A framework to improve evaluation in educational games

Ángel Serrano, Eugenio J. Marchiori, Ángel del Blanco, Javier Torrente, Baltasar Fernández-Manjón
Department of Artificial Intelligence and Software Engineering
Complutense University
Madrid, Spain
aserrano@e-ucm.es, {emarchiori, angel.dba, jtorrente, balta}@fdi.ucm.es

Abstract—**The evaluation process is key for educator's acceptance of any educational action. The evaluation is challenging in most cases but especially when educational games are used. In educational games if in-game evaluation exist it is usually based on a series of simple goals and whether these goals are achieved (i.e. assessment). But we consider that evaluation can be improved by taking advantage of in-game interaction, such as the user behavior during the game and the type and number of interactions performed by the user while playing. In this paper, we propose an evaluation framework for educational games based on in-game interaction data. We discuss how user interaction data is collected in the most automatic and seamless way possible, how to analyze the data to extract relevant information, and how to present this information in a usable way to educators so they achieve the maximum benefit from the experience. The evaluation framework is implemented as part of the eAdventure educational platform, where it can be used both to improve upon traditional basic assessment methods (i.e. goals, scores & reports) and to provide information to help improve interaction with games (e.g. discovery strategies).**

Learning Analytics; Educational video games; framework proposal; case study;

I. INTRODUCTION

In traditional education, either in higher education or in other levels, the main evaluation method is based on written final exams [1]. This method, as some authors have pointed out [2], presents a series of problems. These problems are related not only to the student evaluation, but also to the evaluation of the educational action itself: the amount of data available is limited, and it is usually restricted to students and educators subjective perceptions (e.g. through polls about the past courses). Other metrics, mostly based on exam grades, might not give enough information about the educational action, or whether it was a success or a failure and why. Moreover, these data usually become available when the action is finished or when it is too late to make an intervention, improvement or correction in the ongoing action.

With the emergence of the Web, on-line educational resources have grown exponentially. Many institutions now use LMS (Learning Management System) to organize their courses, to allow students to communicate among themselves and with teachers, and to improve access to educational resources [3]. Still, despite all of these on-line resources, evaluation is still usually performed using traditional methods. Most of the content presented in LMS is finally evaluated through written exams in classrooms, or through online tests or exams.

However, there is a whole new body of data, derived from the student interaction with on-line educational resources. These data can be collected and analyzed not only to improve the evaluation methods, but also to obtain real-time feedback about the progress of any educational action, enabling educators to predict results and react to that progress.

The field studying the use and analysis of this kind of data is known as Learning Analytics [4]. This new field advocates of capturing all the data derived from interaction with on-line educational resources, and analyzing it to assess students, predict future events and act consequently to refine educational actions. These ideas have been successfully applied in other disciplines, like Business Intelligence, a well-extended set of techniques for analyzing business data to support better business decision-making [5], or Web Analytics, where internet data are collected in order to understand and optimize web usage [6].

Currently, LMS are the main target for Learning Analytics systems. Projects like SNAPP [7] or LOCO-Analyst [8] offer statistics about the interactions made by students inside an LMS. SNAPP is focused on analyzing forum activity and creating network diagrams of all interactions among students. From this, it infers which are the most active students in a class, and those students who are "disconnected" or "at risk", among other features. LOCO-Analyst is based on student interaction with learning content (e.g. number of views, time spent with every resource) that is used to infer conclusions about learning content characteristics (e.g. difficulty or importance).

Though LMS activity reports can contribute to better understand students' interactions with content and resources, educational games represent an ideal environment to capture more detailed and diversified student interaction. In the last few years, Game Analytics are being used to let developers know about how players interact with their games. One of their main purposes is identifying where and why a player got stuck during the game[1], so game developers can try to smooth this hardness, to avoid player frustration and thus keep him

[1] http://www.gamasutra.com/view/feature/6155/hot_failure_tuning_gameplay_with_.php

engaged and playing [9]. All these ideas applied to educational games, combined with other Business Intelligence and Web Analytics techniques and guided by the Learning Analytics process, are addressed in this paper.

