BREAKING EDUCATIONAL BARRIERS WITH CONTEXTUALISED PERVASIVE AND GAMEFUL LEARNING

Grant Agreement No. 687676

Innovation Action

ICT-20-2015

D4.7 – Game analytics and adaptation components reference implementation

Due date	Month 24
Actual date	Month 28
Deliverable author(s)	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado, Baltasar Fernández Manjón
Partner(s)	UCM
Version	1.0
Status	Final
Dissemination level	CO

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 European Union – Grant Agreement No. 687676

Version control				
Version	**Date**	**Author**	**Institution**	**Change and where applicable reason for change**
0.1	2017.09.20	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado, Baltasar Fernández Manjón	UCM	First draft
0.3	2017.10.18	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado, Baltasar Fernández Manjón	UCM	Appendix with First Aid Game LAM example added
0.4	2017.11.22	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado, Baltasar Fernández Manjón	UCM	Updated version after internal review. It includes new meta-LAM structure proposal description.
0.5	2017.12.19	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado, Baltasar Fernández Manjón	UCM	Updated version after meetings with partners (mainly meeting on Monday 18th December).
0.6	2018.04.27	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado, Baltasar Fernández Manjón, Massimiliano Cazzaniga	UCM, Imaginary	Updated version including new appendix with BEACONING Analytics requirements. Feedback from Massimiliano.
0.7	2018.05.25	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado, Baltasar Fernández Manjón	UCM	New version after internal reviews.
0.8	2018.06.08	Manuel Freire, Iván Martínez Ortiz, Cristina Alonso Fernández, Iván Pérez Colado,	UCM	Added more information regarding adaptive components (alerts and warnings) and student dashboard API.

		Baltasar Fernández Manjón		

Quality control				
QA Version	**Date**	**QA Responsible**	**Institution**	**Change and where applicable reason for change**
0.2	5/10/2017	Jayne Beaufoy	COVUNI	Language Check
0.3	7/11/2017	Mustafa Ali Turker	SEBIT	Internal Review
0.3	9/11/2017	Ioana Andreea Stefan	ATS	Review. Changes or proposal in comments/revision notes
0.6	15/05/2018	Mustafa Ali Turker	SEBIT	Internal Review
0.6	17/05/2018	Francois Mohier	ORT	Internal Review
0.7	08/06/2018	Sylvester Arnab	COVUNI	Internal Review
0.8	12/06/2018	Francois Mohier	ORT	New review, changes accepted
0.8	12/06/2018	Sylvester Arnab	COVUNI	New review, required changes accepted
0.8	13/06/2018	Mustafa Ali Turker	SEBIT	New review, required changes accepted

Release approval				
Version	**Date**	**Name**	**Institution**	**Role**
0.8	15/06/2018	Jannicke Baalsrud Hauge	BIBA	QM
1.0	15/06/2018	Jannicke Baalsrud Hauge	BIBA	QM

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVATIONS

ABR	DESCRIPTION
AT	Activity Tree
ADL	Advanced Distributed Learning
D	Deliverable
xAPI	Experience API
GLA	Games Learning Analytics
GLP	Gamified Lesson Plan
JSON	JavaScript Object Notation
LA	Learning Analytics
LAM	Learning Analytics Model
meta-LAM	meta-Learning Analytics Model
POI	Point Of Interest
SN	Sequencing and Navigation
SG	Serious Game
UI	User Interface
UMS	User Management System

EXECUTIVE SUMMARY

This document describes a reference implementation of the analytics component as an open source code package. It includes a technical implementation of the required backend services[1]. Analytics is a key part of the BEACONING (Breaking Educational Barriers with Contextualised, Pervasive and Gameful Learning) EU H2020 Project.

BEACONING Analytics reuses, improves and extends analytics modules developed by UCM for the EU H2020 RAGE Project to simplify the creation of better serious games.

The present document describes how this platform currently works including the required Learning Analytics Models (LAMs) that allow data tracking for meaningful analysis and visualizations. LAMs allow Analytics to provide meaningful insights into student actions, helping the different stakeholders to understand how learning is taking place. While LAMs can be provided for each game or mini-game in the BEACONING platform to provide game-specific analysis, they can also be shared among minigames with similar analysis requirements. Without game-specific LAMs, games behave as “black-boxes”, and no meaningful analytics can be extracted beyond generic metrics and visualizations.

Significant changes are still required to bring the existing analytics platform in line with BEACONING’s integrated platform vision. Therefore, together with the reference implementation of analytics, this deliverable describes requirements that must be fulfilled by all actors in the process (learning designers, teachers, game designers, analytics software developers) to achieve a coherent and complete integration. To this end, the deliverable includes detailed examples of game-specific LAMs and their corresponding visualizations, in an effort to document and simplify the development of future BEACONING Learning Analytic Models. Moreover, proposals for possible meta-LAM structures and the overall data analytics workflow are suggested.

The approach presented has been contrasted with the scientific community in (Pérez-Colado, Alonso-Fernández, Freire-Moran, Martinez-Ortiz, & Fernández-Manjón, 2018), (Pérez-Colado, Rotaru, Freire-Moran, Martínez-Ortiz, & Fernández-Manjón, 2018) and (Calvo-Morata, Alonso-Fernández, Freire-Moran, Martinez-Ortiz, & Fernández-Manjón, 2018).

[1] Backend services described in this deliverable and their latest documentation can be found as open-source packages at https://github.com/e-ucm/rage-analytics

1. INTRODUCTION

1.1. BACKGROUND

The Analytics System collects, analyses and visualizes data obtained from games and minigames in the BEACONING platform, with purposes such as revealing student progress and how they actually learn while playing a game; and provides this information to teachers and other stakeholders.

Data gathering and visualization are necessary, but not sufficient ingredients towards actionable analytics. Choosing what to display when, and how, is critical. However, it is also highly game-dependent: there is no general game-universal set of data that, when displayed in a certain manner, is guaranteed to produce actionable results. The goal is to generate visualizations that can be interpreted by teacher/trainer providing control over the deployment of the games in the classroom and insight into student learning, and such insights depend heavily on how the game is designed to teach.

The previous deliverable, *D4.6 Game analytics and adaptation component design***,** described the standards and specifications to be used and detailed the process of data capture and data exploitation, taking into account several issues such as data privacy, anonymization, security and accessibility issues.

This report reflects the work done in the project based on the feedback received and on collaboration with BEACONING partners. With the initial information obtained from the early version of authoring system during the research stay of INESC project members at UCM, and especially with the experience obtained when preparing the demo for the latest review, which featured integrating geolocalized-games and minigames and unifying that information into a single integrated analytics.

1.2. ROLE OF THIS DELIVERABLE IN THE PROJECT

This deliverable describes a reference implementation of the client component, which will be provided as an open source code package. It includes a technical implementation of the required backend services, delivered as an open source code package[2].

Additionally, this deliverable describes how to integrate analytics in a game, provides a full analytics manual for developers and teachers (including API usage) and provides a complete example collection model for a particular LAM and its visualization configuration.

1.3. APPROACH

BEACONING Analytics reuses, improves and extends the analytics infrastructure developed for the RAGE EU H2020 Project[3], which in turn extends the reference platform developed in the GALA EU FP7 Project[4]. BEACONING Analytics is also built on experience from Lea's Box[5] (EU FP7 Project; includes BEACONING participants) and the LACE[6] (EU FP7 Project). While in RAGE some assets related to analytics were created to facilitate the development of applied games, BEACONING's goal is to deliver an integrated platform that integrates game authoring,

[2] Open-source backend packages & documentation: https://github.com/e-ucm/rage-analytics
[3] RAGE Project: http://rageproject.eu/project
[4] GALA Project: http://cordis.europa.eu/project/rcn/96789_en.html
[5] Lea's Box Project: http://cordis.europa.eu/project/rcn/189134_en.html
[6] LACE Project: http://www.laceproject.eu

context-aware techniques, procedural game-content generation, adaptation, learning design and analytics.

To adequately integrate BEACONING Analytics in the complete BEACONING platform, several steps must be taken both at a high-level platform (gameplot) level and at the lower (games and minigames) level by each of the partners involved in plot and content development. For the latter, a complete general Learning Analytics Model has been proposed to describe the prerequisites game and minigames developers must complete to send meaningful data to analytics services. These preparation steps, built on top of the analytics model defined by the developers ensure that: 1) the data collected for the analytics system contains meaningful information about learning and that 2) the BEACONING Analytics System can extract and analyse learning information from the collected data.

As detailed in previous deliverables (*D4.6 Game analytics and adaptation component design*) a complete interaction model was defined and implemented using the Experience API (xAPI) data standard, in collaboration with the ADL Initiative, developers of xAPI as part of an open community. Also, a complete set of visualizations was presented to cover developers and teachers' main interests.

1.4. STRUCTURE OF THE DOCUMENT

Section 2 describes the Learning Analytics Model which includes the requirements that game developers must fulfil prior to sending any data to the Analytics Sever to ensure its meaningfulness, and provides a proposal for the meta-LAM and for the data analytics workflow;

Section 3 describes the process that games should follow to send information to the Analytics Server, and its expected format following the xAPI-SG Profile;

Section 4 describes the analyses and visualizations that are currently available to be automatically performed with the standardized data received from the game as well as details for game-dependent analyses and visualizations.

Section 5 describes how to incorporate analytics inside a game.

Appendix A provides a detailed LAM example for the *First Aid Game*.

Appendix B provides a detailed step by step guide for developers and teachers to use the analytics interface.

Appendix C provides an example collection model whose visualization configuration is detailed in Appendix D.

Appendix E described the steps taken towards the integration of analytics for the demo of month 18.

Appendix F details the final version of the requirements stablished for partners, as previously distributed and agreed in a document, including the definition of the student dashboard and the API provided.

2. LEARNING ANALYTICS MODELS AND CURRENT REQUIREMENTS

2.1. LEARNING ANALYTICS MODEL

The Learning Analytics Model (LAM) is responsible for describing which data is to be collected for a specific educational videogame (also called Serious Game, or SG); and how it should be processed, reported, and interpreted (for example, in terms of success or failure).

The LAM is an integral part of building usable dashboards for the different stakeholders, and is a key component of Learning Analytics for SGs, also termed Game Learning Analytics, or GLA (Freire et al., 2016). Figure 1 illustrates the role of a LAM within GLA.

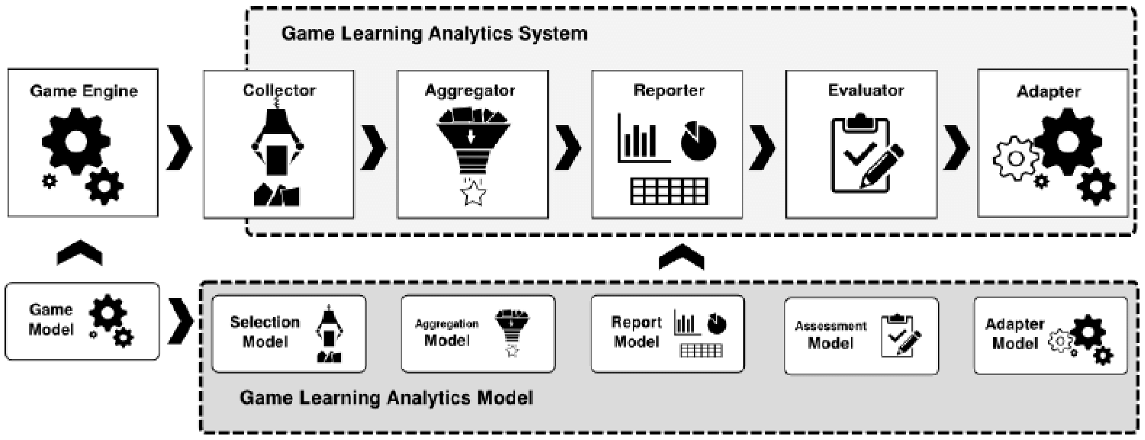


Figure 1. General architecture of a Game Learning Analytics platform. Note the Learning Analytics Model (lower left). Image from (Baalsrud Hauge et al., 2015)

The system, developed as part of the H2020 RAGE Project, is being reused, improved and extended as part of the H2020 BEACONING Project. Therefore, the system will be updated as work continues. For the latest version, please refer to the official open-source entry-point repository in GitHub: https://github.com/e-ucm/rage-analytics.

To adequately identify the features that have been developed for BEACONING, and because the analytics software is open source, we are considering the use of *sponsoring features*, as it is common in open source projects. This will be reflected in repositories that have more than one sponsor by changing its documentation (README.md file) to identify features and their sponsors. Repositories that specifically belong to one project will not need to be changed. With this approach, rage-analytics which had the RAGE H2020 project as main sponsor until BEACONING, will eventually become eUCM-analytics.

Complete information on the system deployment process, status, use and updates can be found in the official GitHub Wiki page: https://github.com/e-ucm/rage-analytics/wiki. Please refer to this wiki page for the last up-to-date documentation of the system.

Figure 2 shows an overview of the Learning Analytics Model: while a student plays a game or minigame, the embedded tracker will send the corresponding traces (as xAPI statements) to the analytics system for analysis and result visualization. Game-independent analyses and visualizations are provided by default for teachers and game developers. If configured, adaptive components like game-dependent analysis and visualizations can be deployed, as well as warnings and alerts for specific situations (using the students' analytics API).

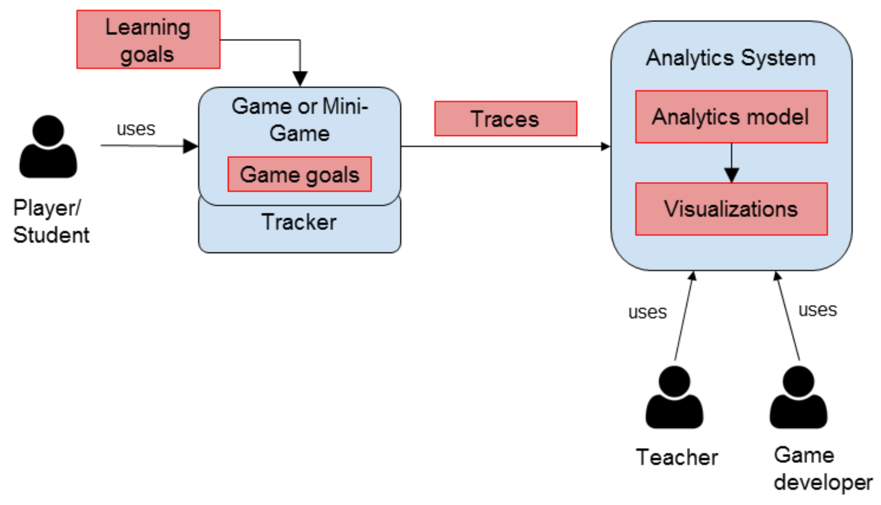


Figure 2. Learning Analytics in BEACONING. The Learning Analytics Model describes what statements are being sent by the tracker; and the analyses and visualizations that will be performed and later used to display results.

With this approach, game developers may communicate in-game information to other stakeholders (for instance, teachers). While the game could e-mail or instant-message results to stakeholders, it is considerably easier to report the information to the analytics system which may perform any required post-processing and provide rendered dashboards or data to the broader platform. This results in visualizations with added value, as the information will be presented in a clearer way after filtering and analysis, simplifying information extraction and making it both more effective and efficient.

Notice that when reporting to students, games can present current data directly to users, without analytics support; or rely instead on the Analytics API as described in Section 5. However, presenting more complex data within the game, such as comparing a single user's performance with the class, or displaying historical (from previous game-play sessions) or GLP-level data (from other games in the same GLP) can only be reasonably implemented by going through the Analytics API. Avoiding analytics, would, for these scenarios, require a much more complex game and game server to provide these services; and would in fact duplicate basic analytics functionality.

For the process depicted in Figure 2 to occur smoothly, every step needs to follow some previously defined rules in terms of format and types. Even then, best results require game-specific analysis and visualizations: while BEACONING Analytics uses defaults to simplify initial deployment and provide example analytics and visualizations to build on, these are generic in nature, and custom-built visualizations for each game can provide much better insights.

Several steps need to be completed for any given game or minigame to actually report meaningful learning information. These steps are as follows:

1. **Define learning goals:** The first step, prior to any other development, is to define a set of learning goals (such as specific knowledge, procedures or tasks to be mastered). These learning goals are to be achieved in the game and result in a specific learning design. Notice that these learning goals may have any meaning including competences, skills, and so on. We expect skills/competences/knowledge/scores to be defined by partners involved in GLP creation, including the original creators and the teachers that edit or customize this GLP.

2. **Define game goals:** the previously defined learning goals should be reflected in game goals (e.g. tasks, levels) resulting in the game design. The easier the learning goals are identified in game, the easier the tracking process is made.
3. **Define traces to be sent:** once the learning and game design are clear, the game designer, with help from the learning designer, can define the information to be traced and sent by the game to the analytics system for analysis and visualization. The traces sent by the game should contain learning information and be formatted according to the xAPI-SG Model. Traces can be sent in any moment in the game, but, for maintainability purposes, we recommend doing so in few, well-defined spots. For example, a learning goal that is satisfied when a quest is completed is easier to track than another learning goal that is satisfied when a specific answer is selected while deep in an in-game conversation.
4. **Define the analysis model:** it is necessary to have a clear analysis model that can extract the meaning (as related to the learning goals) from the xAPI traces sent to determine if the game is actually meeting its expected goals or not. This model about how the traces are to be analysed and interpreted should be provided by learning designers in cooperation with game designers. Based on this model the analytics developers will implement the code for the actual analysis of the tracked data. Notice that if all games in a platform have common analytics needs, a platform-specific "default analysis" can be developed. However, this is so far not the case in BEACONING, as there is no agreed-upon, well-defined set of data that all games support. Coupling between learning design and learning analytics is required, since only learning designers know what learning means in those particular games. Even if working with a limited set of STEM competency goals (this is not universally described or accepted in the literature), these goals could be approached in many different ways in the games and the LA platform would be unable to infer or analyse such data without reference to a suitable LAM or meta-LAM (see more details on Appendix F.7. GLP Definition).
5. **Define the visualizations:** together with the analysis model, visualizations that adequately represent the results of analyses should be designed to be as informative as possible to their stakeholders. Note that the best visualization for a given analysis depends entirely on the type of information that such an analysis generates. To generate the most informative visualizations, we recommend the following steps (full examples of custom visualizations can be found in Appendices B and C):
    a. See all the available visualization types shown here: https://www.elastic.co/guide/en/kibana/current/createvis.html
    b. If none of the visualization types fits the analysis needs, implementing a new type extending the visualization engine (Kibana) is possible. For this process, a detailed specification is required including:
        i. Requirements specification: long text that describes what the visualization should do; how it should react to multiple sessions; how is the data going to be displayed and, if needed, clustered; and how the visualization will change on time.
        ii. Detailed mock-ups: how the visualization is supposed to look, including pictures, drafts, mock-ups, and multiple examples on how the visualization should be presented in different situations.
    c. An analytics developer will code the visualizations for the integrated dashboard and provide the required information for in-game presentation of the data via an API (i.e. students' analytics).
6. Personalized alerts and warnings may also be defined: alerts and warnings (adaptive components) are triggered when certain events occur. This can be added to the dashboard by specifying under what conditions they shall trigger. For example, if a

learner fails in a test, or answers more than half of the questions incorrectly, alerts may be triggered to help teachers to quickly act and help that learner. Use of triggers should also be fully documented as part of game LAMs.

