intended users (game designers; learning designers; teachers; students).

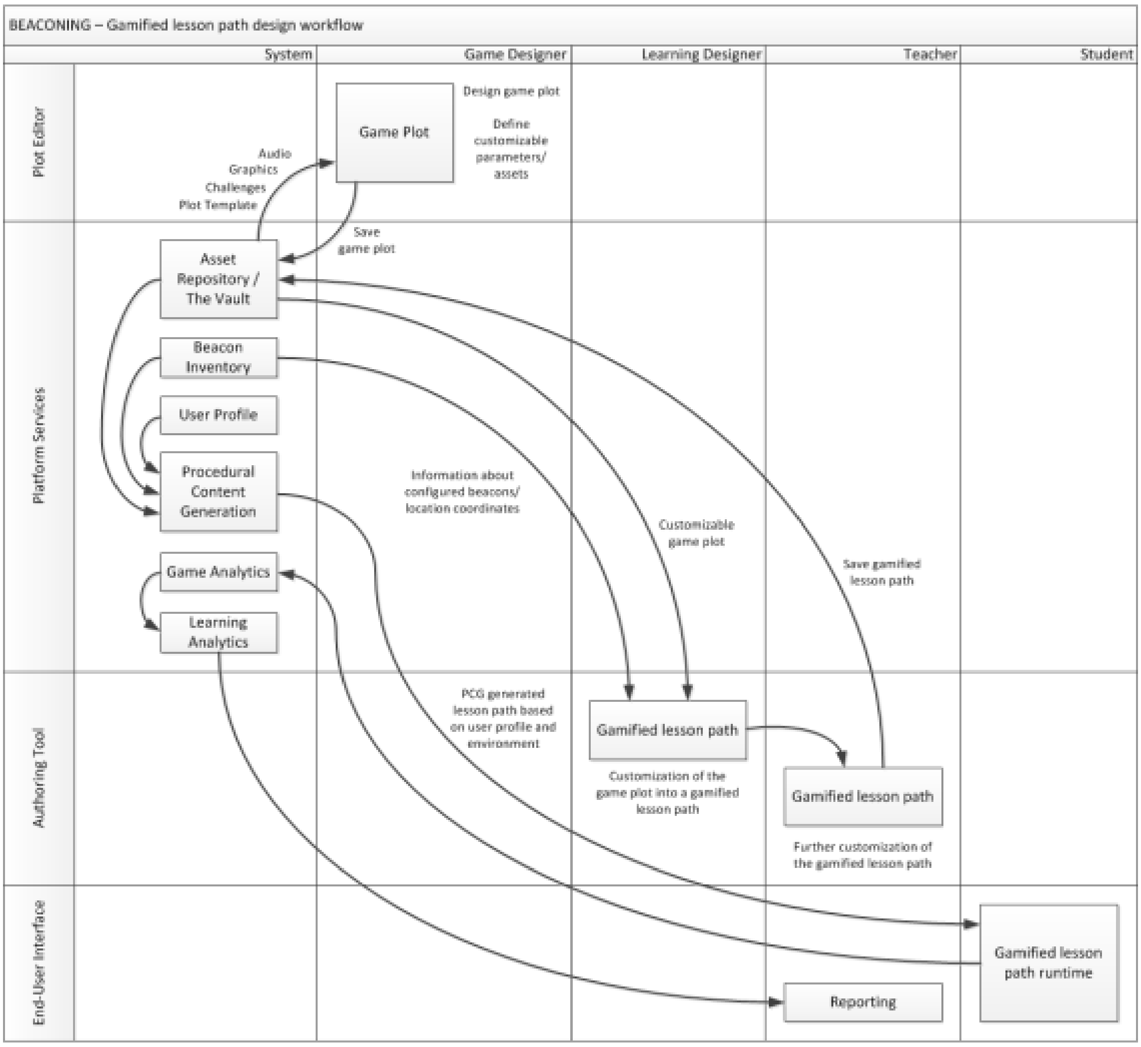

BEACONING – Gamified lesson path design workflow
System
Game Designer
Learning Designer
Teacher
Student
Plot Editor
Platform Services
Authoring Tool
End-User Interface
Game Plot
Design game plot
Define customizable parameters/ assets
Audio
Graphics
Challenges
Plot Template
Save game plot
Asset Repository / The Vault
Beacon Inventory
User Profile
Procedural Content Generation
Game Analytics
Learning Analytics
Information about configured beacons/ location coordinates
Customizable game plot
Save gamified lesson path
PCG generated lesson path based on user profile and environment
Gamified lesson path
Customization of the game plot into a gamified lesson path
Gamified lesson path
Further customization of the gamified lesson path
Gamified lesson path runtime
Reporting

Figure 7. Beaconing Workflows

2.2 BACK-END SERVICES

Backend services, developed as part of task T4.1, will provide a central access point for persistent data storage and enumeration. Providing these services using an open API facilitates the development and integration of new components from other vendors.

In accordance with the specifications defined in deliverable D1.8, the backend service architecture will include a specialized component for storing personal identification data, and the main data structures will store only de-personalized data.

Core component

Table 2 lists the services provided by the core components of the BEACONING platform

Table 2. Provided services core component

Service	**Description**
Student profile	Stores data about student preferences, competencies and achievements. This data is stored in a de-personalized manner, so that it cannot be traced back to a real identity without also accessing the personal data store component.

Asset repository	Repository for game assets (images, sounds, etc.) which can be used during the authoring process.
Mini-game repository	Provides a list of available mini-games and their associated descriptive metadata. The metadata describes the parameters that can be customized as part of the authoring process.
Game plot repository	Manages available game plots that can be instantiated into gamified lesson paths, or further refined by game designers.
Gamified lesson path repository	Manages gamified lesson paths created by learning designers. These can be adapted into new gamified lesson paths as required.

Personal data store

This component provides access to the identity of a user, so that other components, such as reporting, can display this data directly to authorized persons (eg. teachers reviewing students' performance). Having these functions in a separate component provides better control and auditing of access to personal data and also allows the services to be located according to required jurisdiction regulations.

Table 3 presents the user identification services supported at platform level.

