Iria Estévez-Ayres, Jesús Arias Fisteus, Carlos Delgado-Kloos. Lostrego: A distributed stream-based infrastructure for the real-time gathering and analysis of heterogeneous educational data, In *Journal of Network and Computer Applications* (2017), v. 100, pp. 56-68
https://doi.org/10.1016/j.jnca.2017.10.014

Lostrego: A distributed stream-based infrastructure for the real-time gathering and analysis of heterogeneous educational data

Iria Estévez-Ayres , Jesús Arias Fisteus, Carlos Delgado-Kloos

The quick technological evolution of the last decades has also reached learning environments, where the use of networked computing devices such as laptops, smartphones, tablets, IoT devices, servers, etc. is continuously growing. In particular, those computerized learning environments have the potential to track the activity of teachers and students in them, which enables the development of innovative applications that enrich the learning process by analyzing the collected data. The majority of related work in this field has been centered on batch gathering and analysis of the data. However, in order to integrate more reactive applications, there is a need for an infrastructure that enables the real time collection and analysis of data in learning environments. Such an infrastructure should be scalable and flexible enough to cope with heterogeneous data coming from different types of learning settings. This paper presents Lostrego, a stream based, modular, scalable and flexible distributed infrastructure that allows the gathering and analysis of educational data from heterogeneous data sources in a real time fashion. Lostrego applications are composed by interconnected services that can be reused in different courses. The results of the evaluation of Lostrego in two editions of a computer programming course with 233 students and 384,702 gathered events are also reported.

1. Introduction

Since technology irrupted into the learning landscape, understand ing the learning process and providing a more individualized learning experience by using non invasive techniques has become a concern (Suppes, 1968).

There are two closely related research fields devoted to enhancing teaching and learning through the study of the learners' behavior: learning analytics and educational data mining (EDM). Learning analytics is defined as "*the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs*" (Siemens and Gasevic, 2012); while EDM is devoted to "*developing, researching, and applying computerized methods to detect patterns in large collections of educational data that would otherwise be hard or impossible to analyze due to the enormous volume of data within which they exist*" (Romero and Ventura, 2013). Both disciplines rely on collecting learners' data from different sources in order to apply their different techniques.

As the learning process is becoming more ubiquitous and learners engage in different settings through different devices (Pérez Sanagustín et al., 2012), there is a growing necessity to gather and analyze huge volumes of data coming from different sources, platforms and tech nologies, in an effort to capture the whole learning experience (Ferguson et al., 2016). Moreover, although there are tools to analyze the learner's experience, they are usually tightly coupled to specific systems, such as LMS or MOOC platforms, as they need to access to the internal records of these systems to perform their analysis (Del Blanco et al., 2013).

The Advanced Distributed Learning (ADL) Initiative of the U.S. Department of Defense and the IMS Global Learning Consortium are aware of this problem and propose their respective specifications to record learning experiences and outcomes: the Experience API speci fication (xAPI) (Experience, 2014) and the IMS Caliper Analytics Learning Measurement Framework (Caliper) (Haag et al., 2015). They try to maximize the interoperability of services that create, gather, store and process information about learning experiences. Both specifications use the JSON (JavaScript Object Notation) data format, both define an API to send learning records (xAPI) or events (Caliper), both store all this information within repositories (LRS, in the case of xAPI, Event Stores, within the Caliper framework), which will be later accessed by others to perform learning analytics, and both can include privacy controls to access their repositories. In the case of Caliper, the specification is silent about the protocols to transfer the

data, while xAPI follows the guidelines of the REST paradigm. However, both can be viewed as semantic technologies that rely strongly on the use of data repositories. Moreover, the tracking of learning experiences as defined in the xAPI specification requires storing the learning events in a repository in order to make them available to consumers.

One of the challenges in the field of Technology Enhanced Learning (TEL) is adapting and providing appropriate support to the learner in the right place and in the right time (Hwang, 2014). In order to make it possible, there is not only the need to collect, merge and analyze data from different sources, but also to provide real time feedback to both learners and teachers (Kinshuk et al., 2016). In a setting where continuous updates of the application are needed due to the continuous arrival of events, a model where the communication is initiated always by the client can become impractical (Babu and Widom, 2001). It is the case of the xAPI Query interface implemented by the ADL.Collection API,[1] which offers a set of SQL like queries that force the client to perform polling. Data streaming is a more natural paradigm when new data are constantly being collected and need to be processed on the fly (Tatbul, 2010). Moreover, it eases the development of applications where the communication is initiated by the infrastructure as soon as a new event is available (Chandrasekaran et al., 2003), usually following the publish subscribe paradigm (Ghate and Pati, 2016).

