Journal of Universal Computer Science, vol. 21, no. 7 (2015), 976-996
submitted: 10/6/14, accepted: 14/5/15, appeared: 1/7/15 © J.UCS

Tracking Data in Open Learning Environments

Jose Luis Santos, Katrien Verbert, Joris Klerkx, Sven Charleer
Erik Duval
Dept. of Computer Science, KU Leuven
Leuven, Belgium
{joseluis.santos, katrien.verbert, joris.klerkx, sven.charleer
erik.duval}@cs.kuleuven.be

Stefaan Ternier
Welten Institute, Open University of the Netherlands
Heerlen, The Netherlands
Stefaan.Ternier@ou.nl

Abstract: The collection and management of learning traces, metadata about actions that students perform while they learn, is a core topic in the domain of Learning Analytics. In this paper, we present a simple architecture for collecting and managing learning traces. We describe requirements, different components of the architecture, and our experiences with the successful deployment of the architecture in two different case studies: a blended learning university course and an enquiry based learning secondary school course. The architecture relies on trackers, collecting agents that fetch data from external services, for flexibility and configurability. In addition, we discuss how our architecture meets the requirements of different learning environments, critical reflections and remarks on future work.

Key Words: learning analytics, learning record store, personal learning environments
Category: H.3.5, L.3.0, L.3.6

1 Introduction

This paper is situated in the domain of Learning Analytics and presents a simple architecture that tackles the challenge of collecting and managing data from a variety of services and feeds for applications such as Learning dashboards and Educational Data Mining algorithms [Verbert et al., 2013].

We focus specifically on Open Learning Environments where the teacher defines the learning goals and activities and the student can determine the activities he wants to focus on, the tools or both [Nistor et al., 2014].

Personalised Learning Environments (PLEs) [Govaerts et al., 2011] and Social Media Supported Learning Environments [Zhang et al., 2015; Behringer and Sassenberg, 2015] are typical examples that enable this educational approach.

Tracking data from such environments is not easy. We quote one of the characteristics of the PLEs:

From the perspective of the PLE, connection is far more critical than compliance, and it is far better to offer a wide range of services, requiring support for a range of standardisation from formal standards through to fully proprietary (yet publicly available) APIs, than to restrict the connections possible to users [Wilson et al., 2007].

The first challenge of this wide range of services, standards, specifications and APIs is that connecting these components is often complex. In addition, tracking and following learner actions across the boundaries of these components is complicated.

This paper presents our work on the development of open learning architectures with simplicity and flexibility as main requirements. These characteristics enable the architecture to be adaptable to new requirements in dynamic environments.

In order to illustrate how the architecture meets such requirements, this paper describes the deployment of the architecture in two different case studies: a blended learning university course and an enquiry based secondary school course.

This paper presents our work on the development of open learning architectures with simplicity and flexibility as main requirements. Section 2 presents relevant work in the learning analytics field on collecting and modelling learning traces, with special attention to open learning architectures. We present the main requirements and our architecture in Section 3. Sections 4 and 5 present two case studies where our architecture is deployed to collect learning traces, and to visualise these traces in dashboards. The paper ends with a discussion on how our architecture meets the requirements of our case studies and the limitations of this proposal.

2 Background and Related Work

When tracking learning activities, the granularity of the events to capture must be considered. Different approaches have been proposed to model this behavioural information [Wolpers et al., 2007]: a first approach focuses on low-level events such as keystrokes, mouse gestures, clicks, etc. A second approach focuses on higher-level events such as learning activities of the student. An example is reading a resource or answering a question. Current standardisation initiatives like IMS Caliper [Sakurai, 2014] and xAPI [del Blanco et al., 2013] focus on such high level activities.

This section compares the following relevant specifications in the field : a) xAPI [del Blanco et al., 2013], b) CAM [Niemann et al., 2013], c) ActivityStrea.ms [Vozniuk et al., 2013], d) IMS Caliper [Sakurai, 2014], f) NSDL paradata [Niemann et al., 2013] and g) Organic.Edunet format [Niemann et al., 2013].

Table 1: Comparison between seven relevant specifications in the field of tracking learning activity

	Core of the specification	**Educational information**	**Service definition**
NSDL Paradata	Object	No	No
Organic.Edunet format	Object	No	No
xAPI	Event	Yes	Yes
IMS Caliper	Event	Yes	Yes
CAM	Event	No	Yes
Activity Stea.ms	Event	No	Yes

- **Core of the specification:** NSDL Paradata and Organic.Edunet focus on the object capturing aggregated usage data about a resource (e.g. downloads, favourites, ratings [Niemann et al., 2013]). On the other hand, other specifications are event-centred. These specifications share a group of common fields [Niemann et al., 2013; del Blanco et al., 2013; Sakurai, 2014]: a) User, b) Verb, c) Object, d) Timestamp and e) Context. User represents who performs the action. Verb is the kind of action that the user performs, such as commenting on a blog post. Object is the artefact that receives the action. Timestamp is the exact moment that the action is performed. Finally, Context represents additional information that can be captured, such as the course, the kind of device and browser.

- **Educational information:** Specifications such as xAPI and IMS Caliper consider specific educational information such as the score and the result of the course [del Blanco et al., 2013; Sakurai, 2014]. Other specifications such as NSDL Paradata, Oganic.Edunet, Activity Strea.ms and CAM do not explicitly consider this specific information [Niemann et al., 2013; Vozniuk et al., 2013].

- **Service definition:** Specifications such as xAPI, IMS Caliper and CAM do not only define the data schema, the specifications also describe which services apply to the Learning Record Store (LRS) [del Blanco et al., 2013; Sakurai, 2014; Niemann et al., 2013]. The LRS is responsible of storing, managing and exposing the learning activity. The description of such services helps to standardise the exchange of information. Other specifications such as Activity Strea.ms, NSDL Paradata and Oganic.Edunet focus only on the data schema definition.

xAPI is our preferred specification because of the following reasons:

- xAPI is event-centred. Event-centred specifications cover a larger range of actions, as the field verb can define arbitrary actions [del Blanco et al., 2013], while the object-centred specifications focus on social actions such as rating and downloads.
- xAPI provides a service definition in addition to a data model. Such a definition facilitates extensibility and interoperability of learning analytics architectures.
- xAPI has many adopters [Corbi and Burgos, 2014] and is also supported by commercial Learning Record Store services such as Cloud Scorm and Learning Locker [Megliola et al., 2014]. This is clearly an advantage of xAPI compared with other specifications, as we want our architecture to be compatible with existing and successful architectures.