For these activities to be completed, their expected results and the stakeholders that should define them are depicted in Figure 3.

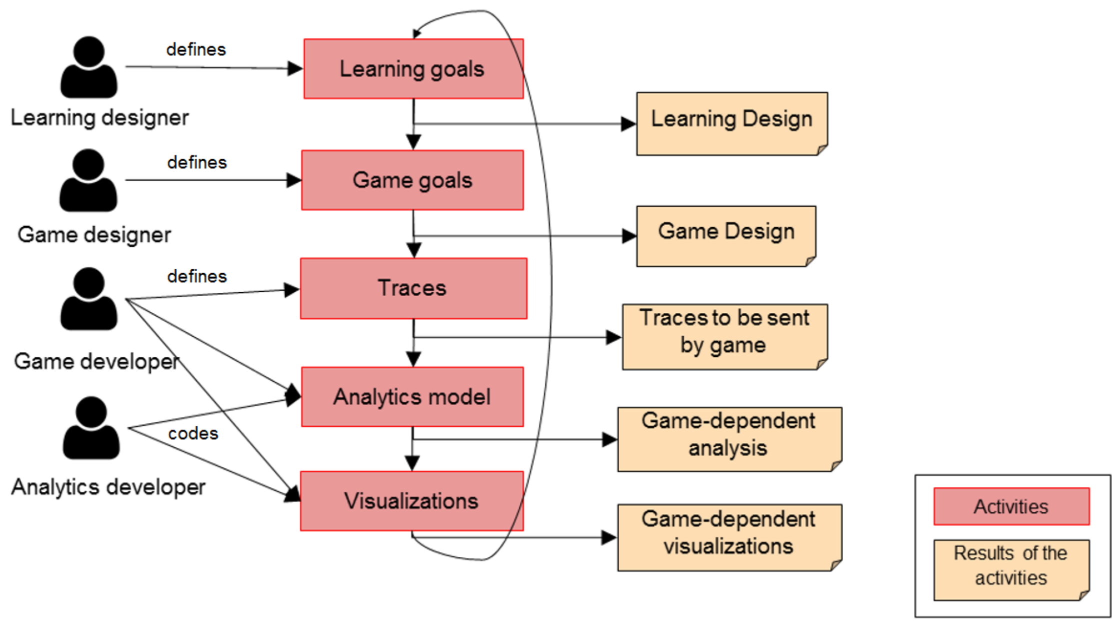


Figure 3. Activities to be carried out by different partners in BEACONING to define learning goals, game goals, traces, analytics model and visualizations.

Section 4 describes default analyses and visualizations for teachers and developers; and how these can be modified and extended to better accommodate particular games. For students, it will make more sense to provide dashboards embedded in the game platform, without breaking the flow of interaction. Games can request these dashboards, or their underlying data, through well-described APIs provided by the LA System. For details on analytics integration in games and hosting platforms, refer to Section 5, "Incorporating analytics to a game".

2.2. INTEGRATION REQUIREMENTS

BEACONING Analytics is part of a larger whole. This section describes the requirements that must be fulfilled to achieve tight integration into the BEACONING Platform, and that are responsibility of other BEACONING partners, as they involve educational, games or minigame issues that are beyond the scope of the Analytics System itself.

To achieve full integration, BEACONING Analytics requires full specifications for the following:

- **Meta-LAM:** detailing how analytics for a single game or minigame should be aggregated for GLPs built of multiple games and minigames.
- **User's management system:** the identity of students is critical to correct analysis of their actions. To avoid duplicating user management, BEACONING Analytics requires access to a robust and well-documented BEACONING User management system which can store user sessions across applications.
- **Data analytics workflow:** while BEACONING Analytics can be queried programmatically to provide in-game adaptation (see Section 5, "Incorporating

analytics to a game”), specifics of when and how adaptation information will be used is necessary to provide full support.

The following subsections describe each requirement in greater detail. Note that, as soon as each requirement is fulfilled, e-UCM will begin working on making any changes to BEACONING Analytics that may be needed.

2.2.1. **Meta-LAM**

The previously described LAM focuses on a single game or minigame. When several of these are connected or embedded in a general platform (e.g. gameplot), a general meta-LAM must be defined. This is the case of the BEACONING gameplot. Therefore, a complete meta-LAM for the BEACONING platform (at GLP level) must be defined and specified by BEACONING partners, as it is directly linked and determined by the BEACONING platform architecture. For this meta-LAM, each activity (e.g. mini-game) will have its corresponding LAM previously defined.

Notice that building the meta-LAM (how results from sub-activities contribute to their super-activity) is a matter of pedagogical design. As Section 7.1.1, “Assessment and Evaluation”, in D4.6 states: “[...] authoring of evaluation models should be done within the authoring tool where lessons paths are created and customized [...]”, therefore, the meta-LAM for a GLP is authored as part of the GLP itself. In this sense, analytics is the mechanism that performs accounting of the pedagogical policy (LAM); but it cannot enforce the policy itself, as it can neither prevent students from accessing activities, nor force them to repeat activities. Analytics is not sequencing, and cannot control what nodes are visited when. Therefore, even though analytics may detect that certain nodes have been "failed" and cannot be completed successfully (for example, due to excess attempts; or because a sub-node needs to be repeated), analytics cannot force nodes to be avoided or repeated. The only choice for analytics is whether to keep or discard incoming data.

The meta-LAM, similarly to particular LAMs for games or minigames, should define the learning goals, how they are going to be achieved by its different components, how the information is going to be traced and what analyses and visualizations should be performed on the data to provide a complete understanding of the learning process in each GLP and the platform as a whole. Once defined, the meta-LAM should allow access to essential information such as overall student progress in the platform per GLP. Any variable definitions by teachers must be made prior to GLP instantiation. During instantiation, each activity is assigned a unique tracking code, and no further changes to analytics can be accepted without invalidating previously-collected data.

Currently there are two options (treated in a different manner): location-based activities can launch mini-games by themselves, and it is also possible that they are return to the metagame, so the meta-game launches the mini-game, and then go back to the location-based game. In the first case, this requires the GLP to contain the information of possible activities to launch (for more information about location-based analytics, check (Pérez-Colado, Rotaru, Freire-Moran, Martínez-Ortiz, & Fernández-Manjón, 2018)).

Minigames will be described by their developers in detail in small documents that will include: name, short description of the mechanics, game goal and specific visualizations that may be interesting for that minigame.

2.2.2. **Users management system**

When different games or minigames are connected, as it is the case of the BEACONING platform, **a User Management System (UMS) needs to be defined** to allow games to maintain information on logged-in users. This UMS is part of T4.1, to be developed so that other components (including the Analytics System) can access user information and identity. Notice that UCM has implemented a user management system for controlling the access and protection of the analytics, which is different from the BEACONING general user management system: the Analytics System will interact with the platform UMS when it is available, so whenever a user is created in the beaconing platform the LA system is expected to be notified (a webhook[7] is a light way and straightforward way to do it) to register the user in the platform too.

In particular, we want to remark that **the same user identifier should be used** for the results of the complete GLP, and each of its component games or minigames, so that student information can later be connected and linked correctly within analytics.

Notice that for any long-term model of analytics, user identifiers should also remain the same so that all data from a single user can be univocally related and suitable long-term analyses can be performed to obtain the corresponding visualizations.

2.2.3. **Data analytics workflow**

Not only a meta-LAM must be specified but also, it must be clear how the architecture will manage analytics data. For instance, games and minigames are going to send some information back to the gameplot.

Our proposal to handle the data analytics workflow is as follows: the architecture should ensure that both the general platform and the games (or minigames) send their own analytics data (as xAPI statements), without any intermediate actor. This will ensure that real time analytics and dashboards can be obtained.

All these issues should be considered when defining the meta-LAM, **stating clearly the flow of analytics data between the components in the BEACONING platform** (who each component receives the data from, who sends the data to BEACONING Analytics, how this data will be formatted, and how should it be analysed).

Additionally, for any data received by the Analytics System that developers would want to anonymously share, game developers or managers must ensure that its collection meets all the data privacy requirements.

We are currently working on a detailed version of the data analytics workflow. We have already provided details on the formatting and type of information needed, including the description of the xAPI-SG Profile in D4.6. Also, work with game developers has been done in several workshops to implement and improve the analytics provided by their games.

2.3. META-LAM STRUCTURE PROPOSAL

As previously stated, a meta-LAM is required for GLP-level analytics, as GLPs are composed of several games or mini-games and a single LAM does not suffice to represent the relations between those.

For our proposal, we consider the case where the GLP is the root of a tree structure that has missions as children, which in turn will have quests as children, which in turn will have games, mini-games or geolocalized games as children. Those three types of games will be the leaves of the tree, and for each of those is where LAMs will be required. For the general GLP structure, a meta-LAM is required.

[7] Webhook: https://developer.github.com/webhooks/

After revising specifications used in related fields looking for a structure that could be suitable for BEACONING's requirements, we settled on the IMS Simple Sequencing specification from SCORM 2004 4th Edition Sequencing and Navigation (SN) (ADL, 2009). This complex standard requires some simplifications for the case of Beaconing, but we consider that after those simplifications it meets the project's requirements. Our proposed meta-LAM structure, described in detail in (Pérez-Colado, Alonso-Fernández, Freire-Moran, Martinez-Ortiz, & Fernández-Manjón, 2018), is based on a simplification of Simple Sequencing SN. This structure suffices for the case (considered in the review, but no longer expected to be used) where a geolocalized game launches several mini-games when positions in the map are reached.

Simple Sequencing SN defines an Activity Tree (AT) as the main hierarchical structure of learning activities (which may or may not actually be strictly hierarchical). This AT will correspond to a GLP in BEACONING. Learning activities (activities for short) are nodes in the AT, and will correspond to games or other level activities in the GLP tree. Activities have completion (with start and finish) and mastery conditions, and can contain sub-activities. As our analysis is defined, only one node in the tree is active (together with all its ancestors). However, the propagation of events to higher-up nodes can only occur when these have already been initiated, so as not to cause an incoherence between the beginnings of each of the attempts of each node. We also expect nodes to be closed in reverse order: children should be closed before their parents, and all non-common ancestor nodes should have been correctly closed in this way before activating a different node.

Any effort to complete an activity is called an attempt, which may be restarted or aborted, removing all information on previous attempt with the exception of an incremented attempt count. Only one activity (together, always, with all its ancestors) may be attempted at any given time. Currently there is no support for suspending and resuming activities within the analytics framework. A student logging back in after a gameplay session will be understood to be beginning a new attempt, and old attempt information will be discarded. If at a future time the GLP and games support some kind of save and restore feature, then we would need to define specifics regarding how sessions are restored from an analytics point of view. The simplest way would be for games and activities to request the analytics state at the point of suspension, and make a request to restore it on resume.

Learning Objectives have no intrinsic semantics: they may refer to competencies, skills, etc; semantics is only specified as the AT is built. There is no direct correspondence between activities and objectives, that is, one activity may help to achieve several objectives and one objective may be achieved through several activities. Failing nodes does not prevent knowledge propagation: a GLP may be considered successfully completed even though all its component activities may have been individually failed.

Figure 4 shows an example activity tree, retrieved from (ADL, 2009) , with the learning activity "A" as root. Some learning objectives are also defined: all objectives are local to their associated activity, except objective 5 that is shared between activities "B" and "C". Some limit conditions may be stablished for activities to determine when they are not allow to be deliver. In SCORM's SN, the only mandatory condition is the Attempt Limit, which will corresponds to the maximum number of attempts to complete an activity.

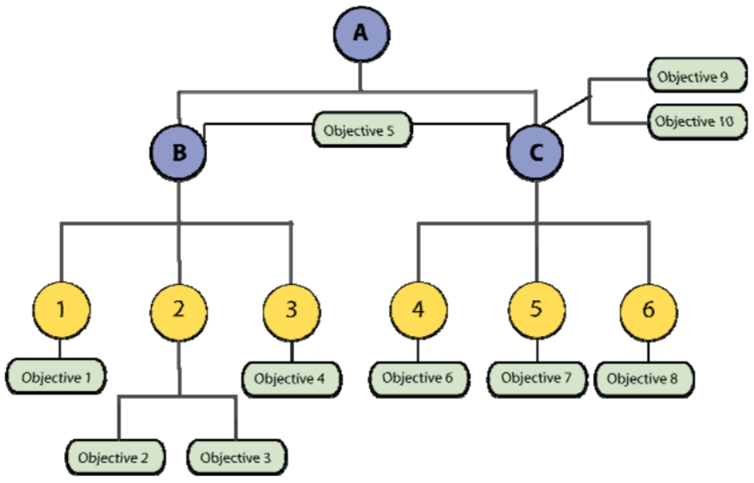


Figure 4. Activity Tree with Learning Objectives shared. Figure retrieved from ADL SCORM 2004 4th Edition Sequencing and Navigation (SN) specification (ADL, 2009).

When activities are repeated, unless instructed otherwise, we will discard any previous data as soon as the repetition starts, and revert any contributions to objectives from the affected node and its descendants. This is easy to implement and understand, but may result in impossible-to-succeed scenarios, especially when coupled with time or repetition constraints.

Every activity will have an associated LAM defining the conditions for its successful (or unsuccessful) completion. Notice that a generic LAM may be defined to cover several games.

As time limits are defined in the GLPs, they are only present at activity level. As the tree structure allows to set time limits for each node, we will consider this possibility in order to increase the flexibility of the system. Note that time limits in this context is used to evaluate pass/ fail status for the GLP activities (quest, missions, etc.) in the context of the analytics system and teacher feedback to do the aggregation. Using this information to modify the game (preventing the student from playing the game) it is out of scope of analytics: students will always be able to play the whole thing, as analytics is powerless to prevent them from doing so. They will, however, be considered to have failed the corresponding nodes. There are certain bonuses for successful completion that depend on time limits. We will apply no time-related bonus unless there is a time limit present.

Success levels between 50 and 70% are considered "partial", and over 70% as "full". The thresholds are applied to scores, learning objectives, and competences. We will consider that all individual learning objectives and competence must pass the corresponding thresholds. See more details on Appendix F.7. GLP Definition.

Within SCORM 2004, Simple Sequencing defines Rollup as the process of passing information from children to parent nodes in the AT. Rollup rules, depicted in Figure 5, define how progress is to be evaluated and consist of:

- a set of child activities to be considered
- conditions to be evaluated against them
- actions to be taken

By default, all children activities are included in parent rollup, unless they are not tracked or do not contribute to rollup because they are not mentioned in any activity set. Leaf activities are not affected by rollup rules. With these definitions, linear-weighted combination of children information will be used to determine progress or completion, and objectives met.

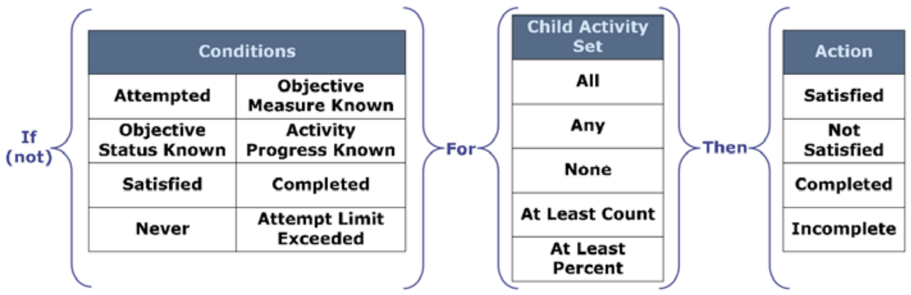


Figure 5. Rollup Rule Child Activity Set, Conditions and Actions. Figure retrieved from ADL SCORM 2004 4th Edition Sequencing and Navigation (SN) specification (ADL, 2009).

Notice how rollup rules are essential for the meta-LAM definition, as they define how information from single activities (games) is to be aggregated up for the GLP. Following the Simple Sequencing proposal and the definition of GLPs, we expect nodes to define their progress and objectives as linear weighted combination of the progress and objectives achieved in their children nodes.

Unlike SCORM 2004, we will perform immediate propagation of knowledge: any new activity points at a given node will propagate to its ancestor nodes. This results in constantly-updated GLP-level knowledge. We will only delay calculation of possible bonuses and success/fail evaluation until end of attempts.

2.3.1. **GLP definition**

We have modelled the flow of analytics on a proof-of-concept implementation to better understand the current GLP model. Because of this implementation, we propose that every node in the GLP include a JSON structure similar to the one below, defining the limits (including thresholds) and the contributions of this node.

```
{
    // from the GLP: limits; if absent, full-success is implied
    limits: {
       "maxTime": 120,
       "maxAttempts": 3,
       "partialThreshold": {
              "learningObjectives": .5,
              "competences": .5,
              "scores": .5},
       "fullThreshold": {
              "learningObjectives": .7,
              "competences": .7,
              "scores": .7},
    },
    // from the GLP
    contributes: {
        "learning-objective1": 1,
        "competence1": .25,
        "competence2": .4,
        "competence3": .25,
        "competence4": .1
    },

    // set during analytics
    active: false,
    success: 0,
    completion: 0,
    startTime: 0,
    endTime: 0,
    elapsedTime:0,
    attempts: 0,
    results: {}
}
```

Internally (within the Analytics System), the results object will contain score, max-possible-score, and activity points for each objective (learning objective or competence) earned in the current node and its descendants.

2.4. ANALYTICS WORKFLOW PROPOSAL

The analytics workflow proposal involves two main steps in the BEACONING platform:

- User management
- GLP assignment and game-plays

The first step, for user management, is that learning designers, teachers and students sign in in Beaconing. Teacher can then create aggregation of students in groups, classes and courses in the teacher UI. All this information will be registered in the Beaconing User Management System (UMS), which will send the required notifications to the Analytics server using the Beaconing webhook, as depicted in Figure 6.