Table 3 Provided services for the personal data store component

Service	**Description**
User identification	Allows other components to retrieve the identity of a user using the surrogate id from the main data store. It should also support multiple identification requests in a single function call for performance purposes.

2.3 CONTEXT-AWARE SYSTEMS AND COMMUNICATION

This section of the deliverable presents the architecture of the context aware system component. This component will be used to construct location based services on the BEACONING platform, enabling teachers to extend the students' "game-based learning experience" beyond the boundaries of the classroom and create context based challenges.

The graph below represents the high-level architecture of the component:

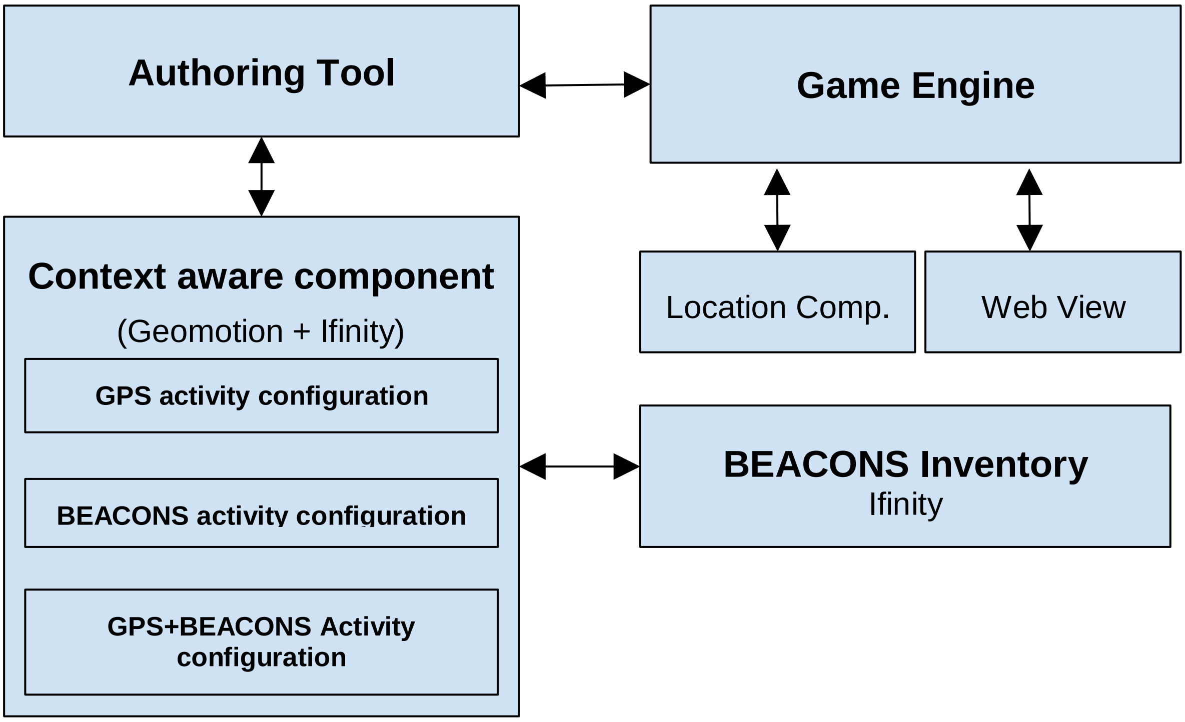


Figure 8. The architecture of the context-aware component

Modules of the context aware component:

- *Context Aware Challenges Authoring Module.* This module of the context aware component will allow users (teachers, learning designers) to create GPS based games and activities. After choosing this type of activity, the user will be able to provide GPS coordinates or pick on the map places he wants to use and personalize the content of each Point Of Interest. The system will use a set of predefined challenges to make activities and no programming skills will be required. This module together with the Beacon Inventory enables the creation of mixed activities using GPS, Beacons, or QR codes.

  Location-based challenges can be of different types:
  - **Follow the Path:** a linear location-based game challenge, where students have to find and check-in at a specific Point of Interest (POI). When they find a POI, the system will show a clue with information about the location of the next one.
  - **Treasure hunt:** a linear exploratory location-based game challenge, where the goal is to find a hidden treasure in the real world. Individually or in groups, students will have to find clues that give them information they have to interpret in order to find the final location of the treasure.
  - **Scout:** a nonlinear exploratory location-based game, where the students have to walk around to find a POI defined by the teacher. Each POI contains information and challenges.
  - **Rat Race:** a linear competitive location-based game, where two or more teams of students have to participate in a race and be the first to reach the finish line. From the starting point to the goal, the students have to solve challenges connected to different POIs.
  - **Conquest:** a non-linear competitive/ cooperative location-based game, where the students have to conquer different zones of the city by solving challenges. The first team to conquer every zone wins.

- **Jigsaw:** a linear competitive/cooperative location-based game, where the goal is to be the first team to arrive to a specific location by solving puzzles in different POIs.

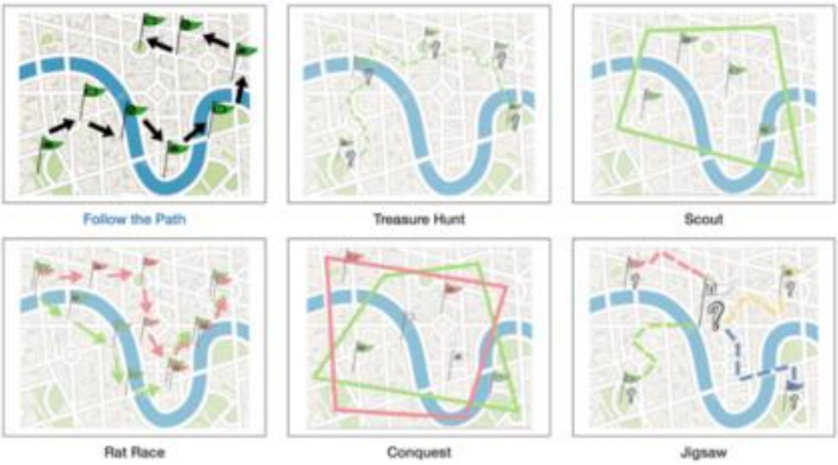


Figure 9. Type of location-based challenges

The teacher can decide if a context aware challenge fits the pedagogical goal of the lesson path setting the condition if the game is still playable, but depending on the scenario the progress may be locked as long as the challenged is not passed.