3 The Architecture

This section describes the different components of our architecture. Section 3.1 starts with the definition of main requirements that were considered when designing the architecture. Section 3.2 describes in detail services and components.

3.1 Design Requirements

The architecture meets two basic requirements: 1) simplicity and 2) flexibility. Fowler [2001] defines both concepts as:

- Simplicity is about keeping things as simple as possible: *"you shouldn't add any code today which will only be used by feature that is needed tomorrow"*. This means that integrating features that will make sense in the future makes no sense adding an unnecessary complexity in the code. Keeping the amount of features an application offers to strict minimum is beneficial to its simplicity.
- Flexibility in the design enables software to be easily adapted as requirements change.

The architecture has been deployed in different open environments that integrate a variety of services. Therefore, the architecture must be easily adaptable to new requirements from data providers and consumers.

Other requirements were also taken into account when designing the architecture:

- Extensibility is defined as the ability to add functionality to a system and to support easy integration of new software components, diverse software packages, etc [Bass et al., 2006].

- Performance relates to timing. Events like interrupts, messages, requests from users, or the passage of time occur and the system should react to these [Bass et al., 2006].
- Scalability refers to the ability of the architecture to grow proportionally to the number of users interacting with the system and the hardware required to support such interactions [Fielding, 2000].
- Functionality is defined as the ability of the system to do the work for which it was intended [Bass et al., 2006].
- Configurability refers to post-deployment modification of components, or configuration of components, so that they are capable of using a new service or data element type [Fielding, 2000].
- Maintainability is about frequency of new releases and how expensive the process of adapting these new releases for the end user becomes [Offutt, 2002].
- Privacy is about a system regulating the process of how personal digital information is being observed by the self or distributed to other observers [Pardo and Siemens, 2014].

Section 6 and 7 discuss the suitability and limitations of the architecture to meet our design requirements.

3.2 Architecture components

This section describes the communication and components of the architecture presented in Figure 1. The first section describes the different components and the specific services that the LRS exposes. Section 3.2.2 describes how the architecture uses xAPI and simplifies the data schema and section 3.2.3 describes the logic of the trackers.

This section focuses on how to collect and push the data in the LRS. Throughout the presentation of the scenarios in Section 4 and 5 is described which components consume the data from the LRS.

3.2.1 Architecture Components

The central component of our architecture is the Learning Record Store (LRS) (label 1 in Figure 1). The LRS is the glue between activity producers and consumers.

Activity producers and consumers push data in and pull data out of the LRS by using two simple REST services, described in Table 2. The push service (label

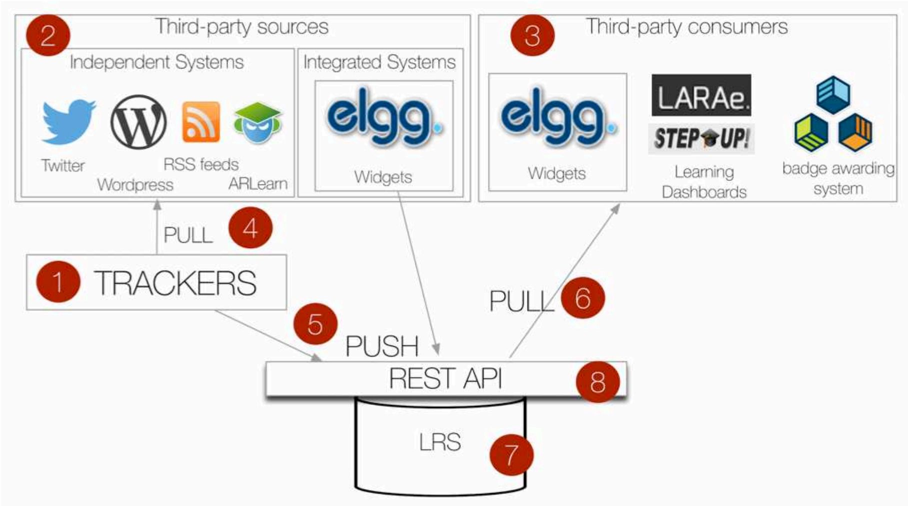

Figure 1: Different components of the architecture

Table 2: REST methods

N	M	Path	Query Params	Produces
1	POST	/event	Parameters described in Table 3 encoded in JSON	LRS generated identifier of the event or error if the message was not properly formatted
2	GET	/events	All parameters specified in Table 3 included in the url	Events containing the fields described in Table 3 codified in a JSON Array

2 in Figure 1 and first row Table 2) requires an event encoded in JSON format following the simplified xAPI specification. The pull service (label 3 in Figure 1 and second row Table 2) makes it possible to retrieve data based on filters specified on the main fields of events.

3.2.2 The simplified xAPI specification

Table 3 describes the simplified xAPI specification. The main field names in Table 3 contain identifiers. Those fields are mainly used for filtering, to retrieve either all actions by a specific user or all instance of a specific kind of action. The original JSON field contains the event in its original format. The purpose of this field is to retain all the original information, so as to avoid information loss when mapping other specifications to xAPI.

Table 3: Specification definition

Field name	Type	Description
Actor	String	An identifier of the user who performs the action
Verb	String	An identifier of the action that the user performs
Starttime	Date	Date when the action starts
Object	String	Identifier of the object/outcome resulted from the action.
Target	String	Target contains the identifier of the target of the action (Eg: John saved a movie to his wishlist, the target of the activity is "wishlist")
Context	String	Context contains the identifier of the course
Original JSON	Text	Open field but should contain JSON.