First, we board the main steps in the Learning Analytics process, and how these steps can be particularized in educational games, proposing a theoretical Learning Analytic Model and a Learning Analytic System. Then, we propose an implementation upon the educational game platform eAdventure and a use case to deploy the system, and finally some ideas and thoughts about the whole process.

II. Learning Analytics steps in educational games

Authors agree that Learning Analytics process [2] begins with selecting the most relevant student data to be captured. Once it is captured, data must be aggregated and transformed into reportable information (for example, using charts or other visual representations). With this information, the educator should be able to judge how the student used the educational resource. Some authors call this step *predict*, since information is converted into knowledge, and knowledge enables predictions. However, in our approach we will use this step to assess the student, and for now on, we will name this step as *assess*. Assessment information can be used, under certain conditions, to dynamically assist the student, and to refine the educational resource, based on the students results. Finally, all the knowledge acquired can be shared with others whom could benefit from it. Table 1 represents a scheme with all the steps and their descriptions.

TABLE I. Learning Analytics Steps

Step	Unit produced	Description
Select	Data	Choose the basis data to be captured
Capture	Data	Collect the selected data
Aggregate & Report	Information	Sort out captured data and convert it in information
Assess	Knowledge	Understand reported information and convert it into knowledge, assess students
Use	Knowledge	Adapt the system based on assessment
Use & refine	Knowledge	improve educational action
Share	Knowledge	Share knowledge for the benefit of others

To support all these steps, we propose a system based on a Learning Analytics Model (LAM) holding all the information required for every step, and a Learning Analytics System (LAS) endued with all the processing power required by the model.

In this approach, focused on educational games, we consider the LAS as a separate system from the game engine, but both are communicated. The LAS also has access to the game model and the LAM (Fig. 1). The LAM is constructed by a set of models, which are directly related to the different modules contained by the LAS (Fig. 2).

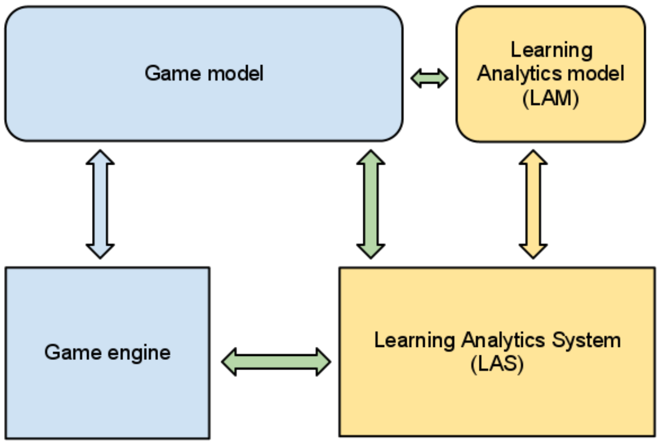
Game model
Learning Analytics model (LAM)
Game engine
Learning Analytics System (LAS)

Figure 1. Relation between the different components involved in the Learning Analytics process. The LAM is dependent on the game model and the LAS. LAS is aware of the LAM and the game model, and can communicate with the game engine.

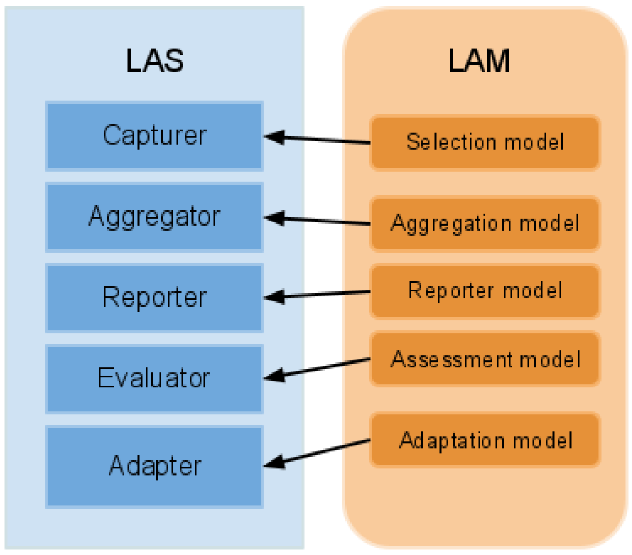

Figure 2. The LAS consists of a series modules that take part in the different steps of the Learning Analytics process. The LAM holds models for those steps which requires defining a model to work. The LAS uses the LAM to process all the data captured and generated.

