Gaming Learning Analytics for Serious Games

Cristina Alonso Fernández

DOBLE GRADO INGENIERÍA INFORMÁTICA – MATEMÁTICAS

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

TRABAJO DE FIN DE GRADO

2015 / 2016

Director: Baltasar Fernández-Manjón

Acknowledgements

This project would not have been possible without the guide of Baltasar, whose continuous help and dedication has been essential to progress with the project throughout the months.

Special thanks to Antonio Calvo, Dan Cristian Rotaru and Ángel Serrano, whose advice and work have been extremely useful for this project.

I would like to thank my classmates for their help in these last 5 years.

And finally, to my friends and family for their incredible support.

This project is especially dedicated to my parents.

Contents

Table of figures

Table of tables

Abstract

Serious games (or educational videogames) are considered an important tool for the future education. This is the reason that lot of effort is being put into the analysis of their correctness and suitability to achieve the intended educational goals. The field of Gaming Learning Analytics aims to provide tools to verify these characteristics improving the quality and effectiveness of serious games. For this purpose, three steps are usually required: 1), to track the data from the player's interaction with the game; 2), to analyze that collected data; and 3), to visualize the results. In this context, there are some key issues to consider: level of game knowledge, stakeholder of the final visualizations or amount and complexity of the data. These ideas are put into practice with two examples of serious games focusing on the last two steps of the process. Several analysis and visualizations are carried out with them considering the different issues mentioned before. Among the conclusion that may be drawn, it stands out that, although there are still a few issues to improve, Gaming Learning Analytics is an essential tool for many users with a wide variety of interests in serious games.

Keywords: e-learning, educational games, learning analytics, dashboard, Elasticsearch, Kibana

Resumen

Los juegos serios (o videojuegos educativos), se consideran una herramienta importante para la educación en el futuro. Por ello, se está invirtiendo mucho esfuerzo en el análisis de su corrección e idoneidad para alcanzar los objetivos educativos pretendidos. El campo de análisis de aprendizaje con juegos pretende proporcionar herramientas que verifiquen estas características mejorando la calidad y efectividad de los juegos serios. Para ello, se necesitan normalmente tres etapas: 1), monitorizar los datos de la interacción del jugador con el juego; 2), analizar esos datos recolectados; y 3), visualizar los resultados. En este contexto, hay algunos asuntos importantes a considerar: nivel de conocimiento del juego, receptor de las visualizaciones finales o cantidad y complejidad de los datos. Estas ideas se ponen en práctica con dos ejemplos de juegos serios centrándonos en las dos últimas etapas del proceso. Se realizan varios análisis y visualizaciones con ellos considerando los diferentes aspectos antes mencionados. Entre las conclusiones que se pueden extraer, destaca que, a pesar de haber algunos aspectos aún por mejorar, el análisis de aprendizaje con juegos es una herramienta esencial para muchos usuarios con una amplia variedad de intereses en juego serios.

Palabras clave: e-learning, juegos educativos, analíticas de aprendizaje, panel de control, Elasticsearch, Kibana

Motivation

The goal of this project is to study the use of learning analytics and visualization tools to extract the relevant information of the interaction of players (commonly students) with Serious Games.

In this context, much data may be collected while the student is playing. There are several options as to how to obtain that data, mainly depending on if it is us accessing the game to extract that information or if it is the game that is sending some information to the outside.

Whichever the way and format in which we obtain the data, once we have it, several analysis may be carried out on it. The most relevant for the learning purpose would be the ones that could provide some insight about the student's learning process. In particular, we may want to look for errors, places where students get stuck or levels of completion and achievement.

Once all the useful information is obtained and analyzed, it is also convenient that it is displayed for visualization so the students and teachers can easily understand it. There may be different visualizations depending on the context and purpose. In a class context, some information would generally be displayed for the teacher so he/she can know at *near* real-time how the students are doing and can help them to continue with their gameplays if necessary.

The students will usually also receive some final report with their results in the game, showing for instance how they did compared to the rest of their class or compared to the mean of all the game plays. This could also be displayed in the case that the interaction is by the students at home as it could happened for some assessment work. In this case, the teacher will also receive the relevant information after the play for marking purposes and to know how the students did.

The key problem to overtake in Learning Analytics is to be able to extract relevant information of the learning process of the students based on their interaction data, ultimately wanting to answer questions as tricky as "is the student really learning?" or "is the game helping the student to learn?". It is important as well for this purpose to give real-time feedback to the student and mainly to the teacher.

The ideal goal of this project would be to have an extensible software independent of the game. That is, a software that is able to perform some analytics and display some visualizations purely based on the signals that it receives from a game which it has no previous knowledge of. Ideally, this software would be configurable without having the need to program. This way, the desired analysis and visualizations could be specified and obtained in an effortless, clear and quick manner.

For this purpose, we must first stablish some basic agreements in the characteristics of the data tracking system. The particular way that we obtain the data in will deeply determine our possibilities both in analysis and visualization, as it will imply the amount of information to be received and how rich that information is. The analysis will be then developed based on that data obtained and their particularities.

The result visualizations should also fit the needs of different viewers: both for students and teachers, as part of the educational use of the game, as well as for developers or managers, who would be more interested in the proper functioning of the game but may also want to know whether the game is fulfilling its intended learning outcomes.

Although the focus of this project is on analysis and visualization, the examples developed could be fit into an architecture that controls the whole process of Gaming Learning Analytics.

Gaming Learning Analytics

Gaming Learning Analytics is the process of analyzing the interaction of students with serious games trying to obtain the relevant information about the learning process of the student. The ultimate goal of this analysis is to understand how a student learns something new and how we can design a game to help students achieve a higher outcome of their interaction with it.

The impact of games is increasing lately with new technologies such as mobile devices, which allow players to access games at any time and in any context. Games provide engaging and motivational content that challenges players, instead of forces them, to perform better. This has naturally increased the interest of games for educational purposes, trying to design games that provide good educational experiences pushing players to go beyond their current competence levels [Koster 2004].

Serious games are not new, but they have gained interested in the last years. Since their origins, they have proved to be educationally effective in several specialized domains (medicine, military or business) and teachers are now starting to use videogames for educational purposes as the students get more engaged on them rather than on the usual homework [Fernández-Manjón 2014a]. The generalization of Massive Online Open Courses (MOOCs) is another point in favor of serious games as they provide interactive content that engages its users. In long-term projects, videogames have also proved to be effective such as in the two year project of the game *PlayForward: Elm City Stories* that aims to make adolescents aware of the risk of the human immunodeficiency virus by teaching them skills and knowledge to prevent it [Fiellin et al. 2016]. This example shows how results obtained with serious games can be empirically tested (in this case, using community-based randomize controlled trials) providing valid and reliable data.

The use of games in education is also interesting as part of the now more common evidence-based education. While manual evaluation is often tedious and subjective, the use of educational technology makes a bigger amount of data available for its analysis through statistics, machine learning and different visualizations techniques. This could simplify the teachers' assessment tasks, but for that purpose it is first needed that teachers trust the games' effectivity in showing learning skills [Freire et al. 2016]. The availability of assessment, tracking or classroom management are some of the reason that could lead teachers to select a game to be used in classrooms, so providing a transparent evaluation system for education is key [Takeuchi and Vaala 2014].

Despite these good features, the educational community is adopting serious games slower than expected. The reasons for this are several. Serious games usually have high development costs and most of them are funded by research projects or governmental agencies. Teachers prefer the traditional evaluation methods as they find difficult to integrate other activities with their teaching in some cases due to a lack of adequate training, having also concerns on how to make games available for students [Takeuchi and Vaala 2014]. As videogames are still seen as non-learning tools, they are usually just a complementary content for motivational purposes with no actual impact on the final mark. But probably the main reason for this slow adoption is the lack of knowledge on how students interact with games and how their learning process actually occurs. This is also a consequence of the lack of existing tools to improve our understanding of the educational impact that games have on students [Freire et al. 2016].

These issues, however, are not unavoidable. Their development and maintenance cost is being reduced by the raising of tools such as Unity3D or Unreal that allow to develop high-quality games in an easy manner, and playable on multiple platforms without changing their details, isolating games from changes in technology [Freire et al. 2016]. Games' deployment in educational settings is also eased by new technologies such as HTML5 which allows games to be played on modern browsers regardless of the operating system and with no further software installation. The previously mentioned extended use of mobile devices also facilitates this, as most players could play games on their own game devices (laptops, smartphones or tablets) without the need for specific laboratories [Entertainment Software Association (ESA) 2015]. Teachers should also be provided with further insight into how students play and learn as well as with supporting tools so they can know what is happening during a game session in the classroom.

The lack of knowledge of these issues could lead to misinterpretation of the data. Teachers' quality is sometimes being measured purely based on the results of tests that do not necessary show neither if the student is actually learning nor if the teacher is good or not. This is an incorrect use of the results obtained from analyzed data that does not take the whole picture into account [Bowe 2016].

Usually, games that are used with educational purposes are simply computer tests in the usual Q&A format acting as message broadcasters to disseminate information. Basic information collection has always been done with Learning Management Systems, providing some insight into students' actions. Nevertheless, actual videogames can also provide lots of information about the students' learning process. As games are more interactive than usual static learning materials (e.g. text documents, slides, quizzes) they have the potential to reveal much more information through the students' interaction with them. Instead of simply analyzing the final scores, the detailed gameplay can provide more valuable information of how players actually interacted with the game, where they got stuck or where they found problems to understand a key concept [Freire et al. 2016].

In order to be able to analyze this information games should no longer be a black box and instead allow the relevant data to be obtained for its later analysis. This can happen either with a white box game that allows access to the game internal state or if the game sends the relevant data to a remote server using some logging framework [Fernández-Manjón 2013].

In this whole process, it may be necessary to work with vast amounts of data (what we usually refer to as *big data*) captured from users behaviors. However, in contrast to what happens in other areas of study such as publicity or economics, for Learning Analytics it is not so important to have a huge amount of data but to have *useful* data, i.e. data that actually shows if the student is closer to achieve the learning outcome of the serious game [Serrano-Laguna et al. 2012].

The key point is not to have a large amount of data but to be able to clean up the data, figure out what useful information is in there and more importantly how to use that information. For that purpose, it may be necessary that the collected data is then revised by a human to add the actual meaning to that tracked information [Bowe 2016]. Still, a vast amount of data may be generated even by short gameplay sessions as game interactions typically comprise a very short feedback cycle of interaction-reaction [Van Eck 2006].

In education, two specific areas of big data are educational data mining and Learning Analytics, whose distinction is not straightforward but that have had different research histories and are developing in different research areas. Educational data mining tends to focus on *developing* new tools for discovering patterns in data, generally about the micro-concepts involved in learning.

On the other hand, Learning Analytics focuses on *applying* known tools and techniques at larger scales, such as in courses and at schools and postsecondary institutions [Bienkowski, Feng, and Means 2012]. Both disciplines work with patterns and predictions, but Learning Analytics aims to understand entire systems and to support decision making. Learning Analytics also draws on a broader set of academic disciplines than educational data mining, as it incorporates concepts and techniques from such different disciplines as sociology, psychology or statistics.

It is key that Learning Analytics is a data-driven process. As an example, the use of online LMS provided some data by consulting the access logs from the web-server [Arnold and Pistilli 2012]. The analysis of those logs could identify behavior patterns that correlate with academic results. This simple issue may still need to deal with big data, and so intelligent data-mining techniques may be able to detect less obvious correlations [Ferguson 2012]. Once we have the data, from the student side self-assessment and motivation are important in online courses, as well as to give some information comparing with others' results. Teachers would need to track progress made by students to adjust both the speed and contents of their courses and to have information for grading [Freire et al. 2016].

While Learning Analytics focus on education, Game Analytics is the application of analytics to game development and research to understand better how users play, find errors and improve their play experience [El-Nasr, Drachen, and Canossa 2013]. The collected data may be considered from the technical view to show number of code bugs or time to fix them, or from the user's perspective to show customer metrics (purchases or downloads within the game), community metrics (interactions of user in forums or social media) and game metrics (purely the interaction with the game). Game developers will also be interested in a variety of information of the game such as knowing if levels are excessively challenging or too easy, identifying potentially unreachable areas or conflicting game states, or detecting popular areas of interaction [Freire et al. 2016].

We define Game Learning Analytics as the combination of the educational goals of Learning Analytics and the technologies from Game Analytics. This field aims to understand how games affect education and training. Although both Learning and Game Analytics use different vocabulary and have their particularities, they both look for games with a better user experience. The data analysis and visualization techniques could be then used for several purposes inside Gaming Learning Analytics such as predicting learners' performance, provide them with personalized learning experiences or improve the design and cost-efficiency of serious games [Freire et al. 2016].

Gaming Learning Analytics is also useful for game design and testing. For designers, it can show how users play and their learning strategies, helping them to fit the preferences of its target audience and their different play-styles. For the reliability of the game, the information extracted could show how a concrete tool improves the students' knowledge but also its effectiveness [Freire et al. 2016].

Usually, to convert player-generated data into analytics, a two-step process is needed:

1. Tracing the actions of play-learners while they interact with the game as evidence of their cognitive process, skills and abilities.
2. Analyzing the actions sequences obtained by way of statistical or machine learning.

After the analysis step, it is both usual and convenient that this process is completed with some visualizations of the results obtained.

But before the very first game design and implementation, it is essential to make some educational goals definition of what the designers want the players/students to achieve in terms of new knowledge acquired while playing the game. This intended educational outcome should lead the development of our whole process of Gaming Learning Analytics as it would determine the analysis that we would want to perform on the data as well as the results that we would want to display in our visualizations.

As a first approach, we may have this general overview of the process:

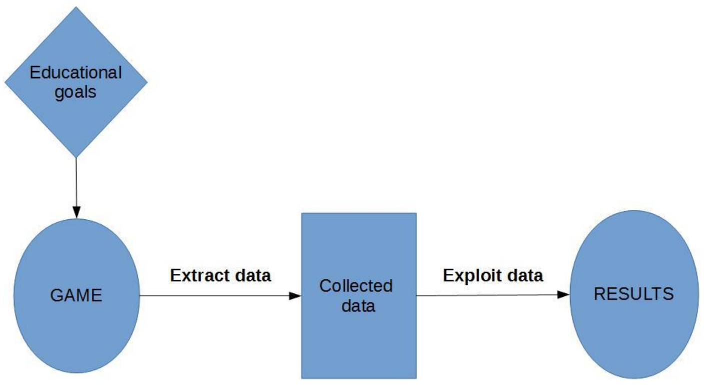


Figure 1. GLA process overview

Data collection

The first step of the process of Learning Analytics ("LA") is to obtain the data that we will analyze later. It is important to first know what data is useful to track, and this would depend on what results we are heading towards. A first approach may be to consider all kind of possible learning activities and track who is performing them, and what happened before, during or after the activity. Not all activities may then be tracked but this could be a good starting point to decide what data should be tracked [Bowe 2016]. It is best at first to include all kind of activities, even the most trivial ones (e.g. logging or logout), as they all may have some utility from the analytics point of view, before going into how to implement data capturing [Gilbert 2015a].

The mining of information is usually made in an undercover way so that the student/player does not realize about the process happening underneath the framework of interaction of the game. Game telemetry is the fundamental element from which measures of player performance are developed [Loh, Sheng, and Ifenthaler 2015]. This mining tool collects data from the user's interaction in the background during a game session and sends it to a remote server for its storage and later analysis.

The collection of data requires a high availability and bandwidth service that guarantees that all incoming traces will sooner or later be processed. This should work together with an infrastructure that classifies and anonymizes the received traces [Freire et al. 2016].

The in-situ collection of data, made inside the system, eliminates subjective data. Once the actions have been captured, it also allows researchers to retrace the actions players performed within the game and visualize their navigational paths. Data can be generated either based on events or by continuous sampling. Events like start, end or quit game traces, as well as phase changes are typically captured events [Fernández-Manjón 2012]. Event listeners are essential for this purpose, along with event tracers that take note of the events that got triggered in the game system.

This real-time analysis of data while the system is in use allows to improve and adapt the learning experience dynamically. Providing real-time information is especially useful if we want to use games in actual face-to-face classes, as this would help the teachers to know that is happening in a classroom when games are used [Serrano-Laguna and Fernández-Manjón 2014]. In that sense, it is recommendable that the students do not know that LA is in place, as they behavior may change making it more oriented to achieve success and less to explore the game.

However, real-time analysis is not the only way to obtain players' information. Offline analysis can discover patterns of use and allow to improve the experience for the future too [Fernández-Manjón 2013]. This offline analysis may not be so directly useful for teachers or students but instead for developers or designers of serious games as they may learn information that helps the improvement of the game as well as their knowledge for future projects.

Again at this stage it is important to recall that collecting big amounts of data is not as important as collecting actual relevant and useful data. Capturing data with low information value may be the result of a poor design of the telemetry system. The ultimate goal is to learn what kind of play-learners actions or behaviors lead to an increase in their performance.

Moreover, as we internally know the structure of the serious game provided, we may have different approaches on the data that we want to collect depending on that structure. That is, depending on the particularities of the interactions that we want to track, we would need to track them with different levels of detail to then obtain the corresponding different levels of information [Bowe 2016].

It is also important to note that not all gamification concepts are appropriate for learning and training. For instance, speed is a usually key measured value in games that in the context of serious games may lead to the player rushing to finish and therefore could be negatively correlated with performance [Loh, Sheng, and Ifenthaler 2015].

The tracked data will then be stored, usually in a Learning Record Store ("LRS"). The LRS could be seen as an application to handle the reception of learning events. This generally includes performing operations such as data validation, data normalization, transactional or long term persistence, and maybe even some processing already related to analytics. The features to consider when choosing a LRS should include its scalability, storage, message format (following a particular standard or not) and data security [Gilbert 2015b].

After its storage, the tracked data should be examined and cleaned. This data cleaning process may comprise removing duplicates or filling any missing data, among others. Once the data is cleaned, it may be exported to a file; usually a simple flat file like a comma separated value ("CSV") file. After that, it could be imported into any suitable statistical program for its later analysis [Loh, Sheng, and Ifenthaler 2015].

Different metrics may be performed on the data but there is still little guarantee of which ones are most suitable for serious games. Typically used metrics include time of completion, number of kills (if it makes sense within the game), amount of points collected, score or experience points gained. Rate achievement may be a more significant metric as it gives the ratio of missions achieved or levels completed compared to the time period.

Data analysis

The analysis process is the key task for Gaming Learning Analytics ("GLA"). Starting from the data obtained from the user's interaction with the serious game, GLA aims to extract the information hidden in that data that may reveal some information about the learning process of the student. For that learning outcome, several different analytics can be carried out through that data and it is not clear which of them best shows the learning process of the student.

The analysis is typically performed by a learning analytics processor. This manager tool consists of three phases. In the first one, the input phase, it would extract data from one or more resources, which will include the LRS. Other sources could be a student information system or a learning management system. This data would be then brought together, transformed and loaded into the processor in the appropriated formats. In the next phase, the analysis itself, the analytics model would be executed on the stored data and it could support either academic analytics or predictive analysis. Finally, the output phase could include, among others, aggregate the results of the execution or perform some other calculation with them. It will typically include persisting the results to some files or a data store too, and often the processor will expose the model output via web service APIs [Gilbert 2015c].

Both general and game-specific analysis may be performed. Some of the general analysis that can be considered to make are the number of students/players that reached each level of the game, the mean time to complete each level or the final ranking of each student compared to the rest of the students in the class (in a classroom context) or to the rest of the tracked history of the game.

In terms of game-specific analysis, the particularities of the analysis should be specified in each single case of serious game, taking into account its characteristics and its particular expected learning outcomes. Both of these analysis can either be performed to the data of a group of players (usually a class) or to a single player's data to obtain more detailed information of his or her particular session. This could include information such as progress in the game, specific errors made or core learning skills gained.

Player's characteristics play an important role, so player behavioral profiling may also be useful. Using both data mining and behavior categorization techniques, user-generated actions can be aggregated into patterns. This patterns are not so easily detected by traditional methods and can be used to later develop player profiles. Using clustering techniques, general profiles can be obtained. These may be representative of certain groups of players/learners [Loh, Sheng, and Ifenthaler 2015]. Users' classification will not only depend on gender or age but also on many different demographic factors such as socioeconomic status [Sánchez and Olivares 2011], cultural differences [Guillén-Nieto and Aleson-Carbonell 2012] or gaming profiles [Manero et al. 2016]. Their results could then be compared and contrasted with the help of GLA.

Based on this information, researchers may predict future users' actions or behaviors and further confirm these predictions using supervised learning. This is particularly useful to prevent situations of students' failure when a misleading pattern is detected. The domain knowledge gained from this approach can be used to inform the design and adaptation of a particular game in order to provide the player with a personalized and adaptive environment [Loh, Sheng, and Ifenthaler 2015].

These predictive analytics require a solid content strategy so we can ensure that when we observe certain behavior X in a student/player we know that certain result Y is likely to happen. Based on that information, we could then provide adaptive flows or recommendations among other strategies [Bowe 2016].

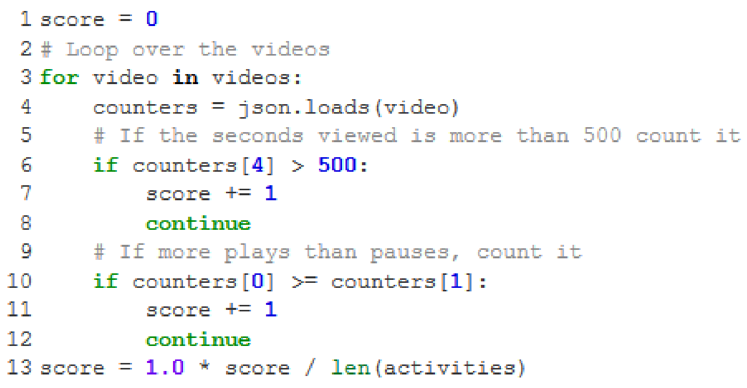
Data visualization

After the analysis is completed, visualization of results is essential to provide a better way of understanding those results. A typical use of it would be to give real-time feedback for students and teachers during a class. The analytics are now transformed into appropriate graphical forms for easy communication and discussion as the students/teachers need not to understand statistics or any other specific mechanism used during the analysis step.

This visualization of analytics frequently takes the form of dashboards for easy communication with stakeholders. They could provide a wide range of information from general overview of key indicators such as completion, grades or educational effectiveness to more game-specific information for developers. They would ideally be configurable to suit the user's needs, for instance to allow new analyses to be set up and launched at real-time.

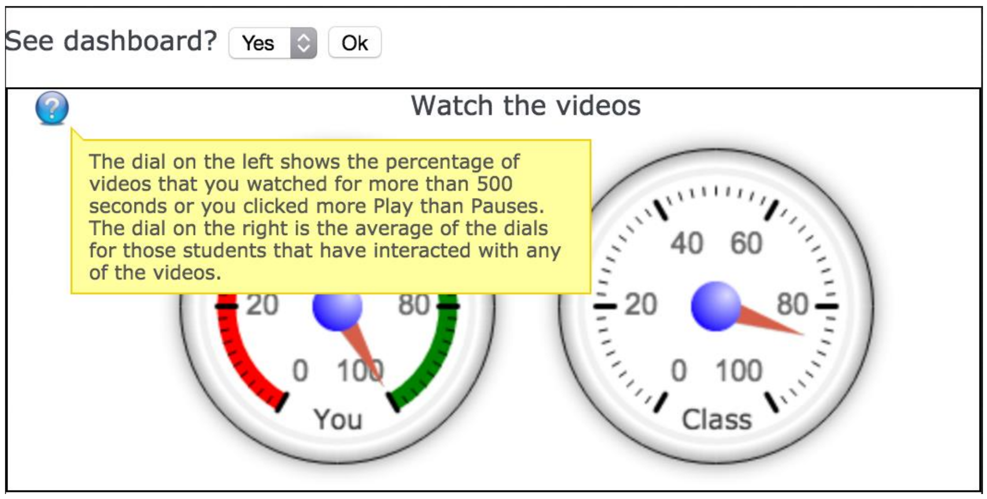
The information displayed in dashboards needs to be clear enough to allow stakeholders to track its origin, i.e. to clearly understand why those results are obtained and no others. This transparency should be a key design criteria, as it should bring the algorithms' decisions into visualization.