3.2.3 The process from the data producers to the LRS.

The data consumption process is simple. Activity consumers gather the data with the GET method described in Table 2.

Collecting the data is a more complex process in open learning environments. The tracking process is different depending on the mechanisms provided by each service. We usually have two possibilities: the so-called wrappers [Scheffel et al., 2010] and collecting agents [Butoianu et al., 2010].

Whilst the wrappers are plug-ins installed in the data producers, trackers are independent components that synchronise the data among components. The implementation of the former is dependent of the plugin architecture and the latter has a more generalisable implementation consuming the data from standard REST services.

This paper summarises the tracker implementation in seven steps. Courses and users are the core of this process. The seven steps are the following:

1. fetch information about the services (i.e. data producers) related to the course. For example, if the students use blogs in the course, the tracker requests what blog platforms and the urls of the blogs,

2. iterate over these different services,

3. fetch the user data (the result is usually an array of user actions), item iterate over this array of user actions,

4. transform every user action to simplified xAPI,

5. enrich this data with some specific information such as the course identifier,

6. push the event to the LRS.

Trackers are scheduled tasks that iterate over these seven steps. These steps are adapted to different requirements. For example, step 1 can get this information from different services. The case study described in section 4 contains the services related to the course in a simple Google spreadsheet, as an LMS is not used in the course. However, the other case study is the LMS that exposes this information. The LMS and the Google spreadsheet expose the information in different formats. The same occurs when trackers fetch the user data from the data producers. For example, the Wordpress API does not use the same data specification than the Twitter API.

This means that a new tracker is deployed when a new data producer, which exposes the user data with its proprietary specification, is integrated in the architecture. This keeps the trackers independent from the different data producer requirements. If a data producer changes requirements, the other trackers keep working without interruption.

In our case studies, the trackers are deployed in the cloud (i.e. Google App Engine) instead of dedicated servers to reduce maintainability and to increase availability of the services. Google App Engine is a Platform as a Service (PaaS), a framework that developers can build upon to develop applications. A cloud based service starts the tracker on regular intervals.

The architecture workflow is extended with real case studies in the following sections.

4 Blended learning university courses

Blended learning courses are an excellent setup for tracking learning activity as students interact with each other using digital means, which means that their 'digital exhaust' can be analysed [Park and Jo, 2015]. Similarly, when they interact in face-to-face settings, their interactions can be captured with microphones, cameras, etc. [Schneider et al., 2015], though we did not deploy such sensors in our experiments.

4.1 Course approach

Figure 2 describes the flow of information in these courses. We illustrate the information flow through a typical interaction scenario.

In these courses, students work in groups of 3 or 4. They learn to brainstorm, design, implement and evaluate tools, following a user centred rapid prototyping approach from paper prototype to fully working digital prototypes. Students evaluate every iteration with representative users to detect usability problems

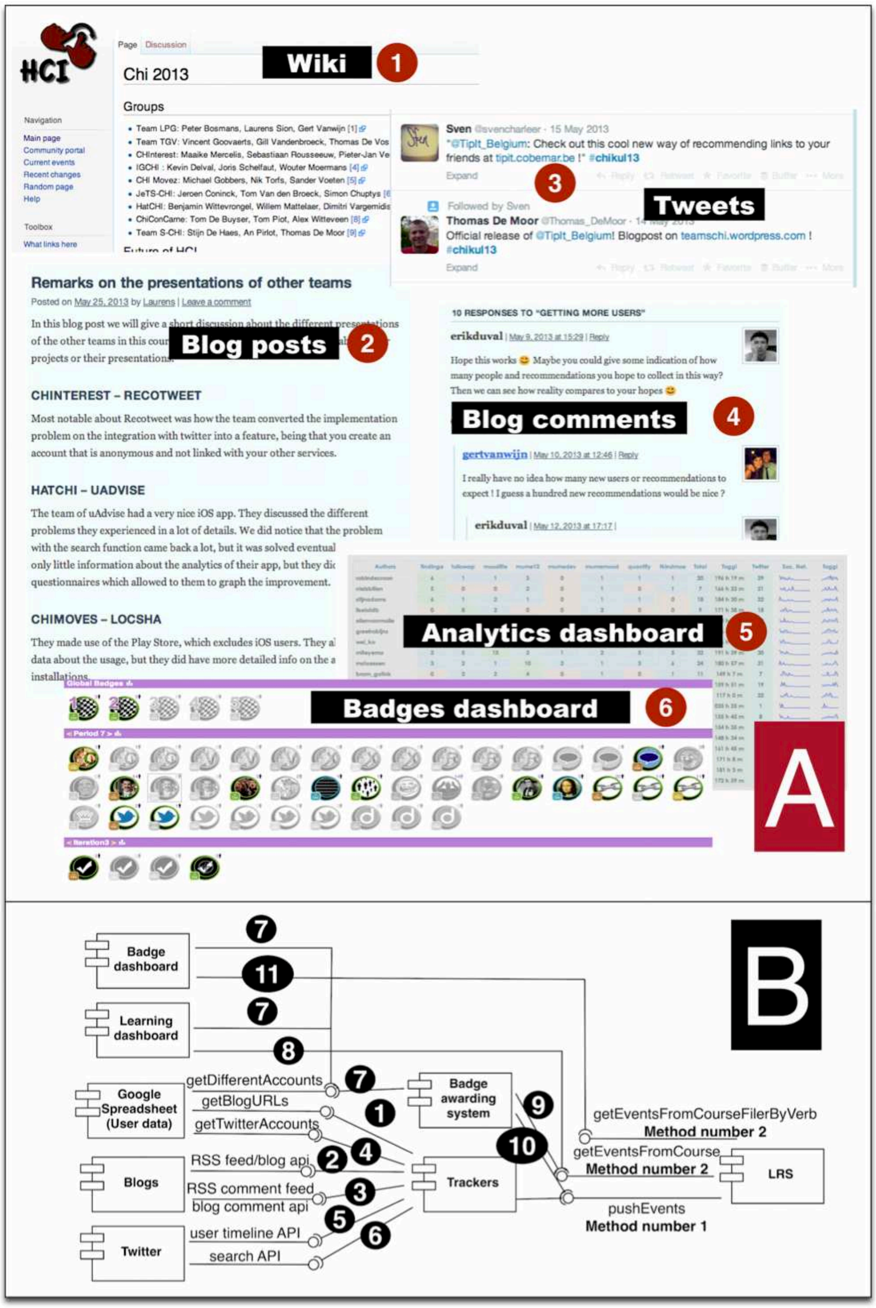

Figure 2: Course architecture - method numbers correspond to the N column in table 2

early on and address them in subsequent iterations. In these courses, a Community of Practice approach is adopted in which students collaborate in an open and transparent way [Duval et al., 2014].