- *The Beacon Inventory.* As shown on the graph above, the Beacons Inventory API communicates directly with the Authoring Tool and will contain information about the beacons used to connect the system with the user localization. After authorization and authentication, the Beacon Inventory will allow users to manage the Beacon and the QR code list. After logging in to the system, the user will get a list of the available and already created locations and a list of the beacons assigned to each location. The user will also be able to create his own places and assign beacons to them. This module together with the system developed by GEOMOTION will allow to create mixed activities using GPS, Beacons, or QR codes.
- *Location Component (game engine extension).* This module will be responsible for receiving the user location and forward it to the Game Engine, and it will allow the activation of context based content on the BEACONING platform. This module will also be responsible for scanning nearby beacons and launching the QR code scanner, which will serve as an alternative for the beacon based solutions.

Location technologies available:

The technologies available to create location-based challenges are unified on the BEACONING platform. The available technologies are:

- **GPS:** should be used for tracking user outdoor location. The precision of this technology might vary depending on the weather. This is why we highly recommend using GPS when the user location does not need to be exactly precise.
- **BEACONS:** can be used both indoors and outdoors. When used outdoors, the system owner needs to pay attention to the device description and restrictions (e.g. temperature, high humidity in the area). Beacons are waterproof and can work in low temperature (been tested in - 15 C). However low temperature may affect the battery life.
- **QR Codes:** The type of surface they are printed on will determine whether they can be used indoors, outdoors or both. We recommend using QR codes only indoors because it is likely that they will be printed on paper.

2.4 THE GAME AUTHORING PIPELINE

The Game Authoring Pipeline incorporates the following functionalities:

- *The Scenario Editor*: an online software developed in the Haxe language and published for FlashPlayer on the web. The role of the Editor is to produce scenario files that will be read by the game engine.
- *The Game engine*: it is an application developed in the Haxe language and published for mobile applications with the Air runtime, and for FlashPlayer on the web. It reads content created by the authoring system that will in turn refer to Scenario Files that will in turn refer to graphic, text and audio assets. It will interact with the context–aware software components with an Adobe Native Extension on mobile, and with JavaScript remoting on internet browsers. It will interact with the mini-games by embedding them in an Air web view on mobile, and with JavaScript remoting on internet browsers.
- *The Server side*: the resources files are hosted and delivered to the game engine on demand. These can be audio files, graphic assets, texts from a database and all the content from the Scenario Editor and Authoring Tool. The game engine will write on the server a trace of all users' experiences that can later be read by any other software to produce analytics.

2.5 THE AUTHORING SYSTEM

The game authoring system and its services are part of the authoring pipeline. It provides means to create, edit and apply gamified lesson plans for learning designers and teachers, respectively. It requires anonymized information relative to students, teachers and courses. This data, together with the gamified lesson plan and configuration parameters for the activities and mini-games available allows for the procedural generation of configurations for the games that are suitable and adapted to the students' limitations and needs. Additionally, it provides, via a web API, the possibility for external tools to make use of these functionalities.

Component Architecture

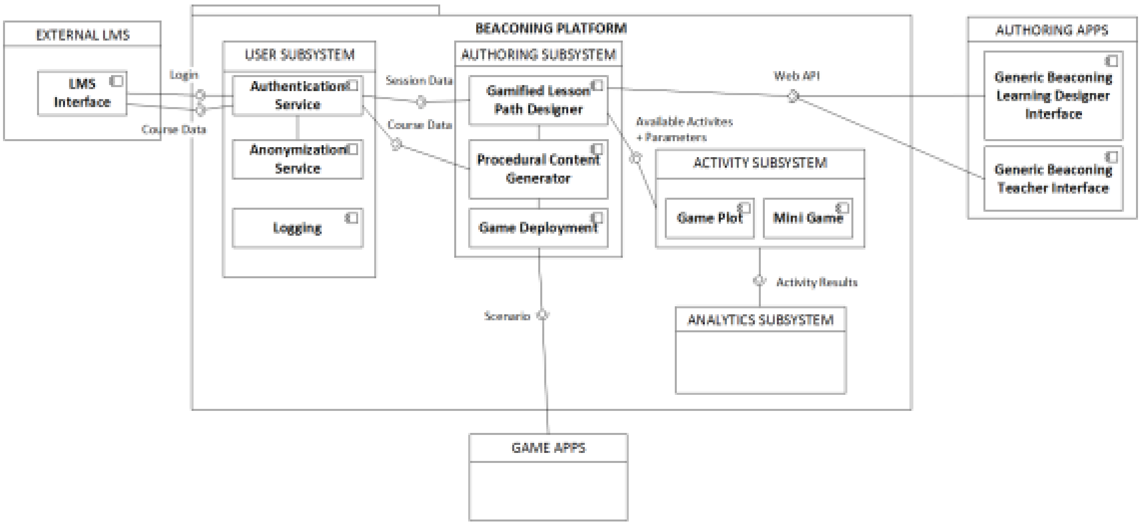
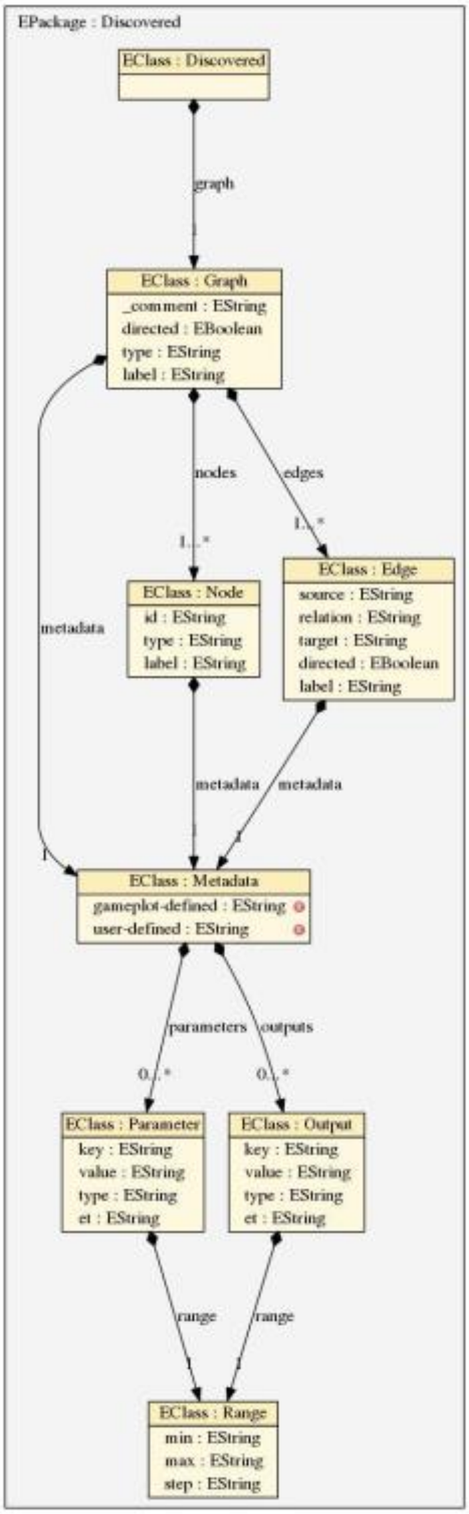