For instance, if we have the following code to calculate the engagement of students with course activities of video watching, the dashboard below could provide the required information maximizing algorithmic transparency [Pardo 2016]:



```
score = 0
# Loop over the videos
for video in videos:
    counters = json.loads(video)
    # If the seconds viewed is more than 500 count it
    if counters[4] > 500:
        score += 1
        continue
    # If more plays than pauses, count it
    if counters[0] >= counters[1]:
        score += 1
        continue
score = 1.0 * score / len(activities)
```

Figure 2. Visualization example: code view



The dial on the left shows the percentage of videos that you watched for more than 500 seconds or you clicked more Play than Pauses. The dial on the right is the average of the dials for those students that have interacted with any of the videos.

Figure 3. Visualization example: dashboard view

This is a simple example of algorithm that basically consists of simple calculus, but in general the algorithms would be more complex so the dashboards' explanations could not be directly obtained from them in such an easy way. However, it is important that we consider these issues at the early stages of the design process so they can be taken into account and then the results' visualizations do contain enough information to be understandable.

Different types of visualizations will usually be necessary depending on its final target:

- For students/players, the main visualization or dashboard will usually contain informative feedback of their personal game performance, for instance, their final score. It is also usual to display some information to compare their results to the ones of other players: ranking compared to their classmates or to the historical plays of the game.
- For teachers/educators, visualizations should include general statistics about the results in the class scope: how many students finished each level of the game, how long it took them to finish or most common mistakes made by the students. They

may also want to go in depth into concrete and specific information of a single student.

- For game designers or developers, general statistics about the use of the game will be required to give a general overview of the game use. Specific information like places where most mistakes are made would usually be useful as well, as they may show a defect or a spot for improvement in the design of the game.
- Managers of the game could be most interested in knowing the peak times of use of the game, to be able to predict them in the future and control the use of the resources of the game.

The information collected may be used for different purposes, again depending on the stakeholder. From the designers' perspective, it could be useful to improve the game design and quality to make the game achieve with greater accuracy the learning outcomes desired. For students, it may lead to personalization and adaptation of each student's learning environment to better suit their necessities. Teachers could also benefit from this information either for formative or summative assessment or for predicting the best course of action in the future. Educators will usually have complete information of initial conditions of the students and their final results so they will be able to track what works and what does not work, and adjust accordingly.

It is also important to notice that different stakeholders would have different levels of access and anonymization, e.g. students should only receive their personal information (maybe together with some metrics that compare their personal results to those of the rest of the users in the class/game history), while the teacher could receive all the detailed information of every student. Also, developers or managers will receive the information of all the game sessions but these will be usually anonymized as they do not need to know the particular details of a user session but the aggregated results of all the users [Freire et al. 2016].

The main three steps previously described as part of the GLA process may be seen conceptually in the high-level diagram below:

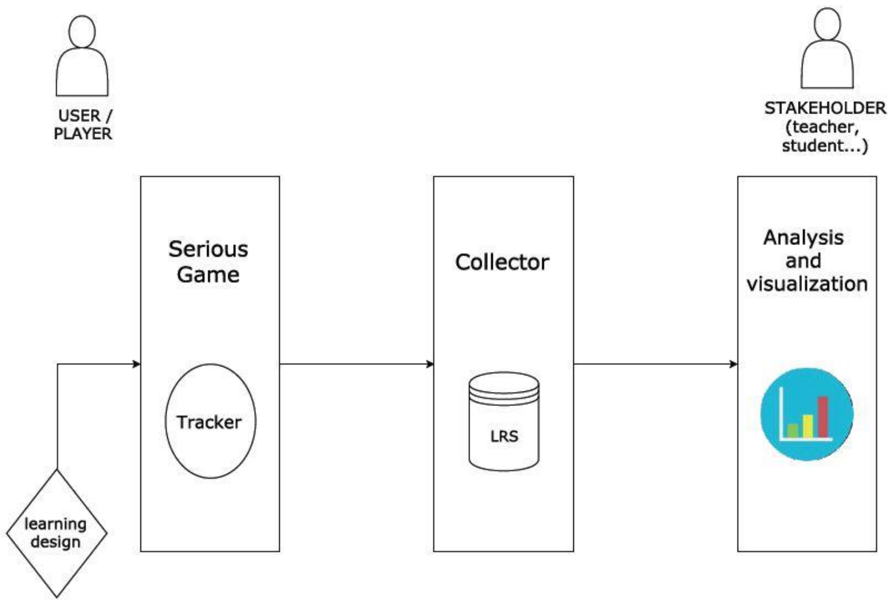


Figure 4. GLA high-level diagram

In the following we would look in detail at tracking and collection tools as part of a complete GLA architecture, but still this project has its main focus on the analytics and visualization part. In particular, we will work in the examples taking into account the different possible stakeholder of the results. This would in most cases be the teacher, as they are the stakeholder that best understands the learning process of the student which is the final aim of LA. Knowing how the students are doing in the gameplay, teachers will be able to both help the students having trouble with the game not to get stuck and continue playing and also provide new tasks for fast learners who have already completed the given task. However, the other stakeholders mentioned (student, developers or managers) will also be considered and dashboards developed to fit their particular needs.

Although the previous scheme may seem clear, it is important to notice that the whole process of analytics and visualization deeply depends on the level of knowledge that we have of the game. That is, different levels of knowledge would lead to different analysis and, therefore, different result visualizations. Consequently, if we know the complete game model we may perform more sophisticated analytics taking it into account and hence our visualizations will then contain broader and deeper information.

However, we would like to be able to perform some analytics on a serious game without having to know its complete model. In this case, assuming that we cannot access the game as a white box, the only information that we would have is the one provided in the traces sent by the game. It is those traces what we would have to analyze trying to obtain information as broad and rich as possible. This shall then be done without knowing the internal structure of the game and therefore we would usually be able to obtain less information than with the previous approach. The variety of the information obtained will then deeply depend on the format that the traces sent by the game follow, and how rich and wide information that format allows to track.

It is still interesting to see how far we can get with the latest approach as the real long-term aim will be to display an analytics and visualization tool independent of the game, which means that we will not have any information of its internal model. With no further data, we will have to obtain all the information to analyze and visualize by means of the traces that that particular game sends. Although our analysis and visualizations will necessarily be more general, we could still provide some visualizations by default that comprise the relevant information that we may obtain from the game traces. These visualizations may also be suited to the different stakeholders previously mentioned.

A more conceptual scheme that captures these ideas may be seen in the graph below.

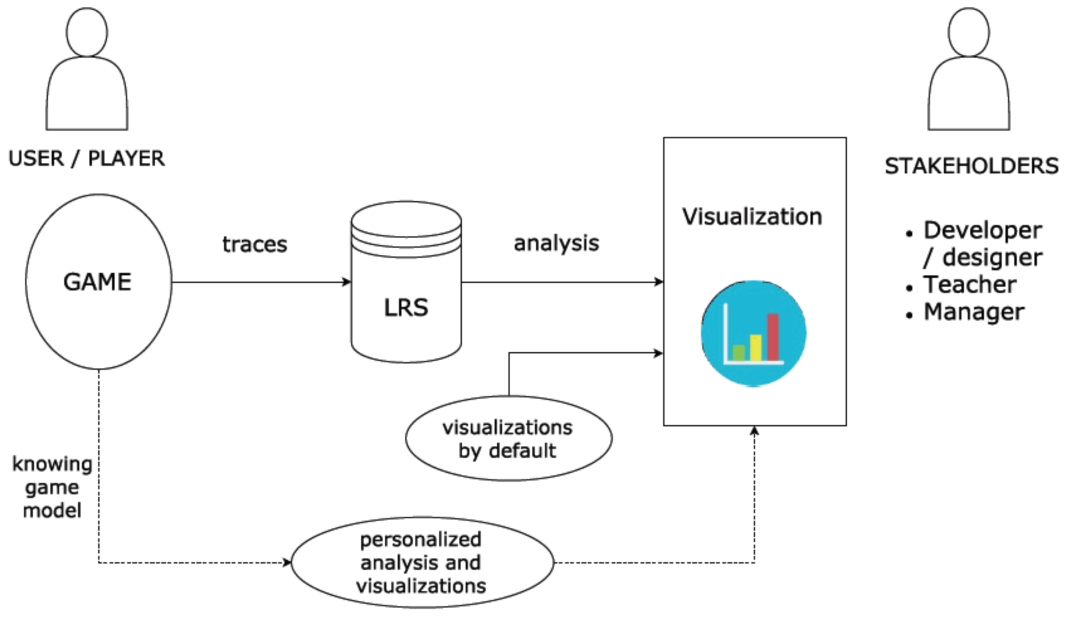


Figure 5. Conceptual scheme of GLA

Current state of Learning Analytics

There are still open issues in the field of LA. For instance, it could be relevant to determine how to automatically distinguish mistakes derived from game exploration from mistakes derived from actual conceptual errors. Also, traces across different types of games or long term learning across games are issues that still need to be explored in greater detail [Fernández-Manjón 2014b].

It is also important to consider some issues such as:

- Ethical and legal aspects.
- Security issues.
- Ownership of information (which we have to inform the user of).
- Anonymization of information.

All those aspects become especially relevant if we are working with kids which will be commonly the case in LA or if we are working in the medical domain [Fernández-Manjón 2013].

Although there are still some problems to overtake, the general perception is that serious games will play an essential role in the future of education so LA is an essential tool to help in its improvement [Serrano-Laguna et al. 2014]. In that sense, it is key to ensure that LA is informed by pedagogy and that support is given for research into the efficacy of LA and into the appropriate pedagogies for it [LACE 2016].

A first example of the work that is being currently done in the field of GLA is the web service GameAnalytics.com [GameAnalytics 2015]. This tool for game designers aims to provide them with a wide variety of already defined dashboards to monitor what players are doing with their game at real time. It has basic free plans and is available for many platforms such as iOS, Android or Unity. It also allows to develop personalized dashboards, obtain the raw data or custom the interactions to track.

An example dashboard developed with GameAnalytics.com is the following:

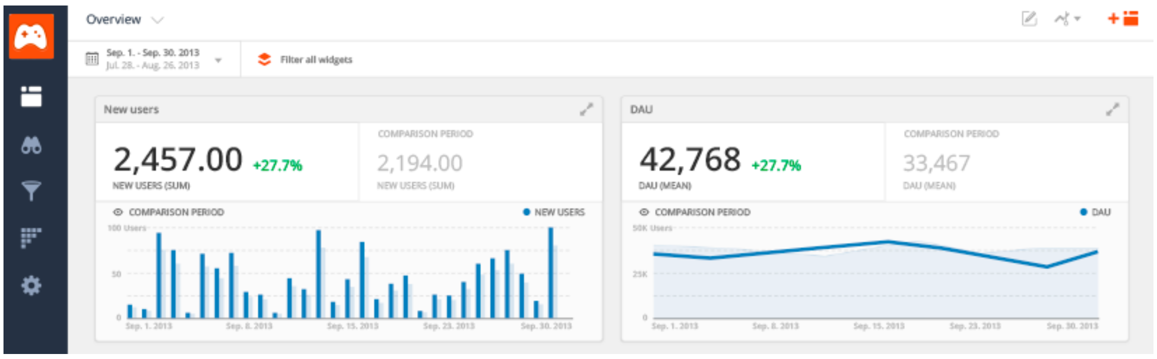


Figure 6. GameAnalytics' dashboard

The current development in the field of LA has raised the interest of several companies and developers. **Apereo** and its Learning Analytics Initiative ("LAI") [Apereo 2016a] is one clear example of it. This initiative aims to create an Open Learning Analytics infrastructure that supports the whole process from data collection to summary and results communication. With their Learning Analytics Processor [Apereo 2015], they aim to manage the whole process from the "academic early alert systems" to the final data visualizations.

Their current work is divided into five areas of LA: collection, storage, analysis, communication and action. These areas are represented as parts of the Learning Analytics Diamond, developed by Aaron Zeckoski, as part of the LAI:

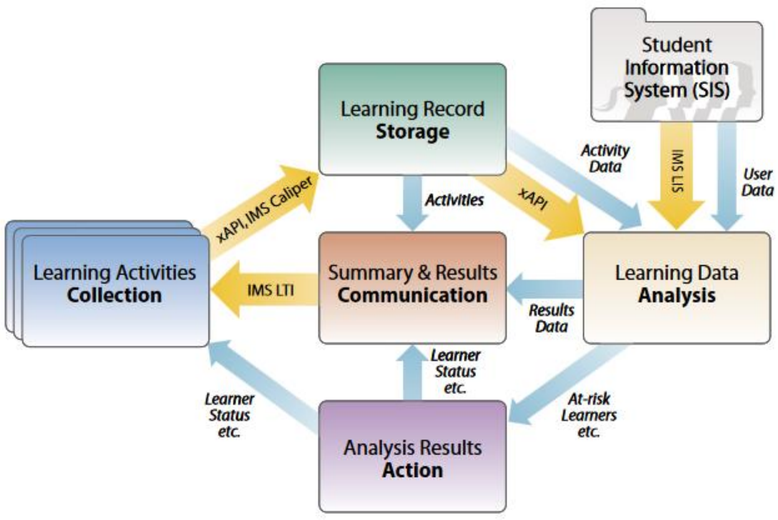


Figure 7. Learning Analytics Diamond

One of their major developed tools is an open source Learning Record Store. Several of their components have been used by different colleges and universities and "through controlled randomized studies", they "found statistically significant evidence that their early alert and intervention system had a positive impact on student success factors such as final course grades" [Apereo 2016].

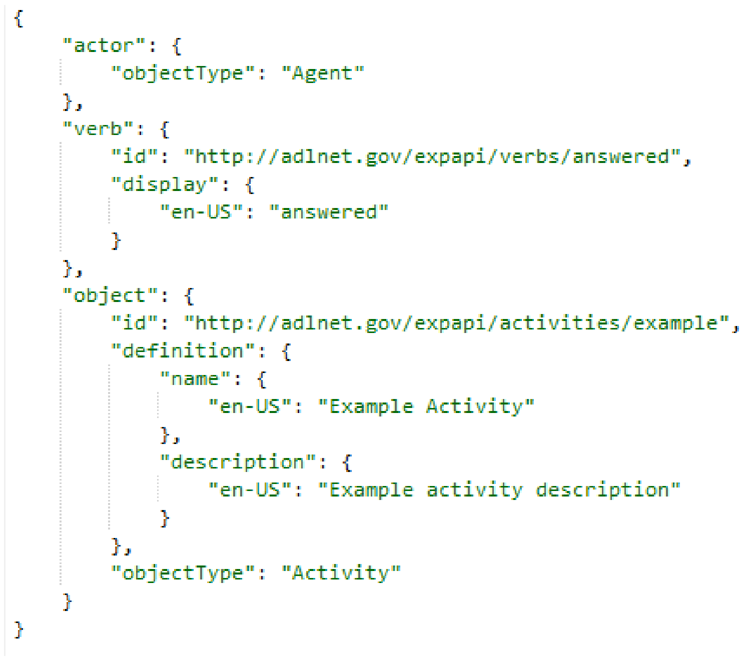
Another sign of this current impact appears in the **ADL** (Advanced Distributed Learning) [ADL 2016a] Initiative, created by the Office of the Under Secretary of Defense for Personnel and Readiness as part of the Department of Defense of the United States of America research and development for learning science and technologies. This initiative seeks to develop and implement tools for high-quality learning. Among their developments there is a broad set of resources that include an open source learning record store, dashboards and others.

All these resources are developed around their key definition of the **Experience API** ("xAPI") standard [ADL 2016b], previously known as the Tin Can API. The xAPI is a widely used e-learning specification. It is quite flexible as it allows to track all kind of learning activities. Although xAPI was initially developed for learning experiences, it may also be used to track all kind of activities. The experience API's set of web-service APIs supports "a simple object-based model for describing, recording and accessing one's experience or performance within a formal or informal learning activity". The specification of the standard allows to work with data about learning experiences including its creation, storage or access.

The use of xAPI is widely extended. Some companies are creating personalized learning experiences for their employees and use xAPI to track them. To start using it there are available platforms that provide a simple way to start a xAPI project. One of them is the xAPI apps [Training Evidence Systems 2016] that allows companies to connect apps and with it create and deliver "xAPI driven learning experiences". There are also several free courses to get familiar with the xAPI syntax, such as the Curatr course's "An Introduction to xAPI" [Curatr 2016].

This standard is open source and uses JavaScript Object Notation (JSON) for its data format. The main communication token in xAPI is a statement. A flow of statements is an activity stream that allows reporting completed experiences and not only final results. A statement contains an `actor`, a `verb` and an `object`. Other properties may also be included like context properties (for instance, the `timestamp`), learning activity results or attachments.

An example of a xAPI statement, containing its three main component (actor, verb and object) can be seen below. This example has been developed using ADL's tool xAPI Lab [ADL 2016c]:



```
{
    "actor": {
        "objectType": "Agent"
    },
    "verb": {
        "id": "http://adlnet.gov/expapi/verbs/answered",
        "display": {
            "en-US": "answered"
        }
    },
    "object": {
        "id": "http://adlnet.gov/expapi/activities/example",
        "definition": {
            "name": {
                "en-US": "Example Activity"
            },
            "description": {
                "en-US": "Example activity description"
            }
        },
        "objectType": "Activity"
    }
}
```

Figure 8. xAPI example statement

RAGE [The Open University of the Netherlands, University of Trento, Hull College of Further Education, Universidad Complutense de Madrid, The University of Bolton, INMARK, Instituto de Engenhariade Sistemas e Computadores, Investigacao e Desenvolvimiento im Lis 2015], that stands for Realising and Applied Gaming Eco-system, is a project funded by the European Union's Directorate-General for Justice and Consumers. It is part of its 'Fundamental Rights and Citizenship Programme' with funds from the European Union's Horizon 2020 research and innovation programme. RAGE aims to "develop, transform and enrich advanced technologies from the leisure games industry into self-contained gaming assets that support game studios at developing applied games easier, faster and more cost-effectively".

As part of the RAGE project, ADL has developed a first vocabulary designed for the Serious Games xAPI profile, aiming to cover all possible interactions that happen within a game environment. This Serious Games vocabulary [ADL 2016d], that also includes some terms not original for the serious games community, is divided into three main categories: verbs, activity types and extensions. These categories directly correspond to the fields: `verb`, `object` and the subfield `extensions` respectively, in the xAPI statements. Verbs aim to cover all possible actions within a game, for instance: started, completed, increased, decreased, set, used or selected. The activity types comprise game features such as: alternative, enemy, health, cut scene, level, menu, score or variable among others. Current extensions include progress and value, to specify the progress in a completable action and the value of a variable in those verbs that require it in their result.

A key open issue in the field of Gaming Learning Analytics is to standardize the way to represent players' interactions. The game industry has not solved this issue as it is protective with their Game Analytics practices and they usually rely on proprietary systems [Serrano-Laguna et al. 2016].

Ángel Serrano started to develop a Serious Games Interactions Model [Serrano-Laguna 2016] to track serious games, as part of his PhD work. This model included a set of game objects and interactions (completable, reachable, variable, target) to full specify the possible actions within a serious game. Each object or interaction has a set of predefined types, a set of actions possible, some requirements and considerations, and metrics.

The model has been revised over the last months and gone through different versions till a more definite one, fully explained in a paper finished as of June of 2016 [Serrano-Laguna et al. 2016]. This includes the collaboration of ADL as xAPI was the choice for a Learning Analytics standard to implement the model, thanks to its perfect capability to fit with the model later described or its lack of constraints on the vocabulary that can be used in statements, apart from its other good features already mentioned and extended use.

After studying several examples of serious games, an event-based interaction model that tries to cover the main necessities identified in games was inferred. Its key elements are the user identifier (*who* performs the action), the action (*what* action is performed), the target of the action (*who or what* receives the action), a timestamp (*when* is the action performed) and an extra optional value comprising the possible parameters of the action.

The types of targets and their related actions are the following:

- *Completables:* something a player can start, progress and complete, e.g. game, session, level or race. Possible actions with it are start, progress, complete.
- *Alternatives*: set of options to choose from, e.g. questions, menus, paths or dialog trees. Possible actions with it are select value (i.e. option) or unlock value (i.e. option).
- *Meaningful variables*: values inside the game with significance, e.g. score, currency or health. Possible action is to set a value.
- *Custom interactions*: to cover the events that are not able to be tracked with the previous events. The model can be extended with new types of targets and a set of associated actions.

This tracking model has been opened to the collaboration of ADL [ADL 2016a], developers of xAPI, aiming to propose an interaction model together with its implementation with the xAPI specification. Although other options of Learning Analytics standards, such as IMS Caliper currently still under development, were considered, xAPI was found to be the most appropriate one to implement this model. For this final purpose, all objects, verbs and values are mapped to their corresponding xAPI value. This full implementation of the xAPI profile (a definition of new vocabularies along with the extensions) was developed as part of the RAGE project and in collaboration with ADL.

Starting from its key elements, they are mapped to their corresponding xAPI statement properties: user identifier to actor, action to verb, target to object, value to result and timestamp to timestamp. The proposed actions (start, progress, complete, unlock, select) are also mapped to xAPI verbs and the targets types (serious game, level, mission, question, menu, dialog tree) to their corresponding xAPI activities' types. The xAPI attribute result allows to specify any game variable updates or any outcome related to the statement. Also, the xAPI extensions help to add extra semantics whenever verbs and activity types are not enough to fully express an interaction. The result extensions defined for the serious game profile are: progress, currency, health and position.

Finally, this complete model and implementation was tried out with a case study: the serious game Countrix, developed by Ángel Serrano. This example illustrates the use of this xAPI profile with a real example of serious game and analyzes the technicalities involved in the xAPI communication, such as the communication with the Learning Record Store. The game contains a xAPI tracker that sends xAPI statements using the profile designed.

Although during the rest of the project we would focus purely on the field of GLA, the analytics and visualizations required for it are no far from those made in different fields with no relation to education. For this reason, we finish this chapter with a brief remark on two examples of the work done in analytics and visualization in other fields. In particular, the fields of interest of these examples are website traffic and software development.

Google is closely related to data analytics. Its **Google Analytics** [Google 2016] service is a tool to track and report website traffic. With this tool, Google aims to show its customers personalized information about their websites in the shape of high-level dashboards as well as more in-depth data. This information for websites includes current activity, number of visits per day and their duration, geographical location, and more high-level information such as how the users found and used the website and how to make them return to it. They also have a specific version for tracking mobile applications' data, from the moment a user finds and downloads the app.