The main hub of the course is a wiki (Figure 2a-1) that is mainly maintained by the students themselves. They report progress on group blogs (Figure 2a-2 as an example of blog post) and comment on the blogs of other groups, as does the teaching staff (Figure 2a-3 as an example of blog comments). Twitter is used to share short status messages about work on the assignment, share interesting resources and ask questions to the teaching staff. Students include in the tweet the hashtag of the course, like for instance '#chikul13'. Examples are shown in Figure 2a-4.

As a simple approach to manage identities of the groups and individuals involved over the different components, the twitter and blog handles are stored in a Google Spreadsheet, in order to enable tracking of the activity.

4.2 Tracking blogging activity

First, we track the blog activity. Figure 2b-1,2,3 summarises the steps. Figure 2b-1 illustrates the tracker fetching the blog urls of the students out of the Google spreadsheet. The tracker then fetches the posts (Figure 2b-2) and comments (Figure 2b-3) from each blog individually.

These steps are extended in Figure 3. In our case study, students use two different blogging platforms: Medium and Wordpress. While the former only exposes blog posts via RSS feeds, the latter exposes blog posts and comments through an API in addition to the usual RSS feed. The data provided by the API is richer than the information provided by the RSS feed.

Table 4: Actual mapping examples

	Actor	**Verb**	**Starttime**	**Object**	**Target**	**Context**
Blog post	robindecroon	post	2012-07-22 14:56:20	http://goo.gl /VaLfGb	-	thesis12
Blog comment	svencharleer	comment	2013-11-13 14:44:33	http://goo.gl /Xcqh3u	-	mume13
Tweet	jlsantoso	tweet	2013-05-09 09:44:01	http://goo.gl /QnpKOx	DVargemidis	chikul13
AR Learn	google_10593 91395511084 73521	response	2014-01-23 10:56:20	http://goo.gl /Dkm9lg	-	26368

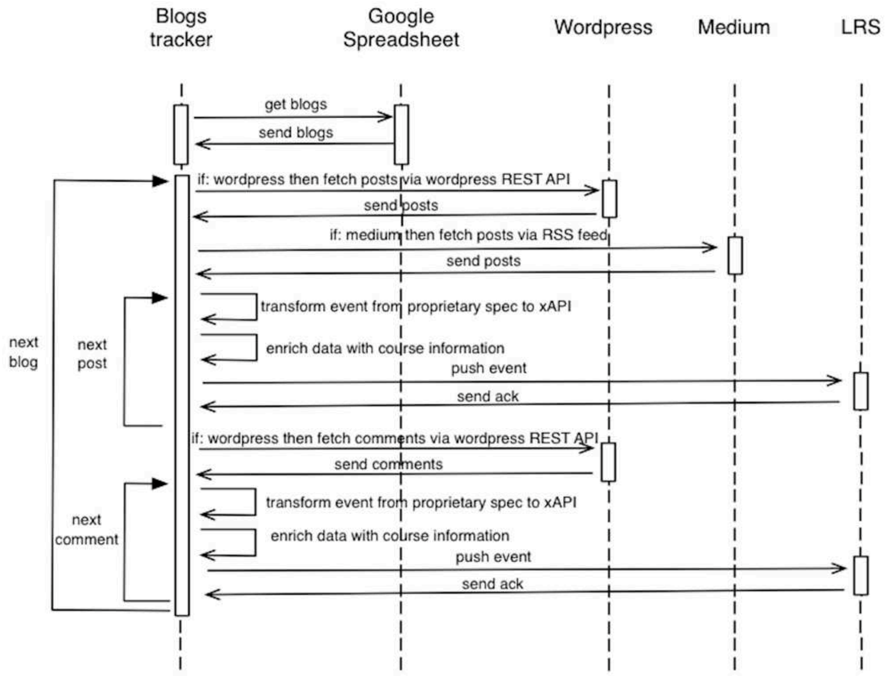

Figure 3: Sequence diagram of the blog tracker (Medium & Wordpress).

Students report the urls of their blogs and their user names through a Google Spreadsheet. Thus, the first action that the tracker performs in Figure 3 is gathering the information regarding the services used in the course (step 1 of the seven steps described in section 3.2.3) - in this case, the blog urls. If the blog is hosted on Wordpress (step 2), we access the REST API of WordPress (step 3). If the blog is hosted on Medium (step 2), the tracker accesses the RSS feed (step 3) since it does not provide a REST API. The tracker iterates over the fetched data (step 4), transforms it to xAPI (step 5), add the identifier of the course to the event (step 6) and pushes it to the learning record store (step 7 - service 1 in Table 2). Then the tracker proceeds with processing blog comments. We only fetch comments from Wordpress, as Medium does not expose the comments.

The first and second row in table 4 present the mapping of a blog post and a comment example respectively: 1) The username is the user handle; 2) The verb is 'posted' or 'commented'; 3) The start time is the date when the blog post comment was done; 4) the object is the url of the blog post or comment; and 5) The context is the course identifier.