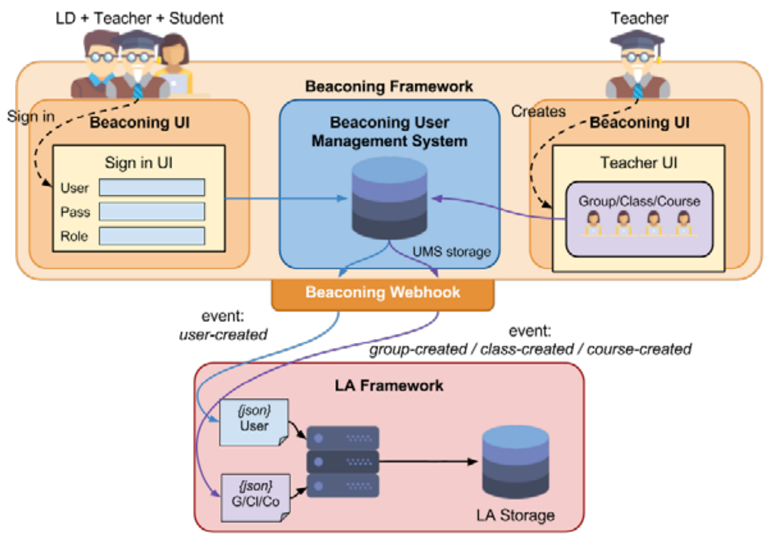


Figure 6. BEACONING Analytics workflow proposal for user management. After stakeholders sign into BEACONING, teachers will create students aggregations.

GLP assignment and gameplay comprises three main steps, which must be carried out prior to any reporting of results via visualizations (dashboards). These steps, depicted in Figure 7, are:

1. Learning designers edit GLPs in the authoring tool. GLPs follow a tree-based structure of missions, quests and activities respectively.
2. Teachers assign a GLP to a group of students from the teacher UI.
3. Students play their assigned GLP, which will typically comprise several activities.

Access to dashboards and the roles of game designers and developers are omitted from Figure 7 for clarity.

When a teacher assigns an instance of a GLP (represented as i-GLP in the figure) to a group, this information will be sent to the Beaconing web hook that will notify every asset listening – including the Analytics server that will receive both the GLP and the group information. The Analytics server will then create a tree structure of activities based on the GLP levels. That tree of activities will be assigned to the corresponding group allowing those users (and only those users) to report traces to the activities in the tree. After this process is completed, the LA framework will return a tree-structured JSON file containing the tracking code for each of the nodes. This information will be passed via the LA web hook that will notify every asset listening with a decorated-GLP that has all the tracking codes (represented as d-GLP in the figure). Those tracking codes will be used in the corresponding games to send their data.

LAM and meta-LAM are essential to analyse the information from activities; and describe how to aggregate the information from each sub-activity. The UMS of BEACONING will ensure that the same user identifier is maintained across the previous process to maintain the information of each student together.

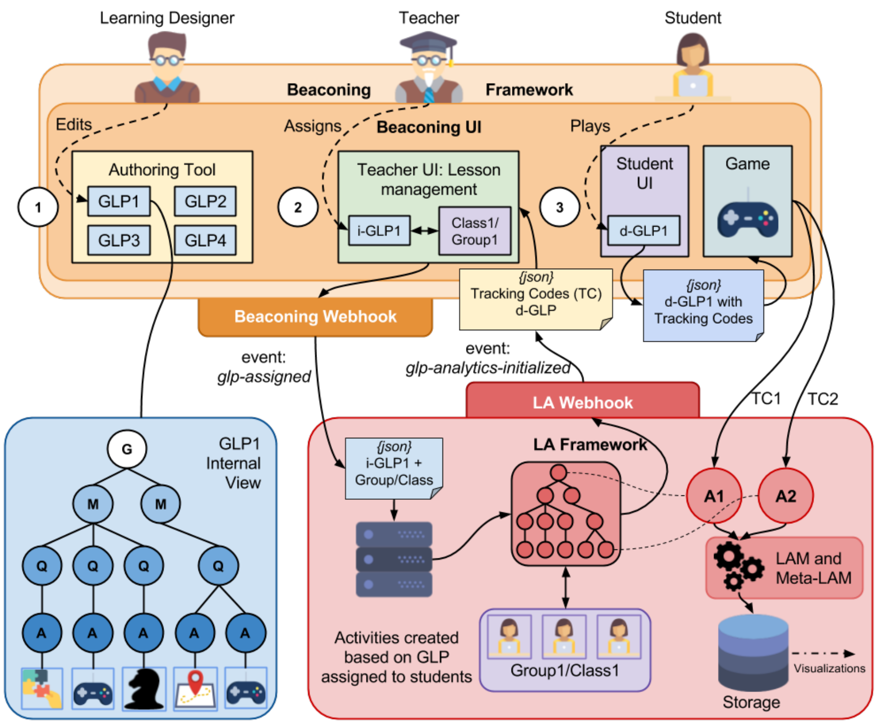


Figure 7. BEACONING Analytics workflow proposal for GLP assignments and gameplay. After the LAMs and meta-LAM are defined, they wil provide the information for analysis of the information tracked from activities and their aggregations.

2.5. ERROR AND ISSUE REPORTING

As in any heterogeneous and complex software development project, errors or bugs may occur, both in code and in documentation. In case of encountering any of these, it is important to adequately report them so the error or issue can be correctly addressed.

In the case of BEACONING Analytics (or e-UCM software in general), errors, bugs and any other issues should be reported by creating an issue in the corresponding GitHub repository (for instance, for issues for the complete Analytics System, report them in the repository https://github.com/e-ucm/rage-analytics). This mechanism is easy to use and provides direct communication between stakeholders. Pull requests from partners are not requested, although we would be happy to receive them.

To create a report, go to the repository and add a new issue[8] with a descriptive title and a detailed description. You may be requested to provide further details if the issue is unclear or we cannot reproduce it. Additionally, you will be notified when the issues' status changes, for example, as soon as it is resolved.

This is the common practice in large community-based software development and will provide more transparency to the software development. This will also simplify uptake by the industry and educational community of the software created by the BEACONING project and contribute to software maintenance beyond the end of the funded project.

[8] https://help.github.com/articles/creating-an-issue/

3. DATA COLLECTION

The analytics process starts with the game. While a student is playing, the game is responsible of observing the actions that the student is performing and sending the information of interest to the analytics server. This first step is key to the whole LAM: the system cannot infer any information that has not been obtained and sent by the game.

The tool embedded in the game to send the information is called a tracker (and in some diagrams, such as Figure 1, "collector"). Currently, there are six open-source tracker implementations available to cover most games:

- JavaScript: https://github.com/e-ucm/js-tracker
- C#: https://github.com/e-ucm/csharp-tracker
- Dot NET: https://github.com/e-ucm/dotnet-tracker
- Unity (adapted from the C# tracker): https://github.com/e-ucm/unity-tracker
- Unity (fully developed in Unity): https://github.com/e-ucm/unity-tracker/tree/0.5.0
- LibGDX (Java) (outdated): https://github.com/e-ucm/libgdx-tracker

However, any tracker that sends traces that conform to the standard xAPI Profile for Serious Games (developed by e-UCM Research Group in collaboration with ADL (Serrano-Laguna et al., 2016), and described below), is acceptable.

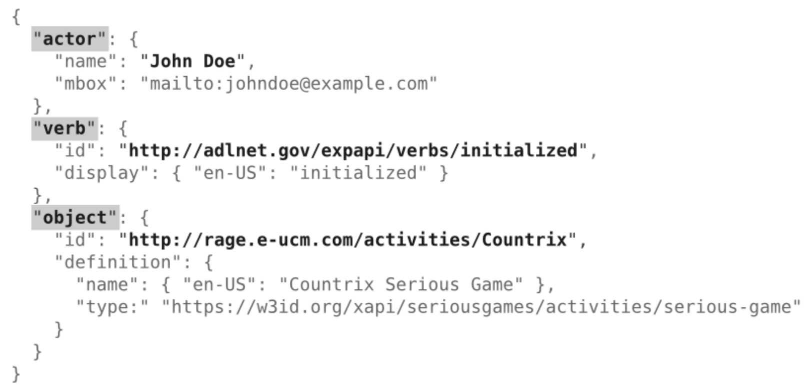
The LAM for a particular game must describe the actual data to be sent from trackers as the game progresses. In particular, it must describe what in-game actions will result in what data being sent in the form of xAPI traces; and its exact mapping into the xAPI serious game profile.

A guide will be provided by partners on how to authenticate the trackers using the Session ID Token; further technical discussions may be needed to implement the necessary changes, either on the trackers (if the Session ID is sufficient for authentication) or to the game launch process (if it is not sufficient, and the trackerId is required instead).

3.1. XAPI-SG MODEL

The xAPI-SG Model describes a general set of interactions to be tracked from SGs. It specifies the verbs, activities and extensions to be used for the information sent by the game. Developed after analysing the most frequent interactions in SGs, it has been implemented in as an Experience API profile.

Experience API (xAPI) is a standard developed by an open community led by ADL to track activities in learning contexts. xAPI traces, called *statements,* correspond to learning activity, and each has three main attributes: an actor, a verb and an object. Additional attributes may be included such as the learning context, the result of the action or the timestamp. Figure 8 shows an example xAPI statement representing that "the actor John Doe started the serious game Countrix".



```
{
  "actor": {
    "name": "John Doe",
    "mbox": "mailto:johndoe@example.com"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/initialized",
    "display": { "en-US": "initialized" }
  },
  "object": {
    "id": "http://rage.e-ucm.com/activities/Countrix",
    "definition": {
      "name": { "en-US": "Countrix Serious Game" },
      "type:" "https://w3id.org/xapi/seriousgames/activities/serious-game"
    }
  }
}
```

Figure 8. Experience API (xAPI) simple statement.

The xAPI-SG Model, developed by UCM together with ADL prior to this project, was fully described in the previous deliverable *D4.6 Game analytics and adaptation component design.*

The xAPI-SG Model includes verbs (e.g. accessed, completed, interacted), activity types (e.g. area, question, serious-game) and extensions (e.g. health, progress). It is accessible online[9].

As required for the H2020 BEACONING Project, an extension of the model is being considered for geolocated games[10].

Additionally, a xAPI trace generator tool has been implemented to help partners to test the exact format that traces should follow. This tracker test app is accessible at http://cdn.rawgit.com/e-ucm/js-tracker/master/test_app.html.

[9] http://xapi.e-ucm.es/vocab/seriousgames
[10] http://xapi.e-ucm.es/vocab/geolocated

4. ANALYSIS AND VISUALIZATION

The analytics system expects to receive statements following the xAPI-SG model. Any other format received may be stored but no default analyses or visualizations can be ensured unless the model is met. By default, some analysis and visualizations are provided. We refer to this as the Default Learning Analytics Model, or Default Model for short, detailed in subsection 4.1.

If any other specific analyses or visualizations are required, as LA systems cannot by themselves determine learning, they should be provided by game developers as detailed in subsection 4.2. Only game developers and designers and GLP authors can decide what is being learnt when, and only with this information in hand do specific visualizations make sense.

Visualizations should be targeted to both roles and games. Notice that the time dimension is already present in data captured by the analytics system, and can be displayed on demand (and indeed, is present in several default visualizations). Other aspects such as Indoor/Outdoor presence are strongly game-dependent: many games are not designed for outdoor use, and displaying outdoor status would make no sense for such games.

Please notice that **the following analytics** (both game-independent and game-dependent) **are specified for the game (or minigame) level Learning Analytics Model**, that is, they work for atomic games. For details about analysis and visualizations for the still-to-be-defined meta-LAM, see subsection 4.3.

4.1. GAME-INDEPENDENT ANALYTICS (FOR THE DEFAULT MODEL)

Game-independent analytics are provided by default. The results of these analyses are then displayed in the default Kibana dashboard. Currently, teachers and game developers obtain a default set of ten visualizations automatically, which provide simple yet useful insights: an overview of the actions undertaken and the results obtained from those actions by each student, in the context of those obtained by classmates.

Information on the default analysis is available online at https://github.com/e-ucm/rage-analytics-realtime and, more specifically, at https://github.com/e-ucm/rage-analytics-realtime/blob/master/default/src/main/java/es/eucm/rage/realtime/simple/topologies/TopologyBuilder.java . This analysis essentially maintains a user-state that keeps track of the progress of the user, and it is always available for display. While much more complex computations are possible (e.g.: the thomas-kilmann example detailed in Appendices B and C), they would only make sense in specific cases, and therefore are not included in the default analysis.

A teacher dashboard (D4.5) is already available outside the Analytics System (of course the data is provided through an API from the Analytics System); the visualizations that compose this dashboard were described in detail in D4.6. Game analytics and adaptation design. However, to improve the data required for specialized (non-generic) teacher visualizations, specific requirements from other partners will be needed. We expect that these requirements will be subsequently updated and refined in the upcoming pilots.

The default dashboard for teachers includes the following visualizations:

- Bar chart with number of times each video has been seen or skipped (see Figure 9 top left).
- Bar chart with maximum score achieved by each player in each completable.
- Bar chart of number of correct and incorrect answers in each alternative.

- Bar chart with minimum, mean and maximum times of completion in each completable.
- Line chart with progress (from 0% to 100%) in each completable along time (see Figure 9 top right).
- Bar chart with alternatives selected in each alternative.
- Bar chart with number of correct and incorrect answers of each player (see Figure 9 bottom right).
- Number of active sessions (see Figure 9 bottom left).
- Number of games started and completed; bar chart with progress (from 0% to 100%) in each completable and in the complete game by each player.

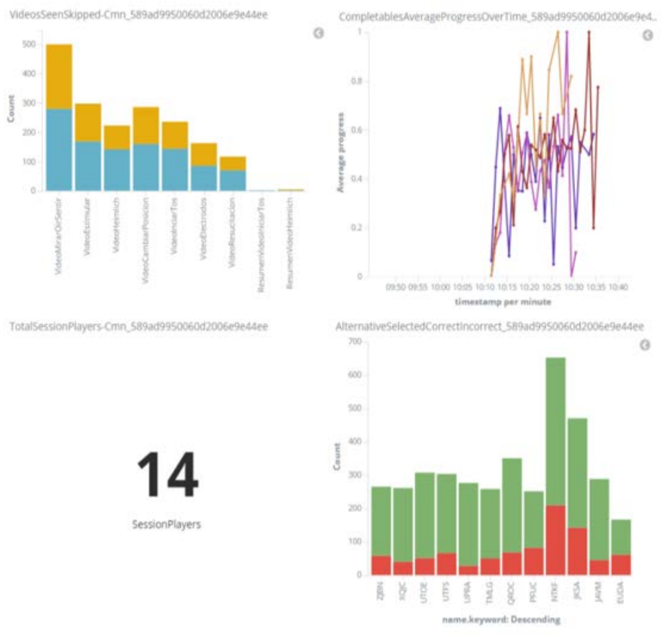


Figure 9. Some of the teacher visualizations available.

Apart from visualizations, specific situations may require that teachers be quickly notified. For this purpose, alerts and warning are included. This is only one of the ways in which the analytics system is actionable: as mentioned previously, a webhook system allows alerts/warnings to be automatically notified to games that register for such notifications. Alerts are situations that require immediate action (e.g. "a student has made an important error") while warnings are less urgent actions (e.g. "a student has been inactive for two minutes"). Both provide teachers with almost real-time feedback on problems that players are having and allow them to act and help the players on the spot.

Alerts and warnings provide an adaptive component for BEACONING: developers can define new alerts and warnings stating the conditions under which they must be triggered and the messages to be displayed when activated. The Analytics System then checks every few seconds if those conditions are true or not, and, when any condition is met, the corresponding alert or warning is added to the general view of alerts and warnings.

For instance, an alert or warning can be specified to be triggered when a scene with identifier *sceneId* has been completed without success:

```
      Message: Player has completed scene sceneId unsuccessfully.
Condition: (this.completed.completable.sceneId.score.normalized < 0.5)
```

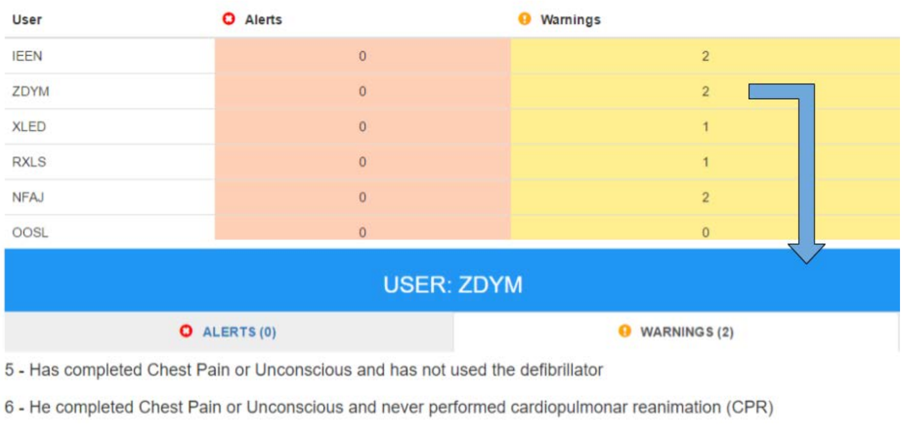
Another example of alert or warning could be to notify when certain scene, for instance the *finalLevel,* has been accessed:

```
Message: Player has reached the final level.
Condition: (this.accessed.accessible.finalLevel)
```

Another option is to notify when the player has interacted with certain item *itemId*:

```
Message: Player has interacted with itemId.
Condition: (this.interacted.item.itemId)
```

The top part of Figure 10 shows the general alerts and warnings view for the Default Model, where the number of alerts and warning each user has triggered is shown next to the user's identification code. More detail information about a single user can be obtained by clicking on the user (bottom part of Figure 10).



User	Alerts	Warnings
IEEN	0	2
ZDYM	0	2
XLED	0	1
RXLS	0	1
NFAJ	0	2
OOSL	0	0

USER: ZDYM

ALERTS (0) | WARNINGS (2)

5 - Has completed Chest Pain or Unconscious and has not used the defibrillator

6 - He completed Chest Pain or Unconscious and never performed cardiopulmonar reanimation (CPR)

Figure 10. General alerts and warning view. Clicking on a user provides more detail information.

We are working in our final validation experiments to test the accessibility of the dashboard by teachers, and the degree to which default visualizations are useful in real-world scenarios (Calvo-Morata, Alonso-Fernández, Freire-Moran, Martinez-Ortiz, & Fernández-Manjón, 2018).