A simple example of the use of this software is to find peak times in cinemas. The following visualization appears automatically in our Google screen when we search for a particular cinema and is developed based on historical visits to the place:

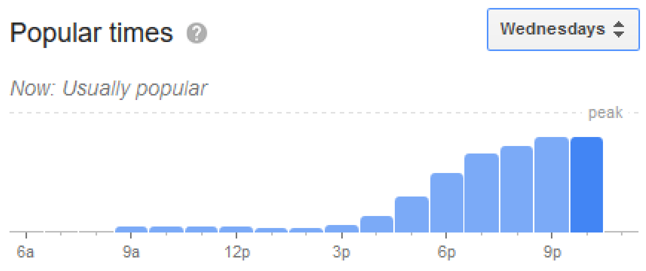


Figure 9. Google Analytics: peak times

But the use of Google Analytics goes beyond this to many other different fields. Another notable example of its use was to help the American Cancer Society to classify and consequently discover the value of the distinct users interacting with their websites and app [Google 2015]. Thanks to the results of those analytics, donations to their website Cancer.org rose 5.4%. We can see one of the results visualizations in the following capture:

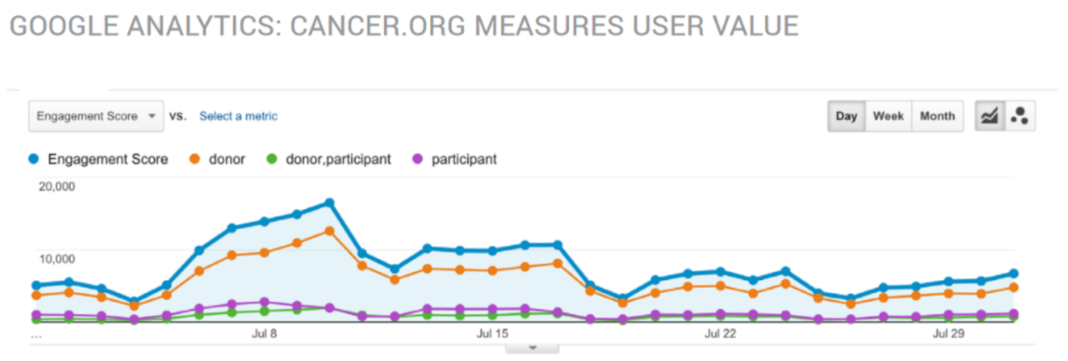
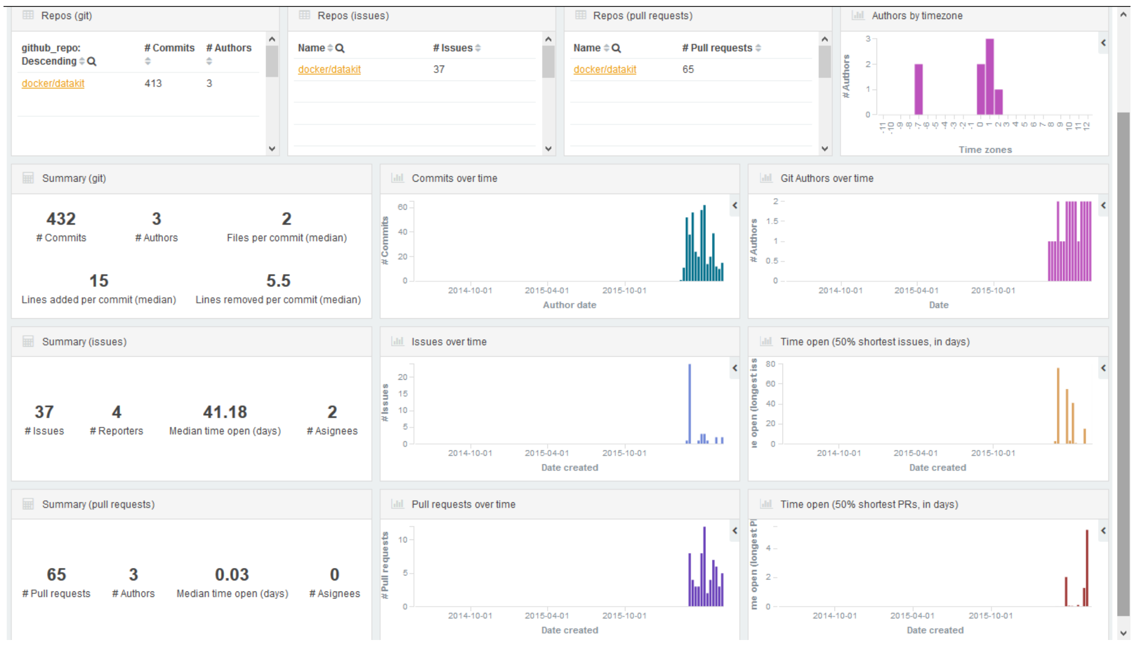


Figure 10. Google Analytics: user value

Bitergia [Bitergia 2016], which the Rey Juan Carlos University closely works with, is another example of the work done in the field of data analytics and visualization. Bitergia provides a technology for software development analytics aimed at companies working with Open Source and Inner Source Software. With their tool Cauldron [Bitergia 2016a] all activities happening within a GitHub repository for software development are analyzed. These include commits, issues and pull requests. The result information is then displayed in dashboards for its better understanding. An example of these dashboards [Bitergia 2016b] can be seen in the following capture:



github_repo: Descending	# Commits	# Authors
docker/datakit	413	3

Name	# Issues
docker/datakit	37

Name	# Pull requests
docker/datakit	65

Summary (git): 432 # Commits; 3 # Authors; 2 Files per commit (median); 15 Lines added per commit (median); 5.5 Lines removed per commit (median)

Summary (issues): 37 # Issues; 4 # Reporters; 41.18 Median time open (days); 2 # Asignees

Summary (pull requests): 65 # Pull requests; 3 # Authors; 0.03 Median time open (days); 0 # Asignees

Figure 11. Bitergia's dashboard

Architecture

The whole process of Gaming Learning Analytics and visualization previously described should be supported by an architecture that controls everything from the data collection to the results visualization.

Any basic implementation of a GLA system would require, at least, the following [Freire et al. 2016]:

- An instrumentation tool that stores the information of interaction in the game, to be typically sent to a server-based collector.
- Collection and storage tools.
- Analysis thorough reports and graphs.
- Some real-time analytics to be able to make interventions during a gameplay session to maximize the learning effectiveness. This should be light weighted, e.g. interactions of each player in the last 5 minutes or number of current players.
- Some aggregated analysis of data from different sessions.
- Key performance indicators such as grades, completion or educational effectiveness.
- Some dashboards or set of analyses and visualizations to provide a general overview of key indicators and ideally being configurable.

The next open code framework, which corresponds to GLEANER, is an abstract overview of a GLA system with these characteristics [Freire et al. 2016]. GLEANER, which stands for Game LEarning ANalytics for Educational Research, is a LA framework to capture and analyze data from interaction. It traces only higher level events considering both generic and game specific traces but only those relevant for our learning objectives [e-ucm 2016a].

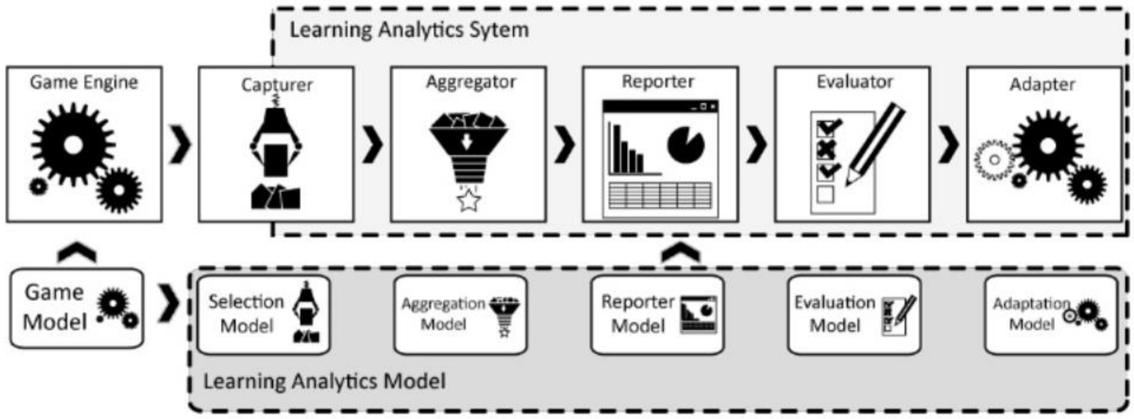


Figure 12. GLEANER framework

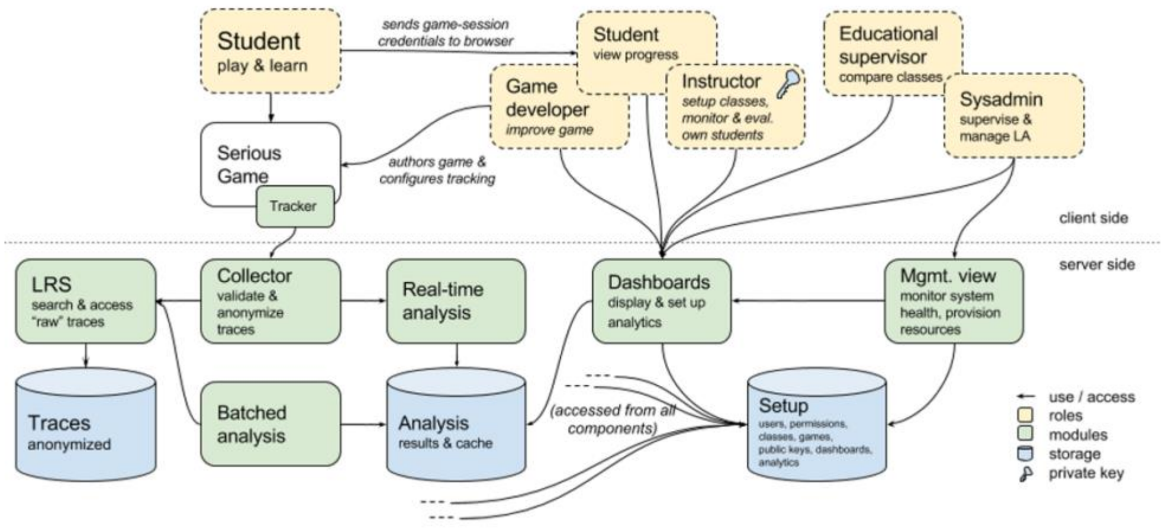
This framework has several components among which the following stand out, as part of the Learning Analytics System:

- The game starts the whole process and sends data to a collector.
- The collector (capturer plus aggregator) consists of a web server ready to receive traces from game sessions. Then it cleans and sorts the data.

- The reporter has access to the database and presents its data through different reports that may consist of graphics, heat-maps or relational tables.
- The evaluator has also access to the database and checks the educational defined goals in the assessment model.
- The adapter completes the loop sending back instructions to the game to adapt it to the player.

This architecture has five main parts and uses several prevailing technologies in each of them such as: MySQL for the real-time results storage, Apache Storm for real-time analysis, MongoDB for traces storage, Apache Spark for batch analysis and Hadoop for batch results storage [Fernández-Manjón 2014b].

In the context of the H2020 Europe Project RAGE already mentioned, a new software architecture is being proposed. The following diagram shows a logical architecture of a generic game LA system for use with serious game. RAGE will provide then several of the required assets to simplify this process [Freire et al. 2016]:



Student play & learn
sends game-session credentials to browser
Student view progress
Educational supervisor compare classes
Game developer improve game
Instructor setup classes, monitor & eval. own students
Sysadmin supervise & manage LA
Serious Game
Tracker
authors game & configures tracking
client side
server side
LRS search & access "raw" traces
Collector validate & anonymize traces
Real-time analysis
Dashboards display & set up analytics
Mgmt. view monitor system health, provision resources
Traces anonymized
Batched analysis
Analysis results & cache
(accessed from all components)
Setup users, permissions, classes, games, public keys, dashboards, analytics
use / access
roles
modules
storage
private key

Figure 13. GLA generic system architecture

As seen in the diagram, this architecture should contain at least a tracker embedded in the game, which sends traces to the collector. The data received may be either analyzed at real-time or stored for batched analysis. The architecture should also accept analysis queries for different stakeholders, with different needs and degrees of access and anonymization [Freire et al. 2016].

This architecture has several components that work in parallel but have dependencies among them. There are three main parts to consider in the following architecture to perform GLA:

1. Configuration: the user specifies the data to be obtained, the analysis to be performed and how the results should be stored for its visualization. An option is to provide an .YMAL file for the analysis and a JSON format file for the visualization details.

2. Analysis: following the format specified by the user, this part performs the analysis on the data captured.
3. Visualization: the results are displayed following the specified format.

Configuration

The developer of the game knows perfectly the game's internal structure and its intended learning goals. In the configuration step, therefore, he/she would be able to specify a topology that tries to show all the relevant information of the variables of the game with learning relevance. This would be the *default* topology of the game analysis. If some other user still wants to specify its own topology, it would also be possible to do using the provided tools, as we will see in the following. The teacher (or any of the other stakeholders) will then see an interface with all the information obtained from the topology. It should also be possible from that interface to choose or filter the information to be displayed, creating different personalized visualizations.

Moreover, in the front-end, the developer will be able to configure some information about the session (alias, score, maximum reachable score, progress) that would help in the following analysis and visualization. As the developer knows all those parameters from the internal structure of the game, according to the number of levels, maximum reachable score, and so on, he/she could easily provide that information. This would give an insight into the game structure and the analysis later performed could be combined with this basic knowledge to provide richer analysis results. A simple example that illustrates this idea is the following: if the developer provides the information of the number of levels that a game consists of, instead of simply showing the maximum level achieved in a game session for a user, the percentage of the game completed could be shown by simply adding the information of that total number of game levels.

At this stage some warning or alerts could also be specified. This could be particularly useful for the teacher during a game session in the class. A warning could be received for instance if some student/player has been inactive for too long, while an alert could be appropriate if a player has made too many or too serious mistakes. The developer could specify the requisites that need to be fulfilled for this extra information to be shown.

Ideally this configuration would be automatic. This could be made, for instance, by drag-and-dropping a .zip file with the configuration in JSON format, or something even more comfortable for the developer, for instance with a plug-in. As the developer would have included the tracker in the game, he/she would know the data that would be obtained with it so both the data and the configuration details could be sent together to the rest of the architecture for the later analysis and visualization.

The whole process described in this section would start when the web **frontend** [e-ucm 2016b] asks for certain information in a specified format. This information will be generated in the **backend** [e-ucm 2016c] using the real-time components. These real-time components will be obtaining data continuously from the tracking tools and performing the required analysis on it when requested. The results produced by the real-time components will then be stored in some suitable storage and displayed with the format specified using the chosen visualization tools.

This whole architecture developed by the e-ucm research group is currently maintained in a server consisting of a virtual machine with different dockers for each of its elements, which we will talk in detail of in the following sections.

Tracking

Starting from the client's side, the **tracker** is the tool that takes the data from the user's interaction with the game. This data may have any format or follow any standard. In particular, as we discussed in great detail in the first chapter, Ángel Serrano has been developing a tracking model to specify the semantics that messages should follow [Serrano-Laguna 2016]. This model follows the format specified in the Experience API standard so the game will send its traces following this complete model.

This interaction data then passes the authentication and authorization filter performed by the **A2** asset, which allows users to "register, login and access server-side assets via proxied requests" [e-ucm 2016d]. After being authorized, it arrives to the **collector** that keeps all the user-generated data. The collector extends the xAPI format of the traces that it receives adding the gameplay identifier and version identifier. This extra information is necessary in order not to mix traces from different gameplays performing a mistaken analysis and visualization. These two identifiers are obtained at the beginning of the game session when both the game and the collector start their communication with a handshake using the unique tracking code of the game. The game will then continue to send traces using this code.

Once the user-generated data is stored in the collector, it can follow two different paths. On one hand, still fulfilling the xAPI standard, the data will be sent to the Learning Record Store ("LRS") for its storage. Some general analysis may already be displayed on the data stored in the LRS but its complexity will be limited as LRS queries do not cover basic needs such as returning aggregate results, retrieving related statements, simple counts of xAPI verbs used or flexible query operators (e.g. not, less than, contains). The reasons for these limitations are that the LRS may be implemented over several backends (Elasticsearch, MongoDB or HBase) and hence the query syntax over JSON document structures may differ from one to another [Bakharia 2016]. Therefore, the most interesting analysis will have to be performed somewhere else, in our case in the other path for the data with the real-time components.

The LRS initially used in this architecture was a fork of the **OpenLRS** [Apereo 2016c], originally developed by the Apereo Learning Analytics Initiative [Apereo 2016a] as explained before. An important feature of this LRS is that it supports data in xAPI format. However, the OpenLRS has no longer support from Apereo, so it is set to be replaced for a different LRS, as it could be the one developed by ADL as part of their research with the xAPI standard [ADL 2015]. The OpenLRS internal database is implemented using Elasticsearch [Elastic 2016a], a highly-scalable open-source full-text search and analytics engine built on top of Apache Lucene. It allows to store, search and analyze big volumes of data in near real time. It is a well-known engine used by notable companies such as Wikimedia, GitHub or Mozilla.

At this point it is important to notice that we can work at two different levels of knowledge of the game. If we are the developers of the game or have complete access to its internal structure, as a white box, then we can know in advance the traces that the game sends. This would make easier the later analysis and visualization of the results. However, we would ideally want to develop a game-independent software that would be launched and it could automatically detect the traces sent by the game. These traces, which will have to fulfill the xAPI standard at least, will provide some information of the game. Based on that information, we could develop some analysis and visualizations for it, without having any access to neither the game nor the tracker of it.

Real-time analytics

For its analysis, on the other hand, the data from the collector will be sent to a chain of infrastructure elements. At this point, it is not strictly necessary that the data continues to be in xAPI format so instead it is mapped to a simpler format containing events, targets and values for its analysis. Still, it is necessary to keep the information added by the collector of gameplay and version identifiers so the analysis performed can be matched later to a game session.

The main analysis occurs in the Storm and Kafka real-time processing part. This element receives requests from the web frontend, performs the requested analysis on the data and stores the results for its later visualization.

Apache **Storm** is a free and open source distributed real-time computation system. Storm allows a quick and simple processing of the stream of data that receives. Storm keeps a variable topology that receives streams from a spout (a source of streams). The processing part takes place in bolts and may consist on filtering, functions, aggregation, joins, accessing to databases and much more [e-ucm 2016e]. This step is essential for LA as we have to consider here which data characteristics show the different learning process ongoing in the student/user.

To make this analysis simpler, Storm **Trident** was used at first. Trident is a high-level abstraction for doing real-time computing on top of Storm. Trident has joins, aggregations, grouping, functions, and filters. In addition to these, Trident adds primitives for doing stateful, incremental processing on top of any database or persistence store. Trident has consistent, exactly-once semantics, so it is easy to reason about Trident topologies.

The computation details may be better specified using Storm **Flux** [Apache 2016]. Flux is a framework and set of utilities for creating and deploying Apache Storm streaming computation topologies in an easier way. Flux aims to allow to package all the Storm components in a single jar, and use an external text file to define the layout and configuration of the topologies. There is no drawback in having used Storm Trident before as Trident defined topologies can be integrated with Storm Flux.

For the topology definition, a .YAML (YAML Ain't another Markup Language) file may be created during the configuration step describing a topology. The Flux topology definition consists of a topology name, a list of topology components and either a DSL (Domain Specific Language) topology definition (containing a list of spouts, a list of bolts and a list of "stream" objects) or a JVM class with the topology definition. Once the file is configured, it is executed generating the analysis requested.

The Storm component receives the data from **Kafka**, [e-ucm 2016f] a distributed commit log service that maintains a persistent and scalable queue with the data. Kafka keeps feeds of messages in categories called topics. Kafka can receive messages from different producers (processes that publish messages to Kafka) and send them to different consumers (processes that subscribe to topics and process the feed of published messages). In our case, the Storm component is the consumer from the data that Kafka provides. Both Kafka and Storm work together to perform the real-time analysis of the data that Kafka obtains from the collector, the producer of the data.

After the analysis with Storm is performed, it is important to keep the information obtained stored somewhere. For this purpose, **MongoDB** [MongoDB 2016] was used at first. MongoDB is a document-oriented NoSQL database system. It provides high performance and availability and it is scalable and open-source. A record in MongoDB is a data structure composed of field and value pairs, similar to JSON format.

After the Storm component finalizes the analysis and obtains the results, it stores them, or at least the necessary part of them, in Mongo. Then Mongo would provide those results to the web fronted. The format in which the results were stored in Mongo would have been specified by the user in the configuration step, usually using JSON format. Thanks to the information added by the collector, Mongo would be able to maintain the data clustered by game sessions so the data of a session will not be overwritten with that of a different session. That way, every time a new trace enters to Mongo, it would be updated or added to the document of its user's session.

However, due to the final choice of visualization tool, it was no longer necessary to store the result information of the analysis in a database like Mongo. Instead, it is now directly stored in a different database (Elasticsearch) that works closer to the chosen visualization tool, i.e. to Kibana.

Finally, the requests made by the frontend will also have to pass the A2 filter before they arrive to the real-time analysis component that will then return the results of the requested analysis to the frontend.

A whole picture of the architecture previously described can be seen in the following diagram, in which we differentiate the two main parts of the architecture; the frontend (client) and the backend (server) as mentioned in the previous description:

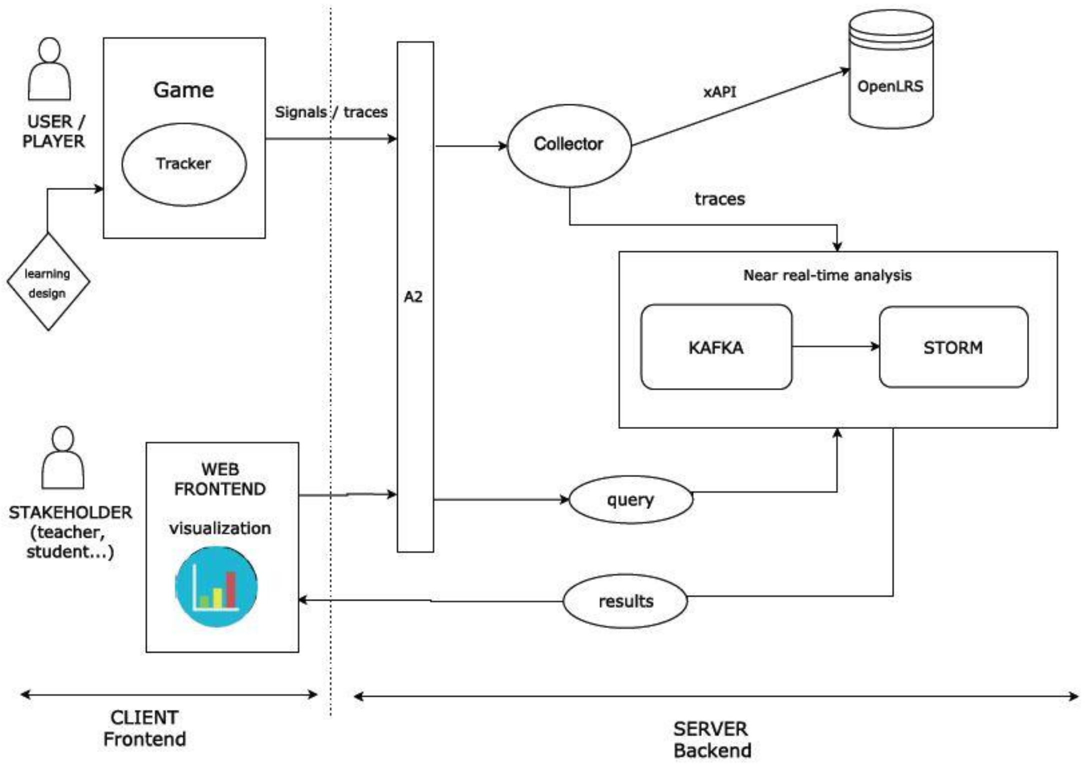


Figure 14. GLA architecture

Visualization tools

The last part in order to complete the GLA process is the visualization of the results. The web frontend asked for certain information in a particular format and this information was passed to the Kafka-Storm system that performs the real-time analysis and stores the results in the proper database. The frontend will then show the information in the form of visualizations and dashboards.