In this paper we present Lostrego, a generic, modular, scalable and flexible publish subscribe infrastructure that facilitates the gathering and analysis of educational data from heterogeneous data sources in a real time fashion. The decoupling nature of the publish subscribe paradigm enables the development of responsive systems (Kim et al., 2010). By taking advantage of these benefits, Lostrego enables the development of applications that require immediacy when dealing with educational data.

The rest of the paper is organized as follows: Section 2 presents the Lostrego infrastructure, its design requirements, its architecture and implementation; Section 3 defines the core services of Lostrego; a case study where Lostrego was deployed during two semesters is described in Section 4; Section 5 presents the results, while Section 6 compares Lostrego with previous approaches; Section 7 discusses the main advantages and limitations of our proposal; and, finally, Section 8 concludes and presents the future work and directions of this research.

2. The Lostrego infrastructure

This section presents the design of the Lostrego infrastructure and discusses some of its possible uses.

2.1. Requirements

As educational environments become more complex, the monitoring of the learning process of students needs to become even more ubiquitous. Students and teachers use different tools (virtual campus, Twitter, IDEs, etc) in many different settings (in the lab, at home, while commuting, etc.). Thus, if the chosen monitoring system is focused on a single tool or environment, a lot of valuable data could be missed. Therefore, there is a needed for an infrastructure that *allows the gathering of many very different sources in an automatic way* (Ferguson et al., 2016).

The usefulness of the gathered data is related to if and when the data analysis happens. Thus, gathering data is not enough. Ideally, such infrastructure should support the implementation and integration of learning analytics tools that, from the gathered data, could give prompt real time feedback to teachers and students (Lewkow et al., 2016). In order to allow on the fly interventions and to make all the actors of the learning process aware *when things happen*, an infrastructure that allows *not only real time data gathering but also real time data analysis* is needed.

Besides, universities are opening their classrooms to the world through different initiatives such as MOOCs and SPOCs (Fox, 2013). The number of students per course is higher in those kinds of courses than in traditional ones and, although different platforms use their own learning analytics tools, students frequently use additional external educational tools in them that should also be monitored. In this context, the proposed infrastructure *should be scalable* enough to cope with large numbers of students.

As different courses require different monitoring (Kinshuk et al., 2016), the *proposed infrastructure should be flexible* to allow the deployment of different reusable modules depending on the requirements of the course. Moreover, the infrastructure should support the dynamic composition of modules to create more complex applications, in order to cope with the inherent dynamism of a course enactment.

The infrastructure should *decouple data gathering from data processing*. In this way, the actual gathering of the data could happen outside the infrastructure (on other platforms or, even, infrastructures) when needed. Similarly, different external tools could be used to process the gathered data, as long as the needed data format converter is provided. Moreover, the infrastructure should be agnostic regarding data formats, thus allowing the coexistence of data from different standards and tools.

Taking into account the discussion above, the design of the Lostrego infrastructure was driven by the following main requirements:

- Gathering of *heterogeneous* data.
- *Automatic* and *real time* data gathering.
- *Automatic* and *real time* data processing.
- *Scalability* in order to cope with large numbers of courses and students.
- *Flexibility* for building *custom applications* on top of it.
- *Loosely coupled* infrastructure, allowing the decoupling between data gathering and processing.

2.2. Architecture

As shown in Fig. 1, students interact with different and heterogeneous educational resources. These resources can be the desktop environment of their computer, the LMS where the course contents are hosted, specialized software such as, in computer programming courses, integrated development environments, and even social networks where students may have conversations about the course. The infrastructure has to monitor the interaction of the students with those resources and, in real time, create and send the data through the infrastructure in the form of *events*.

The Lostrego infrastructure follows the publish subscribe paradigm, which decouples *data publishers* (the educational data sources) from *data consumers* (the analysis infrastructure). Data flows through the infrastructure from educational data sources to the analysis infrastructure in the form of event streams. We follow the definition of *data stream* given by Golab and Özsu (2003): a *real time, continuous, ordered (implicitly by arrival time or explicitly by time stamp) sequence of items*.

Streams are flexible, in the sense that they can easily be filtered, split, merged, etc. Additionally, new streams can be created with data derived from other streams. That is, whereas some streams contain the original events gathered at the educational data sources, other streams may contain higher level events obtained from their processing. The latter are produced by modules within the analysis infrastructure. For example, a session tracking analysis module might process the original streams, infer working sessions (i.e., periods of time during which a learner is interacting with the educational resources) and produce a new stream with events that signal when every learner begins or finishes a working session. Another example would be a module that,

[1] https://github.com/adlnet/xAPI-Dashboard/blob/master/API_collection.md.