Figure 10. UML Component Diagram depicting the proposed Beaconing Authoring Subsystem, its components, and services provided and consumed

Game Plot Mapping Tool



The diagram (Figure 10) depicts a high-level representation of the Beaconing authoring subsystem and its projected integration with the remaining Beaconing platform subsystems and third party systems. Most notably, the authoring subsystem relies on the authentication and/or user subsystems (such as an LMS) to provide both permissions and data that can be used for the creation of procedurally generated beaconing content.

Third party applications can, via a web API, create or instantiate Lesson Paths through the Gamified Lesson Path Designer, through the inclusion of quests, missions and activities with varying learning goals and weights. These activities (such as games) provide a list of available parameters that can be adapted to fit each individual learner's needs. The personalized game configuration is then made available to a gaming application.

The Authoring System aims to provide a tool for bridging the game plot with the lesson plan's activities. It allows the Learning Designer or the Teacher to customize and parametrize activities, mapping them to a game plot.

The game plot and its activities follow a graph structure (see Fig. 11). The corresponding JSON schema is also available in Annex 2.

Each Node of the Graph represents either a game plot section/part or an activity. In this way, it is possible to decide what activities (and their parameters) follow which parts of the game plot. Being a Graph, it also allows for non-linear storytelling to occur. The tool, however, will ensure that the student will complete the gamified lesson plan independent of the paths the story take him/her.

Figure 11. Proposed Game Plot Mapping Tool (UML Class Diagram)

Scenarios

This section provides some scenarios where the learning designer, teacher or student interact with the Authoring Tool or with material it produced. The examples shown follow a chronological order. The creation of a Gamified Lesson Plan (GLP) by the Learning Designer, the instancing of a GLP by the teacher and, finally, the student following said GLP.