In this section the learning analytics steps are described in general but also for educational games in particular, building the LAM and the LAS upon them. As starting point for this definition we consider the available data in games and how to adapt these data to the theoretical model in order to extract the maximum information possible from this media.

A. Select and capture

First, the data to be captured by the LAS are selected. These data will be the raw material that will feed the steps that follow. The data selection criteria are lead by the educational resource objectives and some constrictions such as technical limitations and privacy policies must be taken into account. In educational resources, meaningful data can be selected from personal information about the student (e.g. age, gender, etc.), academic information and any other data provided by the resource context.

While in static resources (e.g. PDF files), the only extractable data are the number of views and the time spent with them, the interactive nature of educational games provides a whole new type of data that can be selected:

- *GUI events performed by the student during the game*: mouse clicks, keys pressed, and other events (joystick movements, buttons interactions), depending on the input method. Not only the event itself can be recorded, but also the time when it occurred and whether it was performed over a target (e.g. some click over a game object). These events can provide clues about the student behavior during the game (e.g. if all GUI events were captured the LAS would be able to recreate the complete game play).

- *Game state evolution*: the game state is a set of variables and their values that specify a concrete status in the game instance. The evolution of variables through time describes the development of the different goals of the game. Depending on the case, the whole game state evolution could be recorded, or it could be recorded only in some points (e.g. when a phase ends, or a goal is achieved).

- *Logic events*: a logic event is anything that moves the game-flow forward. Changing the value of a variable, finishing a phase, launching a cut-scene (i.e. a slide-show or video), losing a life, achieving a defined goal, etc. Some logic events, and their timestamps, can be directly related to the student progress in the game, and thus be relevant for the assessment.

Selectable data are limited by the technologies used to deploy the games: Not every piece of data here proposed will be available in every game platform. These selectable data, then, are platform-dependent and must be defined in the LAM's *selection model*. To avoid unnecessary data capture, every game model should define, among all selectable data, the final data to be captured according to their own purposes.

Once the data are selected, the framework requires a way to capture it. The technology involved will be very important to establish how the data are collected. Access to different internal parts of the game engine is required to capture some of the information. This implies, for instance, that such model cannot be generally applied to commercial games provided as black boxes.

Another issue is the moment when the captured data are passed to the LAS to begin processing. The simplest way is to store all the data locally and send it back to the LAS when the game is finished. Data could be sent in certain significant moments, like when the student ends a phase or achieves a goal. Moreover, all the data could be sent to the LAS as they are being captured. Last two approaches enable real-time assessment that can be used to assist the student during the game. Depending on the needs, all these data might go through a filter in order to make it anonymous.

B. Aggregate and Report

The captured data must be organized in such a manner that it can be shown in human readable formats, like tables or graphics. A more meaningful report can be done if the LAM contains, in the *aggregation model*, semantic rules to interpret all the received data. For example, the system could relate a raw event (e.g. a variable taking a particular value) with a meaningful feat (e.g. the player completed a goal). These semantic rules can be based on the game engine, where some events can have an implicit meaning (e.g. an engine where pressing escape key always brings up the menu) or on the game itself (e.g. if the game variable "hits" is "8", the phase is completed).

Semantic rules can be expressed like conditions producing new data to be reported: when a condition (based on GUI events, a logic event or a concrete game state) is met a new unit of data, defined in the *aggregation model*, is generated. E.g. when in the game state, the variable *score* equals to 10, and the variable *gold* equals to *15*, the LAS *aggregator* produces a logic event "Goal 1 completed". The *reporter* could then treat this event as it would with any other logic event.

The LAS' *aggregator* needs to be endued with mechanisms capable of understating and processing these kinds of rules (Fig. 3).