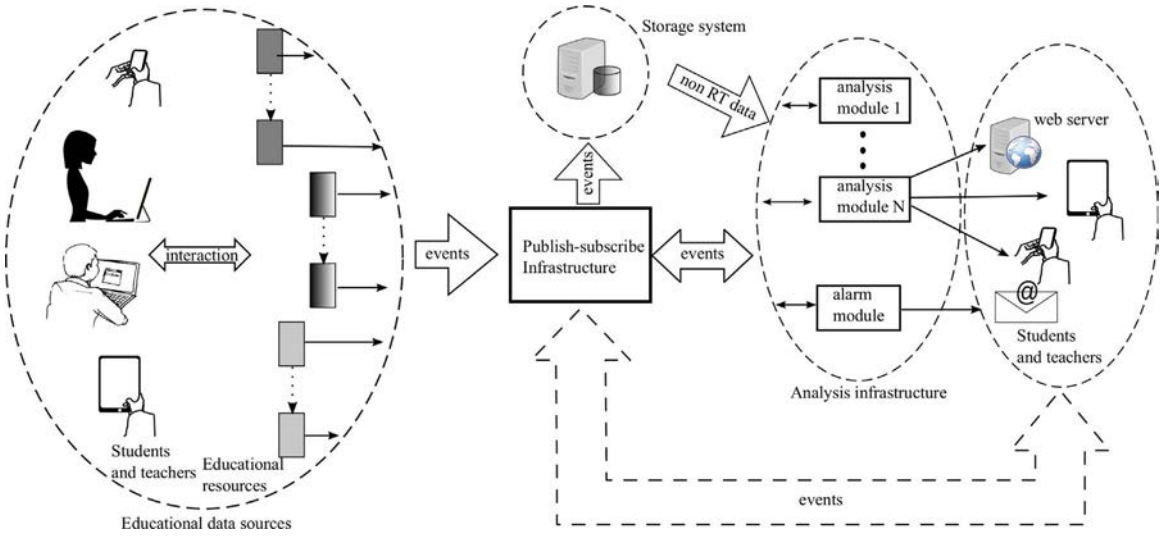

Fig. 1. Lostrego architectural infrastructure.

from that session tracking stream, computes the amount of working time every learner accumulates in the course.

In order to allow analysis modules to use not only the real time streams but also historical data, data streams can also be stored in databases. This is needed, for example, when an analysis module needs to detect potential student dropouts. This kind of analysis needs, besides a model of student behavior, the records gathered in the near past for every student.

This stream based publish subscribe architecture is flexible in the sense that it imposes no restrictions on the analysis modules and their connections. Different deployments could plug custom modules in according to their needs. In addition, end user applications can also consume the original and derived streams. Typical applications would be learning analytics dashboards and alarm systems. It is even possible for end user applications to inject new data back to the system (e.g., annotations entered by the instructor through a dashboard).

2.3. Implementation

We have implemented a prototype of the Lostrego architecture described in the previous section and deployed it as a case study in an actual course. This section describes our implementation and Section 4 presents its deployment in the case study.

2.3.1. Publish subscribe Infrastructure

The infrastructure has been built on top of the Ztreamy[2] publish subscribe stream middleware because of the following reasons (Fisteus et al., 2014):

- It is a scalable platform for publishing data streams on the Web, using HTTP(S) to consume and publish data.
- Communications can be secured by using the HTTPS protocol, since privacy is usually a requirement in educational environments.
- Consumers and producers can be developed in every major application development environment and programming language, with the only requirement of having HTTP(S) support. This simplifies the integration of different educational resources as data sources, since events can be sent to the infrastructure from every major programming language and platform.
- It supports stream duplication, aggregation and filtering, which are basic operations the analysis infrastructure needs.
- It is flexible, in the sense that different network layouts can be deployed depending on the needs of the application.
- It simplifies application development with functionality such as data serialization/deserialization and built in semantic filtering.

Monitoring agents run within the educational data sources (the desktop environment of lab computers, virtual machines used by the student at home, LMSs, web pages that contain class materials, etc.) They track the actions of the student, represent them as events (see Section 2.3.2 and the example in Fig. 2) and publish them by sending HTTP(S) POST requests to an end point within the Ztreamy publish subscribe server. Examples of such monitoring agents would be:

- In lab computers or virtual machines provided to the students, wrappers for the programs of interest are installed in the students' accounts. For example, in a programming course such commands would be integrated development environments, text editors, compilers, debuggers, code analysis tools and version control systems. Depending on the program, data such as the start and end time of their execution, command line parameters, standard output and error streams, working directory and finish status of the processes would be tracked. Our implementation of this part of Lostrego is based on (Romero Zaldivar et al., 2012).[3]
- In a learning management system such as Moodle or in MOOC platforms every interaction of the student would be tracked by a module running within the platform itself.
- When class materials are served in HTML format from a web server the instructors cannot control, a JavaScript agent would be embedded into every page to track visits and send the corresponding events to the infrastructure. If users need to authenticate to access these materials, the identity of the student can also usually be obtained by the JavaScript agent.
- Agents monitoring social networks such as Twitter will get data from the social network API (e.g., by monitoring certain hash tags or users), create events from the relevant data and inject them into the infrastructure.