The visualizations might be any type of graphics or statistics as long as they provide useful information from a learning point of view in a clear and simple way, easily understandable for the viewers of it. It is essential at this point to adapt each visualization to the target of it and its particular necessities as it may not be an audience with high knowledge of statistics or other issues going underneath the analytics performed.

Specifically, the visualization part may use several different tools such as D3js, OpenDashboard or Kibana. Independently of the tool used, it will request and receive the data from the real-time components (Kafka and Storm). We explore some of these tools with different levels of detail in the rest of the chapter.

D3js [Bostock 2016] is a JavaScript library that provides efficient manipulation of documents based on data. It allows us to bind data to a Document Object Model (DOM) to apply data-driven transformations to the document. To create a simple visualization with D3, you may create an html file (as D3 can work together with this and other web standards such as SVG) and use within it any java script files containing the details of the visualization or the functions that bind the data to the graphics. It is also possible to add a CSS file for the style configuration.

The data can be imported from many types of formatted files (plain text, csv, tsv, JSON, xml, html) for its display using D3 functions. Once the visualizations are set up and the data imported, the graphics can be displayed in a browser by simply opening the html file.

Although this tool is very flexible, as it allows to work directly with the code and therefore, develop any desired graphic, it is precisely for this reason that each of these developments requires some quite laborious work. This is needed in order to draw the graphics completely from scratch, having to specify every single detail of them, fixing data limits, margins, data types, and so on, usually having to define several functions to create even very simple visualizations.

Another option for visualization that was considered was **OpenDashboard** [Apereo 2016b], a tool developed by Apereo as part of their Learning Analytics Initiative, which also developed the previously mentioned OpenLRS. OpenDashboard is an open source web application that provides a framework to display visualizations on data. In OpenDashboard, a data view is called a *card*. Cards represent a "single discrete visualization or data view but share an API and data model". The idea of Apereo was to provide a dashboard framework for an open LA environment.

To start using OpenDashboard, it is first needed to add some data providers, i.e. the sources to receive the data from. Once the providers are set, it is convenient to create a set of preconfigured dashboards for common use. For that, new cards need to be deployed. A card may be a chart or any type of visualization graphic. To fill the card information, a JavaScript module (the AngularJs Module) needs to be created containing the basic information and configuration data of the card. Then an html file should be created with the UI markup for the card. All this information needs to be then added to the main page.

Despite all these features, OpenDashboard was not the final choice of visualization tool. Among other reasons, such as the fact that OpenDashboard needs to be executed over an open protocol to communicate with learning management systems, which makes it not so easy to integrate with the current system, the main drawback was that OpenDashboard has no longer support of Apereo as the project entered incubation as of June 2015.

Kibana [Elastic 2016b] is an open source analytics and visualization platform designed to work on top of the search engine of Elasticsearch. It was developed by Elastic, just like Elasticsearch, with the purpose of having a flexible browser-based interface to quickly and easily create new visualizations on the data such as charts, tables and maps and combine those creating dynamic dashboards. Kibana is able to work with vast amounts of data obtained from different sources and later export the interesting bits of data. One of the most interesting features of Kibana is that its dashboards can change dynamically to display results of queries made in real time.

In the rest of this chapter we will describe the several useful and interesting features of Kibana, as it is the fact that it perfectly fits to work together with other useful tools such as Elasticsearch or Logstash. These reasons made it, despite its limitations and some drawbacks which we will also mention, the definitive choice for visualization tool.

To start using Kibana it is first necessary to have **Elasticsearch** [Elastic 2016a] working. As we mentioned before, Elasticsearch is a powerful engine to analyze and search a vast amount of data very quickly – almost at real time. Both Elasticsearch and Kibana are tools in constant development and growth which makes them particularly interesting to work with.

Once we have Elasticsearch ready to work, new data can be added using the syntax of any HTTP request (verb, protocol, host, port, path, query string and body). Queries may also be made to it using this same syntax. To any request, Elasticsearch responds with an HTTP status code and a response body which is encoded using JSON format.

Elasticsearch is document oriented, i.e., it stores entire documents (objects) and indexes the content of each one of them to be able to search within them. The format of the documents stored is JSON, which is the standard format used by NoSQL platforms. Here, each JSON document represents an object. The structure of Elasticsearch is the following. A document belongs to a type and types live inside an index (database), so a cluster can contain multiple indices/databases which in turn contain multiple types (tables). These types/tables hold documents (rows), each one having multiple fields (columns).

A useful feature of Elasticsearch is that it allows to define mappings of fields inside an index and a type before we even have any data inside it. By doing this, the fields of the data that we add to that concrete index and type will be mapped to their correct types (e.g. date, integer or float) for their later correct processing. To start working, we may create an index and a type and add some information to it using the HTTP syntax and JSON format for the body field.

There are several ways to import data into Elasticsearch. Using the simple *curl* command with its http syntax or, if we used Mongo, directly from it using a plugin supported by the community are some available options. A useful tool that we may consider for this is **Logstash** [Elastic 2016c], an open source data collection tool developed as well by Elastic, that perfectly fits to work together with both Elasticsearch and Kibana. Logstash can bring together data from multiple sources and normalize it to the preferred destinations. Logstash allows to read data from several format files such as JSON format, csv and different databases.

After installing Logstash, to bring data to Elasticsearch we only need to edit the Logstash configuration file. In this file, we can specify the desired input, filters and output. For instance, if we want to transfer the data of a csv file to Elasticsearch, we would specify on the input part the path of the file and its type (here we may use an already defined type of Elasticsearch for correct mappings). To filter the data, we could give the csv column names and specify in the output part that we want to index that data to Elasticsearch into a concrete index (that may already exist or be new). Once we save and run the configuration file, we would have the data from the file into the specified index and type of Elasticsearch and so it would be ready for making visualizations with it using Kibana.

After having both Elasticsearch and Kibana installed and running, if we entered our preconfigured local port (typically, http://localhost:5601) we would get access to Kibana's initial configuration page, which would be running on that specified port. This first page asks to configure an index pattern as at least one must be configured to use Kibana. An index pattern is used by Kibana and Elasticsearch to know how to access the data. We may choose our first index pattern to contain time-based events and provide the name of an index that we have previously created in Elasticsearch using any of the ways described. This is, an index pattern in Kibana corresponds to an index in Elasticsearch.

The next step is that for that index, we have to specify the name of a time field. This step is compulsory for Elasticsearch and Kibana to be able to filter the data using the global time filter. For that reason, it is key that our time field, which would usually be a timestamp, has been correctly mapped to a date field when it was saved to Elasticsearch. Either if the data was saved to an already existing index or to a new one with a previously specified mapping for the date field, both options obtain the desired result and the time field is correctly mapped for its use in Kibana. Once the first index is configured, it will be added to the list of index patterns and we can access all the data and fields associated to its types.

With our index configured, its contained data will already be added and ready for its visualization using Kibana. The first option that Kibana offers is *Discover* which allows to see all the data within the current selected index and its fields' content as well as to make searches on it. We may also want to filter the data to just consider some concrete part of it that fulfills some requisite. In the visualizations and dashboards we also have to take into account the time filter set (by default established to the last 15 minutes) as it would only allow to visualize the data whose timestamp field belong to the established time range. Also we may need to narrow or expand the time filter for better data visualization.

The *Visualize* option allows to create new visualizations based on either new or already saved searches. The visualizations' types include data table, line chart, vertical bar chart, area chart, pie chart, or a single metric. They also have tile maps to associate an aggregation with geographic points. Another quite useful option for visualization are markdown widgets that allow to add text in markdown language. This could be used to add explanations of visualizations or any other extra information we want to add in our dashboards in a quick and elegant manner. In the visualizations we have both metrics to choose among (e.g. maximum, minimum or average of any numeric field, count of any field) as well as buckets and sub-buckets to create more complex graphs.

The last option is *Dashboards* which allows to create new personalized dashboards of multiple data. In a created dashboard, we may add any already saved visualization, even if they contain data from different indices. The visualizations may be placed in any position and shape as we prefer to create a dynamic dashboard.

An essential feature of Kibana is that thanks to its connection with Elasticsearch, once we have a visualization created of an index's data, if we add any new data to that index in Elasticsearch, the visualization and all the dashboards it appears in will automatically and almost at real-time be updated to contain the new data. Moreover, although visualizations created with Kibana correspond to a particular index, they would remain in Kibana even after the index is deleted so we could reuse them.

Kibana is having a huge impact in many areas and consequently the preconfigured types of charts that it allows to create are no longer enough to fulfill everyone's needs. However, this is not an unsolvable problem as Kibana allows its users to create their own plugins and custom visualizations. Many consumers of Kibana, therefore, have developed their own plugins, most of them to create new types of visualizations apart from the seven types that Kibana has currently configured by defect.

Some of the existing plugins for Kibana are: Timelion [Elastic 2016d], a time series composer for Elasticsearch, an authentication plugin for Kibana [Hmalphettes 2016], a heat map chart [Stormpython 2016a], a tag cloud chart [Stormpython 2016b], a vector map chart [Stormpython 2016c], a radar or spider chart [Sirensolutions 2016], a slider widget [Raystorm-place 2016a], a widget to add HTML code [Raystorm-place 2016b] or a csv exporter plugin [Minewhat 2016].

Another important feature for the later used of both Kibana's single visualizations and dashboards is that once they are created, they can easily be exported, by default in JSON format. Kibana also provides an html code for any visualization and dashboard created so they can easily be embedded into a website and displayed in a browser by simply copying and pasting the given html code by Kibana into the html code of our website. Another available option is to share a link to them. In any case, the only requisite is that all clients must still be able to access Kibana.

The option of sharing a link to a visualization or dashboard in Kibana could also be very useful in our context of work for the following. In the link that Kibana generates we have several information of that concrete visualization or dashboard including its name, its time range considered, the visualizations contained on it in the case of dashboards and the filters applied to the data. Consequently, this set of information included in any link allows the possibility to create more general visualizations, generate a link to them, and then based on those generic visualizations previously established, working solely with its generated link, add the required filters to that data creating the specific visualizations desired.

The filters that we can add in the links that Kibana generates correspond to the queries that we can make directly from the Kibana web interface. The interesting feature of this is that adding the filters directly to the links makes unnecessary to access the interface to create different visualizations but instead allows to work with generic ones that can be particularized with filters to fulfill our concrete needs.

Some samples of queries that we could directly add to the links are: field (numeric, boolean, string, date) equal to value, numeric or date field greater than (less than) value, numeric or date field greater than (less than) or equal to value, or field (numeric, boolean, string, date) distinct to value.

Despite all its good features, Kibana has also a drawback in its security model as Kibana by itself does not support authentication or restricted access to its dashboards. To overtake this issue, Elasticsearch has developed Shield [Elastic 2016e], a product that provides security in several ways such as encrypted communications, authentication, role-based access control and auditing. With it, it is possible to create different configurable layers of security with both field and document-level security.

Mathematical component of the project

The project has several mathematical concepts ongoing in its whole development from the very first tracking of data to the final visualization of the result information. Although this is clear throughout the whole report of the project, this section aims to make some of the most relevant parts of it even more explicit for their better understanding.

A serious game, first of all, could be usually perceived, as it is in the case of many other non-serious games, as a directed graph. In this graph, each level of achievement of the game corresponds to a node and each path to reach one level from a previous level correspond to an edge. This is a coarse-grained view of the game, as levels would usually contain subparts to be also completed or required goals to be achieved before being able to move forward to the next level.

With this simple approach, however, it is possible to guarantee the end of the game, if we know for sure that several premises are verified. The game model could be then characterized by the graph and each particular situation of a user during a session be described by a state or node on that graph. That is, we will not need to know the whole progress of the session but simply its current situation. This instant picture could provide a complete information about the current state of the session and therefore, if we incorporate to it some new information, with the graph we could univocally determine what the next state of the user is going to be.

This instant information of the game will be given by the xAPI traces, as this event-based specification will provide the data of each interaction as the player moves forward in the game process. An overview diagram of this ideas can be seen below.

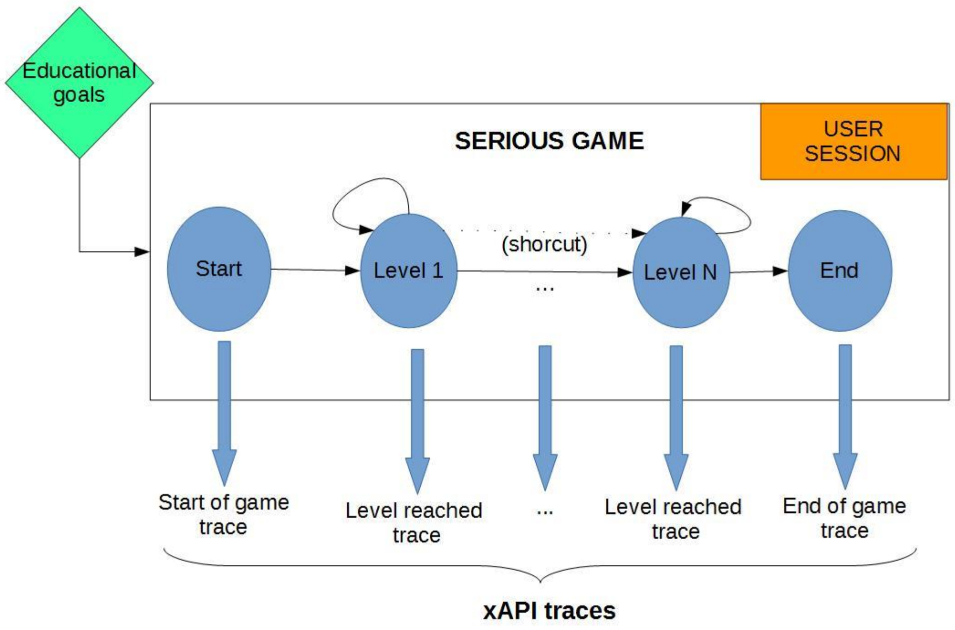


Figure 15. Game structure and traces sent

This is a simplification of many games as we are not considering several other issues such as random events, enemies or time, but in most serious games these are not so common. Random events usually are to be avoided as they could drastically change the conditions of the game from one player to another, and we would ideally want them to have game conditions as similar as possible. Also, the presence of enemies is most common in non-serious games than in this field.

The analysis of data is the key part for the process of Gaming Learning Analytics. Several data needs to be traced in games for its later analysis. This relates to Big Data as the number of traces may become significantly large, even if the number of game players is still quite low compared to non-serious games. For instance, the usual number of players to be considered could only vary between 20 and 30, corresponding to a single class session. However, even with that small number of players, the number of traces each one of them sends may be considerably large depending on both the complexity and depth of the particular game and the peculiarities of each player's interaction with it. Therefore the size of the traces collected may become a problem itself, including issues such as their collection, storage, analysis and visualization.

The specific analysis performed on the data collected compress both the data gathered and its treatment. For its gathering, it may be required to aggregate the data according to different characteristics or ignore some of this features to focus on others. The analysis itself would typically include quite a big amount of calculations such as count of errors made, duration of play sessions, evolution of score over time and other game-dependent metrics. The analytics typically comprise several-steps calculations that may require storage of partial calculus, even if they are later ignored in the final result.

The visualization part of the project also contains many mathematical reasoning. Starting from the results information received to visualize after its analysis, this information is again treated mathematically to develop the desired graphics of the represented data. Commonly used calculus are: average of score or time fields, duration of game sessions, maximum and minimum values achieved in fields like score or time of play, etc. All these calculus are particularly important in GLA as most of the fields of the data that we may consider interesting to visualize are fields with a numeric value (integer or float). This is the case of scores, levels, duration, and so on.

The visualization of the data may also include in many cases data filters. These could be a wide range of filters from numerical intervals to single values of fields. Again for those filters it is essential to work with numeric fields that allow to make searches such as queries on values greater or equal than a given value, or values distinct to a concrete value that we may not want to consider. This type of queries would be useful for the teachers so it is important to consider them at this stage.

Analysis and visualization examples

In this chapter the ideas and discussions previously described are put into practice with a few examples of serious games. In particular, we would work with two different examples of games specifically developed for these issues.

Firstly, several analysis were carried out on the data generated by the games. These analysis were made in Python scripts and their corresponding results stored into csv files. Those were later imported into different Elasticsearch indices and types using the tool Logstash.

The analysis covered some general issues such as results of game, scores, times of play session, ranking of results, personalized data of a user, as well as others particular to the game or to the stakeholder considered. In the first examples, only students and teachers are considered as the information they may want to visualize is simpler, and therefore easier to obtain from little data. However, in later and more specific visualizations other stakeholders are considered as well. For instance, developers could be interested in information of the game sessions (total time, number of interactions per session occurring) while managers would be more interesting in the peak times of use of the game.

The visualizations and dashboards were developed using Kibana. Specifically, the ones for the first example were developed using the Elasticsearch version 2.2.1, as well as Kibana's version 4.4.2, as they were the last versions available as of March of 2016, when those first examples were developed.

For the second example, although all the examples were firstly developed with those same versions, they were then updated to the last released version of Kibana. This last version, Kibana 5.0.0-alpha2, was released on the 3rd of May of 2016, and also needs the last version of Elasticsearch, Elasticsearch 5.0.0.-alpha2, that was released on the same day. Among other features, this last version of Kibana allows to customized labels in graphics (x-axis, y-axis, single metric, and so on). This simple improvement eases the communication of results with stakeholders as each graphic can have now its own explanation within it. This reduces the need of markdown widgets to add extra explanations of graphics and therefore makes the resulting dashboards clearer.

The improvement between Kibana's versions may be seen in the screen captures of the following sections, as the ones of the first examples remain in the old version of Kibana to make the change of version even more explicit. Meanwhile, all the screen captures of the second example correspond to the last available version of Kibana.

First example: QuizDemo

QuizDemo [e-ucm 2016g] is a sample mini-game developed by Antonio Calvo with Unity with the aim to show how traces in xAPI format are captured during a game session. Although it has no complexity, it contains some of the most relevant traces present in the majority of the games such as traces of new level achieved, score updated or option chosen in multiple choice situations.

After a first welcome screen, the game consists of two screens with multiple choice questions. Every time a wrong answer is selected, the score is decreased. When the right answer is finally selected, the score is increased and the next level loaded. The final screen shows the total score. For instance, the second question of the game is a visual multiple choice question:

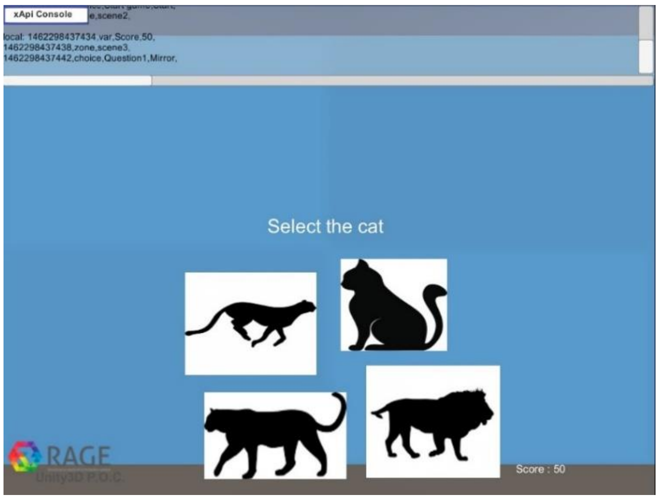

```
e,scene2,
local: 1462298437434,var,Score,50,
1462298437438,zone,scene3,
1462298437442,choice,Question1,Mirror,
```

Figure 16. QuizDemo question

The most interesting part of this game is that we can see the traces sent at real-time in the xAPI console at the top of the screen.

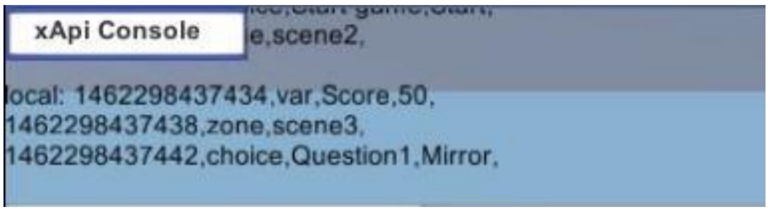

```
e,scene2,
local: 1462298437434,var,Score,50,
1462298437438,zone,scene3,
1462298437442,choice,Question1,Mirror,
```

Figure 17. QuizDemo xAPI console

Every time we interact with the game in some way (e.g. selecting an answer) the new traces are added to that. For instance, a trace like

```
1462298437442, choice, Question1, Mirror
```

is sent when, at timestamp `1462298437442`, in the first question, the option `Mirror` was selected.

In order to create the visualizations, it was first necessary to obtain some further information by means of analyzing the traces sent by the game. The required analysis, that were made in Python scripts, included tracking of scores, calculation of time per session or calculation of errors made by level. Some of the developed analysis made in Kibana, after import the data to Elasticsearch, are the following:

Score trace

This graphic aims to show the progress of the score of a user during a single session.

To create it, we choose a line graph and as buckets, we split the X-Axis by timestamp and then we split lines by the field *score*.

To have only the information of a particular session we filter the data by session, making a query that asks for the field *session* to be equal to this particular session id.

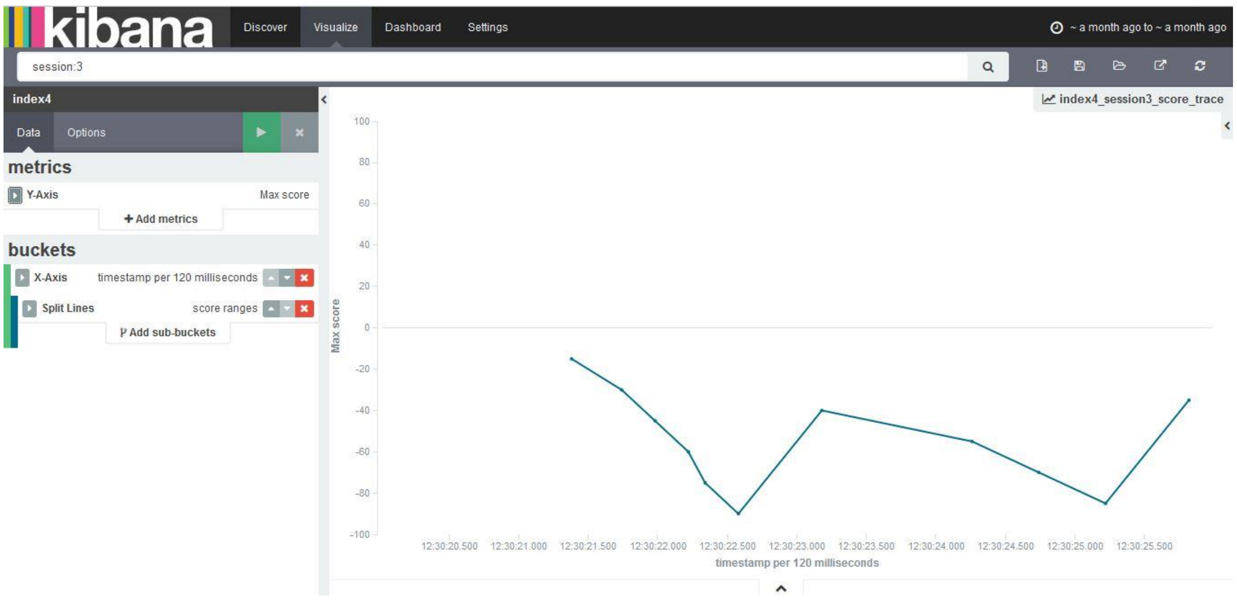


Figure 18. QuizDemo: score trace visualization

Time of play sessions

This dashboard aims to show the time of play of the sessions tracked as well as the mean time of play of all the sessions.