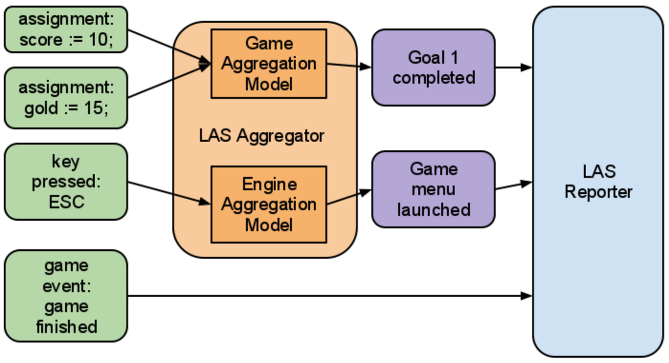

Figure 3. Raw events are passed through the semantic rules contained by the LAM, and converted to more meaningful data. Events can be grouped and simplified through semantic rules. Some events, such as the game finishing, have enough meaning and do not need to be interpreted by any rule.

After aggregation, information can be reported in common ways, such as tables or charts, but also, we can take advantage from the inherent characteristics of games to report information with new representations. For example, "heat maps" could be created for every phase, in which the heat can measure the amount of times the player clicked in every point of the phase, or the places where the player was defeated. If there is enough information of user interaction, an animation recreating how the student played a game phase could be shown.

The *reporter model* contains which information must be reported and which representations must be used. Common reports can be defined at engine level (e.g. heat maps for every phase can be common for all games), as well as reports at game level, holding important information in that particular game.

These reports can be even richer if data from different students are aggregated. Average results can spot which goals took more time or the places where most of the students failed.

C. *Assess and Use*

The information and the reports generated until now can give an overview of how the students are using an educational resource. However, this information should have some practical consequences to be really useful. It is the moment to transform the information received into knowledge. In the educational game context, all the information reported is processed to assess the student in this step.

Games are organized around goals. In educational games, these goals should be based on the success in some educational aspects. In our context, and based on the concepts of the selection and aggregation process, we could have several types of goals, represented by:

- A GUI event or a series of GUI events performed by the student, over a game object or in total.
- A concrete game state, fulfilled fully or partially.
- A variable taking a defined value.
- The launch of a particular logic event.

These classes of goals are platform-dependant, and should be defined by the LAM's *assessment model.*

Compound goals can be defined based on these simple goals. An educational game can define all the necessary goals to cover all the educational aspects that are to be learned by players of the game. Based on these goals and with the reported information the game can be used to assess the student.

This assessment can have two applications: one, just to measure the success of the student in the game, and act accordingly (e.g. enabling the student to access to new educational resources), or, if the captured data are being passed to the LAS during game time, dynamic adaptation through real-time assessment (e.g. if the student got stuck in some point, the system will offer him help). Rules for this assistance are contained by the *adaptation model*, and are processed by the adapter, which is able to communicate with the game engine to perform the adaptation.

Assessment and dynamic adaptation could be more sophisticated. As some authors pointed out [10], propagating information through complex structures, like Bayesian networks, can help to determine what is going on in virtual simulations, and better decide what adaptation profile to choose.

However, our approach pretends to be based on easy principles and stay accessible for as many educators as possible. Complex structures, like Bayesian networks, are normally out of reach for most educators.

D. *Refine and share*

With all the accumulated knowledge from previous steps, an educator can know about the global results (assessed in the previous step) of the educational action and can identify which educational goals were not achieved as expected. Thus, educators can refine the educational resources to improve the results or readjust their expectations.

In educational games, those game goals that were not solved as expected can be detected. Aggregated data from several students can ease this job, pointing out, in average, which goals made more trouble to students. From here, the game could be modified or even redesigned to facilitate its accomplishment. This does not directly imply making the game simpler or shorting the educational goals, it could be enough to smooth the learning curve in the game, or adding some extra help in game points especially hard. Maybe, learning analytics conclusions showed that the student did not get stuck, but stop playing the game after a while, indicating that it was not engaging enough.

Finally, the LAS can share all the knowledge obtained with other systems. These systems cover from LMS to institution administrative systems. Even making the data public can be an option in some cases. In order to be able to share data, some considerations such as privacy policies, what knowledge is shared or which standards are used in the communication, need to be taken into account.