We have implemented three of those types of monitoring agents for our current prototype of Lostrego, leaving the monitoring of learning management systems for future versions. Section 4.2 provides further technical detail about how those agents have been implemented and deployed in the case study in order to track lab computers, virtual machines and web materials.

[2] http://www.ztreamy.org/.

[3] https://github.com/dleony/PLA

```
Event-Id: 7cafe009-086a-4d1c-9895-52c520009e88
Source-Id: 1e35bcb8-d558-4493-b403-a093c2b69cab
Syntax: application/json
Timestamp: 2015-10-09T20:32:55+02:00
X-SentAt: 2015-10-09T20:32:57+02:00
X-StudentId: 42424242
Event-Type: gcc
Body-Length: 601
{
    "user": "astt",
    "pwd": "/home/astt/Project/version1",
    "command": ["gcc", "-g", "-Wall", "-o", "program", "main.c", "menu.c"],
    "status": 1,
    "num_stdout_lines": 0,
    "stdout": "",
    "num_stderr_lines": 8
    "stderr": "main.c: In function 'main':\n
               main.c:12:8: warning: unused variable 'comp' [-Wunused-variable]\n
                            int comp = strcmp(argv[1], inputHelp);\n        ^\n
               main.c:14:6: error: 'comp' undeclared (first use in this function)\n
                            if(comp == 0){\n      ^\n
               main.c:14:6: note: each undeclared identifier is reported only once for each function it appears in\n"
}
```

Fig. 2. Example of an actual Lostrego event.

The Ztreamy publish subscribe server creates a *raw event stream* with the events it receives from the educational data sources. Some analysis modules within the analysis infrastructure consume that stream. Some of them, such as the session tracking analysis module, may produce derived higher level streams, which are also served by the publish subscribe infrastructure. Other analysis modules, as well as the storage system, consume those derived streams and may produce in turn new derived streams.

Finally, end user applications consume the high level streams they need to provide their functionality to students and instructors. They might even produce new events and push them into the infrastructure.

2.3.2. Lostrego event objects

An event represents in Lostrego an action of the learner upon an educational resource. For example, a new event could be published when the learner enters a specific content in the LMS, runs a compiler in a computer programming course, uploads a submission, etc. Those events are modeled on top of the Ztreamy event model, which consists of several headers and a body object. Some of those headers are defined by Ztreamy (a unique event identifier, an event timestamp, etc.), but applications are allowed to define their own extension headers. Body objects are completely managed by applications, and may consist either of text or binary data.

More specifically, events in Lostrego contain the following headers, which are common for every action type (see Fig. 2):

- A globally unique event identifier (the header `Event Id` as shown in Fig. 2).
- A pair of timestamps with the instants in which the action occurred (`Timestamp` header) and the event is actually sent to the server (`X SentAt` header).
- The learner's identity (`X StudentId` header).
- The identity of the environment in which the event was created, e.g. the identifier assigned to a specific computer, virtual machine, etc. (`Source Id` header).

The detailed description of the action is placed in the body of the event as a JSON object. The information it carries is specific to the type of action, although actions may share some fields when appropriate. For example, as shown in Fig. 2, all the actions that involve issuing a command in the command line may include the user name of the student in the system, the current working directory, the command typed by the student, including its command line arguments, the success status of the process, the data written to its standard output and standard error, etc.

3. Data processing in lostrego

Whereas some analysis modules in Lostrego are course or institu tion specific, others provide generic functionality that may be useful in many scenarios. Those reusable modules are called *core services* of Lostrego. Scenario specific applications and analysis modules needing of their functionality may just consume their output. This section describes some core services that are already part of Lostrego.

3.1. Team annotation service

Because team working is nowadays a demanded soft skill (Andrews and Higson, 2008), courses with a strong collaborative component are becoming an essential part of formal education (Sahami et al., 2013). Thus, having the ability to monitor not only how individuals work but also how teams of students collaborate to solve problems is important for instructors.

The *team annotation service* is meant to bridge the gap between individual and collaborative work. It annotates incoming events, which already contain a header with the student's identity, with the identifier of the team the student belongs to, if any. The output stream of this service is a copy of its input stream with an additional header that contains the team identifier. Applications that need to track the progress of teams in collaborative exercises use this header to map events to teams.