To create this dashboard, we first create a bar graph that contains the maximum *time* in the Y-Axis and the *session* in the X-Axis. We also create a single metric showing the average of the *time* field (in milliseconds).

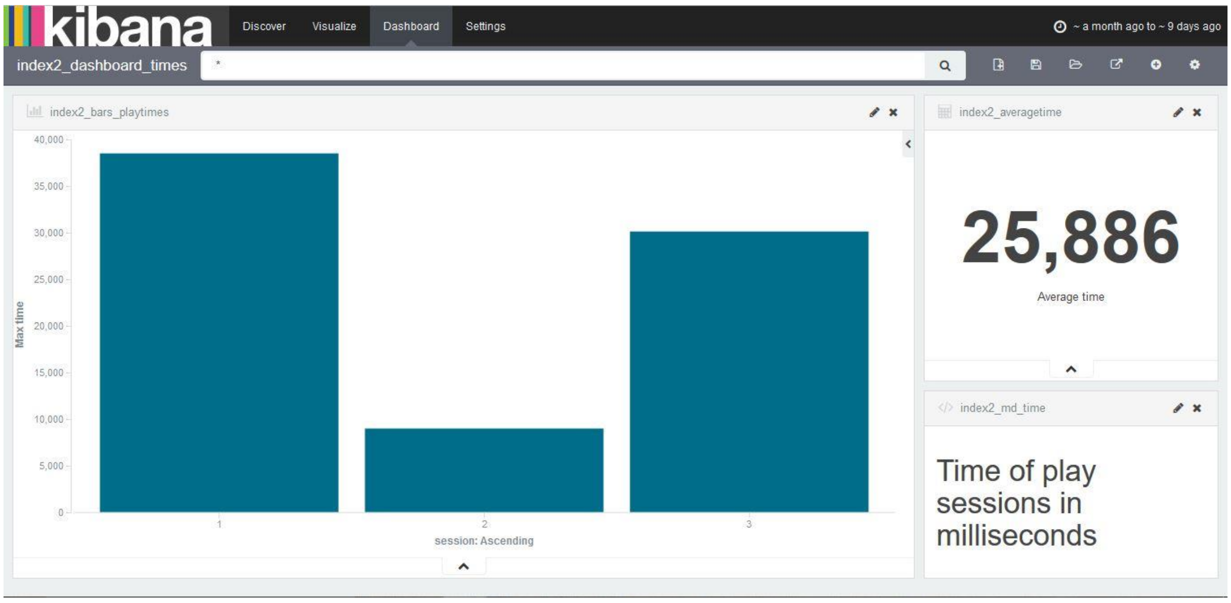
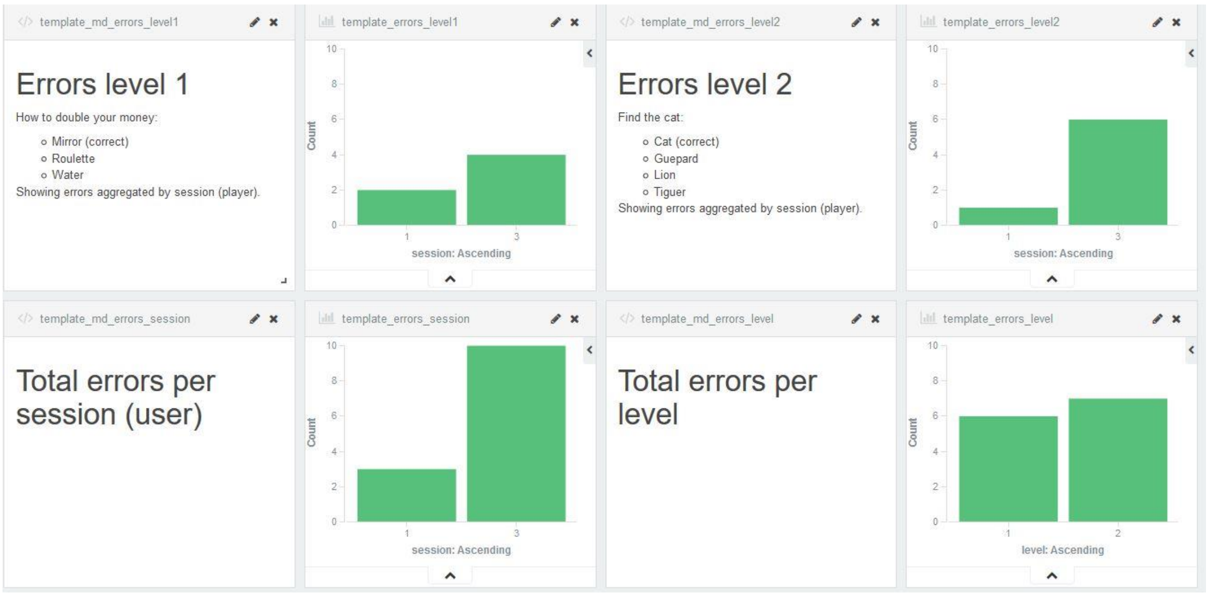


Figure 19. QuizDemo: time of session dashboard

Errors made

This dashboard aims to show the distribution of errors made in the game.

All four graphics contained on it have on the Y-Axis the count of errors, and show, from top left to bottom right: errors made in the first level separated in X-Axis by *session*, errors made in the second level separated in the X-Axis by *session*, total errors per *session* and total errors per *level*. The explanations included in the dashboard were added using markdown syntax.



Errors level 1

How to double your money:

- Mirror (correct)
- Roulette
- Water

Showing errors aggregated by session (player).

Errors level 2

Find the cat:

- Cat (correct)
- Guepard
- Lion
- Tiguer

Showing errors aggregated by session (player).

Total errors per session (user)

Total errors per level

Figure 20. QuizDemo: errors dashboard

Second example: Countrix

Countrix [e-ucm 2016h] is a geography questions game developed by Ángel Serrano and finished to work with as of the end of April of 2016. The purpose of this game was to have a more complex example that tracks data during a game play and sends the corresponding traces in xAPI format to a LRS. The game consists of a sequence of random questions from a predefined set of geography questions that includes four types of questions: which country has a given capital, which country has as capital a given city, which country has the flag given and which continent is a given country in. A sample question of the game is the following:

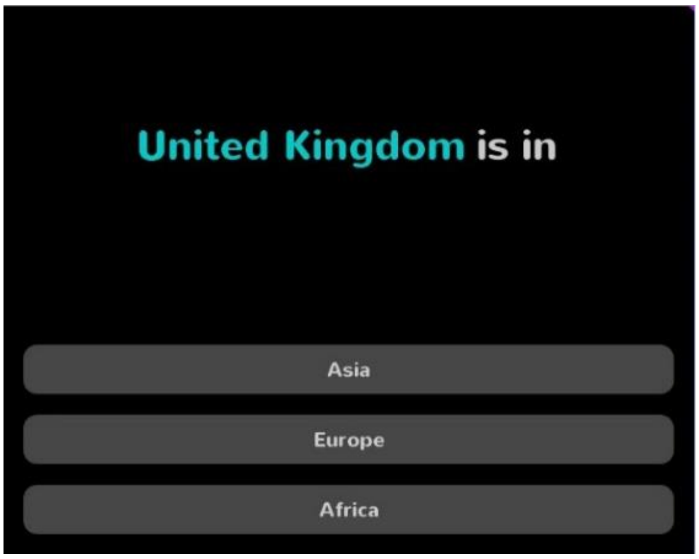
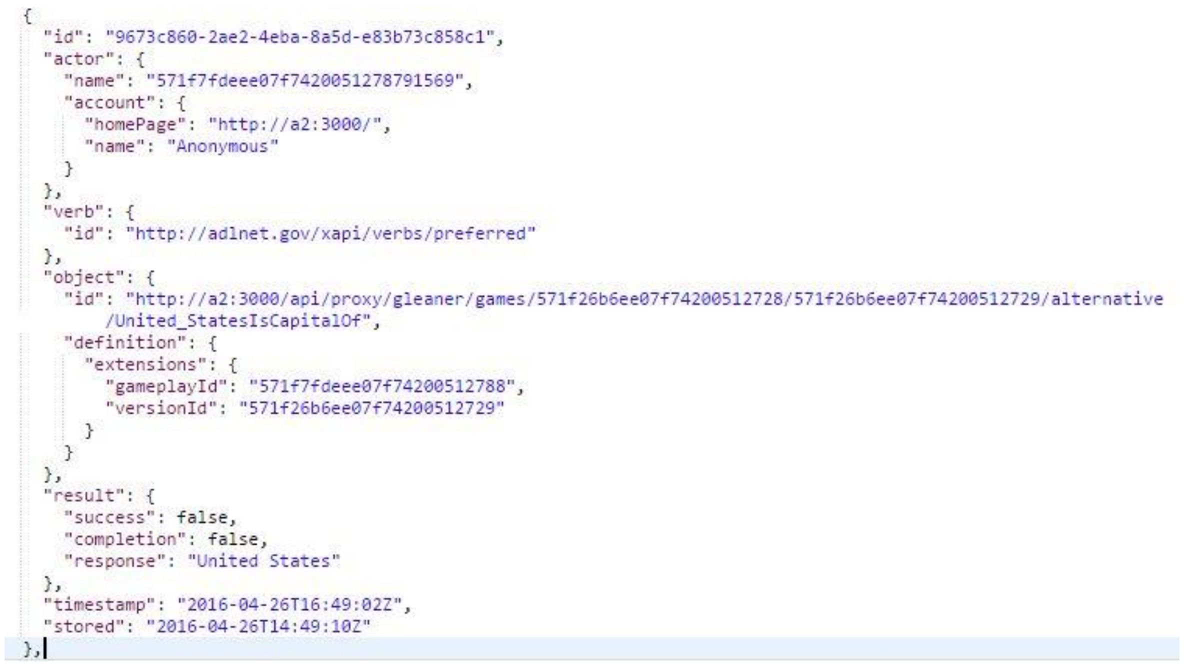


Figure 21. Countrix question

The game has a fixed time for each play. Every time a question is answered correctly, the score of the session is increased; when it is wrongly answered, the remaining time is decreased. The game also gives the possibility to show at the end all the xAPI statements generated during the game session.

For the purpose of this demo, all the traces stored in the server were obtained as of 5th of May. The LRS contained at that moment 1209 xAPI statements generated by the game plays between its start on the 26th of April and the date of its getting. A sample of a xAPI statement sent by the game and stored in the LRS can be seen below.



```
{
  "id": "9673c860-2ae2-4eba-8a5d-e83b73c858c1",
  "actor": {
    "name": "571f7fdeee07f7420051278791569",
    "account": {
      "homePage": "http://a2:3000/",
      "name": "Anonymous"
    }
  },
  "verb": {
    "id": "http://adlnet.gov/xapi/verbs/preferred"
  },
  "object": {
    "id": "http://a2:3000/api/proxy/gleaner/games/571f26b6ee07f74200512728/571f26b6ee07f74200512729/alternative
        /United_StatesIsCapitalOf",
    "definition": {
      "extensions": {
        "gameplayId": "571f7fdeee07f74200512788",
        "versionId": "571f26b6ee07f74200512729"
      }
    }
  },
  "result": {
    "success": false,
    "completion": false,
    "response": "United States"
  },
  "timestamp": "2016-04-26T16:49:02Z",
  "stored": "2016-04-26T14:49:10Z"
},
```

Figure 22. Countrix xAPI statement

The analysis was made, firstly, focusing on the questions of the game. The error ratio of each question was calculated to give an idea of the questions that set out a bigger difficulty for the users. Focusing on the players, the analysis stores each question a player has answered and the result of it, i.e. whether the player answered the question correctly or not.

Due to the amount and complexity of the data, some deeper analysis was now required. It included parsing the statements to get their type and, for the question statements, parsing of the `object.id` field to get the question name and therefore the question type. Besides, both the `actor.name` and `timestamp` fields where necessary to establish the user name and timestamp of the result traces respectively.

The next diagram shows an overview of the analysis developed:

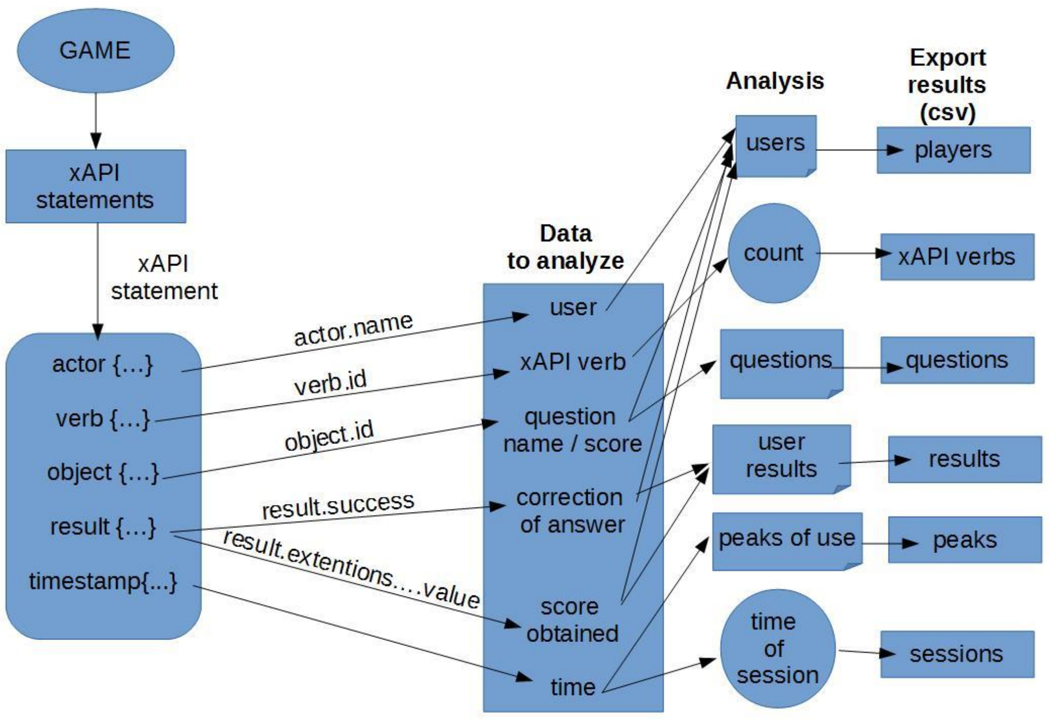
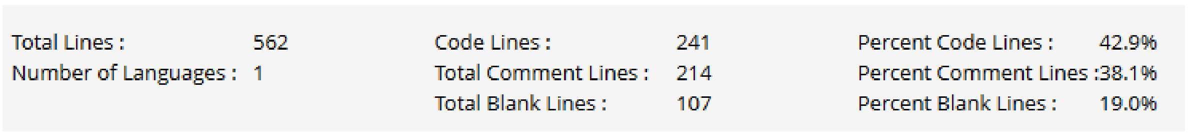


Figure 23. Diagram of Countrix analysis

The code developed for this analysis is accessible in a repository in GitHub as part of the e-ucm development group: https://github.com/e-ucm/countrix-analysis

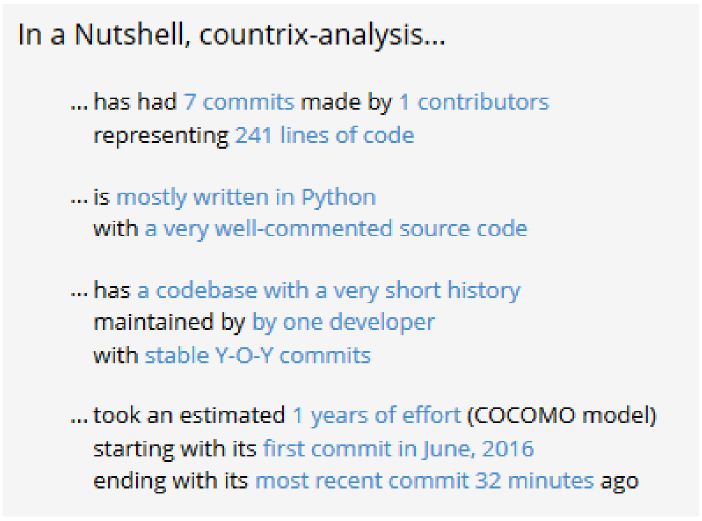
This repository contains the file for analysis, written in Python, the data files (a JSON file containing the xAPI statements from the server and a csv file with the information about countries for finding the correct answers of questions) and the csv result files of the analysis, as described in the previous diagram.

Analyzing the code with the tool OpenHub [Black Duck Software 2016], we obtain the following reports:



Total Lines :	562	Code Lines :	241	Percent Code Lines :	42.9%
Number of Languages :	1	Total Comment Lines :	214	Percent Comment Lines :	38.1%
		Total Blank Lines :	107	Percent Blank Lines :	19.0%

Figure 24. OpenHub: lines of code


In a Nutshell, countrix-analysis...

... has had 7 commits made by 1 contributors
representing 241 lines of code

... is mostly written in Python
with a very well-commented source code

... has a codebase with a very short history
maintained by by one developer
with stable Y-O-Y commits

... took an estimated 1 years of effort (COCOMO model)
starting with its first commit in June, 2016
ending with its most recent commit 32 minutes ago

Figure 25. OpenHub: code information

The csv files containing the results of these analysis were then imported into Elasticsearch. For that, a set of indices was first created in Elasticsearch. The required mappings were established for their fields and the results of the analysis were then imported into their corresponding indices with Logstash. Specifically, the Elasticsearch indices created, with their corresponding fields and types, were the following:

Index name	**Data about**	**Field 1**	**Field 2**	**Field 3**	**Field 4**	**Field 5**	**Field 6**
Countrixp	Players	Timestamp (date)	User (string)	Question (string)	Error (boolean)		
Countrixq	Questions	Timestamp (date)	Question (string)	Qtype (string)	Errors (integer)	Answered (integer)	Ratio (float)
Countrixr	Results	Timestamp (date)	User (string)	Flag (string)	Country (string)	Capital (string)	Continent (string)
Countrixt	Sessions	Timestamp (date)	User (string)	Time (float)	Questions (integer)		
Countrixpk	Peaks of use	Timestamp (date)	Minute (string)	Users (integer)			
Countrixx	xAPI verbs	Timestamp (date)	Verb (string)	Times (integer)			

Table 1. Elasticsearch indices

After that, the following dashboards were developed using Kibana. The first ones focused on questions, the set of users and a single user respectively. After these first dashboards, the aim was set into trying to obtain as much information as possible from the game traces from the two different approaches:

- knowing the internal model of the game
- purely based on the xAPI traces sent by the game, with no further information

Also, the new dashboards were developed considering different stakeholders. Apart from some further information for students and teachers, some dashboards were developed for game designers, developers or managers. In the following we use the term *developer* to comprise dashboards that would be of interest of both developers and game designers, who will work at different levels of the game but could obtain useful information from the same results.

The development of these dashboards began by thinking what data would be of interested to visualize for each of the previous stakeholders, and then it was necessary to figure out how to obtain that information from the xAPI traces (adding to it the knowledge of the game model in the first approach). With those ideas in mind, their development had as starting point which question we wanted them to answer, so we make those questions explicit in the following.

In each of the following dashboards, we explain the required analysis (and the particular index it corresponds to) as well as the different visualizations developed and included on it.

Teacher - top 30 questions with higher error ratio

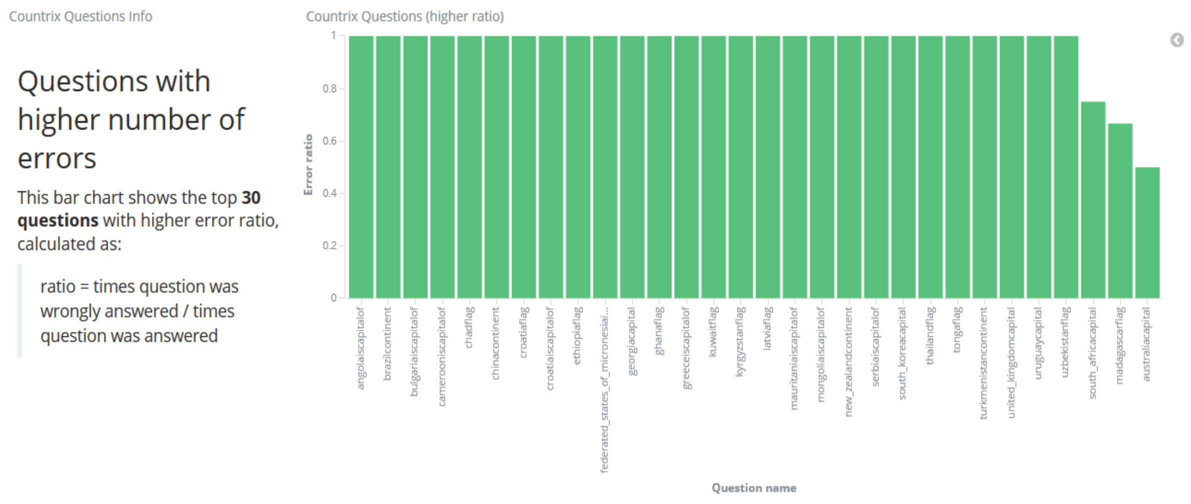
<u>Analysis</u> [questions]:

- The question name is obtained from `object.id`.
- The error ratio of each question is calculated as the number of times the question has been wrongly answered divided by the times it has been answered.

Both were calculated in the analysis increasing the corresponding variables each time the question is answered.

<u>Visualization</u>:

This dashboard aims to show the questions that are most difficult for players. With that purpose the questions were ordered in descendent order of error ratio and the top 30 was selected for proper visualization.


Countrix Questions Info

Questions with higher number of errors

This bar chart shows the top **30 questions** with higher error ratio, calculated as:

> ratio = times question was wrongly answered / times question was answered

Figure 26. Countrix: ranking of questions dashboard

It does not require access to the game model, but to the `object.id` of traces with `verb.id` *preferred* meaning an option was chosen among a set of possibilities. Whether there is an error or not can be obtained directly from the `result.success` field in the statements.