The available student dashboard API (see more details about it on Appendix F.6. Student dashboard) allows to retrieve data from students that will allow the implementation of new and more sophisticated adaptive behaviors.

4.2. GAME-DEPENDENT ANALYTICS

Game-dependent analytics (that is, analytics that is not already included in the Default Model) must be developed for each specific game by developers; since each game is different, details will depend on what is to be collected, and how it should be processed and displayed. Only game developers (or those with access to developers) can link in-game actions with the information to be analysed and displayed. A game-specific analytics model (as part of a defined LAM) is required in these cases for correctly interpreting the traces sent by the game. Sophisticated analyses are possible as long as the corresponding analytics model is available. This dependent analytics can be needed as well for specific GLPs.

One example of an existing analytics plug-in is the Thomas-Kilmann analysis plug-in (https://github.com/e-ucm/tk-kibana-vis). Appendix B and C contain the custom Learning Analytics Model that drive this particular visualization. However, game-dependent analysis does not strictly mean not reusable, that is, game-specific analyses can be carefully designed in order to be reusable in other games; this is, for example, the case of the Thomas-Kilmann analysis and its associated visualization.

4.3. META-LAM ANALYSIS AND VISUALIZATIONS

Some specific analysis and visualizations may be needed for the general meta-LAM, for instance, if a concept of *progress* is defined for a complete GLP in the BEACONING platform.

These meta-LAM analytics should align with the defined model, and therefore, will have to take into account several of the requirements still to be defined, for example as proposed in section 2.3 ("Meta-LAM Structure proposal"). With a suitable meta-LAM, GLP-compliant analytics can be implemented.

Therefore, meta-LAM analysis and visualizations should be defined when the general meta-LAM is completely specified by BEACONING partners.

5. INCORPORATING ANALYTICS TO A GAME

The BEACONING Analytics System is accessible at https://analytics.beaconing.eu/. As described before, any game can send traces to a previously opened activity, belonging to a class (group of students) and a game.

However, it is also possible that games want to present the data retrieved from analytics inside the game interface in a specific way. This could be the case when displaying student metrics without breaking the flow. For instance, Figure 11 depicts in-game display, and previously proposed for BEACONING partners. Game reporting can benefit from querying analytics to retrieve data not only from the current session in the current game, but for overall achievement within the GLP. This is considerably hard to do without an analytics system to query. Getting in-game reports rather than having to open browser windows to an analytics website avoids breaking the game-flow for students - although analytics certainly supports both.

In case any game or minigame wants to show metrics from analytics (such as in Figure 11, retrieved from T4.5), the game will have to make the corresponding calls to the Analytics System using the available APIs, as detailed in the following subsections. Additionally, the Analytics System can only report on data that is actively tracked. In the case of Figure 11, displaying that the “Complex Problem-Solving” skill has increased (the small +1 near the center of the figure) would require a specific analysis to developed; as the default analysis would not, by itself, know to highlight such an increment (would this +1 be displayed only for a short time after a skill is incremented? For how long?).

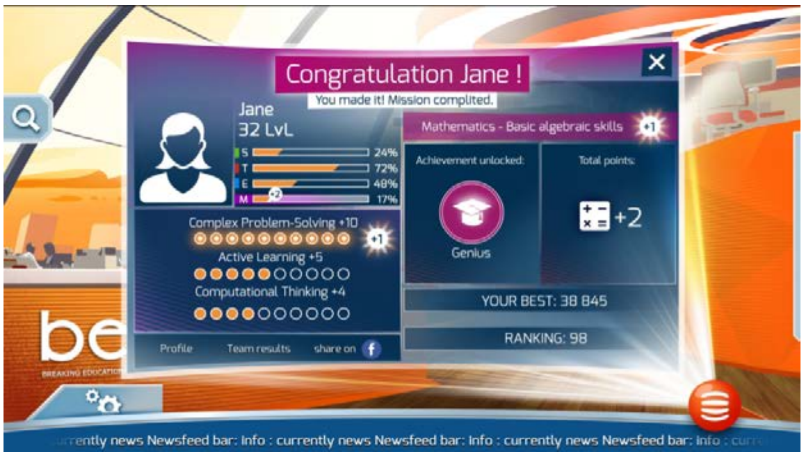

Congratulation Jane !
You made it! Mission complited.
Jane
32 LvL
S 24%
T 72%
E 48%
M 17%
Mathematics - Basic algebraic skills +1
Achievement unlocked:
Total points:
Genius
+2
Complex Problem-Solving +10
+1
Active Learning +5
Computational Thinking +4
YOUR BEST: 38 845
RANKING: 98
Profile
Team results
share on
currently news Newsfeed bar: Info : currently news Newsfeed bar: Info : currently news Newsfeed bar: Info : curr

Figure 11. Sample in-game visualization proposed in BEACONING T4.5.

5.1. INFORMATION REQUIRED FOR IN-GAME ANALYTICS

Recall that games generally have access, within a game session, to the data that they have generated within that session. They can query data sent to analytics or they can use their own copy of the data. However, if they need data from other sessions (say, to display GLP progress), they must have sent it previously, or analytics will be unable to retrieve it for them.

Before starting, we are going to clarify a common issue regarding data modification tracking. For every bar or text in the visualization that displays changes in variables instead of absolute values, a state management system must be created to register those changes. After a detailed analysis, we identified two options:

- Game developers will manage the changes: every time that the game makes a request to the server, the game will keep the state of all the variables. From then on, when a new request is made, the game will compare it with the previously saved state to obtain the changes.

- Analytics developers will create a state management system: this registers changes on the variables until data is requested. We identified some issues regarding parallel data requesting. For example, if the learner is not the only person who requests the changes, they may be unable to track all the changes.

For a more detailed and concrete example, based on the in-game analytics shown in Figure 11, and on the latest agreements in terms of GLP interpretation, we would require the following information:

- **STEM percentages**: we propose the use of four different variables; to be requested from the analytics server individually. However, their values need to be stored first, and for that we propose two options:
  - The meta-game and the minigames will specifically increment or decrement those variables through their traces, for example by adding an extension that specifies: "stem_science: +2" or "stem_technology: -1".
  - The game specifies, as part of the GLP, knowledge objectives it will contribute to. At the end, when the game is completed and the score is processed, an increment or decrement into the objectives will be made by the analysis. This second option is most in line with how the GLP will be interpreted.

  Section 2.1 contains a detailed explanation of how these variable-containing traces are analysed based on LAM definition.
- **Three skills (complex problem-solving, active learning, computational thinking):** in the picture, only the changes to the corresponding variables are shown but, in addition to that, we assume that the changes are being aggregated into a common variable shared between minigames. Again, the objective-containing variables would be updated by the analysis model, based on the GLP's description of scores and weights for the corresponding activity. Thus, "active_learning: +5" could be the consequence of a 10 points-score being reported for an activity with weight 50% towards the Active Learning skill (with the remaining 50% distributed among the other skills).
- **Achievements**: we propose using one variable for each achievement with a special extension that identifies this as an achievement. These variables could then only have two values: achieved or not-achieved (true or false). Then, the minigame or meta-game could specify as an extension the change in the value of this variable. We identify a potential issue in the situation where the analytics server does not know in advance what the badges are. This will make the game unable to retrieve badge values until a change into them is registered. We propose that every badge or achievement should be specified in advance, into the analysis model.
- **Total points**: we understand that the points are gained from the scores of the activities within a GLP. As agreed with GLP stakeholders, all activities will report a score and a maximum-achievable score upon completion, although they may report this more often if so desired.
- **Your best**: we propose to store this metric as the best score of the player, that is, the maximum of the score variables for each mission playthrough. Again, it has been agreed that both scores and maximum-achievable scores will be sent for all activities.
- **Ranking**: this metric, as we understand it, compares the student with the rest of the group (which should also be defined: is it the class, or all previous players?). We would need to know:
  - The total number of players in the ranking, defined as a specific extension in the traces sent, or calculated using the class definition.
  - The scores of all the other players in the ranking. If all the other players are specified, their scores could be obtained from the analytics server.

If the previous information is not provided, the analytics could not infer the ranking of the player and therefore would be unable to return it – and it would be entirely up to the game to generate a ranking for the student.

In-game visualizations must be implemented and provided by the developer along with the educational designer, as well as the traces that the game will send for analysis. It is also necessary to define the required analysis and its outputs to fill these visualizations.

The Analytics System contains two main APIs (A2 and Backend) explained in detail in the following sections, so games can access the required data and, with that data, provide the student dashboard. However, **we strongly recommend using the provided trackers**, submitting feature requests through the GitHub repository if there are missing features. Requesting new features ensures that applications will not break even when the API changes, which will happen in order to provide planned features.

5.2. A2 API

The A2 API is accessible at http://e-ucm.github.io/a2/. It allows several operations for authorization and authentication which include:

- Roles management: creating, returning or deleting roles and permissions.
- Applications management: registering, returning or removing applications.
- User management: listing and removing users, adding or removing roles to users.
- Sign Up of a group of users.
- User login, e-mail based password reset, and password change.
- User logout.
- Returning lists of registered login plugins.
- Checking the API health.

5.3. BACKEND API

The Backend API is accessible at http://e-ucm.github.io/rage-analytics-backend/. Once access is granted through A2, several operations are possible via the Backend API including:

- Game management: adding a game or game version, removing a game, changing game attributes or returning the list of available public games.
- Class management: returning class information and adding or deleting classes.
- Analysis management: adding, deleting or returning analysis for a given version of a game.
- Collector management: starting a tracking session for a client.
- Visualizations management: adding or removing visualizations for a game.
- Kibana management: adding visualizations or dashboards.
- Templates management: adding or returning templates in an Elasticsearch index.
- LTI management: adding or returning a LTI object.
- Checking the API health.

5.3.1. **Current state and upcoming updates**

The current version of the API includes games, classes and sessions management. Each role can obtain information using the corresponding API calls that they are allowed to make (as credentials are required):

- Teachers can create classes and sessions, and obtain the list of classes they belong to and the list of all public games
- Students can obtain the classes they belong to
- Game developers can create games

Session management was modified to include activities management instead of sessions, as corresponds to the newly defined Analytics Model for BEACONING. The API is also being changed to facilitate class and activities creation, simplifying the relation between those and games to make it more usable.

5.3.2. API calls example

In the following section, we present an example with the particular calls that should be made from the game to the Analytics System API. Please notice that some of the following URLs will be changed in the next release, and that some of the calls to obtain player data are not yet available. For the latest up-to-date version of the API, please refer to the corresponding API links[11].

Assuming that the game has been configured together with a class and a session, the videogame should make the following actions through the API calls indicated. In this example, we use analytics-internal nomenclature, where "class" is a group of students. We are currently reworking these descriptors to better map to BEACONING requirements.

- The game can ask for the player username and password to log in. If successful, this will return a bearer-token that must be included in any future authenticated request.

```
POST /api/login

Body:

{
        "username":"username",
        "password":"pass"
}
```

- In case that the user does not have an account yet, the game should register this new user and, after that, proceed to login with the newly created user.

```
POST /signup
Body:
{
       "email": "user@email.com",
       "username": "user",
       "password": "pass",
       "role": "roleName",
       "prefix": "applicationName"
}
```

NOTE: `/api/login` and `/signup` are subject to change in order to align with WP4 core AUTHN / AUTHZ core services.

- The game can show all classes and sessions from a player. This way, it may for instance create a menu where the player should select the class to see the data from.

```
GET /sessions/my

GET /classes/my
```

[11] http://e-ucm.github.io/a2/
http://e-ucm.github.io/rage-analytics-backend/

- The game can get all the data from a single player with id *userId* corresponding to a particular session with id *sessionId.*

```
GET /sessions/:sessionId/results/:userId
```

- The game can see all data from a class and session with id *sessionId*.

```
GET /sessions/:sessionId/results
```

The last two GET queries will return a JSON file with all the collected traces. To be able to create visualizations from these files, it is key that the game developer knows the analyses that are performed on the raw traces, and therefore the structure of the returned data. It is also important to recall that these requests will only be processed if they contain the bearer-token that was returned as a result of a successful login request. That is, the API enforces strict privacy measures and will only allow authorized access to any privacy-sensitive information.

6. CONCLUSION

This document describes a reference implementation of the client and back-end services components of BEACONING's Analytics, that comprise the game analytics and adaptation components as described in DoA. It also includes detail specification of the requirements for its integration with the complete BEACONING platform.

Key design decisions for the implementation, some of them still to be taken by all BEACONING partners, are also presented in the relevant sections. These include choices of architecture, data analytics flow and meta-LAM specification, including analysis model and visualizations required for the complete BEACONING platform.

Learning analytics are both described and exemplified, although not with BEACONING mini-games – we expect authors to create LAMs for each minigame. However, we created an example for the Luxemburg review meeting integrating geolocalized games and minigames (although this line appears not to be the one now used in the GLP).

6.1. RESULTS

The work described in this deliverable shows the complete LAM for games and minigames, the data collection process with the xAPI-SG Model and game-dependent and game-independent analytics. Additionally, details have been provided on how to incorporate analytics to a game using the available APIs.

Further details are also provided in the Appendices: Appendix A describes a complete LAM for the First Aid Game; Appendix B contains the Analytics Manual and API Usage for developers and teachers; Appendix C provides a complete example collection model; Appendix D a complete example custom visualization configuration, both for the Thomas-Kilmann LAM; Appendix E details the analytics integration for the month 18 demo; and Appendix F details the final version of the requirements for Analytics integration, including the definition of the student dashboard and the API provided.

Specific requirements from pilots will also be considered when received.

6.2. IMPACT

The use of the proposed analytics components and its integration in the BEACONING platform facilitate the possibility of all stakeholders to understand and leverage the potential of games inside and outside the classroom, helping them to have a broader picture of how the platform is impacting the students' learning progress.

Also, adaptive components and games are possible since a webhook system allows alerts/warnings to be automatically notified to games that register for such notifications. The possibility of easily configuring new alerts and warnings and to retrieve all the information from students' gameplays from the API provided constitute some of the adaptive components that can be exploited in other scenarios.

6.3. ACKNOWLEDGMENTS

In this work, we have reused, improved and extended some of the Learning Analytics infrastructure of the H2020 RAGE Project.

7. REFERENCES

ADL. (2009). SCORM 2004 (4th Edition). Retrieved from http://www.adlnet.org/adl-research/scorm/scorm-2004-4th-edition/

Baalsrud Hauge, J. M., Stanescu, I. A., Arnab, S., Moreno Ger, P., Lim, T., Serrano-Laguna, A., … Degano, C. (2015). Learning Analytics Architecture to Scaffold Learning Experience through Technology-based Methods. *International Journal of Serious Games*, *2*(1). https://doi.org/10.17083/ijsg.v2i1.38

Calvo-Morata, A., Alonso-Fernández, C., Freire-Moran, M., Martinez-Ortiz, I., & Fernández-Manjón, B. (2018). Making understandable Game Learning Analytics for teachers. 17th International Conference on Web-based Learning (ICWL 2018), August 22nd - 24th, 2018, ChiangMai, Thailand.

Freire, M., Serrano-Laguna, Á., Iglesias, B. M., Martínez-Ortiz, I., Moreno-Ger, P., & Fernández-Manjón, B. (2016). Game Learning Analytics: Learning Analytics for Serious Games. In *Learning, Design, and Technology* (pp. 1–29). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-17727-4_21-1

Marchiori, E. J., Ferrer, G., Fernandez-Manjon, B., Povar-Marco, J., Suberviola, J. F., & Gimenez-Valverde, A. (2012). Video-game instruction in basic life support maneuvers. *Emergencias*, *24*(6), 433–437.

Pérez-Colado, I. J., Alonso-Fernández, C., Freire-Moran, M., Martinez-Ortiz, I., & Fernández-Manjón, B. (2018). Game Learning Analytics is not informagic! In *IEEE Global Engineering Education Conference (EDUCON)*. April 18-20, 2018, Santa Cruz de Tenerife, Canary Islands, Spain.

Pérez-Colado, V.M., Rotaru, D. C., Freire-Moran, M., Martínez-Ortiz, I., & Fernández-Manjón , B. (2018). Learning analytics for location-based serious games**.** In *IEEE Global Engineering Education Conference (EDUCON*). April 18-20, 2018, Santa Cruz de Tenerife, Canary Islands, Spain.

Serrano-Laguna, Á., Martínez-Ortiz, I., Haag, J., Regan, D., Johnson, A., & Fernández-Manjón, B. (2016). Applying standards to systematize learning analytics in serious games. *Computer Standards & Interfaces*. https://doi.org/10.1016/j.csi.2016.09.014

APPENDIX A: FIRST AID GAME LAM EXAMPLE

First Aid Game is a game designed for players between 12 and 14 years old to teach first aid techniques in three particular situations: chest pain, unconsciousness and choking (Figure 12). Each situation appears as a different game level where the player may interact with the main victim or a mobile phone. By answering textual or visual questions, the player learns if the decisions made are correct or not. After completing each level, a mark from 1 to 10 appears according to the errors made and their importance. The complete description of the game can be found in (Marchiori et al., 2012).

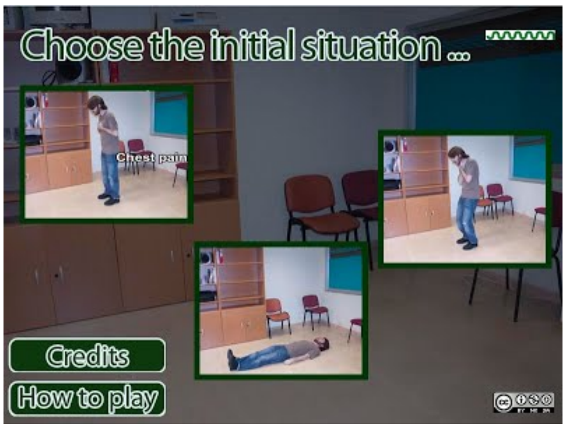


Figure 12. First Aid Game initial screen with three levels.

For the First Aid Game, a complete Learning Analytics Model was defined including the game's learning goals, game goals, traces to be sent, analysis model, visualizations and personalized alerts and warnings. This LAM that was used in the experiments that validated the Analytics Technology for BEACONING.