4.3 Tracking tweeting activity

A similar process is used to fetch data from Twitter: the tracker calls the search API (Figure 2b-6) to collect all the tweets containing the hashtag of the course: '#chikul13'. As the search API does not index all the tweets, the tracker also queries all the user timelines of the students to be sure that we do not lose any tweet from them (Figure 2b-5). Of course, we filter out only the tweets related to the course, based on the hashtag. The search API is also used to fetch tweets from external participants that comment on the course progress. These specific details illustrate the importance of implementing the trackers in specific ways for specific components.

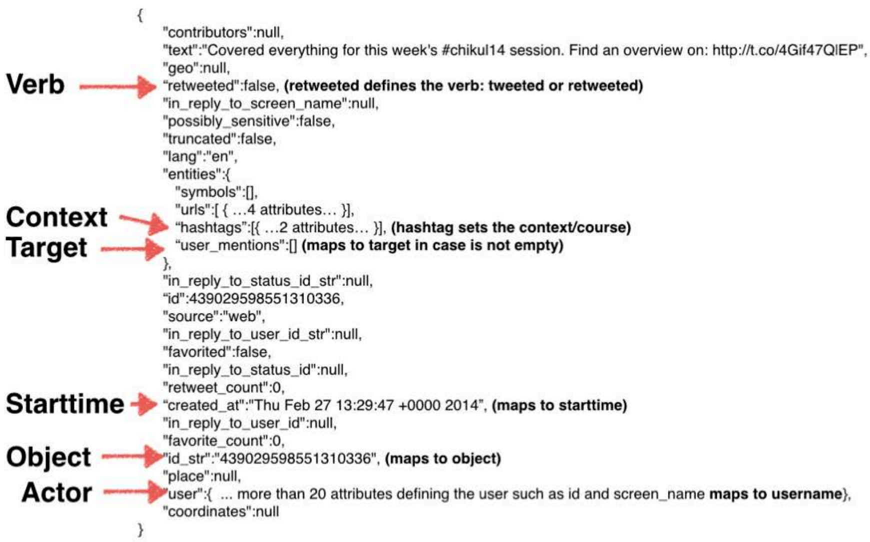

Figure 4: Snapshot of a simplified twitter event. The bold text indicates how the field was mapped.

The third row in table 4 presents a tweet example: 1) The username is the twitter handle, 2) The verb is 'tweet', 3) The start time is the date when the message was tweeted, 4) the object is the url of the tweet, 5) The target is the user who was mentioned in the tweet and 6) The context is the course hashtag.

Figure 4 shows a twitter JSON example and the mapping of the actual attributes into the main fields of the xAPI specification. A tweet has more than 50 attributes.

This means that choosing the right identifier is not straightforward sometimes. The user identifier is the twitter handle in the tweet example. However, twitter also exposes the system identifier, an alphanumeric value. Both, the twitter handle and the system identifier are unique. We choose the twitter handle because, as described before, the users have to provide their own identifiers. Therefore, they probably know their own twitter handles better than their system identifiers.

The complete JSON described in Figure 4 is stored in the original JSON field. The original JSON field ensures completeness of data for future research purposes. This data can also be consumed by additional services that require the information encoded in the original source.

4.4 The activity consumers

Students reflect on their activity with two different applications: a learning dashboard [Santos et al., 2013a] (Figure 2a-5) and a badge dashboard (Figure 2a-6) [Santos et al., 2013b].

The Learning Dashboard and the badge awarding system fetch information regarding groups and individuals from the Google Spreadsheet (Figure 2b-7). Based on that information, the dashboard and the badge awarding system query the LRS using the corresponding filters (Figure 2b-8 and 5b-7 - GET service in Table 2). The badge awarding system evaluates the rules and pushes the awarding of badges as an event if the student meets the requirements of the rules (Figure 2b-10 - POST service in Table 2). The badge dashboard obtains all badge events (Figure 2b-11) in order to display them. The result is illustrated by Figure 2a-6).

5 weSPOT Integration Example

In order to illustrate the versatility of our architecture, this section explains how we have deployed it in a very different context: a high school inquiry based course, as part of the weSPOT project. weSPOT aims at propagating scientific enquiry as the approach for science learning and teaching in combination with today's curricula and teaching practices [Mikroyannidis et al., 2013].

5.1 Course approach

As an example, a teacher may want to raise the inquiry skills of her students on the domain of 'Energy Efficient Buildings'. Thus, she creates an inquiry called 'Batteries discovery' in the Enquiry Workflow Engine (Figure 5a-1). The goal of the inquiry is to raise the awareness on how many batteries there are in a building and on how these batteries are used.

Six phases of learning activities are often discerned in an EBL process model: problem identification, operationalisation, data collection, data analysis, interpretation and communication [Mikroyannidis et al., 2013]. These phases link to the different tabs in the Inquiry Workflow (see Figure 5a-1). In every phase, there are several recommended widgets that are deployed by default (e.g.: a widget to create hypotheses in the first phase - Figure5a-2).

The third phase called 'Collect the data' contains a widget that the teacher can use to create collection tasks for documents such as text, pictures and videos. Therefore, she creates two data collection tasks: collect videos and pictures of batteries in the building. This widget connects to an external system, the mobile Personal Inquiry Manager (PIM) that is based on the ARLearn architecture [Ternier et al., 2012]. This connection enables the widget to automatically create the tasks in ARLearn (Figure 5a-3 shows the ARLearn screen displaying the last documents uploaded to a task). Doing so, the students are able to log on in the mobile app and collect text, pictures and videos. The connection is bidirectional: once the documents are collected, they are pushed to the Inquiry Workflow Engine after which the collected data is immediately accessible from within the widget.

5.2 Tracking collected data

Figure 5b-2,3,4 and 5 summarise the technical workflow of the data collection process. 2 and 3 connect to the LMS and get data related to the courses and data collection tasks. 4 and 5 fetch the data from the data collection tool.