Teacher - ranking of users

<u>Analysis</u> [players]:

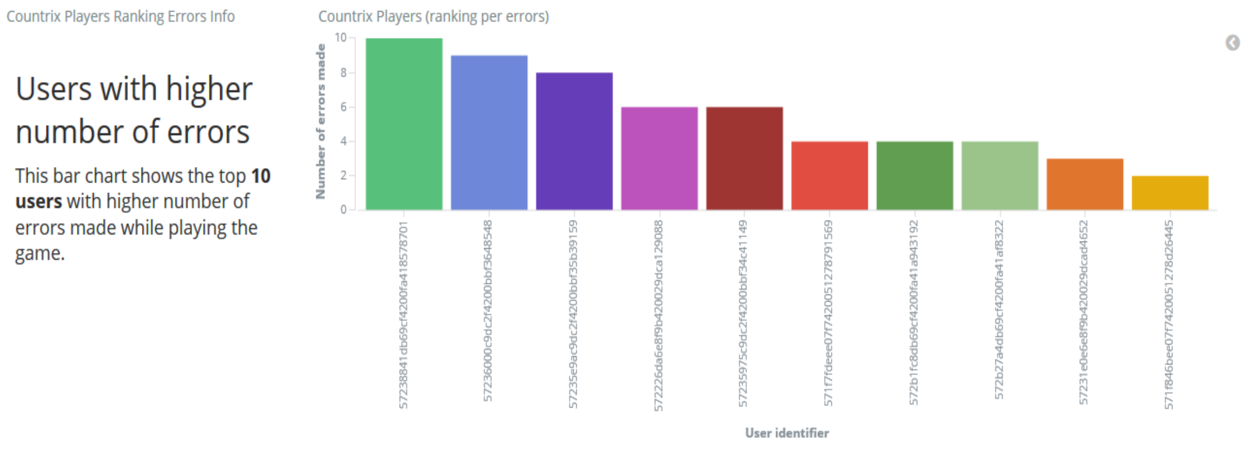
Each time a question is answered, we store

- the timestamp from the `timestamp` field
- the user from the `actor.name` field
- the question name from `object.id`
- and whether there is an error or not is calculated (this should be obtained automatically from the `result.success` field, which it was at that point not set by the game yet)

<u>Visualization</u>:

This dual dashboard shows the top 10 users with higher number of errors (respectively, correct answers) made, with their user id and count of errors (respectively, correct answers).

Here we show the error part of the dashboard, as the other is completely analogous:


Countrix Players Ranking Errors Info

Users with higher number of errors

This bar chart shows the top **10 users** with higher number of errors made while playing the game.

Countrix Players (ranking per errors)
Number of errors made
10
8
6
4
2
0
57238841db69cf4200fa418578701
57236000c9dc2f4200bbf3648548
57235e9ac9dc2f4200bbf35b39159
572226da6e8f9b420029dca1290088
57235975c9dc2f4200bbf34c41149
571f7fdeee07f742005127879 1569
572b1fc8db69cf4200fa41a943192
572b27a4db69cf4200fa41af8322
57231e0e6e8f9b420029dcad4652
571f846bee07f742005127 8d26445
User identifier

Figure 27. Countrix: ranking of users dashboard

As in the previous, there is no need to know the game model in order to develop this dashboard but to the xAPI fields made explicit in the analysis.

Teacher/Student - questions of a single user

Analysis [players]:

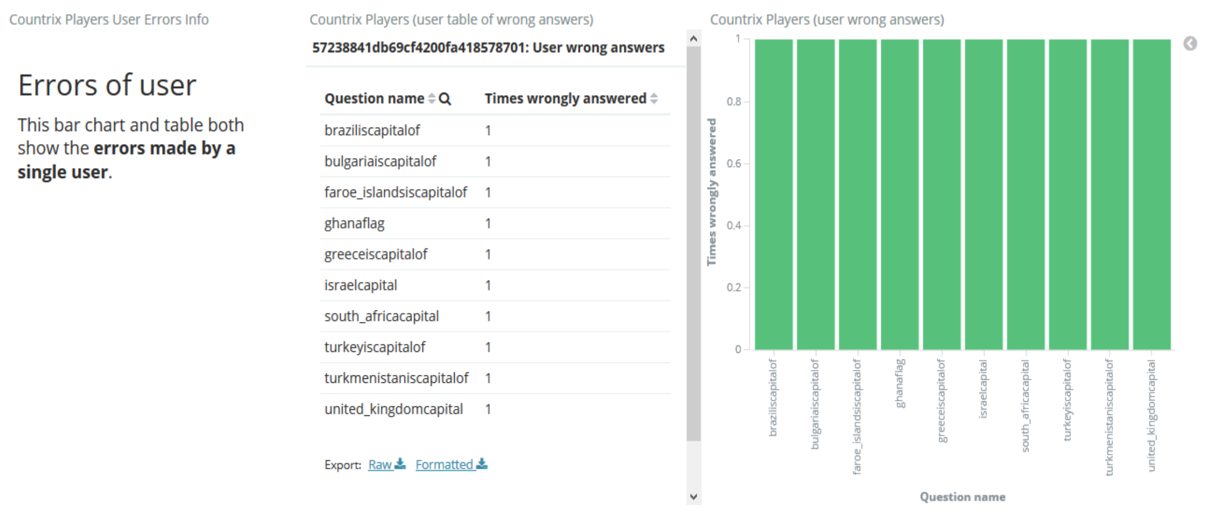
Same analysis as in the previous (ranking of users).

Visualization:

This dual dashboard shows the information of errors and correct answers of a single user in detail. This could be useful for teachers to obtain detailed information of a single student or for the students themselves.

It contains both a table and a bar chart of the questions that were wrongly answered by the particular user; as well as the same visualizations of the questions that were correctly answered by the particular user.

Here we show the error part of the dashboard, as the other is analogous:



Countrix Players User Errors Info

Errors of user

This bar chart and table both show the **errors made by a single user**.

Countrix Players (user table of wrong answers)

57238841db69cf4200fa418578701: User wrong answers

Question name	Times wrongly answered
braziliscapitalof	1
bulgariaiscapitalof	1
faroe_islandsiscapitalof	1
ghanaflag	1
greeceiscapitalof	1
israelcapital	1
south_africacapital	1
turkeyiscapitalof	1
turkmenistaniscapitalof	1
united_kingdomcapital	1

Export: Raw Formatted

Figure 28. Countrix: errors of user dashboard

This dashboard does not require access to the game model.

Student – individual information

<u>Analysis</u> [players]:

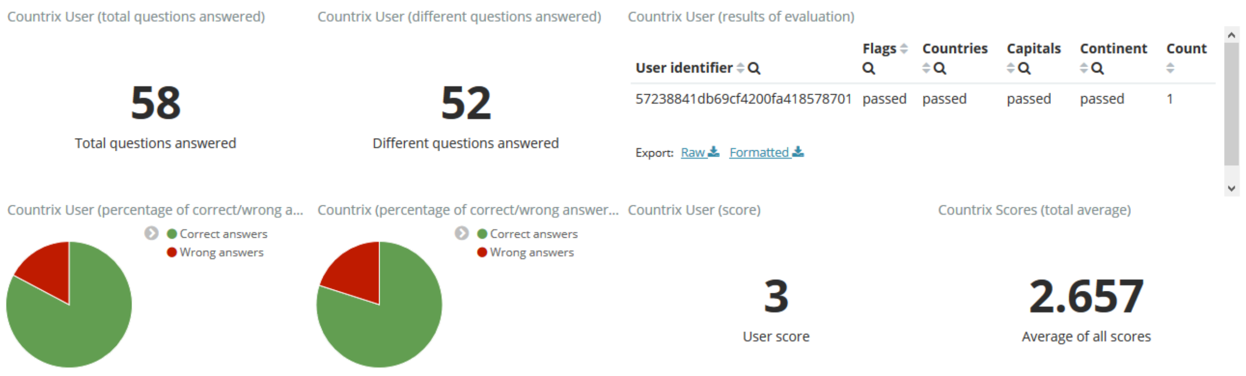
Same as in the previous (ranking of users).

<u>Visualization</u>:

This dashboard provides personalized information for a single user, showing he/she the results of the game session as well as different data of it. The top half of the dashboard includes:

- the total count of questions answered
- the count of different questions answered
- his/her results in the four topics evaluated
- a pie chart with the percentage of correct/wrong answers during the session
- a pie chart with the percentage of correct/wrong answers of all plays of the game
- the user's final score
- the average score of all sessions of the game

These last two visualizations aim to provide a way to compare the users' results with the average of all the sessions of the game.



Countrix User (total questions answered)

58

Total questions answered

Countrix User (different questions answered)

52

Different questions answered

Countrix User (results of evaluation)

User identifier	Flags	Countries	Capitals	Continent	Count
57238841db69cf4200fa418578701	passed	passed	passed	passed	1

Export: Raw Formatted

Countrix User (percentage of correct/wrong a...

Countrix (percentage of correct/wrong answer...

Countrix User (score)

3

User score

Countrix Scores (total average)

2.657

Average of all scores

Figure 29. Countrix: student's personal information dashboard part 1

The bottom half of the dashboard shows:

- the total amount of time spent in the session (in milliseconds)
- the timestamp of the answers of the student
- a bar chart containing all the questions answered by the user correctly
- a bar chart containing all the questions answered by the user wrongly

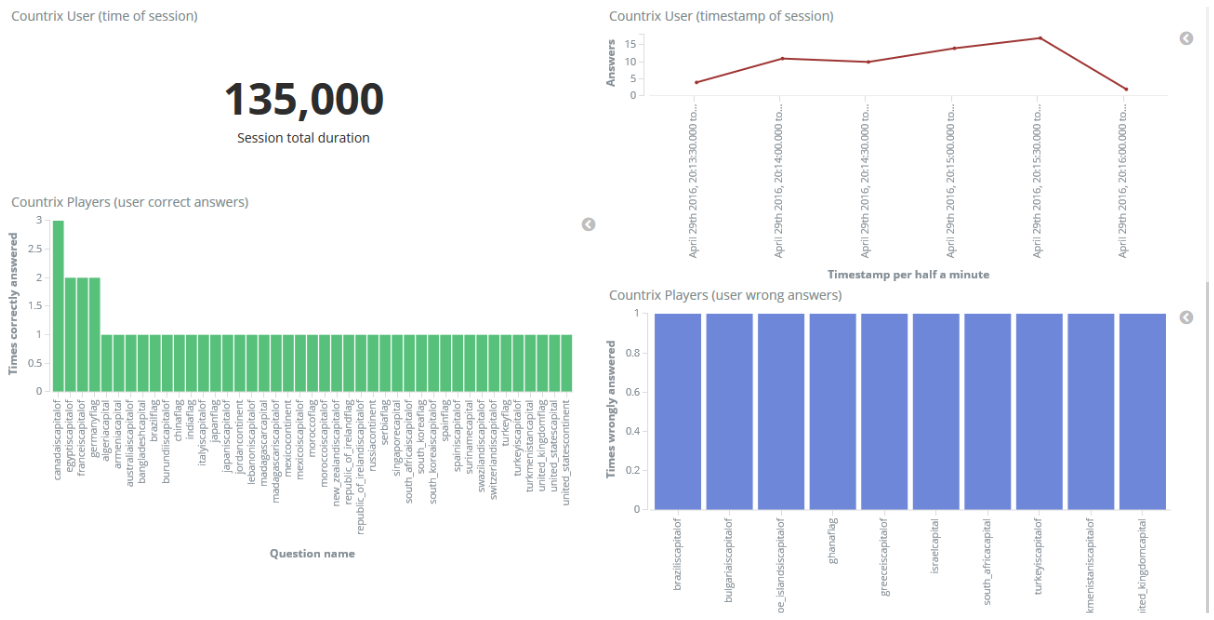


Figure 30. Countrix: student's personal information dashboard part 2

Some part of this dashboard could be developed independently of the game model: timestamp and session duration, errors and correct answers (aggregated by `object.id` of `verb.id` *preferred*, without explicitly knowing that is a multiple-choice question), count of traces of verb *preferred* and different answers (with `object.id` as before).

The student's results in the four topics evaluated, however, does require access to the game model to be able to classify the questions into the four types.

Teacher – results of the students

Analysis [results]:

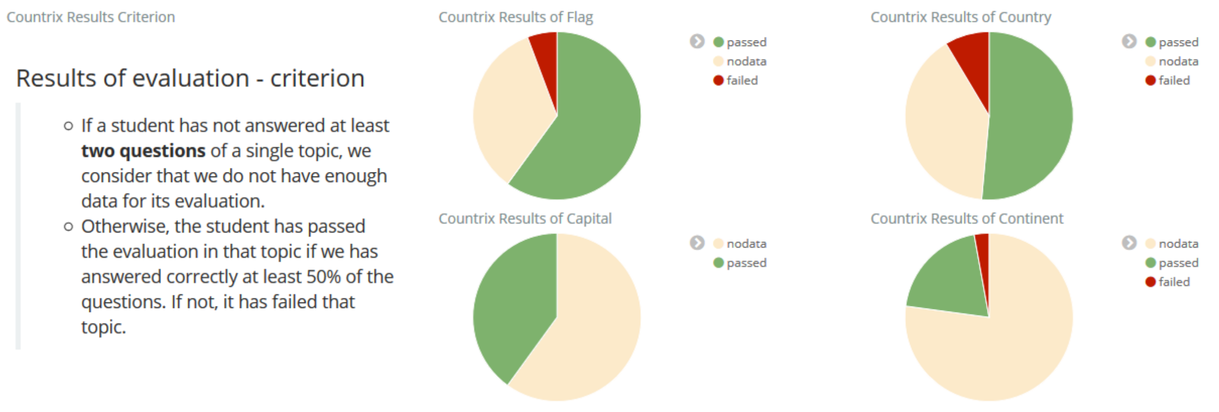
When a new session starts, we create 4 pairs with value (0,0) corresponding to the (correct answers, total answers) for the 4 topics evaluated for the user. Each time the student answers a new question, the pair of its corresponding type is updated accordingly.

Finally, the results are stored following the criterion:

- if number of questions answered is strictly less than 2, no data
- otherwise,
    - if correct answers is strictly less than 0.5, failed
    - otherwise, passed

Visualization:

This dashboard for the teacher aims to provide a classification of the students in terms of academic goals achieved. Assuming each of the four types of questions corresponds to a topic to be evaluated, this dashboard first shows the results of the students in each of the four topics, according to the classification described in the analysis.

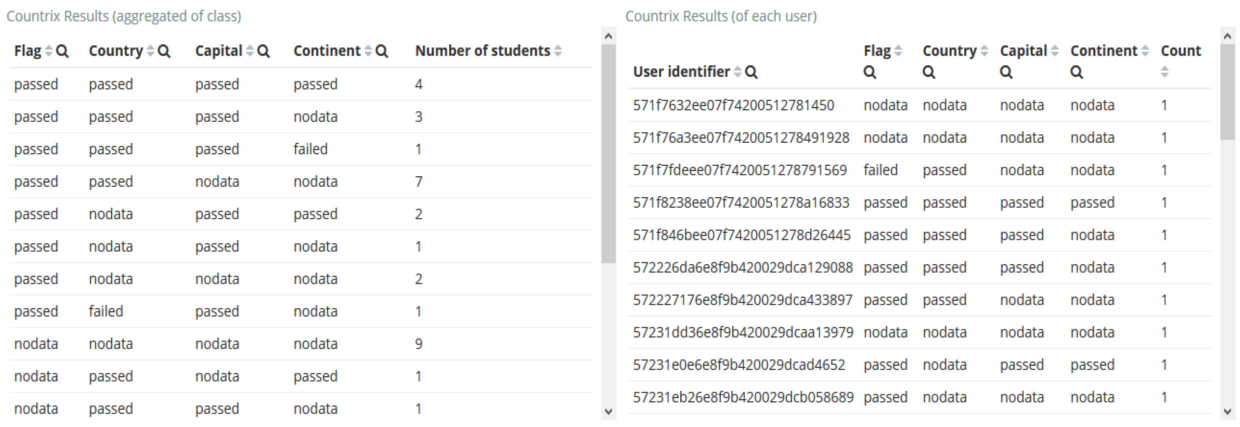

Countrix Results Criterion

Results of evaluation - criterion

- If a student has not answered at least **two questions** of a single topic, we consider that we do not have enough data for its evaluation.
- Otherwise, the student has passed the evaluation in that topic if we has answered correctly at least 50% of the questions. If not, it has failed that topic.

Figure 31. Countrix: results of evaluation dashboard part 1

The second part of the dashboard contains the results data for more detailed evaluation: first, aggregated by results to obtain a general information of the results of the class; and then, with a complete data table with the results of each student in the four evaluated topics.



Countrix Results (aggregated of class)

Flag	Country	Capital	Continent	Number of students
passed	passed	passed	passed	4
passed	passed	passed	nodata	3
passed	passed	passed	failed	1
passed	passed	nodata	nodata	7
passed	nodata	passed	passed	2
passed	nodata	passed	nodata	1
passed	nodata	nodata	nodata	2
passed	failed	passed	nodata	1
nodata	nodata	nodata	nodata	9
nodata	passed	nodata	passed	1
nodata	passed	passed	nodata	1

Countrix Results (of each user)

User identifier	Flag	Country	Capital	Continent	Count
571f7632ee07f74200512781450	nodata	nodata	nodata	nodata	1
571f76a3ee07f7420051278491928	nodata	nodata	nodata	nodata	1
571f7fdeee07f7420051278791569	failed	passed	nodata	nodata	1
571f8238ee07f7420051278a16833	passed	passed	passed	passed	1
571f846bee07f7420051278d26445	passed	passed	passed	nodata	1
572226da6e8f9b420029dca129088	passed	passed	passed	nodata	1
572227176e8f9b420029dca433897	passed	passed	nodata	nodata	1
57231dd36e8f9b420029dcaa13979	nodata	nodata	nodata	nodata	1
57231e0e6e8f9b420029dcad4652	passed	nodata	passed	passed	1
57231eb26e8f9b420029dcb058689	passed	nodata	nodata	nodata	1

Figure 32. Countrix: results of evaluation dashboard part 2

This dashboard requires access to the game model to know that there are 4 types of questions and classify each question in its corresponding type.

Developer – general information about the game

Analysis [players]

It requires:

- count of different `actor.name`
- count of different `object.id` when `verb.id` is *preferred*
- `timestamp`
- analysis of whether there is an error or not (`result.success` field)

Visualization

It contains:

- count of sessions
- count of different questions answered
- count of questions answered
- pie chart showing percentage of errors and correct answers of the total
- line chart of timestamp of answers

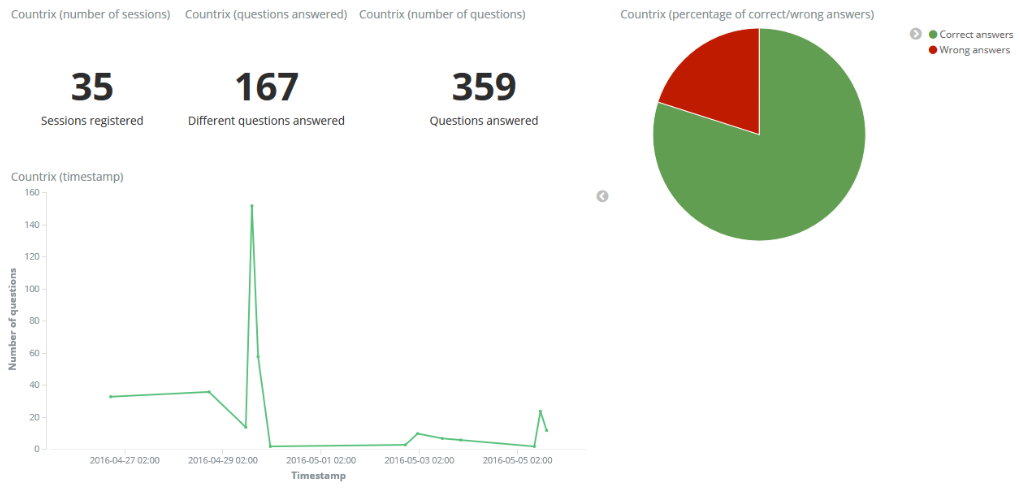


Figure 33. Countrix: general information dashboard

It does not require access to model, only to the fields: `actor.name`, `verb.id`, `result.success` and `timestamp` in xAPI statements.

Developer – scores obtained in the game

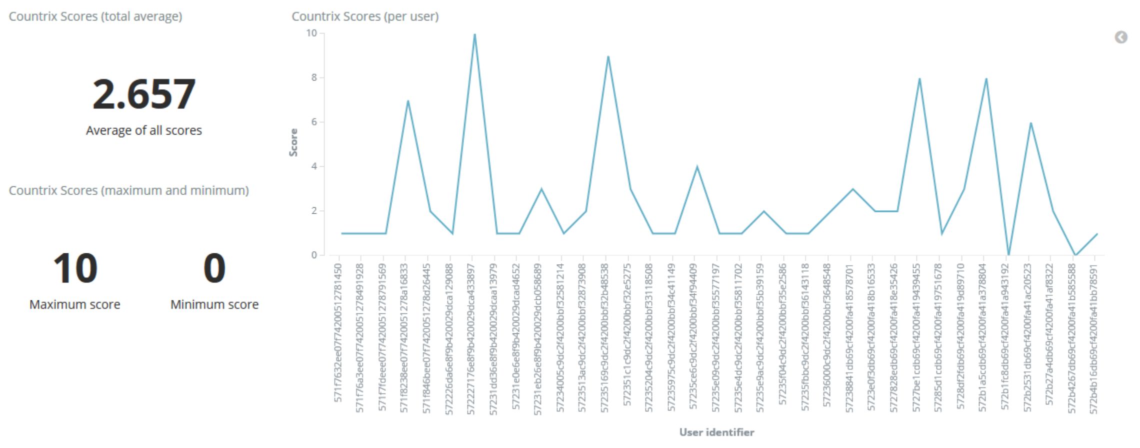
Analysis [players]:

For each user, it stores the score obtain in each statement sent by the game with `object.name` score and keeps only its final value.

Visualization:

It contains:

- the average of all final scores registered in the game
- the maximum and minimum scores registered
- a line chart with the scores obtained in each session



Countrix Scores (total average)
2.657
Average of all scores
Countrix Scores (maximum and minimum)
10
Maximum score
0
Minimum score
Countrix Scores (per user)
Score
User identifier

Figure 34. Countrix: scores dashboard

This dashboard requires access to the `object.id` field in the xAPI statement, check that it is a variable named score and access to its value in the `result.extensions.http://rage-eu.com/xapi/extensions/value` field, so it does require at this point some knowledge of the vocabulary that the traces sent by the game are using.

Developer – how many questions each user answers

Analysis [sessions]:

For each user, count of total questions answered by counting statements with `verb.id` *preferred*.

Visualization:

It contains a line chart of number of questions answered per session and the mean number of questions answered per session.

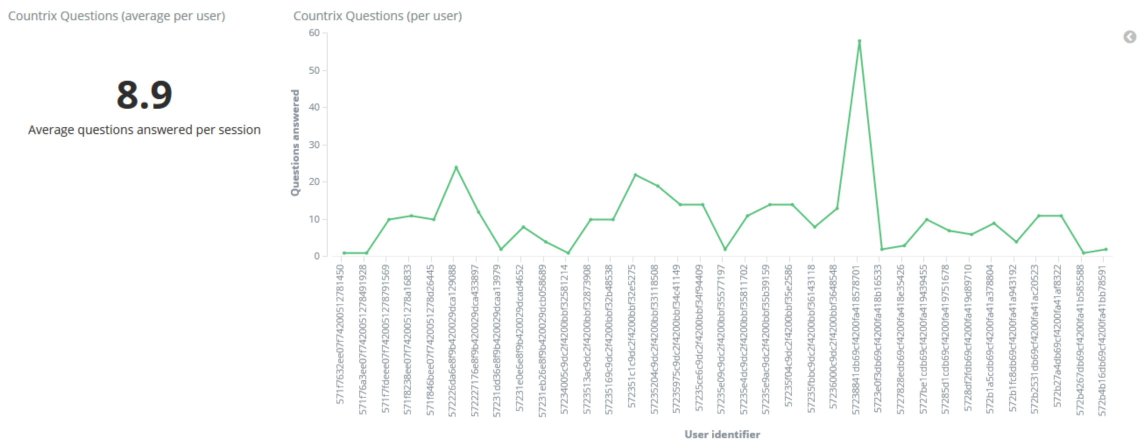


Figure 35. Countrix: questions per session dashboard

It does not require access to model, only to `user.name` and `verb.id` *preferred* in xAPI statements, which means that an action was chosen among several possibilities.