The defined LAM for First Aid Game is the following:

1. **Define learning goals**: the learning goal is to learn first aid techniques. This goal can be subdivided in seven more specific learning goals:
    - LG1. Learn how to react in a first aid emergency situation.
    - LG2. Learn first aid techniques for *chest pain* situation.
    - LG3. Learn first aid techniques for *unconsciousness* situation.
    - LG4. Learn first aid techniques for *choking* situation.
    - LG5. Learn the emergency telephone number (112).
    - LG6. Learn the security position.
    - LG7. Learn the CPR procedure.
2. **Define game goals**:
    - GG1. To satisfy LG1: Successfully complete all three levels of the game.
    - GG2. To satisfy LG2: Successfully complete *chest pain* level.
    - GG3. To satisfy LG3: Successfully complete *unconsciousness* level.
    - GG4. To satisfy LG4: Successfully complete *choking* level.
    - GG5. To satisfy LG5: Successfully answer the specific question about the emergency number.
    - GG6. To satisfy LG6: Successfully answer the specific question about the security position and watch the explanation video.
    - GG7. To satisfy LG7: Successfully answer the questions about: How to open the airway, how to determine if the patient is breathing, correct position for doing chest compressions, rhythm of the chest compressions. Also, watch the video about the entire procedure.
3. **Define traces to be sent**:

   a. Progress in each level from 0 to 1.
      i. *Initialized* trace at the beginning of each level.
      ii. *Progressed* trace with each progress in the level with progress from 0 to 1.
      iii. *Completed* trace when level is completed and extension with score in the level.
   b. Score in each level.
      i. Sent as an extension in the *completed* trace of the level.
   c. For each question, answer selected by the student and if the answer is correct or incorrect.
      i. *Selected* trace of type alternative with response the selected options and success true if the option is correct or false if it is not correct.
   d. Scenes and Cutscenes (Accessible):
      i. Accessed trace that is sent when the player enters a scene or cutscene.
      ii. Skipped trace that is sent when the player decides to skip a cutscene (video).
   e. Interaction with game elements (e.g. game main character or mobile phone).
      i. *Interacted* trace with object the element interacted with.

4. **Define the analysis model**: The analysis model should specify how the traces are going to be analyzed to extract the relevant information to be displayed in dashboards and to derive the previously defined learning goals (Traces are in xAPI format but this is not relevant here. The important is how to interpret that traces). The analysis model for this example includes:
   a. Progress of the whole game increases in ⅓ with each completed level.
   b. For each level, the completed trace and its score will determine if the learning goal (LG2, LG3 and LG4 for the three levels) will be achieved or not. For this purpose, a threshold for the score needs to be defined: for instance, we set the threshold to 5 out of 10, therefore, if the analysis receives a completed trace for level chest pain with score higher than or equal to 5, we conclude that GG2 has been achieved, and therefore, from its definition, we conclude that LG2 has been achieved too.

   Notice again at this stage that we can provide general analytics but for more specific ones there is no alternative to tight coupling with the game - in the game-specific LAM, the learning outcomes from a specific game's traces will be specified and how these traces will look like.
5. **Define the visualizations**: For this LAM, some of the default visualizations available were used as they provided information of interest that matched the needs of this game and that helped to provide some insight into some of the learning goals. These visualizations should be driven by the teachers requirements (the same for alerts and warnings). These include, for instance:
   a. **(LG1, LG2, LG3, LG4) Progress in each of the three levels (completables) and in the complete game** from 0 (not started yet) to 1 (completed): a bar chart with all students in x-axis and, for each of them, four bars containing the progress from 0 to 1 in y-axis of the three levels and the complete game.
   b. **(LG5, LG6, LG7) Correct and incorrect answers in each alternative** (question in the game): a bar chart with all alternatives in x-axis and, for each of them, a stacked bar with count in y-axis of correct answers (depicted in color green) and incorrect answers (depicted in color red).
   c. **(LG6, LG7) Videos seen/skipped by the student**. Vertical bar chart that displays a pair of sliced column: left one displaying the videos seen by the student, right one displaying the videos skipped.

Some game-dependent visualizations were also developed for this game to show further information of interest for teachers that increments the control of the students' gameplay. For instance:

d. **Number of times each students asked for help in the game**: a game-specific question asked whether the player wants the emergency services to stay on the phone to help the player. This game-specific visualization showed a bar chart with all students that asked for some help
e. **Number of times each video has been seen inside summary menu**: the game contains a final menu after each level where it is possible to play again some of the videos seen in the level. This visualization showed how many times each video was played again in this menu. A pie chart was displayed to show the number of times each video (accessible) has been accessed.

6. **Personalized alerts and warnings**:
    a. *"The user has failed, at least, once the question about the emergency number"*: important question the teacher wants to be notified about.
    b. *"The user has completed Chest Pain or Unconscious and has not used the defibrillator":* situation the teacher wants to be notified about.
    c. *"The user has completed Chest Pain or Unconscious and never performed cardiopulmonar reanimation (CPR)*": situation the teacher wants to be notified about.
    d. *"The user has failed Chest Pain game mode*": the level has been completed but failed, so the knowledge has not been acquired.

APPENDIX B: ANALYTICS MANUAL

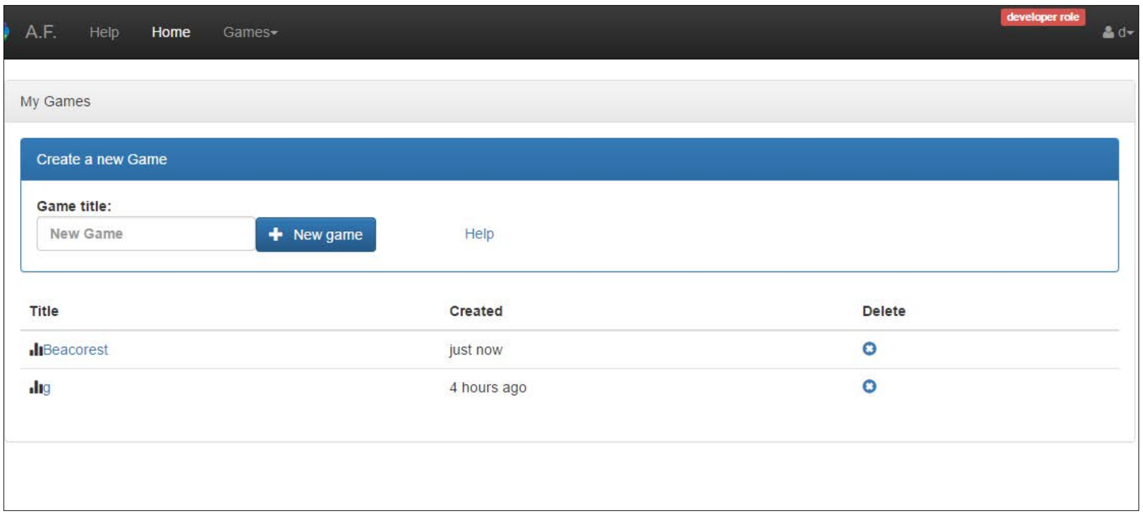
This manual provides both a step by step guide of use of the Analytics System as well as a description of API usage.

B.1. STEP BY STEP GUIDE

Once the analytics server is installed and the framework running, you can start working with all the features it provides. Those features are divided for two roles: Developer and Teacher. Notice that although the interface is aimed to be used for both roles, its use is mainly for programmers.

B.1.1. Developer

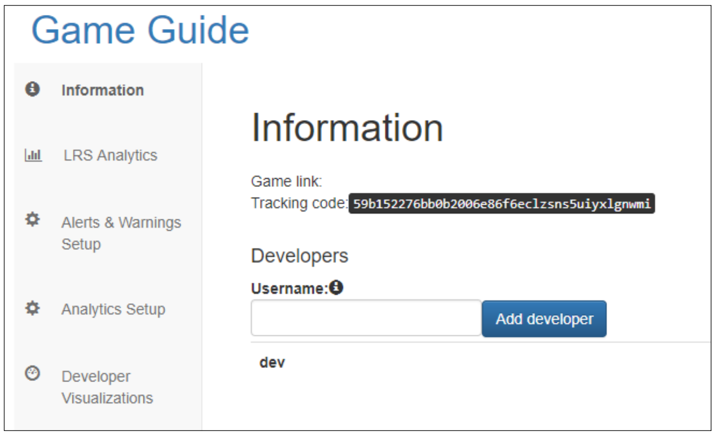
1. If you do not have an account, sign up in the analytics front-end to create a developer account with username, email and password, selecting *Developer* as role.
2. With your account created, log in the analytics front-end.
3. Once you're logged, in the developer homepage (Figure 13), you can create games by using the *Create a new Game* formulary, which needs a game name.



Title	Created	Delete
Beacorest	just now	
g	4 hours ago	

Figure 13.Developer home page with formulary to create games.

4. After creating a game, the game configuration page with information as the *Tracking Code* will be displayed (Figure 14).



Game Guide

- Information
- LRS Analytics
- Alerts & Warnings Setup
- Analytics Setup
- Developer Visualizations

Information

Game link:
Tracking code: 59b152276bb0b2006e86f6eclzsns5uiyxlgnwmi

Developers

Username:

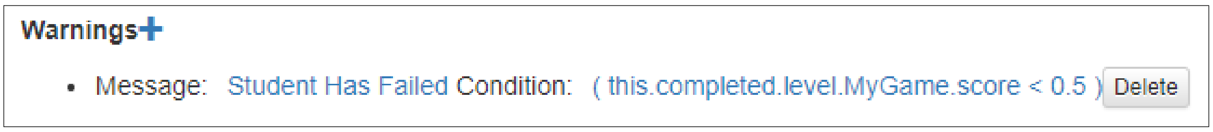
Add developer

dev

Figure 14. Information about a game for developers including tracking code.

5. In the information tab, you have the *Tracking Code* and a small form to add more developers to your game, so they can manage too the game.
6. The Alerts and Warnings setup has more specific configuration. Here you can set up certain assertions to trigger alerts or warnings when something happens. All the variables are analysed for each player. More information in https://github.com/e-ucm/rage-analytics/wiki/Alerts-and-warnings. For example:
   a. If you want to trigger a warning when one of your learners fails a Game (Figure 15), you have to add a warning like this:

```
( this.completed.level.MyGame.score < 0.5 )
```



Warnings

- Message: Student Has Failed Condition: (this.completed.level.MyGame.score < 0.5)

Figure 15. Warnings configuration requires a message and a condition.

   b. If you want to trigger a warning when something more complex happens, like, for example, a student that has failed more than the half of the alternatives (minimum 5 answers given), you have to add a warning like this:

```
( function(res){
      var good = 0, bad = 0;
      for(var i in res.selected.alternative){
            for(var j in res.selected.alternative[i]){
                  if(j == "Incorrect")
                        bad += res.selected.alternative[i][j];
                  else
                        good += res.selected.alternative[i][j];
            }
      }
      return (bad + good ) > 5 && bad > good ;
}(this) )
```

7. Also, in the Alerts and Warning Setup, you have a checkbox that allows you to set the game as public game (Figure 16), so the teachers can create activities for their classes using this game. **If you do not active this option, the game will be only accessible to you as developer. Teachers will not be able to use it.**

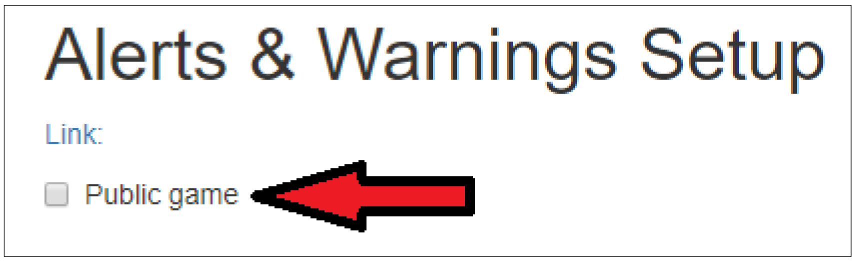
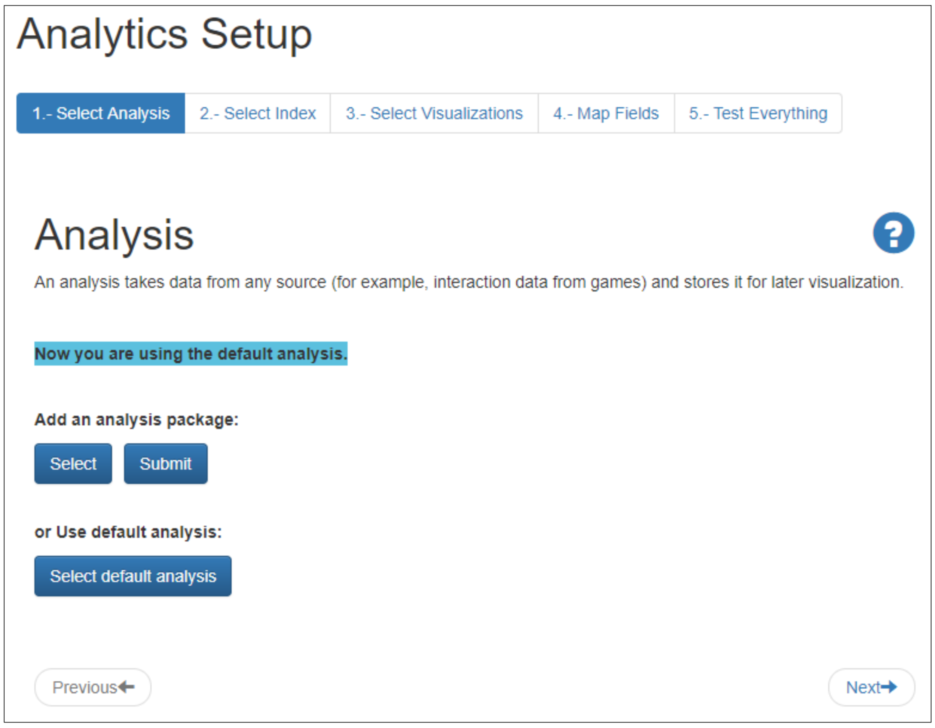


Figure 16. Option to make the game public.

8. Analytics setup tab (Figure 17) allows the developer to manage complex details about how the traces are going to be analysed. As they are considered advanced configuration, we recommend you to read the wiki: https://github.com/e-ucm/rage-analytics/wiki/Analysis-Configuration



An analysis takes data from any source (for example, interaction data from games) and stores it for later visualization.

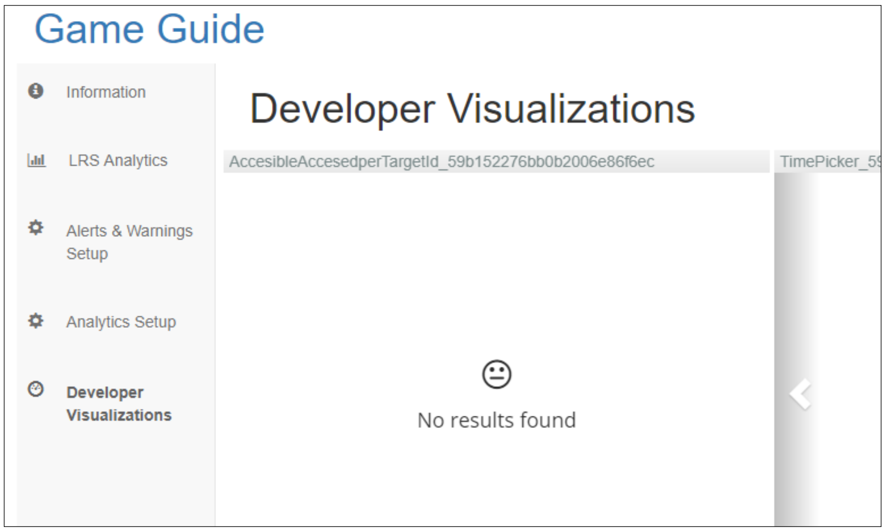
Now you are using the default analysis.

Add an analysis package:

or Use default analysis:

Figure 17. Analytics setup page.

9. And finally, you have developer visualizations (Figure 18):


Game Guide

- Information
- LRS Analytics
- Alerts & Warnings Setup
- Analytics Setup
- **Developer Visualizations**

Developer Visualizations

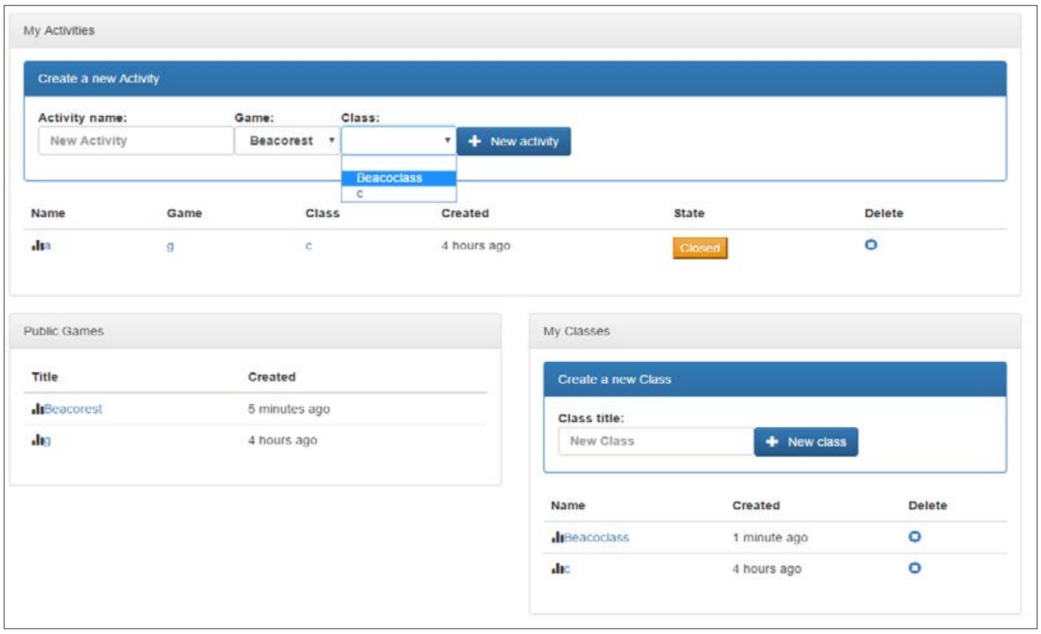
AccesibleAccesedperTargetId_59b152276bb0b2006e86f6ec | TimePicker_59

No results found

Figure 18. Developer visualizations tab.

B.1.2. Teacher

1. If you do not have an account, sign up in the analytics front-end to create a teacher account with username, email and password, selecting *Teacher* as role.
2. With your account created, log in the analytics front-end.
3. Choose the game to use from the list of available games (Figure 19).
4. Create a new class by using the *Create a new Class* formulary which needs a class name or select an already-created class (Figure 19).
5. Create a new activity by using the *Create a new Activity* formulary which needs an activity name, a game and a class, or select an already-created activity (Figure 19).