These steps are extended in Figure 4. First, the tracker fetches the course information, and all inquiries that are created (step 1 of the seven steps described in section 3.2.3). Second, the tracker retrieves the list of associated data collection ids to each inquiry (step 1). ARLearn is used in other contexts than weSPOT. Therefore the tracker only accesses to the associated data collection ids (identified as runIds) (step 2). The tracker fetches (step 3) and iterates over the events from ARLearn (step 4). Every event is transformed to xAPI (step 5), enriched with specific inquiry information (step 6) and, finally, stored in the LRS (step 7 - service 1 in Table 2).

Fourth row in table 4 presents a ARLearn event example: 1) The username is the user system identifier; 2) The verb is response; 3) The start time is the date when the user collected the sample; 4) the object is the url to the sample; and 5) The context is the identifier of the course.

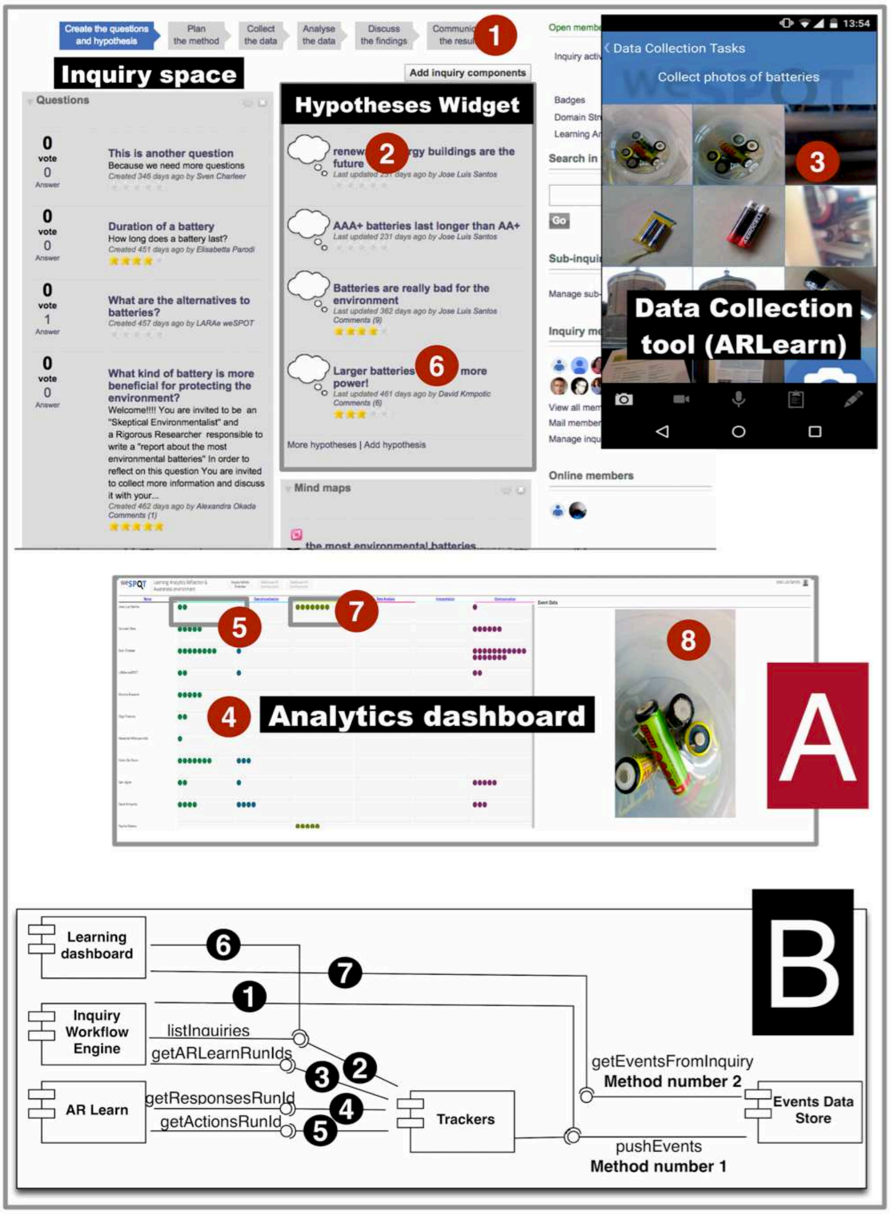

Figure 5: analytics architecture examples - method numbers correspond to the N column in table 2

5.3 Pushing the data from the Enquiry Workflow Engine to the LRS

The communication between the Enquiry Workflow Engine and ARLearn is independent from the learning traces data collection process. As the Enquiry Workflow Engine enables built-in plugins, there is a plugin that pushes the events to the LRS. For example, when a user performs an action such as 'Create hypothesis', the Enquiry Workflow Engine pushes this information into the Learning Record Store (Figure 5b-1 - POST service in Table 2).

5.4 The activity consumer

A dashboard gets the complete list of inquiries from the Inquiry Workflow Engine in which the user is enrolled (Figure 5b-6). As a next step, the dashboard queries the LRS to fetch the events related to the list of inquiries (GET service in Table 2) (Figure 5b-7).

The final result is illustrated by Figure 5a-4 to 9. For example, Figure 5a-5 shows that the user has created one hypothesis and he has replied to a comment of another student within the hypothesis widget (Figure 5a-6). This information is represented with dark green circles in the dashboard (Figure 5a-5). The collected data is represented with light green dots as in Figure 5a-7. These artefacts were collected with ARLearn. The circles are clickable. If the user clicks on the circle, the source is displayed at Figure 5a-8. The url of the object is stored in the event metadata as explained in the examples in table 4.

6 Conclusions

As we describe in Section 3.1, simplicity and flexibility are important factors that enable an architecture to be adaptable a new requirements from data providers and consumers. We illustrate that the architecture meets these requirements by describing the successful deployment in two different case studies and identifying three main components that enable such flexibility and simplicity of the architecture:

- *Simple services*: We implemented a very simple LRS that enables third-party services to push and fetch user data. These services are defined in Table 2 and they rely on the simplified xAPI specification described in Table 3.

- *Simple data schema*: We present a simplification of xAPI. Data consumers can still query and find all the events based on the core set of fields described in Table 3. Moreover, we reduce the effort in the mapping process. This can also be considered as a limitation, as we discuss in section 7.