Instructors configure this service by providing a list of teams and the identity of their members. The service is also configured with a date range. It will not annotate events whose timestamp is outside that range. This supports the reorganization of teams for different activities in a course, since a separate instance of the service would be deployed for each activity, configured with a different list of teams and date range.

3.2. Exercise detection service

This service matches an event to an exercise the student is currently working on. The output of this service is a copy of its input stream where the matched events include a new header with the exercise and session identifiers. The instructor may also specify other exercise related headers, such as whether the exercise is mandatory or intended as a teamwork activity.

This service is performed in two steps. In the first step, the instructor configures the service with the list of resources (directories, file names, web pages, etc.) for each exercise/session in the course. The service scans the body of every input event looking for mentions to these resources.

In the second step, which requires the input stream to be annotated with working sessions (see Section 3.3), those events for which the first step did not detect the exercise are annotated with the exercise of the latest event of the same working session annotated by the first stage. This approach is based on the fact that, in absence of more specific information, there is a high chance that the student continues with the same exercise as in the previous events. Since its output is not always exact, this second step may be optionally disabled by instructors.

3.3. Working sessions detection service

The knowledge of the time on task for each student can be useful for instructors in order to detect problems within a team (Petkovic et al., 2016). Moreover, how a student works on an assignment (e.g., in computer programming exercises, her behavior when facing compila tion errors or the amount of time she spends editing) has been proved to be a successful performance predictor (Jadud, 2006; Watson et al., 2013; Rodrigo et al., 2009). Furthermore, knowledge about how students manage their time, i.e., when they work on course activities, the length of those working sessions and their behavior during them can be used as early predictors of students' performance (Willman et al., 2015).

A student's working session is defined in Lostrego as a period of time during which a student is continuously working on one or more activities of the course. The objective of the *working session detection service* is grouping events into working sessions. The service takes an input event stream and generates two output streams: an event stream composed by events signaling the beginning and end of sessions, and a copy of the input in which events that have been successfully associated to a session are enriched with a new header that contains their session identifier.

Given a student with no active working session, the service determines that a working session begins with the first event tracked for her. It determines that a working session ends when a configurable amount of time (e.g., 30 min) passes without further events from her. In order to do that, the service keeps a table of active sessions in memory. Each session in the table is described with a unique identifier, the identity of the student, the timestamp t_0 of its first event and the timestamp t_1 of the most recent event associated to this session. For each input event with timestamp t_e, the service looks for the active session associated to its student:

- If there is no such session, a new session is added to the table and t_e is assigned to both t_0 and t_1. An event signaling the beginning of the session is published.
- If there is such session and $t_0 - \mu < t_e < t_1 + \tau$, the event is asso ciated to this session. If $t_e < t_0$, t_0 is updated to t_e. If $t_e > t_1$, t_1 is updated to t_e. The configurable parameter μ accounts for the fact that events are not guaranteed to arrive in order, for example when they come from separate monitoring agents or they get delayed because of network issues (e.g., when the monitoring agent cannot connect to the server when the event is created, it needs to be re transmitted later.) The configurable parameter τ represents the maximum period of time without receiving any event a session can be considered active.
- If there is such session but $t_e \leq t_0 - \mu$, the event is considered to be too old and is not mapped to any session.
- If there is such session but $t_e \geq t_0 + \tau$, the previous session is closed and a new one is added to the table. Two new events are created: one signaling the end of the previous session, which includes a summary with its start time, end time, duration, number of events, exercises the students worked on, etc., and one signaling the beginning of the new session.

All the sessions in the table are checked periodically in order to close those that have been inactive for too long. Being t the current timestamp, sessions with $t > t_1 + \tau$ are removed from the table. Similarly to the last case above, a new event that signals their end is published.

Since there is no consensus on how to compute the time on task (Kovanovic et al., 2016), the maximum inactivity period τ can be configured by the instructor.

Due to the inherent flexibility of the platform, in environments where instructors preferred to apply alternative session tracking algorithms, a custom module could be developed and used instead.

4. Case study

As a proof of concept, the Lostrego system was deployed in a second year computer programming course belonging to the Bachelor's Degree in Telecommunication Technologies Engineering at the University Carlos III de Madrid during the Fall semesters of 2015 and 2016.

This course introduces operating systems (focusing on UNIX) and discusses how to manage multiple processes and tasks that execute simultaneously and share resources. Students should be able to use both the Java and C programming languages. Additionally, students are expected to practice some traversal abilities in this course, namely teamwork (including the use of version control systems), time manage ment and verbal communication.

The course follows an active learning approach and applies a continuous assessment scheme. Therefore, students are expected, before face to face sessions, to work at home on several activities related to the concepts those sessions target. The course design has been improved following a methodology that involves feedback gather ing from students and teachers at specific milestones, as well as iterative refinement (Estévez Ayres et al., 2015).