My Activities

Create a new Activity

Activity name: New Activity | Game: Beacorest | Class: (Beacoclass, c) | New activity

Name	Game	Class	Created	State	Delete
a	g	c	4 hours ago	Closed	

Public Games

Title	Created
Beacorest	5 minutes ago
g	4 hours ago

My Classes

Create a new Class

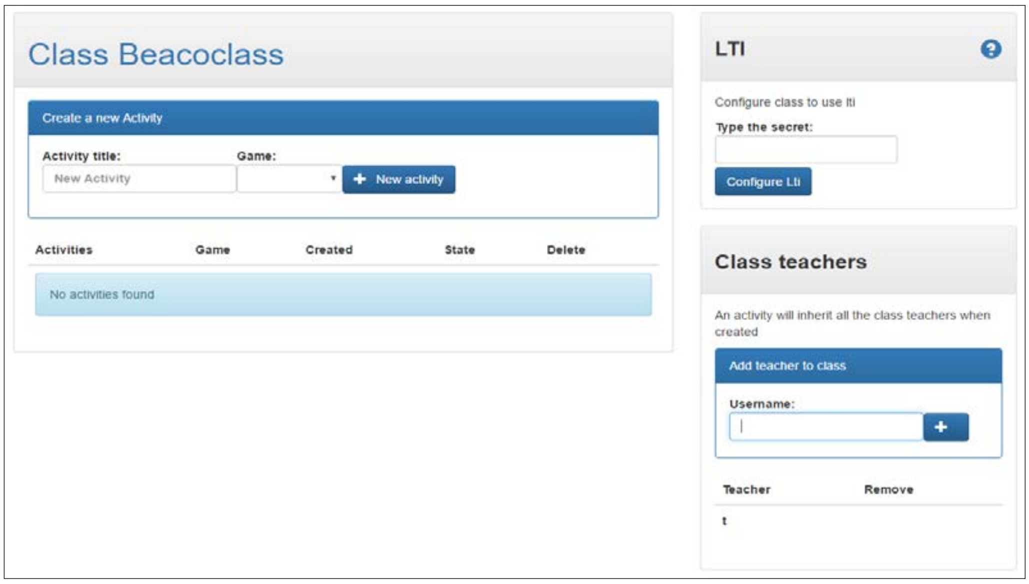
Class title: New Class | New class

Name	Created	Delete
Beacoclass	1 minute ago	
c	4 hours ago	

Figure 19. Teachers page to access available games, create classes and activities.

6. Open the activity to start tracking students' information (initial state for the activity will be *Closed,* see Figure 19).

7. If required, LTI can be configured from the class main view by typing the secret (Figure 20).
8. If more teachers are involved in a class, they can be added from the class main view (Figure 20).



Class Beacoclass

Create a new Activity

Activity title: New Activity

Game:

New activity

Activities	Game	Created	State	Delete
No activities found				

LTI

Configure class to use lti

Type the secret:

Configure Lti

Class teachers

An activity will inherit all the class teachers when created

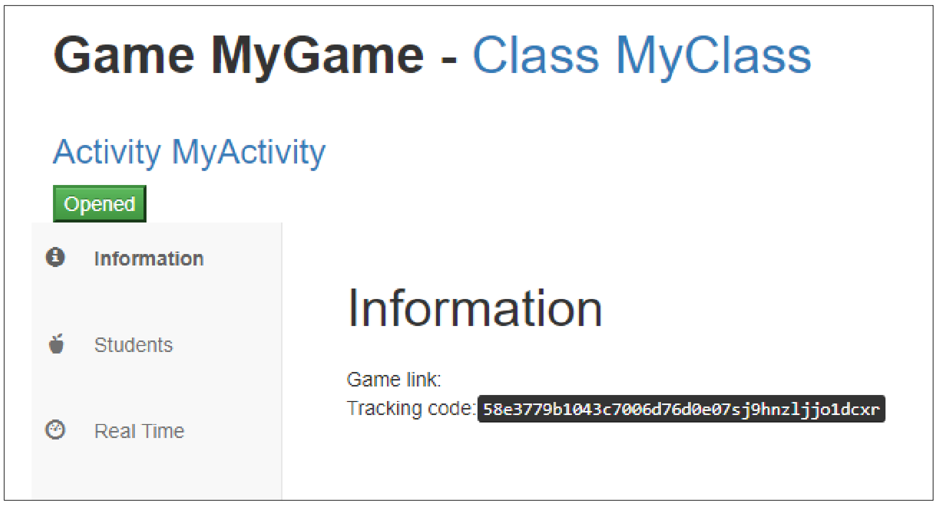
Add teacher to class

Username:

Teacher	Remove
t	

Figure 20. Class view where activities can be created, teachers can be added and LTI can be configured.

9. The activity main view (Figure 21) provides information about the game (link and tracking code) in the *Information* tab.



Game MyGame - Class MyClass

Activity MyActivity

Opened

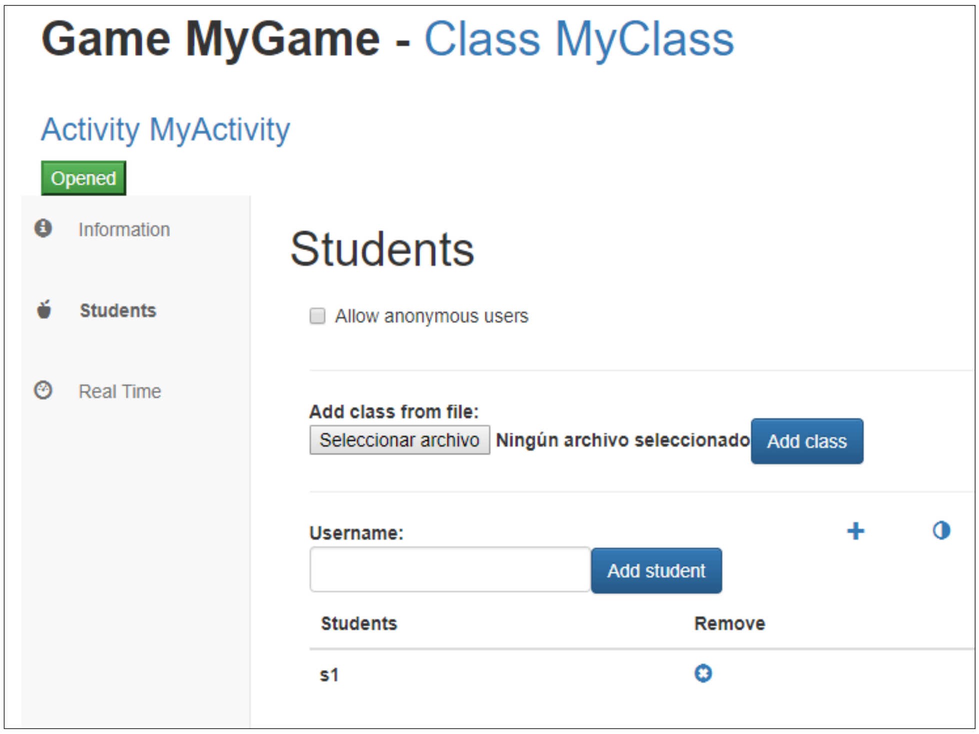
Information

Game link:

Tracking code: 58e3779b1043c7006d76d0e07sj9hnzljjo1dcxr

Figure 21. Information tab of the activity with information about the game including tracking code.

10. In the *Students* tab (Figure 22), teachers may allow (or disallow, by default) anonymous users. A complete class may be added by uploading a class file. Also, specific students can be added to the activity.


Game MyGame - Class MyClass

Activity MyActivity

Opened

Students

- [ ] Allow anonymous users

Add class from file: Seleccionar archivo Ningún archivo seleccionado

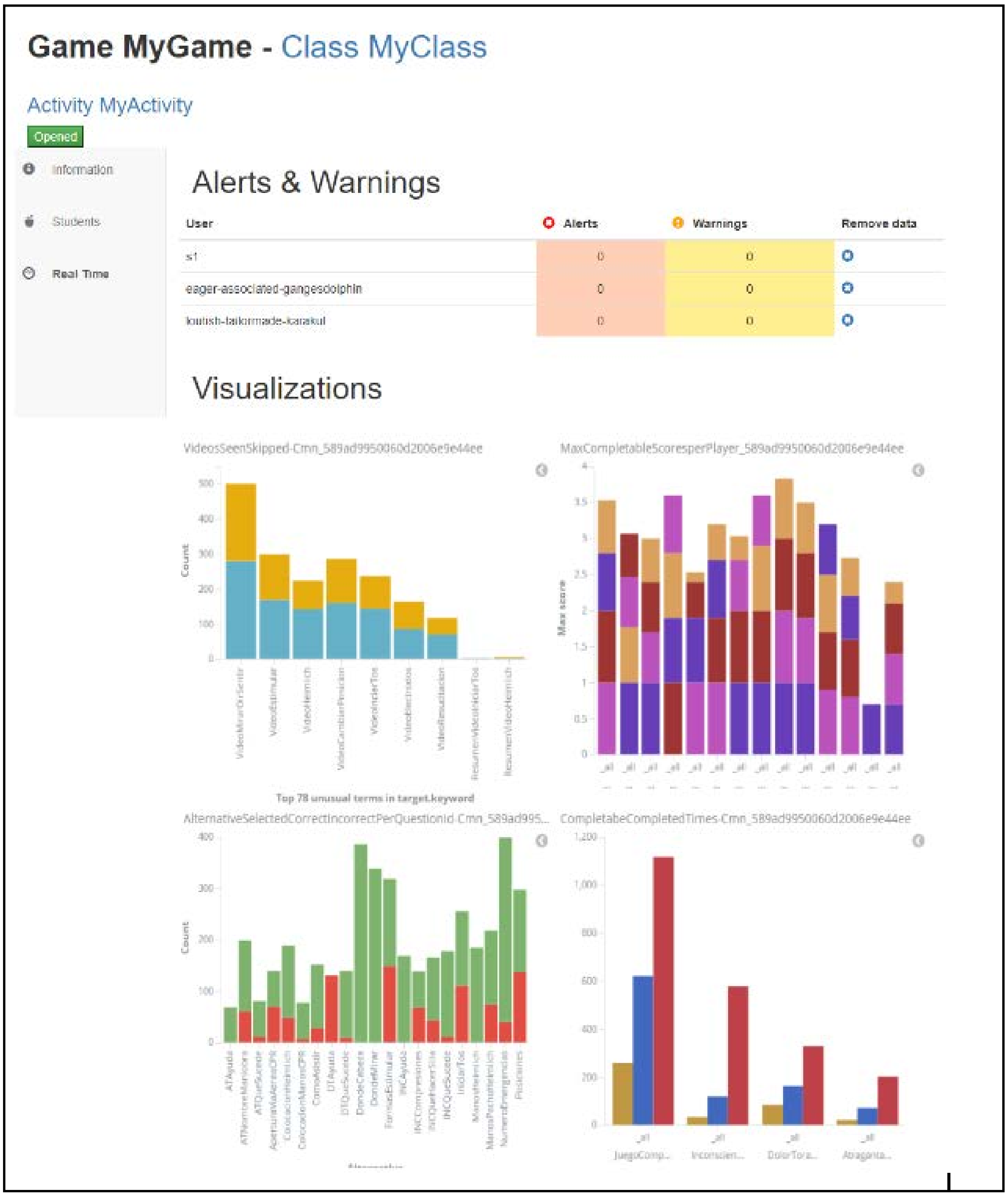
Username:

Students	Remove
s1	

Figure 22. Students tab of the activity where teachers can allow anonymous users or include students or a complete class.

11. In the *Real Time* tab (Figure 23), teachers will receive real time information tracked during the activity.
    a. First, teachers can see the number of Alerts and Warnings triggered in the activity for each student.
    b. Below, they can see the teacher visualizations[12] .

[12]As described in https://github.com/e-ucm/rage-analytics/wiki/Default-visualizations-teacher



Game MyGame - Class MyClass

Activity MyActivity

Opened

Information

Students

Real Time

Alerts & Warnings

User	Alerts	Warnings	Remove data
s1	0	0	⚙
eager-associated-gangesdolphin	0	0	⚙
loutish-tailormade-karakul	0	0	⚙

Visualizations

VideosSeenSkipped-Cmn_589ad9950060d2006e9e44ee
Count
Top 78 unusual terms in target.keyword
MaxCompletableScoresperPlayer_589ad9950060d2006e9e44ee
Max score
AlternativeSelectedCorrectIncorrectPerQuestionId-Cmn_589ad995...
Count
CompletabeCompletedTimes-Cmn_589ad9950060d2006e9e44ee

Figure 23. Real time tab of the activity where teachers can see alerts, warnings and visualizations displayed with the information tracked from students.

B.2. API USAGE

B.2.1. Developer

The API calls for creating a game, and for creating a game version, are available in http://e-ucm.github.io/rage-analytics-backend/#api-Games-PostGames and in http://e-ucm.github.io/rage-analytics-backend/#api-Games-PostVersions.

Here there are two possibilities:

1. You can create only 1 game, and send all the traces to this game. The problem with this is that you have to make sure you tag every trace with an extension of, for

example, which minigame has sent this trace. This way the Kibana dashboard will be hard to read unless you filter the traces.

2. On the other hand, you can register a game for each actual game, and for each minigame. This way every dashboard will be useful by itself, with no need to filter:
    - If you decide to do this, you have to provide tracking codes to each minigame (this is easy to automate).
    - Remember also you have to create an activity for each minigame and class.

B.2.2. Teacher

After creating a game, log in as teacher, and create a class.

- To create a class, you have to make a POST to https://analytics.beaconing.eu/api/proxy/gleaner/classes/ with the class name as body:

```
{ “name”: “class name” }
```

- To add and remove students from a class, use a PUT request to https://analytics.beaconing.eu/api/proxy/gleaner/classes/:classid.
  More documentation can be found at http://e-ucm.github.io/rage-analytics-backend/#api-Classes-PutClasses.

Then, when you have a class and a game, you create an activity.

- To create an activity you have to make a POST request to https://analytics.beaconing.eu/api/proxy/gleaner/activities/ with the following body:

```
{“name”: “activityname”, “gameId”: “ID”, “versionId”: “ID”,
“classId”: “ID”}
```

- Once you have the activity, you have to start the activity sending a post request to https://analytics.beaconing.eu/api/proxy/gleaner/activities/:activityId/:event.

APPENDIX C: EXAMPLE COLLECTION MODEL FOR THOMAS-KILMANN LAM

A full LAM includes:

- A collection model (describing what will be collected when, and how it will be sent to the server; in this case, always as xAPI traces)
- An analysis model (describing how it will be processed in the server, if necessary)
- A presentation model (describing how it will be visualized in dashboards)

This is a collection of 3 trace templates that together conform a collection model. They were developed by e-UCM for a game where a Thomas-Kilmann style visualization was to be used (https://github.com/e-ucm/tk-kibana-vis). It describes how to format xAPI traces that will be sent to the analytics server; and more importantly, exactly when these traces will be sent, in response to user choices in a dialog-driven SG.

C.1. TRACES TYPES

We describe the format for the following trace-types:

- **Completion traces**: to be sent when a level or game-phase, or even the game itself, is finished. We will be using this to update scores.
- **Selection traces**: to be sent when the player selects a reply from among a set of possible replies. Should include information regarding the Thomas-Kilmann classification of the answer, and any biases evidenced or avoided by the chosen answer.
- **Variable traces**: to be sent to indicate changes in variables tracked by the dashboard, such as number of shipped games or average office morale.

C.2. COMPLETION

Use when levels / activities are completed.

C.2.1. Template

```
{
   "actor": <yourActor>,
   "verb": {
      "id": "http://adlnet.gov/expapi/verbs/completed"
   },
   "object": {
      "id": "http://<yourUrl>/<yourGame>/<yourLevel>",
      "definition": {
         "type": "http://curatr3.com/define/type/level"
      }
   },
   "result": {
      "extensions": {
         "https://rage.e-ucm.es/xapi/ext/value": <resultsForLevel>
      },
    "score": {
      "raw": <floatScore>
    },
    "success" : <boolSuccess>
   },
   "timestamp": <iso8601dateAndTimesString>
}
```

C.2.2. Placeholders

- <yourActor>: obtained from “start” REST request
- <yourUrl>/.../<yourLevel>: should be unique, used to identify exactly what was completed.
- <resultsForLevel>: overall score as an integer. Useful for Giel's score comparisons
- <floatScore>: can be omitted; 1.0 by default
- <boolSuccess>: can be omitted; true by default
- <iso8601dateAndTimesString>: example - "2016-12-31T13:05:12Z"

C.3. SELECTION

Use when an answer to a question is chosen.

C.3.1. Template

```
{
   "actor": <yourActor>,
   "verb": {
      "id": "http://adlnet.gov/expapi/verbs/selected"
   },
   "object": {
      "id": "<yourUrl>/<yourGame>/<yourAlternative>",
      "definition": {
         "type": "https://rage.e-ucm.es/xapi/seriousgames/activities/Alternative"
      }
   },
   "result": {
      "response": <chosenAnswer>,
      "extensions": {
         "https://rage.e-ucm.es/xapi/ext/thomasKilmann": <tkClassification>
         "https://rage.e-ucm.es/xapi/ext/biases": { <listOfBiases> }
      }
   },
   "timestamp": <iso8601dateAndTimesString>
}
```

C.3.2. Placeholders

- <yourUrl>/.../<yourAlternative>: used to uniquely identify the question.
- <chosenAnswer>: used to uniquely identify the chosen answer
- <classificationOfAnswer>: describes the classification of the answer, in terms of Thomas-Kilmann and of biases. The expected format is a JSON object that complies with the following structure, in pseudo-BNF (where “|” represents a choice, “<” and “>” define a non-terminal, and all other symbols are to be interpreted as JSON syntax):
- <tkClassification> ::= "avoiding" | "competing" | "accomodating"

- | "compromising" | "collaborating"
- <listOfBiases> ::= <bias> | <listOfBiases>, <bias>
- <bias> ::= <biasName> : <biasValue>
- <biasName> ::= "gender" | "race"
- | "ability" | "occupation" | "fashion" | "otherSocial"
- <biasValue> ::= true | false

Examples of biases

{ "ability": true, "gender": false } }

ability-bias exhibited, gender-bias avoided; true indicates that the bias was exhibited, false that the bias was not exhibited.

{ }

question had no biases to either exhibit or avoid. In this case, the biases extension could have been omitted.