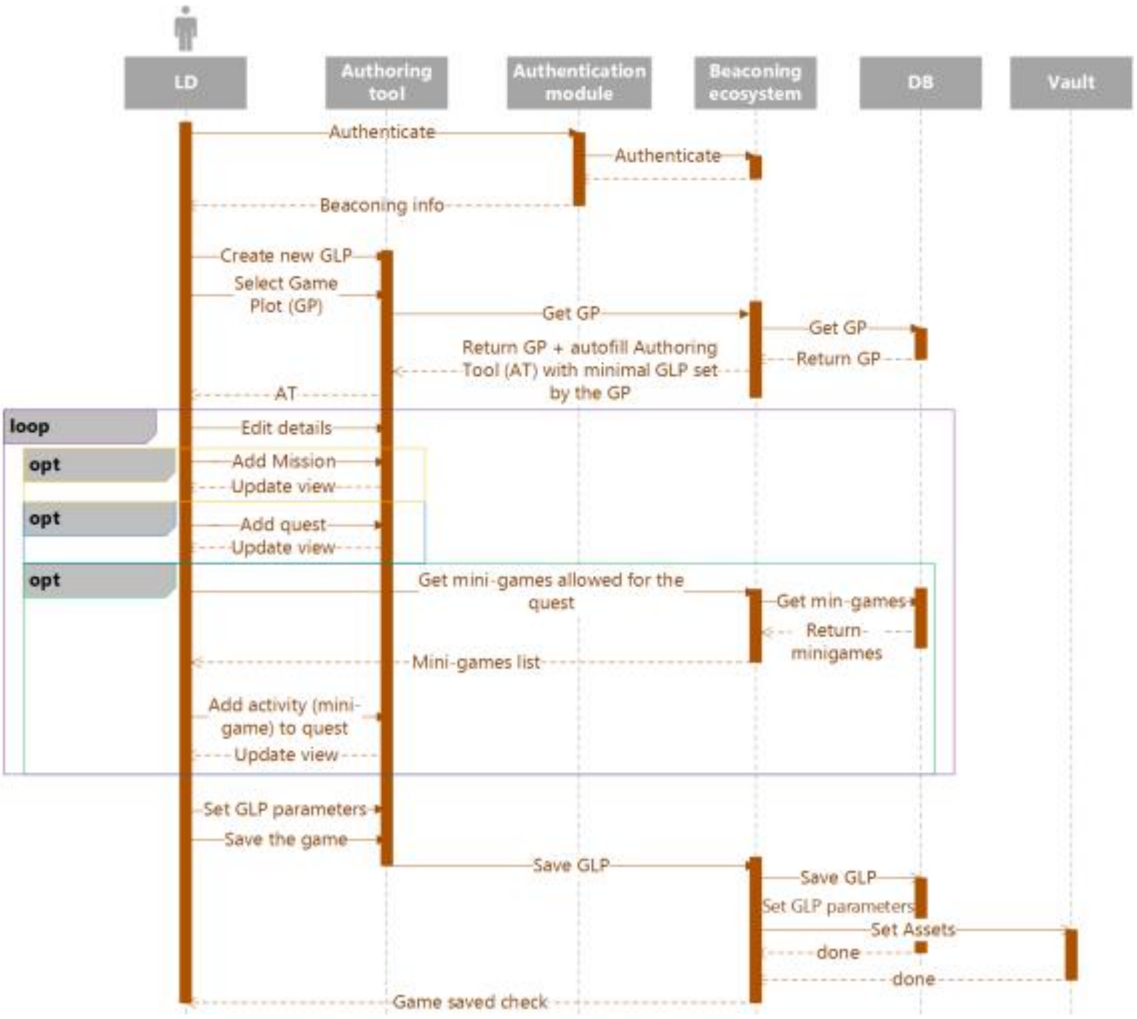


Figure 12. Learning Designer creates a new mapping between a Game Plot and Activities (Gamified Lesson Plan).

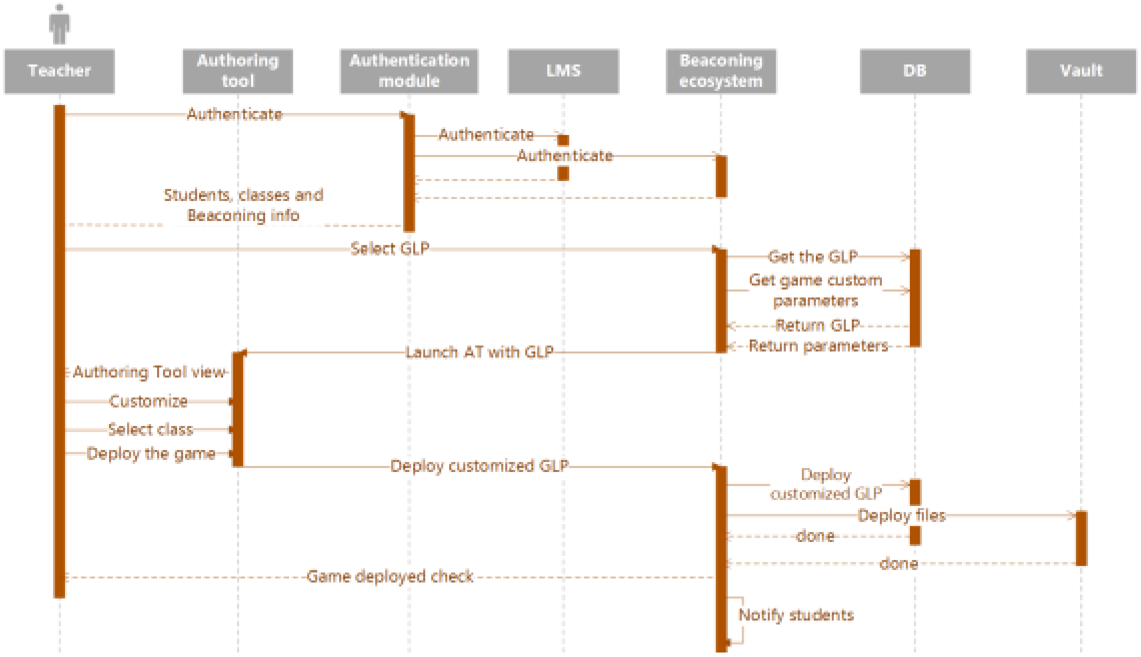


Figure 13. Teacher instantiates a Gamified Lesson Plan and deploys it.

In Figure 12, the Learning Designer, after a successful authentication, creates a new GLP, by choosing one of the available Game Plots (GP) and creating the supporting structure (Missions, Quests and Activities). The structure creation is restricted by the possible activities/mini games supported by the game plot and possible parameters. After finishing the creation, the GLP is saved.

The Teacher may, then, choose from a list of GLPs, which will be deployed to the students. After choosing the GLP, the Teacher proceeds to customizing the GLP (a process similar to the one followed by the Learning Designer) and finally deploys the game to the chosen students. These students are then notified they have pending activities.

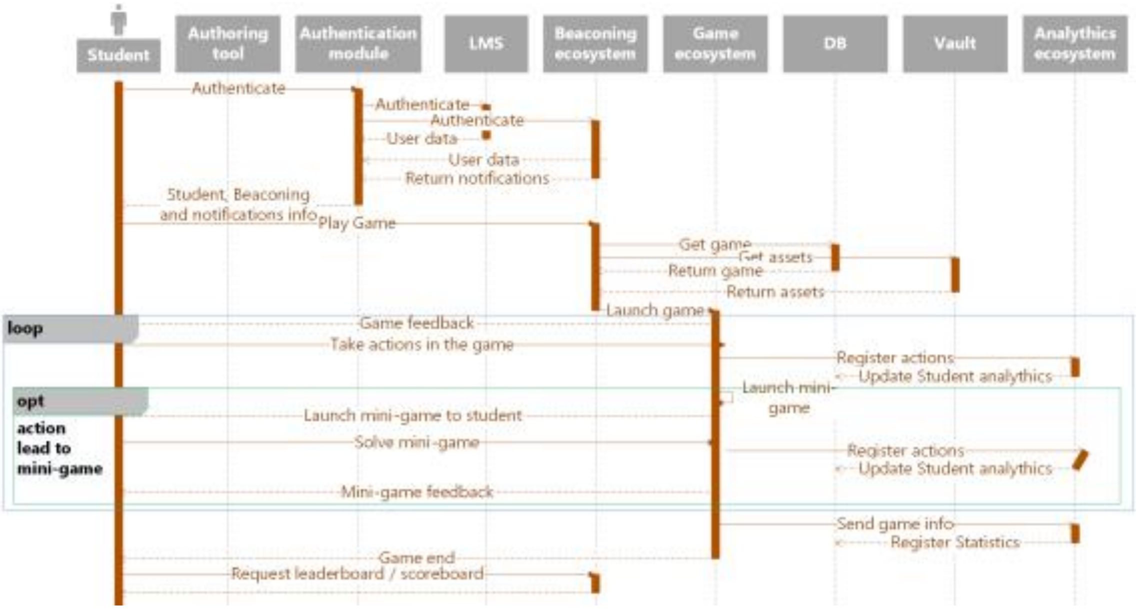


Figure 14. Student plays an assigned game (UML Sequence Diagram)

The Student, after login, is presented with a pending task. After choosing to engage in gaming, the system starts transferring assets for the game. Upon starting, and depending on the choices of the Learning Designer and Teacher, the session may be comprised of several mini-games or activities. All of the student's activity and action is logged to the analytics sub-system. When the student is finished some feedback is provided.