III. Implementation Proposal: eAdventure

eAdventure is an educational game platform developed and maintained by the <e-UCM> research group at the Complutense University of Madrid for the last 5 years. This platform includes a game engine and an easy-to-use editor, targeted at educators. eAdventure is currently undergoing the development of the 2.0 version, where new features are being added. Some of these include support for multiple platforms [11] and an easy to use narrative representation of games [12]. Moreover, we propose to implement the framework presented in this paper in this new version of the system.

eAdventure games are composed of scenes, which can represent from a simple scenario in an adventure game where the player's avatar moves to a more complex slide-show, going through an array of mini-games and other content. These scenes are always composed of simpler parts referred to as scene elements, each of which will usually have a graphical representation, a position in the screen, behaviors, etc. The current scene and the status of elements in the screen are defined as the game state. It is the flow from one scene to another, behaviors of the scene elements and effects (changing current scene, showing text, launching videos, assigning values to variables) that make up a game, by continually changing the game state until a final state is reached.

The LAS is implemented on a server. It is initialized with the Game Model, (containing the Adventure LAM) and the Engine LAM. The LAS has several modules to satisfy the requirements for every step of the Learning Analytics process (Fig. 4). The relation between the modules and the steps is detailed below.

A. *Select*

Given that eAdventure is intended to be a general game engine, including its own editor, our proposal tries to make selectable the biggest amount of data, letting to the game designer choose between all the available options. The eAdventure Learning Analytics Model define three units of selectable data:

- *LAGUIEvent*: represents a detailed GUI interaction. It holds the GUI event (mouse action, drag & drop, keyboard action) with its properties (mouse button, key pressed) and the target scene element, if exists.
- *LALogicEvent*: represents the launching of a game effect. It holds the generic effect data and additional information about the concrete instantiation (e.g., in the changing scene effect, the initial scene and the final scene).
- *LAGameState*: represents a game state in a certain moment. It contains a map holding all the game variables associated with their current value.

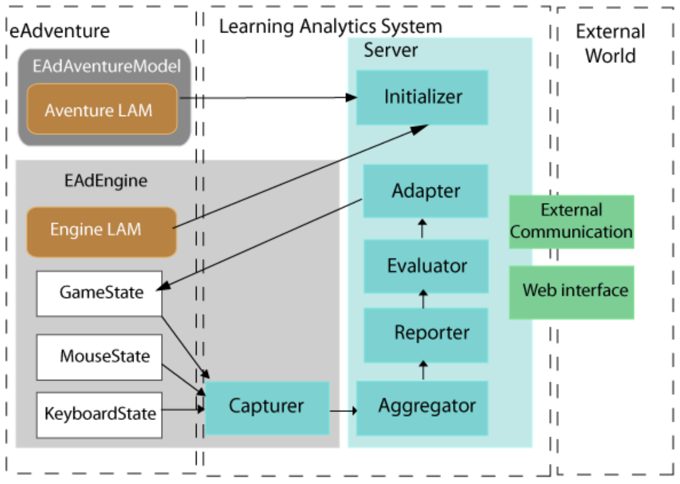

Figure 4. General organization for the LAS integrated with eAdventure. The LAS is mostly deployed in a server, an can communicate with the game engine. A web interface and external communication with other systems is offered as well.

Every of these units has associated a timestamp, representing the moment the event occurred since the game was started.

In the editor, we select where and when these data must be captured. For example, we can mark which type of GUI events (mouse, keyboard) we want to capture for every scene element (*LAGUIEvent*), and there is a special option to record all the GUI interactions performed in the game, allowing the recreation of the whole game play. It is possible also to capture those game effects that have relevance in the game flow *(LALogicEvent)* and, if desired, produce a *LAGameState* with the current game state. *LAGameStates* can be configured to be automatically generated periodically, when a game condition is met or when the game ends.

All these options are added to the *selection model* as part of the game's LAM, which will be used by the LAS' *data capturer*.

B. *Capture*

To capture all these data it is required a *data capturer* with access to all the relevant parts of the eAdventure game engine. The game engine has three main elements that are involved in this process: the *input listener*, which processes all GUI input from the user; the *game state*, which stores all the values for all the variables in the game at any given point in time; and the *effect handler*, which processes all in-game events (such as scene changes). All these elements communicate any relevant change to the *data capturer*, which then captures this information according to the current game *selection model*. The captured data are instances of the selectable data units presented before.