The course material comprises two types of resources: course notes, available to the students at the website of the course, and practical material, which includes handouts and auxiliary files. All the practical material is organized in different folders and delivered to students through the Subversion[4] version control system. Those Subversion repositories are the *de facto* workspace for the students. At the beginning of the course, each team of students is given a personalized URL that points to their repository. Exercises and assignments are delivered to them through those repositories as the course advances, and they are expected to submit their solutions by the same means (Pardo and Kloos, 2011).

Being this course eminently practical, students are expected to work not only at the University laboratories but also at home. Since the exercises require a Linux environment and some specific software, they are provided at the beginning of the course with a Virtualbox[5] virtual machine that replicates the configuration of the computers of the laboratories. This way students can work at home without having to install the environment themselves.

4.1. Compliance with data privacy legislation

In the context of this course, in order to comply with the Spanish data privacy legislation, Organic Law 15/1999 on Personal Data Protection (Boletín Oficial del Estado, 1999), a document is shown to the students at the beginning of the course, and before downloading the virtual machine and the tracking tools. It explains the monitoring mechanisms, how to temporarily enable and disable them, how to permanently uninstall them, the events that are recorded, and the use of the gathered data. Students must agree with these conditions before continuing. As events were not only used and processed on the fly, but also a copy of them was stored at university facilities, a contact e mail was available for the students to exercise their rights to review, amend or delete the gathered events at any time during or after the course.

4.2. Deployment of the monitoring agents

Since this course follows a practical approach in which doing programming exercises on a computer is the most important student's activity, the case study was focused on tracking those programming activities. Additionally, during the Fall 2016 edition of the course the system tracked also their access to every page in the website of the course, which hosts all the course materials. This section provides further detail about how the monitoring agents (see Section 2.3.1) that

[4] https://subversion.apache.org/.
[5] http://www.virtualbox.com.

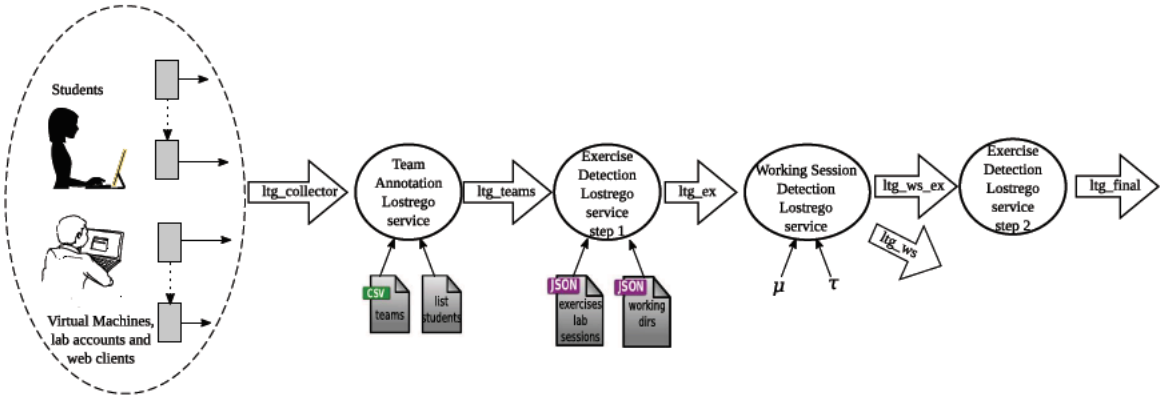

Fig. 3. Deployed Lostrego application.

track those students' actions have been deployed, since similar techniques may be useful in the deployment of other courses.

The virtual machines provided to students, which they used mainly at home, had monitoring agents already deployed. In addition, students were also offered the possibility to install them on their lab accounts by following a quick and straightforward procedure. Those agents were implemented as wrappers for the tools they use in their programming exercises, with around 100 lines of Python code each: text editors (Emacs and Kate), compilers (GNU C Compiler and Java compiler) and the Java virtual machine, debuggers (GDB), profilers (Valgrind) and version control systems (Subversion). Each wrapper was declared as a shell alias for the command it tracks, and captured its command line arguments, standard input, output and error streams, start and end times, working directory, student identity, etc. Wrappers for short duration commands, e.g. compilers, create a single event, whereas wrappers for long duration commands, e.g. a text editor, create one event at the beginning of their execution and another one at the end. In order to prevent the transmission of the event from delaying the actual execution of the command, wrappers just store the events they create in a queue within the file system and fork a parallel process to immedi ately dispatch them. This mechanism also provides reliability, because if the transmission of an event fails, e.g. due to a temporary loss of network connectivity, it remains in the queue and its transmission is attempted later, in the next execution of the dispatching process.