These are the main usage scenarios that are supported by the Authoring Tool.

Technological Stack

The following technologies were selected to be used for the development of this system. These were chosen due to their flexibility, stability, scalability, interoperability, ease of use, and available documentation: PHP (Laravel); HTML (Blade); CSS; Apache Web Server; MySQL/Postgres/SQLite/SQL Server (depending on the choice of Laravel database driver); Bootstrap; JQuery.

Some additional, specialized dependencies may be added to the stack over time, in addition to those listed above. Some technologies may be used to ensure syntactic interoperability (ex: JSON, XML) before semantic interoperability can be attained.

Table 4, 5 and 6 describes the core functionality of the Authoring System, as well as the services it produces and consumes at platform level.

Table 4. Core components of the Authoring System

Service	**Description**
Gamified Lesson Path Designer	Component responsible for creating, instancing and customizing gamified lesson paths.
Procedural Content Generator	This component parametrizes game plots or mini game configurations, suitable to the needs and limitations of each student
Game Deployment	Component that provides game scenarios to the game applications. These scenarios have been previously created by the gamified lesson path designer and correctly parametrized in the procedural content generator components

Table 5. Services provided by the Authoring System

Service	**Description**
Gamified Lesson Path Designer CRUD	Service responsible for creating, instancing and customizing gamified lesson paths.
Game Deployment Scenarios	This service provides the scenario to be loaded by a game application.

Table 6. Services consumed by the Authoring System

Component	**Service**	**Description**

Personal data store	User identification	Requires user identification and permissions
Personal data store	Course Data	Requires course data (if available).
Personal data store	Students Data	Anonymized student data is necessary for the procedural content generator to work
Game Plot Repository	Available mini-games and configuration	Lesson Path Designer needs to know what mini-games are available and how each mini-game can be configured
Game Plot Repository	Available Game Plots and Configuration	Lesson Path Designer needs to know how can each game plot can be configured

2.6 THE MINI GAME ENVIRONMENT

As stated in the deliverable D3.5 Game design document mini games (challenges) will be short game challenges to assess or train players in specific STEM topics.

Considering the limited time frame of the project, during the project lifetime a fixed number of mini games will be developed to address as many of STEM topics as possible. Afterwards, the Beaconing Ecosystem will be opened to 3rd parties' developers and stakeholders to let them contribute to it, providing new challenges and mini games.

Taken into account this architectural requirement, all the mini games developed inside the project scope will be designed and developed in order to be as reusable and configurable as possible to give learning designers and teachers the chance to reuse the same mini game in different learning scenarios by simply authoring its configuration during the setup of the game plot.

The other mentioned requirement of having an open ecosystem of mini games triggered the technical decision to split the overall game experience of the player in two different layers:

- Native 3D gamified environment
- Web mini games environment

The former will be a native App built for different desktop and mobile platforms, the latter will be a web environment included into a native web view component.

This architecture allows a seamless inclusion of new mini games into the Beaconing Ecosystem, without the need for the player to update the App to the latest version, and for the developers to rebuild and republish new build file.

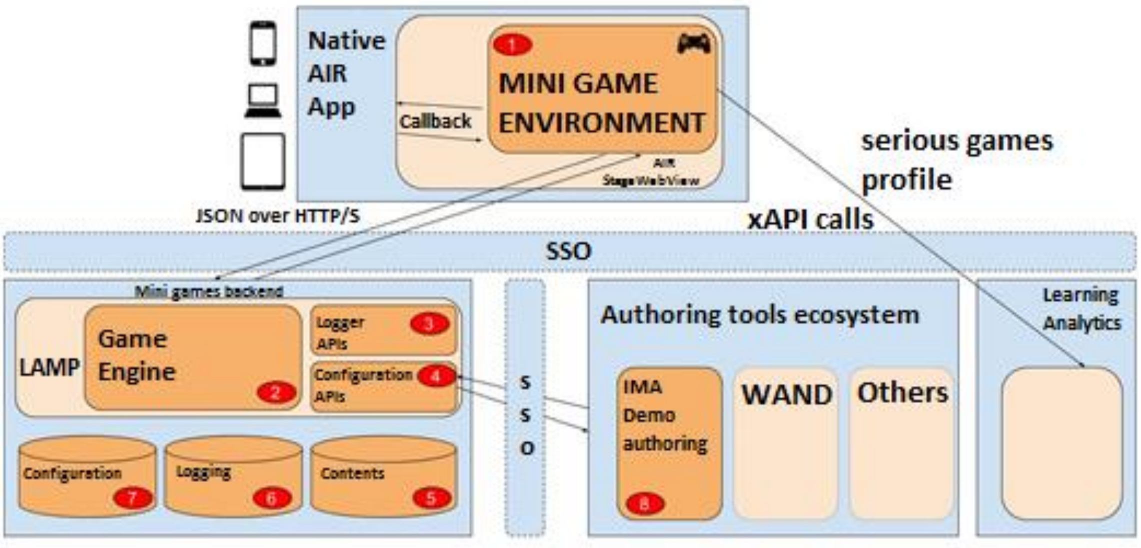


Figure 15. The mini game environment

Considering the scenario above, the two components of the mini game environment will be:

- HTML5 game client
- Game backend: Game Engine; Logger APIs; Configuration APIs; Configuration database; Logging database; Contents database.

In the diagram below the numbered orange elements are the mini games components represented inside a viewport of the whole Beaconing ecosystem.