All these collected data are sent to the LAS *aggregator* (Fig. 5). Due to the multi-platform nature of the eAdventure game engine, different implementations take care of the communication with the *aggregator*. The *data capturer* can be configured to send out the captured data when the game is finished, a scene change happens, a defined condition is met or in real time.

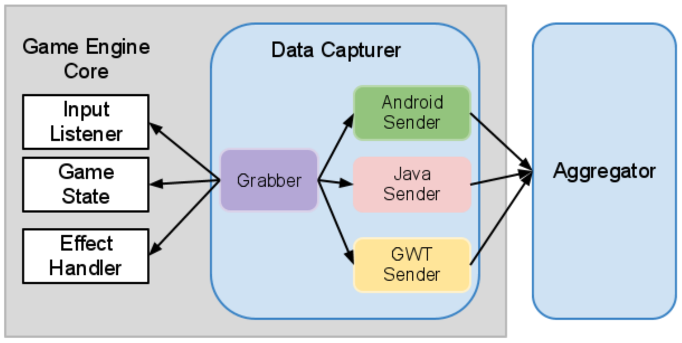

Figure 5. The *data capturer* is compound by two elements: a grabber connected with all the elements producing significant events in the game engine, and a sender managing communication with the aggregator.

C. *Aggregate*

The data sent are received by the *aggregator*, who makes a first data processing based on the semantic rules defined by the *aggregation model*, contained by the Adventure LAM, converting the basic units into more semantic pieces of data. This new units can be defined through the editor, as well as the rules of conversion from basic units (presented in the II.c section).

Aggregator also groups all the GUI events by type and scene element, filters redundancies and stores all the data in the LAS database.

Previous versions of eAdventure provided a basic mechanism for data aggregation and report generation. This basic mechanism allowed for information to be written in a textual report based on the values of variables in the game. This way, the game author could define a set of rules that would, for instance, write in the report that the player failed to complete a goal if a variable to indicate this was given a certain value. [13]

The same sort of data aggregation can be performed with the new LAS system. The rules in this case use a syntax that establishes the meaning of the data that were captured during the game to generate a report. The most basic reports will only include basic textual information, such as "Goal X was completed at time Y". However, more complex information can be aggregated to generate detailed information about the goal, such as "Goal X was completed by time Y, after Z attempts where the player failed to solve problems A, B, C, etc."

D. *Report*

The reporter represents in a web format all the information stored in the database. Among many others, the reports can be:

- A table relating scenes with the total time spent in any of them.
- Heat-maps showing where the player is hovering with the mouse most (Fig. 6).
- Screen capture recreated from game states.
- Game animations built from captured GUI events.
- Graphics showing the evolution of chosen variables in time.
- Tables with direct queries to the database.

All these data can be shown for every student or for groups of students. The system can be extended to add new reports from the stored data.

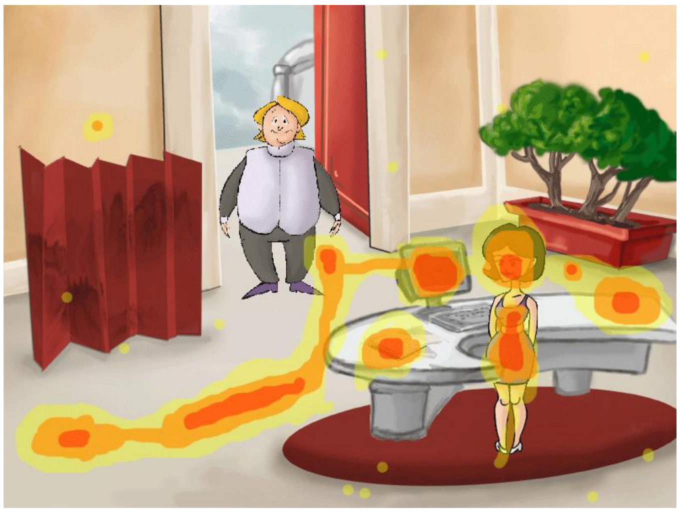

Figure 6. Heat map showing the concentration of left mouse clicks in a scene. Main heat zones are situated in interactive elements of the scene.