Note that it is important to list both biases that are exhibited ("gender": true) and biases that are avoided ("gender": false), because when calculating the frequency of responses that exhibit a certain bias, we will be using the formula

frequency = n_answers_selected_exhibiting /

(n_answers_selected_exhibiting + n_answers_selected_avoiding)

That is, if a player has answered 1000 questions, of which 200 have answers that can exhibit a gender bias, and the user has exhibited this bias in 120 answers, then players' gender bias will be computed as 60% (= 120 / 200) instead of 12% (which would be the result of 1000 / 120).

C.4. VARIABLES

Use for changes in global state

C.4.1. Template

```
{
  "actor": <yourActor>,
  "verb": {
    "id": "https://rage.e-ucm.es/xapi/seriousgames/verbs/set"
  },
  "object": {
    "id": "<yourUrl>/<yourGame>/<yourVariable>",
    "definition": {
      "type": "https://rage.e-ucm.es/xapi/seriousgames/activities/Preference"
    }
  },
  "result": {
    "extensions": {
      "https://rage.e-ucm.es/xapi/ext/value": <value>
    }
  },
  "timestamp": <iso8601dateAndTimesString>
}
```

C.4.2. Placeholders

- <yourUrl>/.../<yourVariable>: used to uniquely identify the variable being changed. We expect it to be one of
- “gamesShipped” - with a positive integer as <value>
- “awardsWon” - with a positive integer as <value>
- “officeMorale” - with an integer from 0 to 100, both included, as <value>

<value>: an integer value; see above point for details on ranges.

APPENDIX D: EXAMPLE VISUALIZATION CONFIGURATION FOR THOMAS-KILMANN

In this example, we are going to configure the Thomas-Kilmann visualization and see results as a teacher sending some sample traces following the template described in Section 0.

D.1. PREREQUISITES

With rage-analytics installed, access the docker-compose file: https://github.com/e-ucm/rage-analytics/blob/master/docker-compose.yml#L143

In the line selected, change the default line:

```
image: eucm/rage-analytics-realtime:1.3.1
```

For the following line, which refers to release https://github.com/e-ucm/rage-analytics-realtime/releases/tag/1.3.1-hull:

```
image: eucm/rage-analytics-realtime:1.3.1-hull
```

Then, restart rage-analytics:

```
./rage-analytics.sh restart
```

D.2. VISUALIZATION CONFIGURATION

To configure and see results using the Thomas-Kilmann visualizations, you need to do several steps both as a developer and as a teacher. Most of these are described in detail in the Analytics System manual, so please refer to Section 0 for further details.

D.2.1. As developer

1. Create a user with developer role
2. Log in with developer role
3. Register a game
4. Make the game public
5. Take note of the game Tracking Code
6. In the *Analytics Setup* tab, go to step *3-Select visualizations* and tick the Thomas-Kilmann visualizations you wish to include. The four visualizations for Thomas-Kilmann available are:
    a. *ThomasKilmannClassificationBoxesWidget*
    b. *ThomasKilmannOverTimeSwimline*
    c. *BiasesBarPercentageViss*
    d. *ThomasKilmannClassificationPieChart*

D.2.2. As teacher

1. Create a user with teacher role
2. Log in with teacher role
3. Create a class
4. Create an activity for that class and the previously created game
5. Open the activity
6. Allow anonymous users in the *Students* tab
7. Go to the *Real Time* tab and wait to see the visualization results

8. If needed, reload page to see results

D.2.3. Send sample traces

1. Send POST to start using the previously obtained game Tracking Code

   POST http://<your-ip>:3000/api/proxy/gleaner/collector/start/<tracking-code>

Expected response:

```
{
   "authToken": <auth-Token>,
   "actor": {
      "account": {
         "homePage": "http://a2:3000/",
         "name": "Anonymous"
      },
      "name": "contorted-unrestrained-conure"
   },
   "playerAnimalName": "contorted-unrestrained-conure",
   "playerId": "59b694dd4e343a006e022f6a98810",
   "objectId":
"http://a2:3000/api/proxy/gleaner/games/59b68db84e343a006e022f60/59b68db84e343a006e02
2f61",
   "session": 1,
   "firstSessionStarted": "2017-09-11T13:51:25.715Z",
   "currentSessionStarted": "2017-09-11T13:51:25.715Z"
}
```

2. Send some sample traces

   POST http://<your-ip>:3000/api/proxy/gleaner/collector/track

Including the previously received authorization token in the header:

Authorization: <auth-Token>

And body with the extensions wanted and the actor previously received:

```
[
{
   "actor": {
      "account": {
         "homePage": "http://a2:3000/",
         "name": "Anonymous"
      },
      "name": "full-necromantic-hairstreak"
   },
   "verb": {
      "id": "http://adlnet.gov/expapi/verbs/selected"
   },
   "object": {
      "id": "<yourUrl>/<yourGame>/<yourAlternative>",
      "definition": {
         "type": "https://rage.e-ucm.es/xapi/seriousgames/activities/Alternative"
      }
   },
   "result": {
      "response": "testAnswer",
        "extensions": {
            "https://rage.e-ucm.es/xapi/ext/thomasKilmann": "avoiding",
            "https://rage.e-ucm.es/xapi/ext/biases": { "otherSocial":"true" }
        }
    },
    "timestamp": "2017-09-11T13:33:50.459Z"
}
]
```

Expected response:

```
{
    "message": "Success."
}
```

D.2.4. Results visualizations

After sending some sample traces, the teacher will see in the *Real Time* tab the selected Thomas-Kilmann visualizations with the results of the analysis performed on the sent traces. The following four visualizations are available: *ThomasKilmannClassificationBoxesWidget* (see Figure 24), *ThomasKilmannOverTimeSwimline* (see Figure 25), *BiasesBarPercentageViss* (see Figure 26) and *ThomasKilmannClassificationPieChart* (see Figure 27).

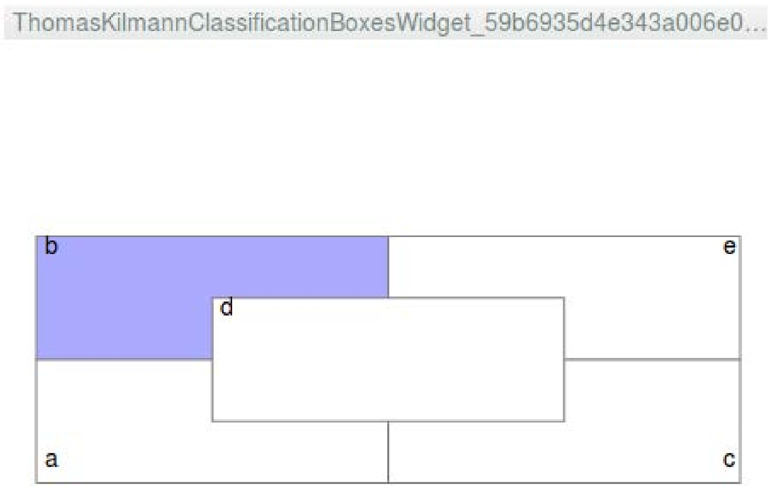

ThomasKilmannClassificationBoxesWidget_59b6935d4e343a006e0...
b
e
d
a
c

Figure 24. Visualization for Thomas-Kilmann classification boxes.

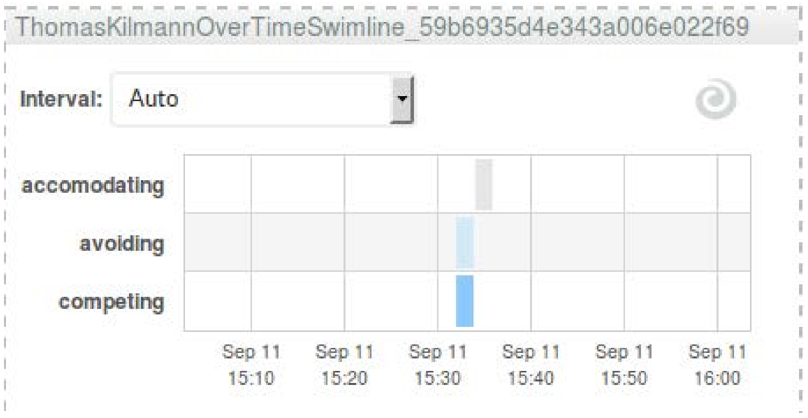


Figure 25. Visualization for Thomas-Kilmann over time.

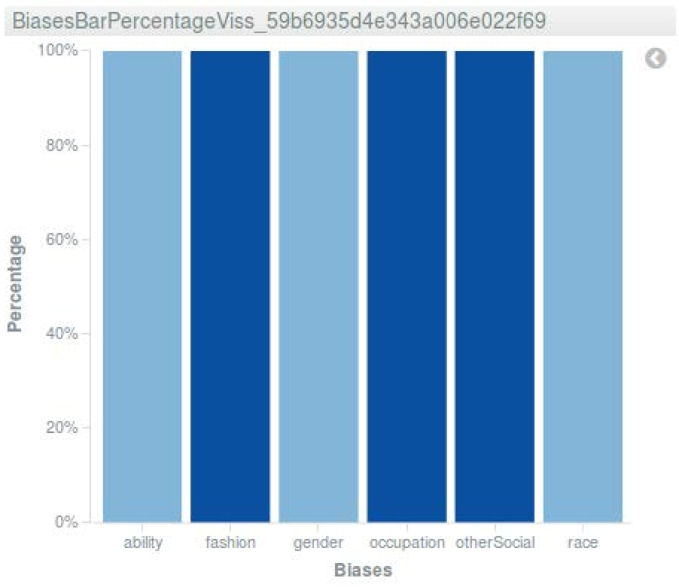


Figure 26. Visualization for Thomas-Kilmann biases bar percentage.

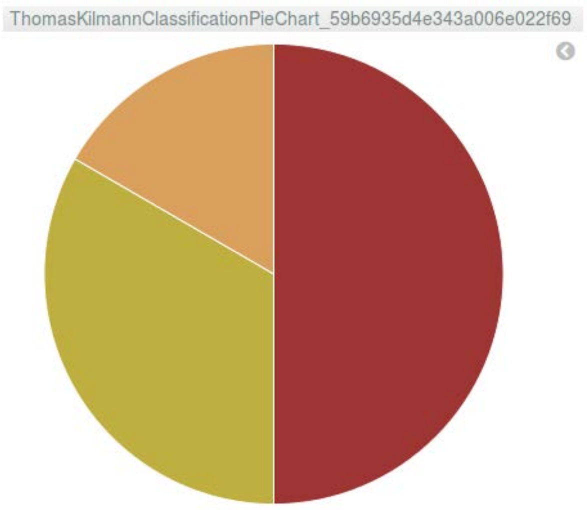


Figure 27. Visualization for Thomas-Kilmann classification pie chart.

APPENDIX E: ANALYTICS INTEGRATION INTO MONTH 18 DEMO

This appendix details the analytics integration performed for the month 18 demo.

E.1. TRACKER MODIFICATIONS

Two main modifications have been made to the tracker:

- For session managing, the tracker is now able to login using a user token, which can be an anonymous PlayerID or a Bearer auth token.
- To improve traces management, the tracker has to be started and received a valid actor object before it allows to send any. Once the actor object has been received, that actor will be automatically appended to the upcoming and already generated (but not sent) traces.

E.2. IMAGINARY MINIGAMES

First, we set up all the analytics for the Swipe and Seek minigame. The trace identification and generation hasn't been done in this order.

Those analytics include:

- Minigame ID is sent as: game_name + "_" + session_id: "SNS_GAME_1234"
- Completable initialized when the minigame starts
- Completable progressed every time a question is answered
- Completable completed when the minigame is completed, including:
    - Time, Total_questions, correct_answers, wrong_answers, skipped_answers
- Alternative selected every time a question is answered including:
    - Generated answer (DOG|CAT or LION|OCTOPUS) as Response
    - Success true or false if the question is correct or incorrect.
- Every time a trace is sent, the game_name and the session_id are appended as extensions.

After having all the traces, similar traces are generated for the Generic Quiz minigame.

The game has been modified to receive, in the url as parameter, the user token. Before starting the tracker, the userToken is added to Tracker.settings.userToken, what keeps the same session as in the meta-game.

E.3. GEOMOTION META-GAME

We already had integrated some traces into the location-based game, including:

- Completable initialized when the minigame starts
- Completable progressed every time a point of interest (POI) is visited.
- Completable completed when the minigame is completed, including:
    - Time, totalAverageSpeed and totalDistance.
- Moved to POI with the POI_ID, lat and lon.
- Every 3 seconds: the current position as geo-point.

We have updated some of the traces due to bugs on the code:

- Progress has not been sent. Now current progress is completed_pois/total_pois
- As minigame closes and opens constantly, time between POIs is lost every time. This has been corrected.

As game closes and opens, also the tracker session is lost. For keeping the session, user token is extracted from the tracker after starting it and saved into a cookie. The next time the game is opened, it will look for that cookie, and, if it exists, will load that user token before starting the tracker.

Finally, for keeping the same session between minigame and meta-game, the user token is sent when summoning the minigame, into the url as query parameter.

E.4. ANALYTICS SETUP

A custom index pattern has been created including all the variables from both the minigame and meta-game with a total of 16 variables:

Table 1. Variables identified from the minigame and the meta-game.

ext.game_name	*ext.correct_answers*	*ext.wrong_answers*	*ext.skipped_answer*
ext.total_questions	*ext.timeSpent*	*ext.poiId*	*ext.poiId*
ext.poiAverageSpeed	*ext.averageSpeed*	*ext.poiDistance*	*ext.distance*
ext.totalDistance	*ext.session_id*	*ext.session_id.keyword*	*ext.location*

Four visualizations have been created:

- *SelectedPlusCorrectIncorrectPerGame*: it shows, for each of the minigames (in a different pie chart), the options selected in the inner circle and whether the option is correct or not (green or red in the outer circle, respectively).

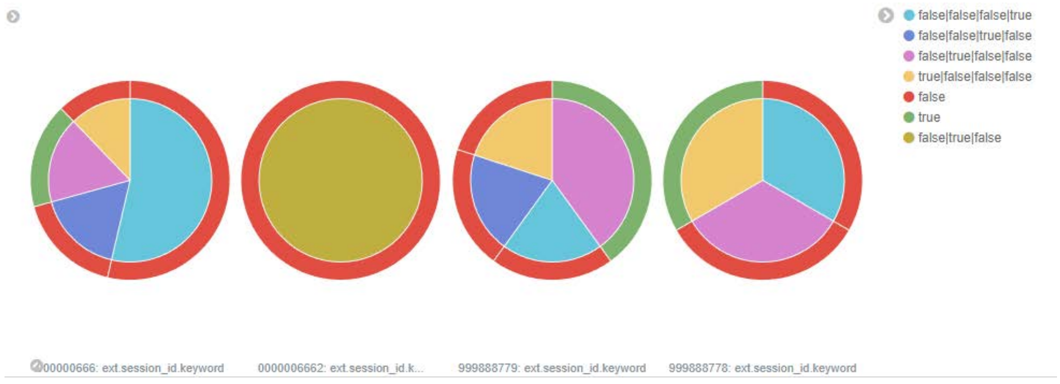


Figure 28. Visualizations for minigames correct and incorrect options selected.

- *POIProgressedTime*: it shows, for each POI, the minimum, average and maximum time spent from the previous to the current POI.

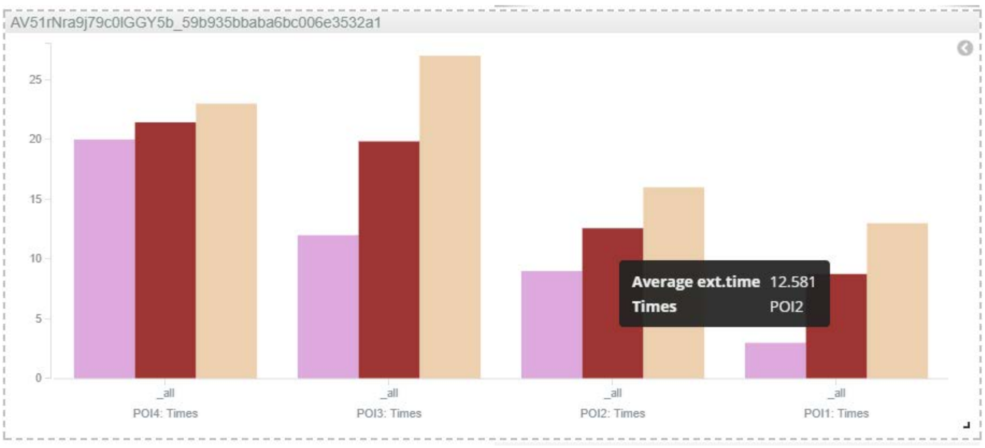


Figure 29. Visualization showing min, average and max time between POIs.

- *GamesCompletedInitialized*: it shows, for each player in a different pie chart, the number of minigames started and completed.

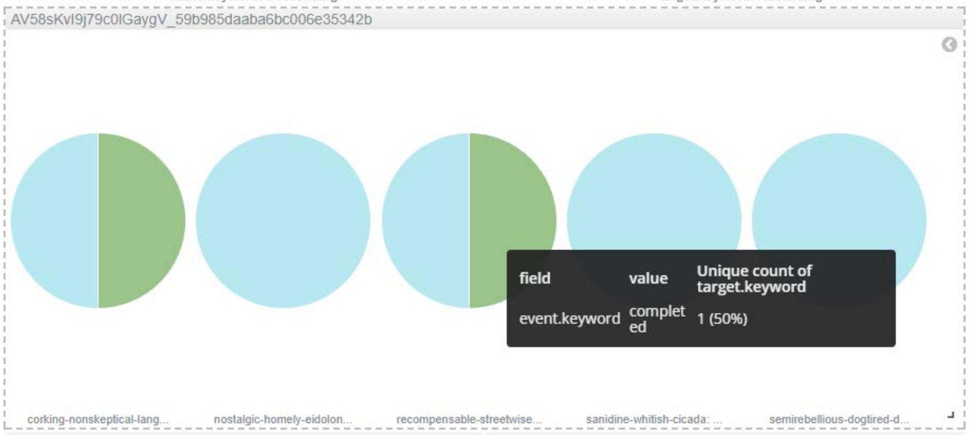


Figure 30. Visualization showing minigames started and completed for each player.

- *TotalCorrectIncorrectSkipped*: it shows, per player and in total, in four bars, the aggregation of *ext.correct_answers, ext.wrong_answers, ext.skipped_answer, ext.total_questions*.