2.6.1 Game client

The game client will be an HTML5 game environment running inside an Adobe Air web view component on mobile platforms, and directly instantiated using JavaScript remoting on web PC platforms. A local custom URL call-back paradigm will ensure workflow communications between the native container and the game content to allow the container to open and close the game content according to the overall game dynamics driven by the container.

The same custom URL call-back paradigm will be used by the game client to send xAPI analytics data to the native App that will enrich them with his own analytics data before sending the whole stream to the analytics platform.

More in detail, the container to launch the game will trigger the web view to call an URL providing one or more query string parameters needed to retrieve the specific configuration for each session.

When the mini game session will be ended, the HTML5 environment will invoke the web view through the above mentioned custom URL paradigm to allow the container to close the mini game environment according to the current game plot.

The game client will also be in charge to log all the relevant actions of the player and send them to the internal mini games backend logging system and to the learning analytics platform using the xAPI paradigm.

The tables below describes the Game client component and the services it requires.

Table 7. Game Client components

Service	Description
Game web client	Client web component responsible for providing the mini game play on mobile and desktop environments. On mobile this component will be an HTML5 web page running inside a native Adobe Air web view.

Table 8. Services Consumed by the Game Client

Component	Service	Description
Game Engine	load game configuration	Loading of the current mini game session configuration JSON file.
Native web view	custom URL communication protocol	Game analytics data passing and mini game close and open events handling.
Native web view	Query string URL to launch the mini game session	Query string URL to launch the mini game session

2.6.2 Game Engine

The game engine will be the owner of the configuration logic of the game, and it will be built upon a LAMP stack. This choice, versus a more complex and resources consuming architecture such as J2EE, reduces deployment and TCO costs, keeping the potential integration with other components smoother and easier to maintain. The Game Engine will communicate with the game client using a JSON notation over an HTTP/S protocol. The Game Engine component will provide two sets of REST APIs: one to expose all the READ and WRITE game configuration functions to the authoring environment, and one to let the mini game client to log all the actions inside the internal logger database. All these APIs functions will be federated inside the Beaconing SSO system. Another layer of the game engine is the “configurations database”: the main function of this component is to host the configuration files for each mini game session. This component will expose also a set of APIs to let the authoring tool author mini games configurations through a set of REST calls.

In the tables below a more detailed description of the game engine component is provided with all the key data flows involved.

Table 9. Core components of the Game Engine

Service	Description
Game engine	Component responsible to host the JSON configuration files for each mini game session and provide them to the game client.

	This component will also provide a set of APIs to let the JSON configurations be authored by the authoring tool.

Table 10. Services provided by the Game Engine

Service	Description
Create or update game session configuration	Exposes an API to the Authoring tool to save or update a configuration for a specific game session.
Get game session configuration	Exposes an API to let the game client retrieve the configuration for a specific game session.
Internal logger	Exposes an API to the game client to log activities in the game engine database.

Table 11. Services consumed by the Game Engine

Component	Service	Description
Core Services	Mini Game repository service	Create or update a new mini game towards the Beaconing Ecosystem

Below the steps involved in the mini games configuration workflow are described.

1. Each developer of mini games will use the mini games repository core service in order to register or update a new mini game in the Beaconing Ecosystem.

   Since each developer company that will contribute in developing mini games for the Beaconing Ecosystem will host his game backend (Game Engine), when a new mini game is registered, some information such as the runtime URL and the JSON configuration bundle file will have to be provided to the mini game repository service. The configuration bundle for each mini game is supposed to be a couple of files: one JSON schema descriptor file and one lookup metafile with all the resources needed by the specific mini game. With this information in the mini game repository, the Authoring Tool will be able to call the mini game repository service to retrieve the list of available mini games with all the related information.

2. The Authoring Tool, after the teacher has made the selection of the desired mini game, will use the mini game specific configuration bundle to generate the authoring form.
3. The Authoring Tool will create and save an INSTANCE of the configuration for that specific mini game session.

In the diagram below a picture with the above mentioned data flows:

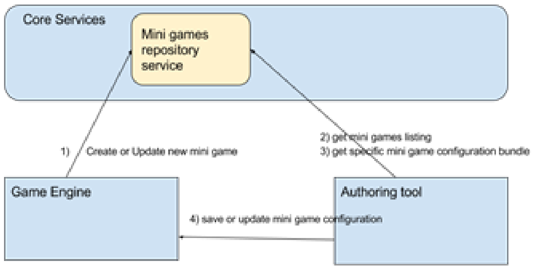

Core Services
Mini games repository service
1) Create or Update new mini game
2) get mini games listing
3) get specific mini game configuration bundle
Game Engine
Authoring tool
4) save or update mini game configuration

Figure 16. The mini games configuration workflow

2.7 USER INTERFACES

There are a number of differing interfaces that are to be incorporated into the Beaconing Platform. These include the visual UI experience both for the student and teacher screens. These will be programmed and integrated using a combination of HTML + CSS styling. Accessibility interfaces will be slightly different. Accessabar is an accessibility interface, which holds a number of assistive tools for the user. These include screen tinting, font changes, magnification, reading bars, text to speech facility and so on. It will primarily be built using HTML for structuring and CSS for styling. A Webpack will be used to create an ES6 JavaScript bundle file to import into the Beaconing site/s itself.

Part of the Accessabar includes a speech recognition feature. This Speech recognition is based on web speech API with the functionality built upon HTML and JavaScript.

The MapMyProcess accessibility tool incorporates its own interface again which follows similar protocols. HTML is used for structuring and CSS for styling. The coding language used for functionality is 'Go', in order to create the server side application. Again we will use a Webpack to create an ES6 JavaScript bundle file to import into the Beaconing site. The Mind Genie Mind Mapping application, which is under development will either be a separate entity with its own interface or incorporated into the authoring tool itself. Following similar programming it will be HTML for structuring and CSS for styling with the use of 'Go' programming language and Web Sockets.

A freeware library of assistive software is to be included in the directory and help files. Links to download sites will be referenced or if licences allow the .exe files can be hosted for download on to the Beaconing platform.