E. *Assess*

As established by the theoretical approach, this step is when the student is assessed. The assessment model contains all the goals established for the game. Goals can be defined in the editor as variables taking certain values at given times. The *evaluator* takes these goals and checks them against the stored information.

The accomplishment of these goals can be viewed through the reporter, and can be sent to the adapter to enable dynamic adaptation or to be shared with external systems.

F. *Use*

When the data are being captured in real time, dynamic adaptation can be used in the game. Adaptation rules are defined in the adaptation model. These rules can be defined in the editor and contain:

- *An effect*, which is considered the adaptation event and could be any eAdventure game effect (showing a text, changing a variable's value, launching a video...)
- *A condition*, establishing when the effect should be launched. Conditions can be the general conditions offered to create game logic in eAdventure games, or conditions based on goal accomplishment (e.g. a goal is not completed when the time for doing it expires).

The adapter takes the current game state and the goals information offered by the evaluator to check adaptation model conditions. When a condition is met, it communicates to the game engine the effect to be launched.

G. *Refine*

To support the refine task the LAS offers, through the web LAS *reporter*, information about the individual goals. This allows, for instance, the goals that lead to the worst performance to be identified. How the performance of students can be improved based on this is up to the game designer.

However, to ease this task, results obtained for the games' goals can be compared through time (checking if results improved after student played several times the game) and between different versions of the game.

H. *Share*

Nowadays, the selection and adhesion to standards for the content interoperability is an essential matter in the development of e-Learning contents. Current e-learning standards, like SCORM [14], are not prepared to communicate all the information collected by our LAS with other systems. For this reason, the best way of taking advantage of the full potential of our approach is to develop specific ad-hoc communication solutions for the systems that take into account all these data (e.g. a Moodle plugin). This idea can also be carried out in the eAdventure activity in LAMS [13], where all the information can be gathered and shown to educators, and use them to modify the lesson flow in an automatic or monitored way.

In the near future, it will be feasible to implement our ideas in compliance with next generation standards. For example, one the last initiatives leaded by the IMS Global Consortium, the IMS Learning Tool for Interoperability (IMS LTI), goes in that direction. This specification allows for the execution of learning tools hosted in external servers. Until other promising standards mature [15], our LAS is able to export all the information contained in the database along with the LAM required to interpret it into a exchangeable XML-based format.

IV. USE CASE: BASIC MATH GAME

We propose using the framework described in this paper in a basic math game targeted at school children. This game covers basic addition, subtraction and multiplication concepts, challenging the students to solve different problems within a given time-frame. This game is intended to provide increasing difficulty in the challenges presented to the students (e.g. the number of digits in the numbers involved is increased gradually).

The main goal of the game is sub-divided into simple goals: learn to operate with numbers of 1, 2 and 3 digits. Each time one of the goals is met, the game is adapted to provide the next level of challenge. Moreover, to provide help to the students and limit the chances that students could get stuck in any particular level, a help button is always accessible. The use of the help button is part of the information selected to be collected by the system.

In the *selection model* we marked the help button to capture all left-clicks performed over it. We also added to the model all the keyboard interactions, since these will be the input method used by the students to give their answers.

In the *aggregation model* a rule is added to convert every left-click over the help button into an “ask for help” event. Another rule is added to transform a sequence of numbers typed in the keyboard, followed by an “enter” into an “answer given” event, which its value is the introduced number and if the answer was correct.

The *reporter* shows the number of times the “ask for help” event occurred and the number of right answers. For the wrong answers, the invalid value introduced by the student is also shown. Average time for every operation is also displayed.

In the *assessment model* three goals are added: a percentage of all the given answers, after a minimum number of operations, must be correct (without using the help button) for operations with 1, 2 and 3 digits.

In the *adaptation model* an effect that changes the difficulty level (operations with 1, 2 or 3 digits) is added when the goal for the current event is achieved.