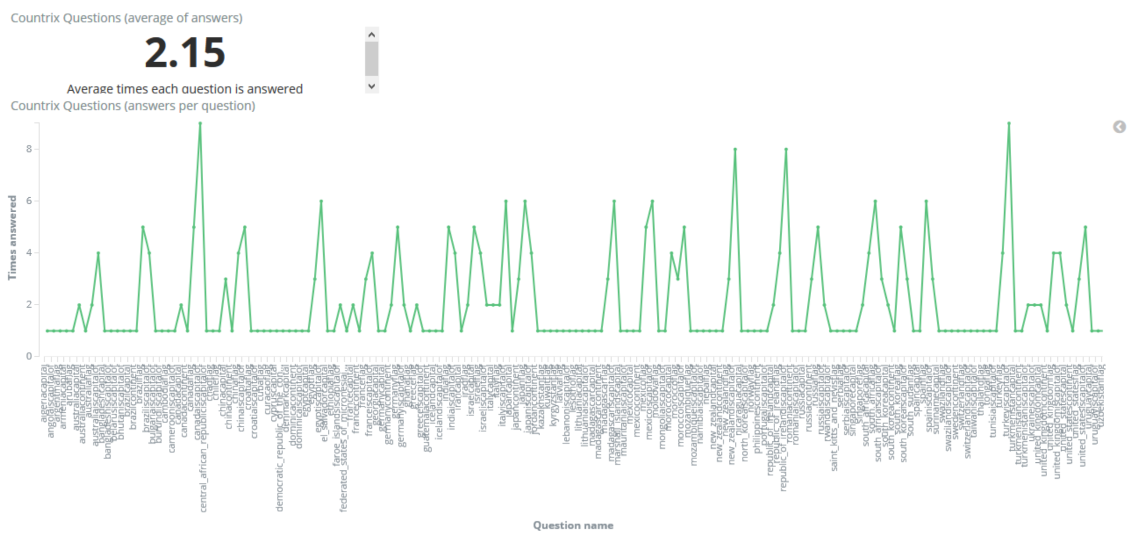
Developer – how many times each question is answered

<u>Analysis</u> [questions]:

For each question (given by object.id when `verb.id` is *preferred*), count of times answered.

<u>Visualization</u>:

This dashboard aims to give an idea of how randomly are questions selected. It contains a line chart with the times each question has been answered and the mean of those times.



Countrix Questions (average of answers)
2.15
Average times each question is answered
Countrix Questions (answers per question)
Times answered
Question name

Figure 36. Countrix: answers per question dashboard

It does not require access to model but to the `object.id` in the xAPI statements of `verb.id` *preferred*, meaning there is a possibility among several options, and takes the name of this multiple-choice from the `object.id` field.

Developer –how many questions of each type, correctly and wrongly, are answered

Analysis [questions]:

The analysis adds to the information required for the previous dashboard (how many times each question is answered):

- the count of errors per question
- its question type obtained by parsing the question name

Visualization:

It contains four pie charts showing the proportion of correct answers and errors for each type and a bar chart showing the number of questions answered of each of the four types of questions.

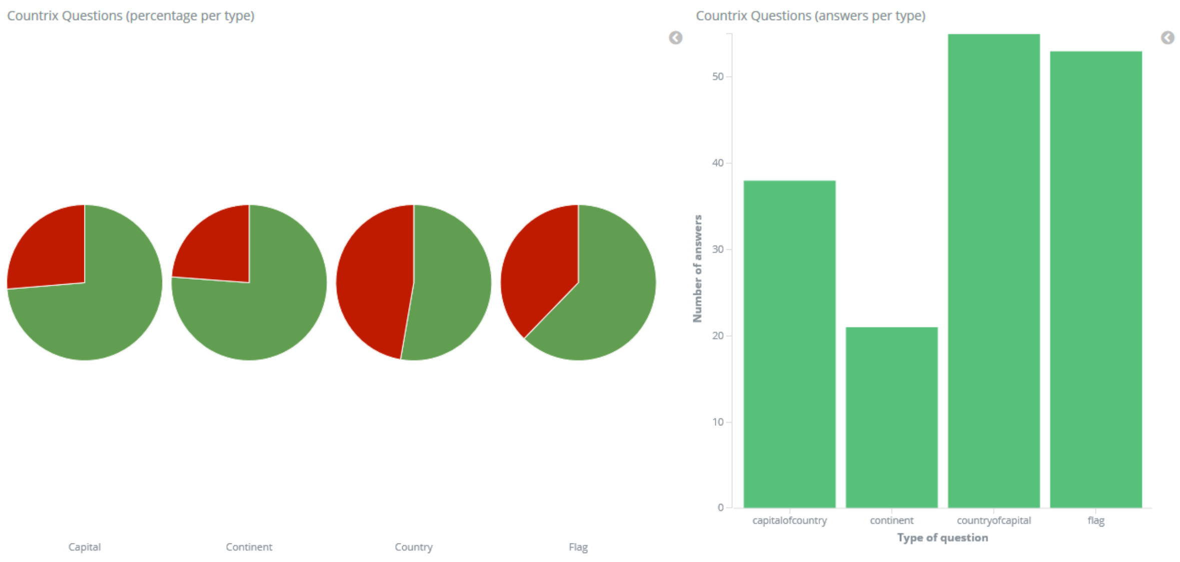


Figure 37. Countrix: questions per type dashboard

It requires access to the game model to parse the question names to classify them according to the four known types of questions.

Developer – how long game sessions last

<u>Analysis</u> [sessions]:

For each different `actor.name` it stores the first and last `timestamp` to calculate the duration of the session by subtracting the former to the latter.

<u>Visualization</u>:

It contains a line chart with the duration of each play session and the mean of those times, both expressed in milliseconds

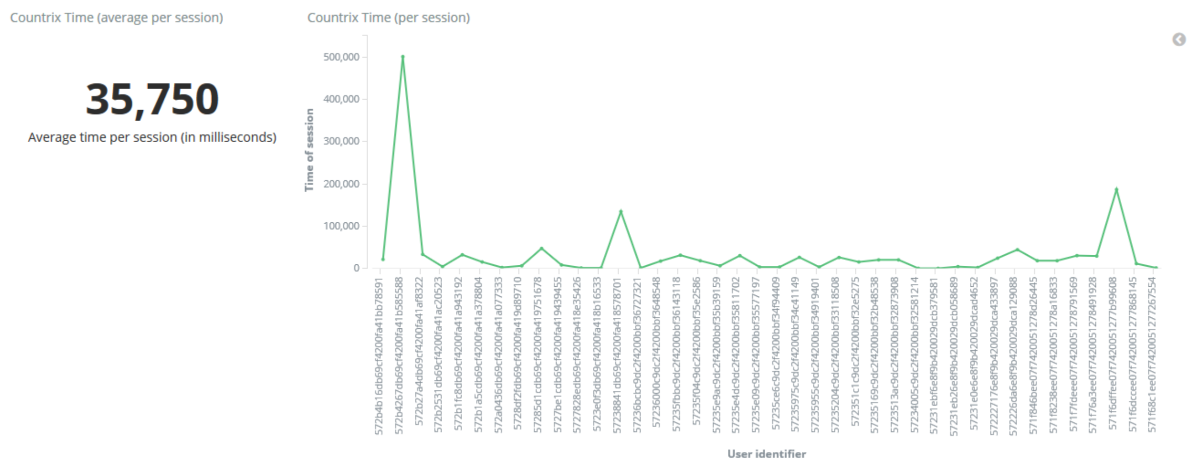

Countrix Time (average per session)
35,750
Average time per session (in milliseconds)
Countrix Time (per session)
500,000
400,000
300,000
200,000
100,000
0
Time of session
User identifier

Figure 38. Countrix: time of session dashboard

It does not require access to the model, but only to the `timestamp` and `actor.name` fields in xAPI statements.

Developer – how many times each xAPI verb is used

Analysis [xAPI]:

For each statement, we store the `verb.id` used counting its appearances.

Visualization:

It contains the count of different verbs used, an area chart with the number of times each verb is used and the mean of those times.

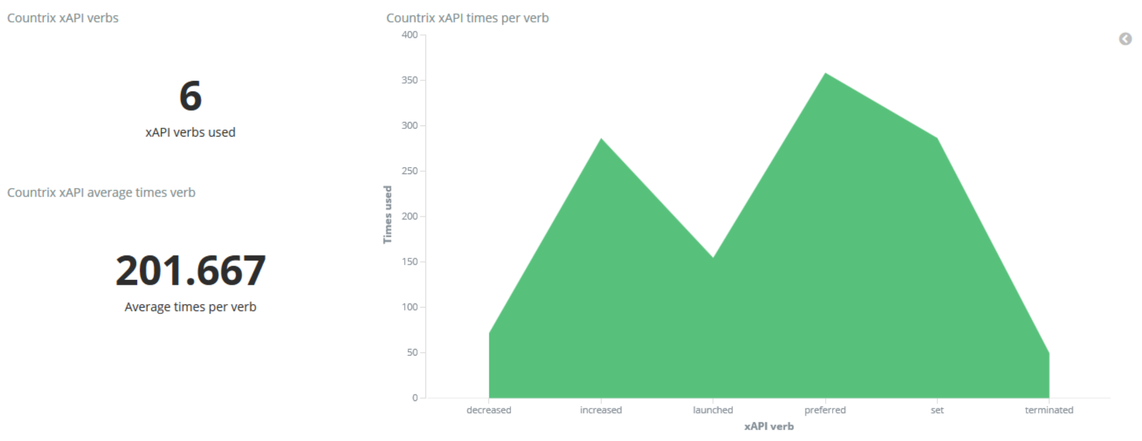


Figure 39. Countrix: xAPI verbs use dashboard

It does not require access to the model, only to the `verb.id` field in the xAPI statements.

Manager – peak times of game use

Analysis [peak times of use]:

For each statement, we calculate its minute of the day from the `timestamp` field. We keep in a 24x60 matrix the number of users per minute registered.

Visualization:

This final dashboard aims to give the manager the information of the use of the game, in terms of the number of users connected to it per minute. It shows:

- the maximum number of users connected in one minute registered
- the average number of users per minute
- a general chart showing the distribution of users per hour in the game registers
- a bar chart of number of users connected to the game aggregated per minute of the day

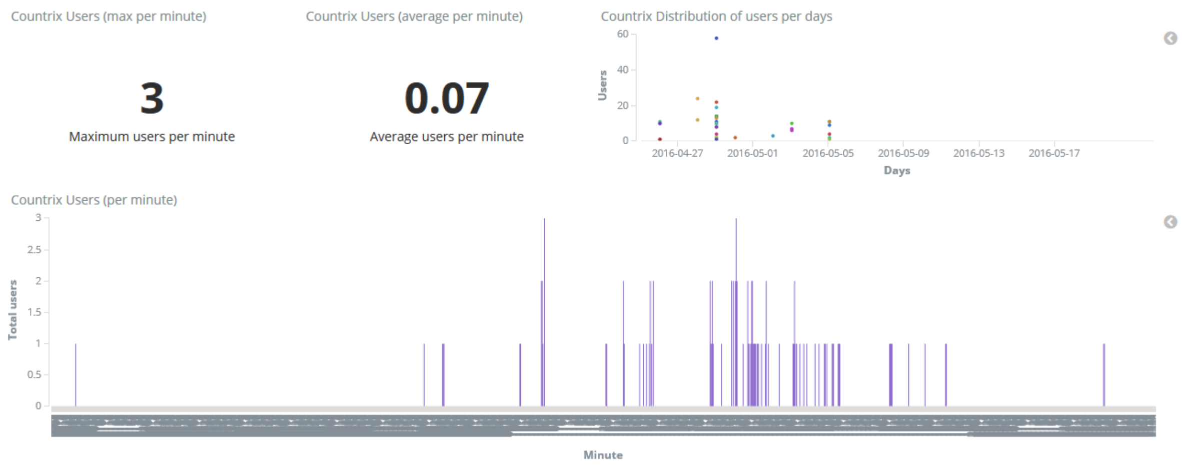


Figure 40. Countrix: peak times dashboard

It does not require access to the game model, but simply to the `timestamp` field.

Integration with RAGE project

The final desired step is that these developed visualizations and dashboards are to be actually used having a greater impact. For that, it was intended to complete their integration with the architecture previously described, as part of the RAGE project. For this correct integration, we count on the help of Antonio Calvo and Dan Cristian Rotaru, who are working closer to the architecture.

As we mentioned before, RAGE is a European project to improve the development of gamed applied to education. Its aim is to provide self-contained gaming assets together with several resources of high-quality knowledge through a common social space that connects every part with an interest on it: research, gaming industries, intermediaries, education providers, policy makers and end-users [The Open University of the Netherlands, University of Trento, Hull College of Further Education, Universidad Complutense de Madrid, The University of Bolton, INMARK, Instituto de Engenhariade Sistemas e Computadores, Investigacao e Desenvolvimiento im Lis 2015].

As we made explicit in the previous chapter, the visualizations and dashboards were initially developed in a now-old version of Kibana, and later moved to a newer version. Besides the reasons previously mentioned for this change, it was mainly looking forward to this integration why the developed visualizations and dashboards in Kibana were updated to this last version, as it is the version currently working in the RAGE architecture.

The main focus is now set into the visualizations and dashboards developed for the game Countrix. The traces sent by this game are richer and follow the xAPI standard and, therefore, the graphics developed based on them contain greater information, and could be of interest for the aim of this project.

After transferring the visualizations to Kibana 5.0.0-alpha2, in order to be able to integrate them with the architecture, it was needed to provide some further information about the analysis and visualizations developed. For that, the analysis made in Python was specified in a detail document containing the required fields in the xAPI statements needed for each of the analysis performed and what calculations were made with that data. The analysis was then to be translated into the corresponding analysis as topologies of Storm Flux.

For the visualizations, the whole list of Elasticsearch indices and types was provided, specifying for each of those, all the contained fields and their corresponding types (date, integer, float, bool or string). These indices were then replicated in the RAGE server. For the visualizations and dashboards, a plugin is being developed to upload a template of a visualization or dashboard and obtain the different visualizations from it.

Conclusions and future work

Finally, we summarize here the conclusions and contributions of the project, as well as some ideas for future work that were not possible to include in it.

Conclusions

The field of Gaming Learning Analytics is still growing and several issues are still to be solved. Many companies, however, are already working on it as it is perceived by many people as the future of education. The focus is then set into achieving a complete architecture that covers the whole process from capturing the interactions of a student with a serious game to visualizing the results in the shape of dashboards.

In order to complete this process, it is important to ensure a solid common ground of knowledge and tools that allow to achieve better and more reliable results. One of the key tools for this is a common vocabulary and tracking model for specification and a standard API for the traces sent by a game and stored into a LRS. The model proposed by Ángel Serrano and its implementation with the Experience API aim to cover this step.

To develop the analysis and visualizations, there are several open options and it is still difficult to find one that suits all the needs for our visualizations. In spite of this, Elasticsearch, Logstash and Kibana have proved to be a good combination of tools to store a big amount of data, filter it, aggregate it or perform different calculations with it, transfer it, and develop different visualizations and dashboards with it.

The characteristics of the information obtained depend on several factors, among which we have seen the following that stand out: how much knowledge do we have of the game, how much information is the game sending, how complete is this information, or how much level of knowledge are we obtaining the information with.

Some of these issues are still difficult to solve. The possibility of having access to the game model could provide richer information to visualize but it would make the analysis and visualization dependent of the game, and therefore much less scalable and reusable. It is for these reasons that we consider that the perspective of not having access to the game model should get the biggest attention, and with it, try to obtain as much information as possible of the game. As we have seen in the examples, this is indeed feasible as xAPI statements are quite complex statements that comprise much information. They provide an instant report of what is happening inside the game at every moment, given the information of every interaction that happens between the player and the game.

The results of these analysis and visualizations may also have different uses. Different stakeholders may benefit from the same information, with maybe different levels of knowledge. While teachers could see results of what is happening in a class as well as having algorithm criteria for students' evaluation, the students may see at the end of each play how much knowledge of a topic they have, either with individual data or by comparison with the rest of their class. Developers of the game could also obtain a lot of information to determine whether the game is working for its intended purpose or not, of if there is some spot to fix on it. The results information could also be useful for managers of the game, giving them an idea of the use of it to provide the needed infrastructure so there are no problems of collapse if too many users are playing at the same time.

Contributions

In the several stages of the process of GLA we are heading towards having as general structures as possible to be able to provide some information for any given serious game. In this sense, the tracking model proposed by Ángel Serrano together with its implementation using ADL's Experience API standard [Serrano-Laguna et al. 2016] achieves that intended goal for the first step of obtaining the data from the users' interaction with the game.

Following this same idea, this project aims to be a first step towards a similar generalization in the analysis and visualization steps. Particularly, the developed analysis and visualizations try to show how much information we are able to obtain having no further knowledge of the game than the one provided by the xAPI traces. The shown dashboards aim to cover the variety of possible stakeholders with their different interests in the use of serious games to prove that this approach still can work for each one of them, and that interesting results may be obtained even considering the game as a black box that merely sends traces to the outside.

In particular, developers or game designers may use the obtained information to improve the game by accessing data such as verbs' frequency of use or mean times of play sessions. The use of xAPI verbs in games could provide an insight to the models' accuracy, as we may discover the real usefulness of the proposed verbs, if some of them are too less used may be unnecessary while if some are overused may be better split into more specific ones.

The percentage of errors made by users could also show if there is some bug in the game or if some of the tasks proposed are too difficult (or too easy) for players and should be redesign. Another interesting information for developers is the use of the different objects provided. As we showed in the example with Countrix, although the questions were selected randomly from a predefined set, we found some questions with significant more appearances than others. This simple idea could extend to other examples to find defined objects in the game that are never used and therefore are not necessary and may be eliminated.

The whole aggregated information of the traces sent by the game also provides an overview of the *areas* in the game that are explore by players, and how many times this occurs. In the examples, these corresponded to questions but in different cases may correspond to levels or zones in the game. The information collected of their use may show their reachability, potentially identifying parts of the game defined in the code but that are never actually reached in the gameplays.

For managers, it is important to know that games are supplied with a stable infrastructure that can manage the amount of users that may want to play at a given moment of time. For this, the information of use of the game and peak times is essential. Identifying information such as mean number of users per minute of the day and its maximum achieved values will help managers to predict a potential risky situation and provide the game with the appropriate resources to cope with this issue.

Both teachers and students will focus on the level of achievement of the intended educational goals of the game. The developed dashboards show the variety of information that we could provide to help teachers in their evaluation task but also to give them information for directly being able to help students having trouble with some concepts or to provide extra work for advanced students. Whichever the situation of the students, the feedback provided by the dashboards will give them an idea of how they are doing in the course and therefore help them to continue or change their approach to the study for the course.

The generalization of the process of GLA that we are seeking was another reason to try to integrate these visual results with the RAGE project. The developed architecture of RAGE covers the whole process from tracking data of the users' interactions to visualizing results and is applicable to any given example of serious game.

For this long-term purpose it was necessary to first study the whole architecture of RAGE. Being part of a European project, RAGE's architecture is a valid and reliable example of a GLA architecture. Although it is already working and stable, the architecture is also in continuous development and improvement with the work of the e-ucm research group. Therefore, it was necessary to keep track of the changes on it, as some of them were occurring during the course of this project, and to update consequently the information gathered from it to the latest version.

To get familiar with this architecture, it was necessary to deal with a broad variety of tools developed from many different companies and developers, most of them of recently creation. Although the focus of this project was set into the analysis and visualization stages of the process, the study and comprehension of the tools of all the different stages of the process was important to have a complete overview of the GLA process and its implementation in this architecture.

The integration of the developed analysis and visualizations as part of this European project was key to ensure the real use and applicability of the work developed in this project. With that achievement, it is likely that the work done in this project will have a broader range of uses and could be used with different examples of serious games, providing valid results for different users.

Specifically, the integration of the project with the RAGE architecture requires two steps. The analysis developed are set to be replied as Storm topologies, while the visualizations made in Kibana can be directly reused as part of the architecture with different examples of serious games. For this reason, it was also interesting to develop as many visualizations as possible with no knowledge of the game model, trying to extract all the information from the xAPI traces.

To summarize, the personal contributions in the project were:

- Study of the GLA process and its variations
- Study of the architecture of the RAGE project
- Practical application of the process of GLA with two examples of serious games

This last step required the following contributions:

- Design of an analysis model for the Experience API.
- Implementation of the analysis in Python.
- Definition of a set of Elasticsearch indices to comprise the information.
- Implementation of visualizations and dashboards in Kibana.
- Consideration of different stakeholders for visualizations.
- Consideration of different levels of game knowledge for visualizations.

Future work

There are several improvements that could be incorporated into the work developed but were not feasible due to the lack of time or their complexity, as they could not be easily brought to work together with the chosen tools.

Some of them are made explicit in the following:

- Include pop-up alerts and warnings to be shown to the teacher during a game session in the class for special cases: for instance, if a user has been inactive for too long or if a user has made too many mistakes. This way, the teacher could immediately act accordingly to help those students. Also, this could be used for advanced students who may finish too soon with their given tasks and may need some extra tasks to work on.
- Incorporate tools for students' motivation. Analyzing the students' use of the game, they could receive motivation messages: for instance, notifying a good streak and encouraging them to continue with it. This could be used as well to give some message to students that have stayed too long without playing the game in order to motivate them to come back to it.
- Broaden the scope of interactions with dashboards allowed. For instance, it could be interesting to be able to make more complex queries in a displayed dashboard such as fixing intervals of data to be considered or values to aggregate data.
- Try out the set of analysis and visualizations stablished as not needing access to the game model with other already working examples of serious games and check if we could broaden the range of visualizations or dashboards to provide further general information.
- Try out the set of dashboards developed with students and teachers in a real classroom context to check if the information provided covers all their interests or if some additional dashboards could be developed.
- Study examples of serious games to see if we could find some common features that will allow to classify them in some categories according to their characteristics. Then we could develop some analysis and dashboards that, although they will not be game-independent, could be used for all the games on each of the categories found. This could be a half-way approach between our goal of completely independent of the game analysis and visualizations and the ones developed for a specific serious game.

Appendix 1: Project timeline

This appendix aims to give an overview of the progress of the project development during the last months focusing on the several meetings made. Most of the meetings were with the director of the project, Baltasar Fernández-Manjón, with Ángel Serrano, PhD student working in his research group and with Antonio Calvo and Dan Cristian Rotaru, two of the Master students also working in his research area.

1. February 15th, 2016

Meeting with Baltasar. The project will focus on Serious Games Learning Analytics and Visualization. Goal of the project (coarse-grained): starting from some data collected of the interaction of users/students with a serious games, perform different analytics on that data to obtain the relevant information for the learning purpose. The result of the analysis will finally be shown to provide information about the learning process.

Start by reading paper [Freire et al. 2016] and book [Loh, Sheng, and Ifenthaler 2015] to get used to the new topic.

2. February 24th, 2016

Meeting: presentation of Ángel Serrano's theoretical model in his thesis about Learning Analytics with the xAPI standard. This model is an attempt to describe the format that the data should be sent to the server from the game (client) using the xAPI standard. We can see the draft of the model in GitHub [Serrano-Laguna 2016]

3. March 2nd, 2016

Meeting with Baltasar. We fixed the language of the project to be English. Things to-do:

- Start to write down the timeline of the project (meetings, tasks, and so on).
- Start to write down the project report:
    - 0: Description (goal and motivation) (improve from past version)
    - 1: Gaming Learning Analytics
    - 2: Infrastructure of the project
- Get familiar with the architecture of the whole project in one week.
- Have a little demo of analysis and visualization working in two weeks.