Comprehensive Tutorials for the use of Accessibility features and interfaces are to be integrated into Beaconing. These will be sourced in the relevant settings. They will be created using iSpring E-Learning authoring tool and can be exported and uploaded in a variety of formats. They can create a SCORM Package that can be uploaded to the relevant LMS or Beaconing LMS, Flash player, HTML5 or .exe files.

2.8 LEARNING SEMANTIC SYSTEM AND LEARNING ANALYTICS

Learning semantics is accounted for via two mechanisms:

1. The use of the xAPI Serious Game recipe (a joint development of e-UCM and xAPI's ADL) enforces the use of a standardized vocabulary to describe game interactions[1]. This

[1] Ángel Serrano-Laguna, Iván Martínez-Ortiz, Jason Haag, Damon Regan, Andy Johnson, Baltasar Fernández-Manjón, Applying standards to systematize learning analytics in serious games, Computer Standards & Interfaces, Volume 50, February 2017, Pages 116-123, ISSN 0920-5489, http://dx.doi.org/10.1016/j.csi.2016.09.014.

forces any game-traces to conform to a particular well-described model. The initial recipe need to be extended for Beaconing as it did not considered geolocalization information.

2. The semantic understanding that can be authored into the Learning Analytics infrastructure to adapt both analyses and visualizations to particular games and learning experiences.

Since the scope of games considered in Beaconing is very broad, and potentially encompasses the whole STEM curriculum, we have not attempted to adopt any overarching semantic ontology to cover all possible game subjects. Even if such an overarching ontology could be found, acceptance, coverage, and applicability of this ontology for all potential game domains would be infeasible, and outside the scope of Beaconing.

2.8.1 Learning Analytics: Inputs and Outputs

Inputs:

- During system setup: Setup of delegated authorization & authentication using Beaconing SSO. This requires per-deployment editing of server-side JavaScript files; all other steps can be performed via REST API calls.
- During game & analysis configuration: Real-time alerts & web call-back setup; Analysis: how should data be processed (defaults available); Display: What will be shown to whom (defaults available).
- During analysis itself: xAPI statements, using the Serious Games application profile (aka recipe), sent from clients. The format is well-documented at the ADL website[2].

Outputs:

- In-game: Games can query current analytics state. In this case, the requesting game is responsible for adequately presenting the resulting data (JSON)
- Dashboard: Authorized users can request to see dashboards. Underneath, this results in multiple queries to the analytics back-end, performed transparently. With the returned data, semi-real-time dashboards are displayed.
- Alerts: When web call-backs have been configured, an alert will be sent to the configured call-back should the conditions to the trigger be met. All active alerts are visible in the corresponding dashboards for authorized users.

Learning analytics relies on REST JSON requests to provide all services, and expects REST API calls for all incoming requests. Initial configuration (and, specifically, SSO setup) is the only part that requires editing of (JavaScript) server-side files:

- Documentation for backend APIs is available online at http://e-ucm.github.io/rage-analytics-backend/
- Documentation for the A2 SSO setup is available online at https://github.com/e-ucm/rage-analytics/wiki/Login-Plugins

2.8.2 Analytics internal structure

The internal setup of analytics is as follows:

[2] http://xapi.e-ucm.es/vocab/seriousgames - vocabulary for serious game xAPI traces expected by the Analytics components

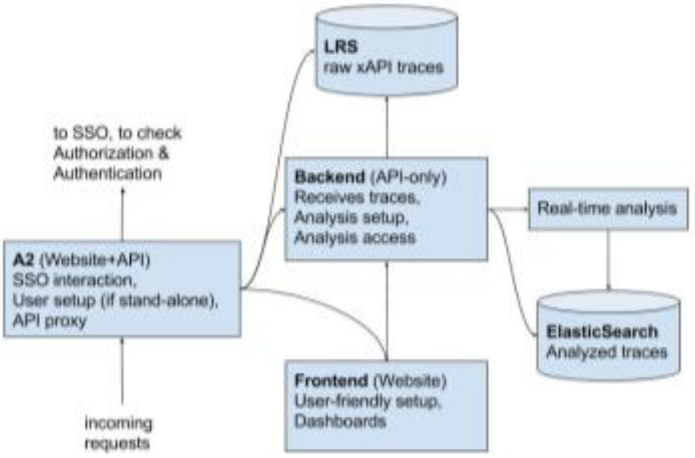


Figure 17. Internal analytics structure

Note that all requests are mediated by the A2 API proxy. This gives us a single point of contact that validates all requests against the SSO, and delegates these requests to the corresponding analytics module. A2 is not analytics-specific – the analytics backend is, in this sense, the critical component, providing full API access to the range of analytics actions, from configuration and trace-collection to alert setup and querying. The frontend provides a user-friendly website which provides access to some of the backend's functionality.

Data can also be queried directly from the LRS (which stores and accesses the raw xAPI traces received by analytics) and ElasticSearch (which stores the processed traces, using the per-game configured real-time analysis).

2.8.3 Analytics requirements and deployment

Analytics can be deployed as a set of linked Docker containers, and its only software dependency is docker and docker-compose itself. An administrator-friendly launch script assumes that bash and other linux utilities are available.

Recommended software: Ubuntu >= 14.04; Docker >= 1.13 and Docker-compose >= 1.7.1

Recommended hardware: >= 12 Gb free HDD space after installation; >= 4 Gb RAM; >= 2 CPU cores.

3 CONCLUSION

This document describes the architecture of the Beaconing platform, the components that are being integrated, and the dependencies and communication between the components. The proposed architecture is constructed using principles of openness and extensibility that allow simple integration of new components and seamless upgrade of existing ones.

3.1 RESULTS

The work presented herein represents a reference point for the design of pervasive learning platforms and ecosystems that reflects the challenges of the integration process, as well as of large scale piloting.

The deliverable describes detailed architecture of the Beaconing core components and six integration scenarios that will be implemented within the platform.

3.2 IMPACT

The activities carried out in defining the architecture provides a working framework for the development activities in WP4 and the testing in WP5. The work provides insights into the infrastructure and operations required to maintain the system, as well as future cases for exploitation and reuse.

A revised version of the architecture will be produced after the development effort has concluded, and will address specific implementation issues and improvements that arise during pilot testing.