- *Trackers*: This work puts special emphasis on the description of a generalisable implementation of trackers (Section 3.2.3) and their independency of the data store. The trackers hold the responsibility of dealing with the connection among the activity provider, the learning management system and the data store.

The presentation of successful case studies supports the idea that the architecture meets the **functionality** requirement enabling the collection and the consumption of learning traces.

Different to other studies such as ([Ruipérez-Valiente et al., 2013], [Friedrich et al., 2011], [Corbi and Burgos, 2014], [Hecking et al., 2014], [Leony et al., 2013]) that report on other particularities of the learning traces management using other specifications such as ActivityStrea.ms and CAM, this study reports on the benefits of deploying the trackers outside of the LMS.

The presentation of successful case studies supports the idea that the architecture meets the **functionality** requirement enabling the collection and the consumption of learning traces.

Trackers are a key of the architecture to meet the **extensibility**, **configurability** and **maintainability** requirements. Section 3.2.3 shows how is possible to reuse the tracker logic and to extend the architecture to other services. Sections 4 and 5 show how trackers are independent components that consume the necessary information from other services such as the learning management system and a simple Google Spreadsheet. Trackers do not require a re-deployment when the external information changes.

7 Limitations and future work

Simplicity usually comes with certain limitations. This section describes our concerns about the limitations of our work.

Collecting data about an event in its original format ensures the completeness of the data and keeps the semantics of the original specification. However, this mapping process includes much of the detail of the original specifications in an open JSON field. This can be addressed extending the GET service. Similarly to other services that enable either to return the data in JSON or XML format, this extension would return the details of the original specifications fully xAPI encoded. This would require the implementation of a small module per specification that provides such mapping. This solution would enable to keep the events stored in their original formats and increase the interoperability of our architecture with other systems that make use of other xAPI fields not included in the described data schema.

This work focuses on defining a set of simple services, but we do not discuss the software and hardware that supports the architecture. In our case, the architecture centralises all the traces in a central LRS. However, as we are aggregating data from several systems, **performance** and **scalability** issues may appear and management of identifiers across systems can become complicated. This is an issue we have not addressed in this paper. It has not been a problem in our deployments, but a larger scale roll-out of our architecture would probably encounter this problem.

We have tackled this issue experimenting with different technologies, Java vs JavaScript, Tomcat vs NodeJS and PostgreSQL vs NodeJS. Our final decision is JavaScript, NodeJS and MongoDB. This solution is very similar to Learning Locker and Go-labs [Hecking et al., 2014] that also use MongoDB. The solution presented in this paper remains also compatible with xAPI solutions such as Cloud Scorm and Learning Locker.

Similarly, **privacy** issues are out of scope for this paper. The architecture deploys a simple authentication mechanism to ensure that the data will be exclusively consumed by tools deployed and used in the scope of the courses. Therefore, the data is not open to unknown third-party services. Aligned with Pardo and Siemens [2014], we also highlight that privacy and authentication has many ethical dimensions: users need to be aware what systems are doing with their data and we need to provide mechanisms to enable them to control who does what with which data in a usable way.

Note - The complete code of our architecture is open and freely available NodeJS and MongoDB[1]. We hope that our work can be useful to researchers and practitioners in this field to enable flexible and simple collection and management of learning traces in open learning environments that go beyond current practices.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No 318499 - weSPOT project. Katrien Verbert is a postdoctoral fellow of the Research Foundation Flanders (FWO).

References

[Bass et al., 2006] Bass, L., Clements, P., Kazman, R.: Software architecture in practice; ISBN 0321154959; Addison-Wesley Professional, 2006; second edition edition.

[1] https://github.com/svencharleer/datastore

[Behringer and Sassenberg, 2015] Behringer, N., Sassenberg, K.: “Introducing social media for knowledge management: Determinants of employees? intentions to adopt new tools”; Computers in Human Behavior; 48 (2015), 0, 290 – 296.

[Butoianu et al., 2010] Butoianu, V., Vidal, P., Verbert, K., Duval, E., Broisin, J.: “User context and personalized learning: a federation of contextualized attention metadata”; J. UCS; 16 (2010), 16, 2252–2271.

[Corbi and Burgos, 2014] Corbi, A., Burgos, D.: “Review of current student-monitoring techniques used in elearning-focused recommender systems and learning analytics. the experience api & lime model case study”; International Journal of Artificial Intelligence and Interactive Multimedia; 2 (2014), 7, 44–52.

[del Blanco et al., 2013] del Blanco, A., Serrano, A., Freire, M., Martinez-Ortiz, I., Fernandez-Manjon, B.: “E-learning standards and learning analytics. can data collection be improved by using standard data models?”; Global Engineering Education Conference (EDUCON), 2013 IEEE; 1255–1261; 2013.

[Duval et al., 2014] Duval, E., Parra, G., Santos, J. L., Agten, S., Charleer, S., Klerkx, J.: “Learning hci and infovis in the open”; Building Bridges: HCI, Visualization, and Non-formal Modeling; 8–16; Springer, 2014.

[Fielding, 2000] Fielding, R. T.: Architectural styles and the design of network-based software architectures; Ph.D. thesis; University of California (2000).

[Fowler, 2001] Fowler, M.: “Is design dead?”; SOFTWARE DEVELOPMENT-SAN FRANCISCO-; 9 (2001), 4, 42–47.

[Friedrich et al., 2011] Friedrich, M., Wolpers, M., Shen, R., Ullrich, C., Klamma, R., Renzel, D., Richert, A., von der Heiden, B.: “Early results of experiments with responsive open learning environments”; J. UCS; 17 (2011), 3, 451–471.

Govaerts, S., Verbert, K., Dahrendorf, D., Ullrich, C., Schmidt, M., Werkle, M., Chatterjee, A., Nussbaumer, A., Renzel, D., Scheffel, M., et al.: “Towards responsive open learning environments: the role interoperability framework”; Proceedings of EC-TEL’11; 125–138; Springer, 2011.