The project will have different visualizations depending on the context, user, and so on. There will be general analysis as well as game-specific ones.

The ideal goal of the project would be to have a software that, once a game is running, can automatically detect on the Learning Record Store (LRS) every message the game sends. And so, from those messages, it can perform an analysis and show some visual results.

(Future goal: think about Angel Serrano's theoretical model and how we could formalize it, for instance with an automata with actors and so on that makes the whole representation compact.)

Also, arrange a meeting with Dan and Antonio to explain to me the infrastructure of the project. In return, I will help them writing in English the description on GitHub of the architecture they are working in.

4. March 4th, 2016

Meeting with Dan and Antonio who are working in the infrastructure of the project. They explain the details of the infrastructure as detailed in the GitHub repositories and in the project report: tracker, A2 [e-ucm 2016d], collector, LRS [Apereo 2016c], real-time analytics, database, web frontend [e-ucm 2016b] and backend [e-ucm 2016c]. It all can be split into three connected blocks: configuration, analysis and visualization.

Specifically, I get to know tools such as Kafka [e-ucm 2016f], Storm Flux [Apache 2016] for analysis, MongoDB [MongoDB 2016] for storing and D3.js Data Driven Documents [Bostock 2016] and Kibana [Elastic 2016b] for visualization. They also gave me some hints for the demo.

5. March 15th, 2016

Meeting with Dan and Antonio to solve doubts about the architecture explanation: the use they do of Storm Trident, which can be integrated with Storm Flux and so on. They show me an example of the whole process from data collection to visualization: in server https://rage.e-ucm.es it shows statements with information of the data collected. Goal: being able to specify which data, which analysis and how to store it in Mongo.

They also show me OpenDashboard, a visualization tool they have just discovered made by Apareo (which also made OpenLRS) that will read the data from MongoDB. They think at this point that this tool may solve two of the three stages of the process: the configuration and visualization.

To-do: investigate both Kibana and OpenDashboard and look for similarities and differences.

6. March 17th, 2016

Meeting with Baltasar. Discussion of the whole context of the project: we will have different visualizations depending on the stakeholder (student, teacher, manager). We will focus on the teacher. I explained the demo done in D3js.

Goal: make something extensible (clear architecture, easy to develop –API-).

Things to-do (Holy Week):

- Investigate Kibana, OpenDashboard and make some more visualizations using different tools than D3js. See pros and cons of them.
- Make graphic of the process to clarify ideas.
- Investigate ADL (xAPI) and Apareo (OpenLRS and OpenDashboard) and write down about them.
- Re-read and re-write report of project, update with new papers and information resources.

7. March 31st, 2016

Meeting with Baltasar. Report of progress made, basically working with Kibana and documenting everything I'm doing and reading about.

Things to-do for the next two weeks:

- Explore other uses of Kibana (Bitergia read the presentations [Gonzalez-barahona 2016a] and [Gonzalez-barahona 2016b]).
- Keep working and exploring visualization tools.
- Investigate about the Apareo initiative.
- Add to the report as a final appendix this document of meetings.
- Have a demo working of the whole process: take a game that may be simple at first (e.g. QuizDemo [e-ucm 2016g] which sends traces in xAPI), detect that traces and make some visualization with information about them, for instance, the mean play time.

8. April 4th, 2016

Meeting with Dan and Antonio to talk about Kibana and how their progress is going. Things to check about Kibana (both things work fine in Kibana):

- Visualizations are automatically updated at *almost* real-time when new data is added to the corresponding index in Elasticsearch.
- Visualizations and dashboards can easily be exported and embedded into a webpage in html.

They also gave me an executable file of the game QuizDemo that sends traces of game sessions in csv format, to make the demo based on this game. Idea of the demo process: make some script (e.g. in Python may be interesting) to split the traces file into sessions, obtain the useful information of them and store that information into Elasticsearch (using Logstash) and make some visualizations of it using Kibana.

Another issues to work with can be related with Storm Flux, as it may be interesting to make some topologies for it. In particular, thinking about what analysis may be most interesting to visualize: mean time of play sessions, histogram of scores, levels where most mistakes are made, most common mistakes, final and mean score and so on…. Think about metrics that are interesting to see for the teacher. And once we have that metrics clear, think about what data requisites we will have then: for instance, to visualize the mean score we need the score file to be numeric.

9. April 12th, 2016

Meeting with Dan to check how the work with Kibana is going. We make some sample visualizations using the data in rage with Kibana. Issues to work on:

- How to make visualizations with strings using Kibana, for instance: graph that shows the final score aggregated by user id.

10. April 20th, 2016

Meeting with Baltasar. We talk about the progress of the report so far. Specifically:

- References need to be checked using the DOI in Mendeley so they are completed.
- Another high-level diagram may be added, showing the division in the three main parts of the architecture (collection, storage, analysis and visualization).
- Extend the xAPI explanation.

- Add a section in the report specifically explaining the mathematical component of the project so that it is clearer (data analysis and big data, visualization and so on).
- Optionally, it may also be interesting to add a brief outline of the demos that I am working on including some snapshots of them.

In the next week or two, besides fixing those things of the report, focus on continuing working with the analysis and visualization part.

11. April 25th, 2016

Meeting with Antonio and Dan to show them the work done with Kibana so far (visualizations of scores, errors, time of play). They gave me some ideas to continue working with:

- Kibana plugins: to create new visualizations. List of plugins to investigate: tag cloud, heat map, vector map, html code plugin, and so on. It may not be necessary to create our own visualization graphics but good to know them.
- In the link that Kibana provides to share a visualization, we can configure filters so we could have general visualizations and then filter them directly in the link, rather than on the web interface. Investigate how broad this option is: e.g. filter a numeric field on greater than or equal to values.

They also suggest that I write down some sample cases of the visualizations made, explaining the steps made to configure them, so they can easily be reused and also add that work to the report.

12. May 3rd, 2016

Meeting with Baltasar and Ángel Serrano. Presentation of the new game Ángel has developed: Countrix [e-ucm 2016h]. It is a Q&A game that shows the xAPI traces following Ángel theoretical model definition.

I also show them the new visualizations made as well as the use of Kibana filters to have more general visualizations that can be particularized later on.

To-do: write down the work done in the first example with QuizDemo and start working with Countrix to have visualizations ready for next week.

13. May 10th, 2016

Meeting with Baltasar. We talk about the theoretical aspects of the game. Specifically, there are two levels we can approach the data from: either if we know the game model or if we do not. With that in mind, it is interesting to see how much information we can extract from the xAPI traces depending on the level of knowledge.

Following this discussion, things to-do in the next week:

- Write down the conceptual idea of the project, deepen into the theoretical basis of it.
- Related to that, create a conceptual graphic to clarify those ideas.
- Check out advances in Angel's model with xAPI.

- As of analysis and visualization in practice, continue to work with Countrix trying to obtain the most possible information of the traces, from the two approaches mentioned above, and considering different stakeholders apart from the teacher: developer, manager, and maybe researcher.
- Talk to Dan and Antonio about how to integrate it with the architecture.

14. May 17th, 2016

Meeting: discussion of Ángel Serrano's theoretical model for tracking using xAPI format. Review of the model and discussion of suggestions made by ADL of changes about verbs, activity types and extensions.

Some issues that appeared were: having a more general set of verbs to cover more types of games versus having a more specific subset that covers less games and can then be extended; the use of the state API to store the state of the game and not only the progress of interactions as it has been done so far; activity types too general such as *completable* or *reachable* to better become extensions of activity, being able to have more information about the same statement. Also, it is mentioned that the OpenLRS has no longer support by Apereo so it should be changed later to a different LRS, for instance the one of ADL that fully supports xAPI.

15. May 17th, 2016

Meeting with Baltasar. While going through the new dashboards developed for different stakeholders, we discuss several things: consider different scenarios of use for dashboards (for instance, in a class, the teacher or the student); visualizations including time of game session should consider whether the game has a limited time to play or not (as it is the case of Countrix) and how this influences the sessions; the stakeholder of manager would be clearer if we consider several games, but in the case of a single one the manager could also be interested in visualize information such as pick times of plays i.e. when do most students play and also the mean number of users playing per minute for instance.

Next things to do: write down the details of the new dashboards developed, taking into account the level of knowledge of the game that they require; extend the explanation of Google Analytics as a clear example, maybe adding some graphics for it; general review of the report including the evolution of the project (problems found and suggested solutions).

16. May 24th, 2016

Meeting with Baltasar. We discuss the new visualizations developed (single user dashboard for student and peak times). Several things to work on:

- Update to last version of Kibana to change dashboards with customized labels and tags.
- Meet with Dan and Antonio to try to incorporate these visualizations to the project.
- Visualizations: student, add comparison with the mean of the class. Discussion of criteria of results, adding helps or explanations to dashboards. Peak times, try to make it interactive selecting both the time interval and the aggregations. Also, make more explicit the difference between the ones that require access to game model and the ones that do not.

- Add a section in the report with conclusions and future work with interesting things that could complete the project. Some of them may be feasible so they would later be moved to the main sections.

17. May 26th, 2016

Meeting with Dan and Antonio to see how we could integrate the visualizations developed with the current RAGE project. For that, it is preferred that I first change the visualizations developed for Countrix to the last version of Kibana. After that, send them their information in JSON format (obtained with a Postman query) including dashboards, visualizations and index patterns. Moreover, they ask me to extend the explanations about the xAPI fields and analysis required for each visualization/dashboard so they could reply it in their analysis. Focus first in the visualizations that do not require access to the game model as they could be used for other exampled distinct to Countrix.

18. June 2nd, 2016

Meeting with Baltasar and Ángel Serrano. We discuss several things to improve or add in the report:

- Extend the state of the art section.
- Extend the other types of visualization tools' descriptions.
- Incorporate the new vocabulary as described in the article developed together with ADL.
- Add a subsection about the final integration of the dashboards with RAGE.
- Add diagrams to describe the analysis developed.
- Add a first scheme that show the GLA project before the description: data extraction, exploit data, results.
- Incorporate the descriptions of the analysis required to develop the visualizations.
- Other improvements: titles, font, conclusions, contributions and future work as subsections.

After these improvements, send the report to Ángel for his revision. Also, comment the code developed and upload it to GitHub as part of the e-ucm group, and check its code volume with some profiler to give an idea of its measure. Finally, start thinking about the public presentation: in English, less than 15 slides.

Bibliography

ADL. 2016a. “ADL.” Accessed March 22. https://www.adlnet.gov/about-adl/.

ADL. 2016b. “Experience API.” Accessed March 20. https://www.adlnet.gov/adl-research/performance-tracking-analysis/experience-api/.

ADL. 2016c. “xAPI Lab.” Accessed June 3. http://adlnet.github.io/xapi-lab/.

ADL. 2016d. “Serious Games Vocabulary.” Accessed May 16. http://xapi.vocab.pub/datasets/seriousgames/.

ADL. 2015. “ADL LRS.” https://www.adlnet.gov/adl-lrs/.

Apache. 2016. “Storm Flux.” Accessed March 18. https://github.com/apache/storm/blob/v0.10.0-beta/external/flux/README.md.

Apereo. 2016a. “Apereo LAI.” Accessed April 1. https://www.apereo.org/communities/learning-analytics-initiative.

Apereo. 2016b. “OpenDashboard.” Accessed March 22. https://www.apereo.org/projects/opendashboard.

Apereo. 2015. “Learning Analytics Processor.” https://www.apereo.org/projects/learning-analytics-processor.

Apereo. 2016. “Apereo Open Analytics.” https://www.apereo.org/sites/default/files/projects/Brochures/Apereo Analytics Briefing 26Apr16.pdf.

Apereo, e-ucm. 2016c. “OpenLRS.” Accessed March 18. https://github.com/e-ucm/OpenLRS.

Arnold, Kimberly E., and Matthew D. Pistilli. 2012. “Course Signals at Purdue.” In *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge - LAK ’12*, 267. New York, New York, USA: ACM Press. doi:10.1145/2330601.2330666. http://dl.acm.org/citation.cfm?doid=2330601.2330666.

Bakharia, Aneesha. 2016. “Could GraphQL Bring Better Query Capability to xAPI and Learning Record Stores?” https://medium.com/@aneesha/could-graphql-bring-better-query-capability-to-xapi-and-learning-record-stores-3c27deb6bc7a#.nwvhyb4og.

Bienkowski, Marie, Mingyu Feng, and Barbara Means. 2012. “Enhancing Teaching and Learning through Educational Data Mining and Learning Analytics: An Issue Brief.” *Washington, DC: SRI International*: 1–57. http://www.ed.gov/edblogs/technology/files/2012/03/edm-la-brief.pdf.

Bitergia. 2016. “Bitergia.” Accessed May 19. https://bitergia.com/.

Bitergia. 2016a. “Cauldron.” https://cauldron.io/.

Bitergia. 2016b. “Bitergia Analytics Example.” https://t.co/7n7KfVbWiE.

Black Duck Software, Inc. 2016. “OpenHub.” https://www.openhub.net/.

Bostock, Mike. 2016. “D3js.” Accessed March 18. https://d3js.org/.

Bowe, Megan. 2016. “Data Strategy and xAPI.” https://vimeo.com/158669808.

Curatr. 2016. "Course: 'An Introduction to the xAPI.'" https://curatr3.com/free-courses/intro-to-xapi/.

Elastic. 2016a. "Elasticsearch." Accessed March 20. https://www.elastic.co/products/elasticsearch.

Elastic. 2016b. "Kibana." Accessed March 18. https://www.elastic.co/products/kibana.

Elastic. 2016c. "Logstash." Accessed April 2. https://www.elastic.co/products/logstash.

Elastic. 2016d. "Timelion Plugin." Accessed May 2. https://github.com/elastic/timelion.

Elastic. 2016e. "Shield." Accessed May 19. https://www.elastic.co/products/shield.

El-Nasr, Magy, Anders Drachen, and Alessandro Canossa. 2013. "Game Analytics Maximizing the Value of Player Data." London ;;New York : Springer,.

Entertainment Software Association (ESA). 2015. "2015 Essential Facts About the Computer and Video Game Industry." http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf.

e-ucm. 2016a. "GLEANER." Accessed March 20. http://e-ucm.github.io/gleaner/.

e-ucm. 2016b. "Rage-Analytics-Frontend." Accessed March 18. https://github.com/e-ucm/rage-analytics-frontend.

e-ucm. 2016c. "Rage-Analytics-Backend." Accessed March 18. https://github.com/e-ucm/rage-analytics-backend.

e-ucm. 2016d. "a2." Accessed March 18. https://github.com/e-ucm/a2.

e-ucm. 2016e. "Dockerized Storm." Accessed March 18. https://github.com/e-ucm/dockerized-storm.

e-ucm. 2016f. "Kzk." Accessed March 18. https://github.com/e-ucm/kzk.

e-ucm. 2016g. "QuizDemo." Accessed May 3. https://github.com/e-ucm/QuizDemo.

e-ucm. 2016h. "Countrix." Accessed May 3. https://github.com/e-ucm/countrix.

Ferguson, Rebecca. 2012. "Learning Analytics: Drivers, Developments and Challenges." *International Journal of Technology Enhanced Learning* 4 (5/6): 304. doi:10.1504/IJTEL.2012.051816. http://www.inderscience.com/link.php?id=51816.

Fernández-Manjón, Baltasar. 2012. "Learning Analytics in Serious Games." http://es.slideshare.net/BaltasarFernandezManjon/tracingalittlebit-balta.

Fernández-Manjón, Baltasar. 2013. "Learning Analytics Applied to Serious Games, Invited Talk at ECGBL 2013 Porto, Portugal." http://es.slideshare.net/BaltasarFernandezManjon/balta-learning-analytics-in-serious-games-invited-talk-at-ecgbl-2013-porto-portugal.

Fernández-Manjón, Baltasar. 2014a. "Applying Learning Analytics to Simplify Serious Games Deployment in the Classroom." http://es.slideshare.net/BaltasarFernandezManjon/applying-learning-analytics-to-simplify-serious-games-deployment-in-the-classroom.

Fernández-Manjón, Baltasar. 2014b. "Applying Learning Analytics in Serious Games." http://es.slideshare.net/BaltasarFernandezManjon/applying-learning-analytics-in-serious-games.

Fiellin, L. E., T. C. Kyriakides, K. D. Hieftje, T. M. Pendergrass, L. R. Duncan, J. D. Dziura, B. G. Sawyer, and D. A. Fiellin. 2016. "The Design and Implementation of a Randomized Controlled Trial of a Risk Reduction and Human Immunodeficiency Virus Prevention Videogame Intervention in Minority Adolescents: PlayForward: Elm City Stories." *Clinical Trials*: 1–9. doi:10.1177/1740774516637871. http://ctj.sagepub.com/cgi/doi/10.1177/1740774516637871.

Freire, Manuel, Ángel Serrano-Laguna, Borja Manero, Iván Martínez-Ortiz, Pablo Moreno-Ger, and Baltasar Fernández-Manjón. 2016. "Learning Analytics for Serious Games."

GameAnalytics. 2015. "GameAnalytics." http://www.gameanalytics.com/.

Gilbert, Gary. 2015a. "Getting Started with Open Learning Analytics - Data Collection." https://www.unicon.net/about/articles/getting-started-open-learning-analytics-data-collection.

Gilbert, Gary. 2015b. "Getting Started with Open Learning Analytics - Storage." https://www.unicon.net/about/articles/getting-started-open-learning-analytics-storage.

Gilbert, Gary. 2015c. "Getting Started with Open Learning Analytics - Analysis." https://www.unicon.net/sites/default/files/files/LA_Analysis_article.pdf.

Gonzalez-barahona, Jesus M. 2016a. "Open Development Analytics: A Step Towards More Project Transparency" (March). http://schd.ws/hosted_files/collabsummit2016/36/slides.pdf.

Gonzalez-barahona, Jesus M. 2016b. "GrimoireLab: A Toolset for Open Development Analytics" (March).

Google. 2016. "Google Analytics." Accessed April 15. http://www.google.com/analytics/.

Google. 2015. "Google Analytics Example." https://static.googleusercontent.com/media/www.google.com/en/us/analytics/customers/pdfs/americancancersociety.pdf.

Guillén-Nieto, Victoria, and Marian Aleson-Carbonell. 2012. "Serious Games and Learning Effectiveness: The Case of It's a Deal!" *Computers & Education* 58 (1) (January): 435–448. doi:10.1016/j.compedu.2011.07.015. http://linkinghub.elsevier.com/retrieve/pii/S0360131511001734.

Hmalphettes. 2016. "Authentication Plugin." Accessed May 2. https://github.com/hmalphettes/kibana-auth-plugin.

Koster, R. 2004. *Theory of Fun for Game Design*. Edited by Scottsdale Arizona: Paraglyph.

LACE. 2016. "Visions of the Future, Horizon Report." http://www.laceproject.eu/wp-content/uploads/2016/02/LACE_D3_2.pdf.

Loh, Christian Sebastian, Yanyan Sheng, and Dirk Ifenthaler. 2015. *Serious Games Analytics*. Edited by Christian Sebastian Loh, Yanyan Sheng, and Dirk Ifenthaler. Cham: Springer International Publishing. doi:10.1007/978-3-319-05834-4. http://link.springer.com/10.1007/978-3-319-05834-4.

Manero, Borja, Javier Torrente, Manuel Freire, and Baltasar Fernández-Manjón. 2016. "An Instrument to Build a Gamer Clustering Framework according to Gaming

Preferences and Habits." *Computers in Human Behavior* 62 (September): 353–363. doi:10.1016/j.chb.2016.03.085. http://linkinghub.elsevier.com/retrieve/pii/S0747563216302564.

Minewhat. 2016. "CSV Exporter Plugin." Accessed May 2. https://github.com/minewhat/es-csv-exporter.

MongoDB. 2016. "MongoDB." Accessed March 18. https://www.mongodb.org/.

Pardo, Abelardo. 2016. "Pushing Transparency in Student-Facing Learning Analytics." http://abelardopardo.blogspot.com.es/2016/05/pushing-transparency-in-student-facing.html.

Raystorm-place. 2016a. "Slider Plugin." Accessed May 2. https://github.com/raystorm-place/kibana-slider-plugin.

Raystorm-place. 2016b. "HTML Widget Plugin." Accessed May 2. https://github.com/raystorm-place/kibana-html-plugin.

Sánchez, Jaime, and Ruby Olivares. 2011. "Problem Solving and Collaboration Using Mobile Serious Games." *Computers & Education* 57 (3) (November): 1943–1952. doi:10.1016/j.compedu.2011.04.012. http://linkinghub.elsevier.com/retrieve/pii/S0360131511000935.

Serrano-Laguna, Ángel. 2016. "Tracking Model." Accessed March 18. https://github.com/e-ucm/tracking-model/blob/master/Tracking Model.md.

Serrano-Laguna, Ángel, and Baltasar Fernández-Manjón. 2014. "Applying Learning Analytics to Simplify Serious Games Deployment in the Classroom." In *2014 IEEE Global Engineering Education Conference (EDUCON)*, 872–877. IEEE. doi:10.1109/EDUCON.2014.6826199. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6826199.

Serrano-Laguna, Ángel, Iván Martínez-Ortiz, Jason Haag, Damon Regan, Andy Johnson, and Baltasar Fernández-Manjón. 2016. "Applying Standards to Systematize Learning Analytics in Serious Games:" 16.

Serrano-Laguna, Ángel, Javier Torrente, Pablo Moreno-Ger, and Baltasar Fernández-Manjón. 2012. "Tracing a Little for Big Improvements: Application of Learning Analytics and Videogames for Student Assessment." In *Procedia Computer Science*, 15:203–209. Elsevier.

Serrano-Laguna, Ángel, Javier Torrente, Pablo Moreno-Ger, and Baltasar Fernández-Manjón. 2014. "Application of Learning Analytics in Educational Videogames." *Entertainment Computing* 5 (4) (December): 313–322. doi:10.1016/j.entcom.2014.02.003. http://linkinghub.elsevier.com/retrieve/pii/S1875952114000111.

Sirensolutions. 2016. "Kibi Radar Plugin." Accessed May 2. https://github.com/sirensolutions/kibi_radar_vis.

Stormpython. 2016a. "Heatmap Plugin." Accessed May 2. https://github.com/stormpython/heatmap.

Stormpython. 2016b. "Tag Cloud Plugin." Accessed May 2. https://github.com/stormpython/tagcloud.

Stormpython. 2016c. "Vector Map Plugin." Accessed May 2.

https://github.com/stormpython/vectormap.

Takeuchi, Lori M., and Sarah Vaala. 2014. "Level up Learning: A National Survey on Teaching with Digital Games." http://www.joanganzcooneycenter.org/wp-content/uploads/2014/10/jgcc_leveluplearning_final.pdf.

The Open University of the Netherlands, University of Trento, Hull College of Further Education, Universidad Complutense de Madrid, The University of Bolton, INMARK, Instituto de Engenhariade Sistemas e Computadores, Investigacao e Desenvolvimiento im Lis, Randstad. 2015. "RAGE Project." http://rageproject.eu/.

Training Evidence Systems. 2016. "xAPI Apps." http://xapiapps.com/.

Van Eck, R. 2006. "Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless." *EDUCAUSE Review*. 41 (2): 16 – 31.