The agents that monitor access to course materials were programmed in JavaScript and embedded in every HTML page of those materials. Every time students visit them, the agent is run within their browser and transmits an event to the server infrastructure. Since students need to authenticate in order to access those materials, the agent is also able to read the student's identity and embed it within the events.

4.3. Deployment of the analysis infrastructure

The implemented Lostrego application is shown in Fig. 3. The monitoring agents described in Section 4.2 publish the events they gather in the `ltg collector` stream. The first processing stage consists in annotating those events with the team annotation service described in Section 3.1. The resulting stream is processed then by the first stage of the exercise detection service presented in Section 3.2. After it, the stream passes through the working session detection service described in Section 3.3, configured with $\mu = \tau = 30\ min$. Finally, the second stage of the exercise detection service uses working session information to infer the exercise and lab session of the events that were not annotated in the first stage of the service.

4.4. Practical use of Lostrego

In order to illustrate the usefulness of the platform, this section presents some of the many potential features that can be built on top of the Lostrego infrastructure. The examples show actual data that have been gathered in the pilot deployment of the system for the case study.

Suppose it is 9 a.m. of the 4th Thursday of the course, two hours before the beginning of the 4th lab session of group `a`, one of the student groups of the course, which Olivia teaches. Olivia wants to get insight into the work of her students since the 3rd lab session, which took place one week before. During that one week period, students were expected to complete and submit the exercises of the 3rd lab (the submission deadline was the previous night: Wednesday at 11:59 p.m.) and work on a preliminary exercise for the 4th lab. The analysis application built on top of Lostrego would present a plot such as the one in Fig. 4. Olivia already knows that all teams except teams `a 02` and `a 09` worked during the 3rd lab session, but the plot lets her notice other interesting facts:

- Teams `a 02` and `a 10` worked a bit every day during the previous week. However, most teams were inactive until the weekend.
- Most teams left a lot of work for the last day before the deadline, and about half of them needed to work that night until immediately before the deadline.
- Teams `a 05`, `a 09` and `a 11` show almost no activity. Although it might just mean that they opted out of tracking, the instructor should take it as a sign they might be at risk of withdrawing the course especially if she finds they did not submit the assignment either. Talking to them in that case might help her to better understand the problem and to try to mitigate it.

Olivia wants more detailed information about the exercises on which each team has been working during the week. The platform could display a table such as Table 1, which shows that 8 teams worked

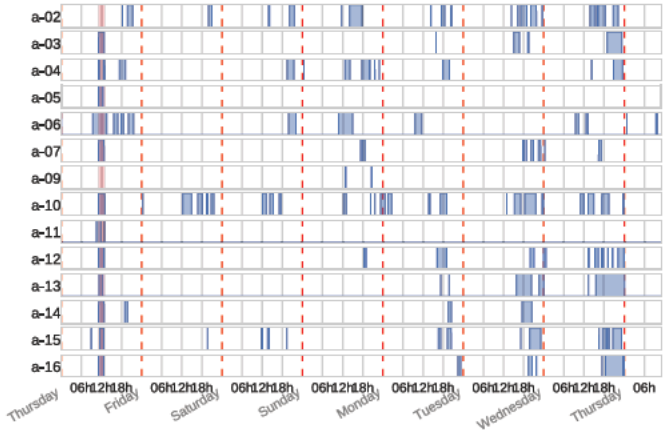

Fig. 4. Teams' working sessions for group a between the 3rd and 4th lab sessions. The in-class period of the 3rd lab session is shown in light red. Working sessions are shown in blue.

Table 1
Exercises done by each team during the 3rd week of the course and accumulated working time, as a sum of the working time of every team member.

Team	# S	Exercises			Worked time
		Previous	Lab 3	Lab 4	
a 02	21	2.3	all		17 h 50 min 56 s
a 03	8		all		8 h 22 min 38 s
a 04	14	2.5	all		13 h 6 min 17 s
a 05	2		3.1, 3.3		1 h 16 min 53 s
a 06	14	1.1, 2.3, 2.4, 2.5	all	4.1, 4.2	19 h 8 min 47 s
a 07	9	2.3, 2.4, 2.5	3.1, 3.3, 3.6	4.1	4 h 53 min 53 s
a 09	2		3.1, 3.3		10 min 12 s
a 10	29	2.3, 2.4, 2.5	all		27 h 45 min 59 s
a 11	2		3.1		1 h 18 min 57 s
a 12	11		3.1, 3.2, 3.3, 3.4		10 h 17 min 1 s
a 13	12	2.3, 2.5	all	4.2	17 h 25 min 37 s
a 14	4		3.1, 3.3, 3.4, 3.5	4.1	6 h 6 min 13 s
a 15	19	2.1, 2.3, 2.4, 2.5	all	4.1	12 h 17 min 21 s
a 16	7	2.4, 2.5	all		9 h 7 min 53 s

Table 2
Team by team individual work during the 3rd week of the course. Teams in which at least one member worked less than the others are highlighted.