V. FINAL REMARKS

Learning Analytics, unlike Business Intelligence or Web Analytics, is still an emerging field that has a great potential. In this paper we tried to focus on a single target (i.e. educational games) in order to develop concrete methodologies trying to clarify some of the steps involved and better define the whole process. But, we think most of the ideas proposed for educational games can be extended to analyze data from other types of interactive educational resources as well, if more information about how they are being used becomes available.

The processing and logic involved in Learning Analytics can be used for other purposes different than education. The proposed LAS can help games in tasks like debugging the game design (statistics could show game points with no return), and testing (the LAS could spot if the user is playing the game as it was specified in the design).

Finally, it is important to note that the ultimate goal of Learning Analytics is to improve educational actions. We believe that learning analytics can help in establishing the educational value of games that use it. It is also important to take into account that monitoring students presents several ethics problems and privacy issues. Therefore transparency must guide all the design decisions.

REFERENCES

[1] L. Communiqué, Towards the European higher education area: Responding to Challenges in a Globalised World. Techreport. 2007.

[2] T. Elias, “Learning Analytics : definitions , processes and potential.” Learning, 23. vol: 6, issue: 4, pp. 134-148. 2003

[3] M. F. Paulsen, “Experiences with learning management systems in 113 European institutions.” Educational Technology & Society. 2003.

[4] M. Brown, “Learning analytics”. Learning Circuits Retrieved March, vol. 2, issue 1, pp. 1-4. 2010.

[5] S. Williams and N. Williams, “The business value of business intelligence”. Intelligence, vol. 8 issue 301, pp. 30-39. The Data Warehouse Institute. 2003.

[6] M. Hassler, “Web analytics.” Redline Heidelberg, vol. 3, issue 1, pp. 1-14. 2010 doi:10.1080/19322900802660292

[7] S. Dawson, A. Bakharia, E. Heathcote, “SNAPP : Realising the affordances of real-time SNA within networked learning environments.” Learning. pp 125-133. 2010.

[8] J. Jovanovic, D. Gasevic, C. Brooks, V. Devedzic, M. Hatala, "LOCO-Analyst: a Tool for raising teachers' awareness in online learning environments”. The 2nd European Conference on Technology Enhanced Learning, Crete, Greece, 2007, pp. 112-126.

[9] K. Gilleade, A. Dix. “Using frustration in the design of adaptive Videogames”. Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology ACE 04 vol: 74, pp: 228-232. 2004

[10] V. Shute and J. M. Spector, “SCORM 2.0 white paper: stealth assessment in virtual worlds”. Learning. pp. 1-10. 2008

[11] E. J. Marchiori, Á. Serrano, J. Torrente, I. Martínez-Ortiz, B. Fernández-Manjón, “Extensible multi-platform educational game framework.” In press, The 10th International Conference on Web-based Learning. 2011

[12] E. J. Marchiori, J. Torrente, Á. del Blanco, P. Moreno-Ger, P. Sancho, B. Fernández-Manjón, “A narrative metaphor to facilitate educational game authoring”. Computers & Education vol 58 pp. 590–599. 2012. (doi: 10.1016/j.compedu.2011.09.017). 2011

[13] Á. del Blanco, J. Torrente, E. J. Marchiori, I. Martínez-Ortiz, P. Moreno-Ger, B. Fernández-Manjón, “Easing assessment of game-based learning with <e-Adventure> and LAMS”. In proceedings of the ACM International Workshop on Multimedia Technologies for Distance Learning (MTDL 2010), pp 25-30. In Conjunction with the ACM International Conference on Multimedia, October 2010, Firenze, Italy. . 2010

[14] Á. del Blanco, J. Torrente, I. Martínez-Ortiz, B. Fernández-Manjón, “Análisis del Uso del Estándar SCORM para la Integración de Juegos Educativos”. IEEE-RITA Vol. 6, Núm. 3, pp. 118-127, Ago. 2011.

[15] D. Dagger, A. O’Connor, S. Lawless, E. Walsh and V. P. Wade. “Service-oriented e-learning platforms - From monolithic systems to flexible services”. IEEE Internet Computing, vol. 11, issue 3, pp. 28-35. IEEE Computer Society. 2007 doi:10.1109/MIC.2007.70