APPENDIX F: BEACONING ANALYTICS REQUIREMENTS FOR PARTNERS

This appendix elaborates on the work done on evaluation on the games and game-plots (including geolocalized games and combination of games and mini-games) and describes the information that BEACONING Analytics requires from other partners in order to ensure that dashboards adequately provide meaningful information about the learning process of students.

The aim of this document is to ensure that every partner understands Learning Analytics Models (LAMs) and agrees on the issues mentioned below that directly affect the Analytics System.

F.1. FROM GLPS' CREATOR PARTNERS

We expect skills/competences/knowledge/scores to be defined by partners involved in GLP creation, including the original creators and the teachers that edit or customize this GLP. GLPs are "instantiated" when they are assigned to a class or group of students. Once instantiated, changing a GLP would render any pre-change analytics data useless; therefore, once instantiated, GLPs must not be modifiable.

Any variable definitions by teachers must be made prior to GLP instantiation. During instantiation, each activity is assigned a unique tracking code, and no further changes to analytics can be accepted without invalidating previously-collected data.

Regarding variable definitions, partners need to consider the LAM definition (see Section on Learning Analytics Model), including any specific processing that Analytics should perform to incoming traces. For example, if all activities contribute equally to a given knowledge variable, this should be stated. If they should contribute differently from each other, then specific weights (assuming linear combination) or specific formulae (for more complex calculations) should be specified.

This should be considered for GLP, mini-games and geolocalized games.

F.2. FROM GAME AUTHORS / DEVELOPERS

Regarding learning-objectives / skills / competencies (which we handle very similarly for analytics purposes, and will refer to as Objectives from here onwards), there are two levels:

- Game objectives: Games that send analytics data must specify, if any, what objectives are accomplished by this data. It is assumed that game developers, in collaboration with teachers, have defined some objectives that are accomplished through the game. Once the game is developed these objectives are fixed. The arithmetic game has only arithmetic goals and this can not be modified by the teacher in the game nor the GLP without game developer help.
- GLP objectives: The GLP can also specify objectives to be associated with success in a game. However, this information is very coarse-grained; for example, a game with a score of 70% may be associated with a GLP-defined knowledge gain of 10 points in Learning Objective 1 and 20 in LO2. In this case, the gain would be +7 LO1 points and +14 LO2 points. But from a single score, the balance between LOs would clearly always be constant. These coarse grained objectives may be changed / customized by teachers before the GLP is assigned / instantiated to students.

We expect game developers to define the game objectives, based on the learning design of the game. Those game objectives should then define the traces to be generated from the game. The traces to be generated should be communicated to UCM to ensure that they will be correctly interpreted. In the same way, for template-driven games (those with game objectives that depend on runtime variables), the templates themselves should also be communicated to UCM to ensure correct interpretation.

As an example of the degree of detail required for these definitions, see the following extract from UCM's discussions on analytics for games **already developed and included inside the code** by different partners:

- Geomotion:
  - Traces:
    - When the game starts as Completable.Initialized(game);
    - When the game ends as Completable.Completed(Score);
    - When the player Moves as Places.Moved(POI_ID, Lat, Lon)
      - with extensions: Position(GeoPoint)
    - When the player reaches a POI: as Completable.Progressed(game,
    - Every 3 second Places.Moved(Location);
  - Meaningful variables:
    - time: Time between POIs
    - poiId: Determines which POI has been visited in order to progress into the game.
    - averageSpeed: Speed between POIs
    - distance: Distance walked between POIs
    - totalDistance: Total distance walked in the game
- Imaginary (GenericQuiz and Swipe&Seek minigames):
  - traces:
    - When the game starts as Completable.Initialized(game_name + "_" + session_id);
    - When the game ends as Completable.Completed();
      - With the game result as a full set of meaningful variables
    - When the uses a Game Item as GameObject.Interacted(ItemName)
      - with extensions: Game_ name + "_" + session_id so traces can easily filtered
    - When a question is answered:
      - as Alternative.Selected(questionName, selectedAnswer)
      - with extensions:
        - Skipped (If learner decides to skip the question)
        - Success: determines if the question is correct or incorrect
  - Meaningful variables:
    - correct_answers: total number of correct answers
    - skipped_answer: total number of skipped answers
    - time: the total time in seconds
    - total_questions: the total amount of questions in the game
    - wrong_answers:0
    - game_name: as a common identifier for all the interactions

        - session_id: as a common identifier for all the interactions
- Succubus and Playsoft (Meta-game):
    - Traces described in the publication "Integrating learning analytics into a game authoring tool", section 3 - http://www.e-ucm.es/drafts/e-UCM_draft_315.pdf

Game authors or game developers should ensure that the required traces for the Learning Analytics Model (LAM) are sent from the game (as described in Section on Learning Analytics Model). These traces should provide all the required data that may be needed during analysis. The analysis will not be able to retrieve any meaningful information unless the necessary data has been received from the game, or if the analysis is not correctly defined. Traces should follow the Experience API for Serious Games Model (xAPI-SG). The above extracts are partial, and full LAMs will be needed for each game.

F.3. FROM THE USER MANAGEMENT SYSTEM

To allow the Analytics System to maintain information of every single user together, a unique user identifier in the BEACONING framework needs to be defined. This unique user identifier should be maintained across all games, so the information of a single user can be adequately aggregated. On section “Analytics workflow proposal”, it is proposed an analytics workflow where the LA framework is notified by the Beaconing User Management System using a Beaconing webhook.

F.4. TRACKING CODES

Section “Analytics workflow proposal” proposes an analytics workflow for notification between the Teacher UI and the LA framework. In the proposal, when GLPs are assigned to a class, group or students, the Beaconing webhook will notify the LA framework. A decorated GLP JSON file, containing the tracking code for each of the nodes (mission, quest, activity, etc.) is returned to the BEACONING framework using the analytics webhook. Each of the tracking codes will send information to a specific bucket in the analytics server. Activities need to be informed of the tracking codes that they need to use, and must supply the” correct tracking codes, or their input will not be available to analytics.

F.5. TEACHER DASHBOARD

A teacher dashboard (D4.5) is already available outside the Analytics System (of course the data is provided through an API from the Analytics System); all the visualizations that compose that dashboard were described in detail in *D4.6. Game analytics and adaptation design*. However, to correctly improve the current set of teachers’ data required for visualizations, specific requirements from other partners will be needed. We expect that these requirements will be subsequently updated and refined in the upcoming pilots.

F.6. STUDENT DASHBOARD

On Section “Incorporating analytics to a game” it is described how the Analytics API is to be used so games can access the required data and, with that data, provide the student dashboard.

In the description of action, partners agreed to display in-game visualizations and not to display a student dashboard because, as described on Section “Incorporating analytics to a game”: "Getting in-game reports rather than having to open browser windows to an analytics website avoids breaking the game-flow for students - although analytics certainly supports both".

Student Dashboard API is ready for using it and it is located in

https://analytics.beaconing.eu/api/proxy/gleaner/data/glp_results/:activityId/:studentId

The *activityId* is obtainable from glp.analytics.json.analytics.activityId. All requests need to be authenticated. Take into account that the API will return 404 errors either if the studentId is missing or *activityId* data have not been populated yet.

According to the final student dashboard definition, it is divided in 3 sections:

1. **Competences**: the competences achieved for the student are to be displayed using a 3 element spider chart (one element for each of the three competences agreed on), which will show how much of each competency the student has got along the gameplay.
2. **Performance** : the performance is calculated in three categories: score, speed and accuracy. Those will be displayed using 3 dial charts, one for each performance category. On the lower part of each dial, an adjective should be displayed that qualifies the performance obtained in each category according to the values specified below.
    a. Score: always displayed in a range between 0 and 1; compared with average, score can be: High or Low.
    b. Speed: time taken to complete the GLP (in seconds); compared with average, speed can be: Fast or Slow.
    c. Accuracy: number of times the minigame has been passed divided by the total number of tries; compared with average, accuracy can be: Accurate or Sloppy
3. **Minigames**: the last part displays a row for each minigame, and in each row, three visualizations are displayed containing information about:
    a. Score: shows your score compared with the average score of the players in your class that have completed the minigame.
    b. Time: shows your time compared with the average time of the players in your class that have completed the minigame.
    c. Attempts: shows your number of tries compared with the average number of tries of the players in your class that have completed the minigame.

Data provided by analytics

```
{
	"competencies": {
		"communication-and-collaboration": 0.1,
		"problem-solving": 0.3,
		"information-fluency": 0.7
	},
	"performance": {
		"score": {
			"own": 0.7, "min": 0.2, "avg": 0.6, "max": 0.4
		},
		"time": {
			"own": 3403, "min": 2340, "avg": 3045, "max": 4340
		},
		"accuracy": {
			"own": 0.5, "min": 0.2, "avg": 0.6, "max": 0.9
		}
	},
	"minigames": [
            {
                    "name": "minigame 1",
                    "score": {
                            "own": 0.8, "avg": 0.5
                    },
                    "time": {
                            "own": 45, "avg": 34
                    },
                    "accuracy": {
                            "own": 0.33, "avg": 1
                    }
            },
            {
                    "name": "minigame 2",
                    "score": {
                            "own": 0.6, "avg": 0.55
                    },
                    "time": {
                            "own": 124, "avg": 167
                    },
                    "accuracy": {
                            "own": 1, "avg": 0.5
                    }
            }
    ]
}
```

F.7. GLP DEFINITION

This section attempts to provide a complete specification based on our current understanding, and factoring in feedback from a previous version of this document.

As the GLP is a tree structure-based game definition, the activities created in order to analyse what happens along the gameplay will require some information to be propagated between child and parent nodes: we refer to this as propagation / knowledge update. The specification spreadsheet does not describe when the propagation occurs; this is addressed by defining a run-time model which describes the exact steps to take in response to user actions. UCM has actually built a working run-time proof-of-concept to test different hypothesis on how the GLP will work, and document our findings here.

Regardless of GLP implementation choices, it is important to make the analysis and the game sequencing mirror each other, such that, for example, if GLP conditions on a node make the analysis mark it as failed, the same conditions must trigger the restart of the node inside the game to avoid incoherent results. Analytics is not sequencing, and cannot control what nodes are visited when. Therefore, even though analytics may detect that certain nodes have been "failed" and cannot be completed successfully (for example, due to excessive attempts; or because a sub-node needs to be repeated), analytics cannot force nodes to be avoided or repeated. The only choice for analytics is whether to keep or discard incoming data.

Proposed changes and clarifications to the GLP spreadsheet

From feedback for the previous version of this document, we understand that:

- In the Score Distribution sheet,
    - Number of Failures => Number of Attempts
    - Good Answers Total => Score
- Activities return scores and maximum-possible scores when completed. The scores are used to contribute, using teacher-defined weights found in the GLP (see Score Distribution, Math Example) towards the activities' Objectives (either Learning Objectives or Competencies), and the contributions are then termed Activity Points. So a score of 10 in an activity that contributes 30% towards LO1 and 70% towards LO2 would yield 3 Activity Points in LO1 and 7 in LO2. This calculation to obtain Activity Points from score and competencies percentages is done from the analysis side.
- Scores for activities (and **per-activity activity maximum scores**, needed to calculate percentages of maximum) can be propagated to ancestors, allowing their success to be calculated along with other Objectives (LOs, CSs).
- An attempt refers to the time during which a student tries to complete a node. Only one activity (together, always, with all its ancestors) may be attempted at any given time.
- Failing nodes does not prevent knowledge propagation.
- The formula used for level of success and/or failure can only be modified in the specific cutoff percentages for each level.

When to propagate objectives

There are two main options regarding propagation. We can either

- immediately propagate any new activity points at a given node to all ancestors of this node. This results in constantly-updated GLP-level knowledge, but also results in counter-intuitive outcomes, such as GLPs being considered successful even when no child nodes have actually ended successfully.
- wait until nodes are considered to be finished (and their success can be measured) to perform propagation. This is linked with the "do failing nodes prevent propagation" issue: if failing nodes prevented propagation, then propagation would only occur once sufficient data to conclude failure/success is available. Since failure does not prevent propagation, the question remains open: regardless of failure state, we can still wait until activity end before we propagate objectives. This would make objective updates more blocky and stable.

Unless instructed otherwise, UCM will perform **immediate propagation**. UCM will only delay calculation of possible bonuses and success/fail evaluation until end of attempts.

Repeated activities

When nodes are repeated, we can either

- discard any previous activity data, and revert any contributions to objectives from the affected node and its descendants. This is easy to implement and understand, but may result in impossible-to-succeed scenarios, especially when coupled with time or repetition constraints.
- keep previous activity data around, and when the repetition ends, decide whether to keep the old or the new attempt. Criteria on which to keep would need to be provided.

Unless instructed otherwise, we will **discard** any previous data as soon as the repetition starts.

Time limits

As time limits are defined in the GLPs, they are only present at activity level. As the tree structure allows to set time limits for each node, we will consider this possibility in order to increase the flexibility of the system.

Time limits can either refer to total time in an node, including sub-nodes; or total time in the current attempt on the node. For example, if a quest has 3 mini-games, the goal may be to finish all three in less than 5 minutes. However, it may make sense to allow the player to retry all three. Should time limits be per-attempt?

It was agreed with partners that the time limit is the sum of the times of all activities. Time limits are especially problematic in a hierarchy. If the student spends too much time on a sub-activity, it is possible to blow the time-budget for the whole hierarchy. And it would then no longer be possible to complete the hierarchy in time. For example, if Node A has children B, C, and D, each with 5-minute limits, and the student spends 20 minutes on B, therefore failing it, then any actions on C and D are irrelevant for success, since A will never be completed "in time".

Note that time limits in this context is used to evaluate pass/ fail status for the GLP activities (quest, missions, etc.) in the context of the analytics system and teacher feedback to do the aggregation. Using this information to modify the game (preventing the student from playing the game) it is out of scope of analytics: students will always be able to play the whole thing, as analytics is powerless to prevent them from doing so. They will, however, be considered to have failed the corresponding nodes.

There are certain bonuses for successful completion that depend on time limits. What time limits should be considered for such bonuses, if none have been specified? There are two obvious choices here:

- Apply no time-related bonus unless the node has a set time limit
- Apply maximal time-related bonus when no time limit was set

Unless specified otherwise, we will **apply no time-related bonus** unless there is a time limit present.

Level of success and bonuses

Success levels between 50% and 70% are considered "partial", and over 70% as "full". The thresholds are applied to scores, learning objectives, and competences. However, it is unclear whether

- each and every learning objective and competence must pass the 70% (or the 50%) threshold to be considered full (partial) success
- it is sufficient for the average of learning objectives, and then the average of competences, to pass each threshold; and individual learning objectives and competences are not examined.

Unless specified otherwise, we will consider that **all individual** learning objectives and competence must pass the corresponding thresholds.

Activity suspend and resume

Currently there is no support for suspend and resume within the analytics framework. A student logging back in after a gameplay session will be understood to be beginning a new attempt, and old attempt information will be discarded. If this is the desired behaviour, there is nothing to change.

However, if the GLP and games support some kind of save and restore feature, then we would need to define specifics regarding how sessions are restored from an analytics point of view. The simplest way would be for games and activities to request the analytics state at the point of suspension, and make a request to restore it on resume.

Node starts and ends

As our analysis is defined (inspired by SCORM 2004 4th Ed. Sequencing and Navigation), only one node in the tree is active (together with all its ancestors). However, the propagation of events to higher-up nodes can only occur when these have already been initiated, so as not to cause an incoherence between the beginnings of each of the attempts of each node.

For instance, consider the case where a student wants to start activity 1 of quest 1 within mission 1. In order to be able to access the activity, the student must first start mission 1, then start quest 1 and, after that, the student will be able to start activity 1. All these nodes can start arbitrarily close in time, but it is important to respect the order in which they are opened.

We also expect nodes to be closed in reverse order: children should be closed before their parents, and all non-common ancestor nodes should have been correctly closed in this way before activating a different node.

Suggestion for GLP definitions

We have modelled the flow of analytics on a proof-of-concept implementation to better understand the current GLP model. As a result of this implementation, we propose that every node in the GLP include a JSON structure similar to the one below, defining the limits (including thresholds) and the contributions of this node.

```
{
      // from the GLP: limits; if absent, full-success is implied
      limits: {
             "maxTime": 120,
             "maxAttempts": 3,
             "partialThreshold": {
                    "learningObjectives": .5,
                    "competences": .5,
                    "scores": .5},
             "fullThreshold": {
                    "learningObjectives": .7,
                    "competences": .7,
                    "scores": .7},
      },
      // from the GLP
      contributes: {
             "learning-objective1": 1,
             "competence1": .25,
             "competence2": .4,
             "competence3": .25,
             "competence4": .1
      },
      // set during analytics
      active: false,
        success: 0,
        completion: 0,
        startTime: 0,
        endTime: 0,
        elapsedTime:0,
        attempts: 0,
        results: {}
}
```

Internally (within the Analytics System), the results object will contain score, max-possible-score, and activity points for each objective (learning objective or competence) earned in the current node and its descendants.

Mini games open points.

Competences and Learning objective tags to be authored by the teacher

The GLP already provides, in their JSON schema, a couple of fields (topic and subtopic) that the teacher can author, along with a bundle of data for mini-games. If those 2 fields will be used the state of the art is the following: "*topic*" is a drop-down menu with values listed in the JSON schema, while "*subtopic*" is a free text field. The AT provides all the possible "*topic*" values to fill in the drop-down menu.

Decision taken from the success full or partial of an Activity inside a GLP.

The state of the art is that each mini-game sends back to the caller component (meta game or another activity like a location-based activity) a success true or false if the successPoints treshold defined for each mini-game has been reached or not.

In order to check the limits defined in the AT, results from mini-games are sent to the LA where decision are taken regarding progress and success.

GLP node Meta-Data needed for multi-level Analysis

```
analytics : {
        "limits": {
                "maxTime": 120,
                "maxAttempts": 3,
                "partialThreshold": {
                        "learningObjectives": 0.5,
                        "competences": 0.5,
                        "scores": 0.5
                },
                "fullThreshold": {
                        "learningObjectives": 0.7,
                        "competences": 0.7,
                        "scores": 0.7
                }
        },
        "contributes": {
                "learningObjectives": [
                        {
                                "name": "learningObjective1",
                                "percentage": 0.4
                        },
                        {
                                "name": "learningObjective2",
                                "percentage": 0.3
                        }
                ],
                "competences": [
                        {
                                "name": "critical-thinking",
                                "percentage": 0.2
                        },
                        {
                                "name": "problem-solving",
                                "percentage": 0.6
                        }
                ]
        },
        "trackingCode" : "5a96c04319b95500764797902ieooikvww7"
}
```