[Hecking et al., 2014] Hecking, T., Manske, S., Bollen, L., Govaerts, S., Vozniuk, A., Hoppe, H.: “A flexible and extendable learning analytics infrastructure”; E. Popescu, R. Lau, K. Pata, H. Leung, M. Laanpere, eds., Advances in Web-Based Learning – ICWL 2014; volume 8613 of Lecture Notes in Computer Science; 123–132; Springer International Publishing, 2014.

[Leony et al., 2013] Leony, D., Glvez, H. A. P., Muoz-Merino, P. J., Pardo, A., Kloos, C. D.: “A generic architecture for emotion-based recommender systems in cloud learning environments”; Journal of Universal Computer Science; 19 (2013), 14, 2075–2092.

[Megliola et al., 2014] Megliola, M., Vito, G. D., Wild, F., Lefrere, P., Sanguini, R.: “Creating awareness of kinaesthetic learning using the experience API:

current practices, emerging challenges, possible solutions"; Proceedings of the 4th Workshop on Awareness and Reflection in Technology-Enhanced Learning In conjunction with the 9th European Conference on Technology Enhanced Learning: Open Learning and Teaching in Educational Communities, ARTEL at EC-TEL 2014, Graz, Austria, September 16, 2014.; 11–22; 2014.

[Mikroyannidis et al., 2013] Mikroyannidis, A., Okada, A., Scott, P., Rusman, E., Specht, M., Stefanov, K., Boytchev, P., Protopsaltis, A., Held, P., Hetzner, S., Kikis-Papadakis, K., Chaimala, F.: "wespot: A personal and social approach to inquiry-based learning"; J. UCS; 19 (2013), 14, 2093–2111.

[Niemann et al., 2013] Niemann, K., Wolpers, M., Stoitsis, G., Chinis, G., Manouselis, N.: "Aggregating social and usage datasets for learning analytics: Data-oriented challenges"; Proceedings of the Third International Conference on Learning Analytics and Knowledge; LAK '13; 245–249; ACM, New York, NY, USA, 2013.

[Nistor et al., 2014] Nistor, N., Trăuşan-Matu, Ş., Dascălu, M., Duttweiler, H., Chiru, C., Baltes, B., Smeaton, G.: "Finding student-centered open learning environments on the internet: Automated dialogue assessment in academic virtual communities of practice"; Computers in Human Behavior; (2014).

[Offutt, 2002] Offutt, J.: "Quality attributes of web software applications"; IEEE software; 19 (2002), 2.

[Pardo and Siemens, 2014] Pardo, A., Siemens, G.: "Ethical and privacy principles for learning analytics"; British Journal of Educational Technology; 45 (2014), 3, 438–450.

[Park and Jo, 2015] Park, Y., Jo, I.-H.: "Development of the learning analytics dashboard to support students' learning performance"; Journal of Universal Computer Science; 21 (2015), 1, 110–133.

[Ruipérez-Valiente et al., 2013] Ruipérez-Valiente, J. A., Muñoz Merino, P. J., Kloos, C. D.: "An architecture for extending the learning analytics support in the khan academy framework"; Proceedings of the First International Conference on Technological Ecosystem for Enhancing Multiculturality; TEEM '13; 277–284; ACM, New York, NY, USA, 2013.

[Sakurai, 2014] Sakurai, Y.: "The value improvement in education service by grasping the value acceptance state with ict utilized education environment"; S. Yamamoto, ed., Human Interface and the Management of Information. Information and Knowledge in Applications and Services; volume 8522 of Lecture Notes in Computer Science; 90–98; Springer International Publishing, 2014.

[Santos et al., 2013a] Santos, J., Charleer, S., Parra, G., Klerkx, J., Duval, E., Verbert, K.: "Evaluating the use of open badges in an open learning environment"; Proceedings of EC-TEL'13; 314–327; Springer, 2013a.

[Santos et al., 2013b] Santos, J. L., Verbert, K., Govaerts, S., Duval, E.: "Addressing learner issues with stepup!: an evaluation"; Proceedings of the LAK'13; 14–22; ACM, 2013b.

[Scheffel et al., 2010] Scheffel, M., Friedrich, M., Niemann, K., Kirschenmann, U., Wolpers, M.: "A framework for the domain-independent collection of attention metadata"; Proceedings of EC-TEL'10; 426–431; Springer, 2010.

[Schneider et al., 2015] Schneider, J., Börner, D., van Rosmalen, P., Specht, M.: "Augmenting the senses: A review on sensor-based learning support"; Sensors; 15 (2015), 2, 4097–4133.

[Ternier et al., 2012] Ternier, S., Klemke, R., Kalz, M., Van Ulzen, P., Specht, M.: "Arlearn: Augmented reality meets augmented virtuality."; Journal of Universal Computer Science; 18 (2012), 15, 2143–2164.

[Verbert et al., 2013] Verbert, K., Duval, E., Klerkx, J., Govaerts, S., Santos, J. L.: "Learning analytics dashboard applications"; American Behavioral Scientist; 57 (2013), 10, 1500–1509.

[Vozniuk et al., 2013] Vozniuk, A., Govaerts, S., Gillet, D.: "Towards portable learning analytics dashboards"; Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on; 412–416; 2013.

[Wilson et al., 2007] Wilson, S., Liber, O., Johnson, M. W., Beauvoir, P., Sharples, P., Milligan, C. D.: "Personal learning environments: Challenging the dominant design of educational systems"; Journal of e-Learning and Knowledge Society; 3 (2007), 2, 27–38.

[Wolpers et al., 2007] Wolpers, M., Najjar, J., Verbert, K., Duval, E.: "Tracking actual usage: the attention metadata approach"; Educational Technology & Society; 10 (2007), 3, 106–121.

[Zhang et al., 2015] Zhang, X., Gao, Y., Yan, X., de Pablos, P. O., Sun, Y., Cao, X.: "From e-learning to social-learning: Mapping development of studies on social media-supported knowledge management"; Computers in Human Behavior; (2015).