Team	Member 1			Member 2			Member 3		
	# S	Time	% t	# S	Time	% t	# S	Time	% t
a-02	16	17 h 44 min 41 s	99.4%	5	6 min 15 s	0.6%	-		
a-03	2	5 h 35 min 28 s	66.7%	6	2 h 47 min 10 s	33.3%	-		
a-04	9	8 h 56 min 49 s	68.3%	5	4 h 9 min 28 s	31.7%	-		
a-05	1	1 h 16 min 53 s	100.0%	1	0 s	0%	-		
a-06	10	17 h 19 min 32 s	90.5%	4	1 h 49 min 15 s	9.5%	-		
a-07	4	3 h 32 min 58 s	72.5%	5	1 h 20 min 55 s	27.5%	-		
a-09	2	10 min 12 s	100.0%	No tracked sessions			-		
a-10	15	14 h 33 min 49 s	52.4%	14	13 h 12 min 10 s	47.6%	-		
a-11	2	1 h 18 min 57 s	100.0%	No tracked sessions			-		
a-12	8	5 h 44 min 24 s	55.8%	3	4 h 32 min 37 s	44.2%	-		
a-13	6	13 h 37 min 9 s	78.1%	6	3 h 48 min 28 s	21.9%	-		
a-14	4	6 h 6 min 13 s	100.0%	No tracked sessions			-		
a-15	9	7 h 59 min 52 s	65.1%	7	4 h 16 min 29 s	34.8%	3	1 min	0.1%
a-16	5	8 h 44 min 47 s	95.8%	2	23 min 6 s	4.2%	-		

on all the exercises of the 3rd laboratory session, but only 5 worked on the preliminary exercises for the 4th laboratory. In also confirms the problem detected with teams `a 05`, `a 09` and `a 11`.

Since worked time within a team is sometimes too unbalanced, Olivia would also want to detect teams in which some members are not working enough and might be acting as *lingerers*. Table 2 shows the number of sessions and total tracked time spent by each team member. It shows that work in some teams has been too unbalanced. Note, however, that a possible interpretation in that case is that team members worked together from the same computer or that one of them has opted out of tracking. Olivia should be careful to further investigate the issue with those teams before reaching any conclusion.

Olivia is now in the middle of the 4th lab session. It is 12:30 and she wants to review what the teams have been doing in the last 30 min. Fig. 5 shows the events per team from 12:00 to 12:30. The different exercises of the session are marked by colors. She can see that, although students were expected to complete exercise 1, i.e., the preliminary exercise, at home, the plot shows that six teams were working on it during this period of the session. There were even two teams working on exercises from a previous lab session. On the other hand, team `a 07` was already working on the last exercise. The instructor sees that teams `a 05` and `a 14` had almost no activity during those 30 min.

Some time after the session, Olivia asks the system for the activity registered during the whole session (Fig. 6.) She sees that only a few teams reached exercises 3, 4 and 5. In addition, from the three at risk teams the instructor had identified before the class only team `a 05` attended the session, but they stopped working too early. Moreover, it seems that team `a 10` had trouble to finish exercise 1, as they needed

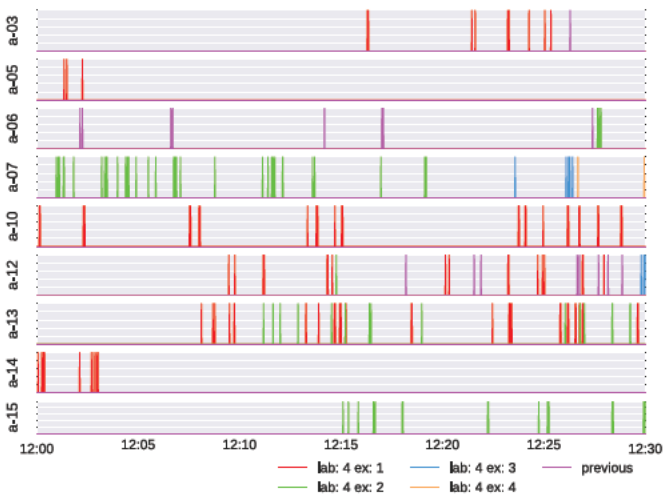

Fig. 5. Group a: events by exercise at 12:30 (4th lab session) during the last 30 min. Each line represents an event.

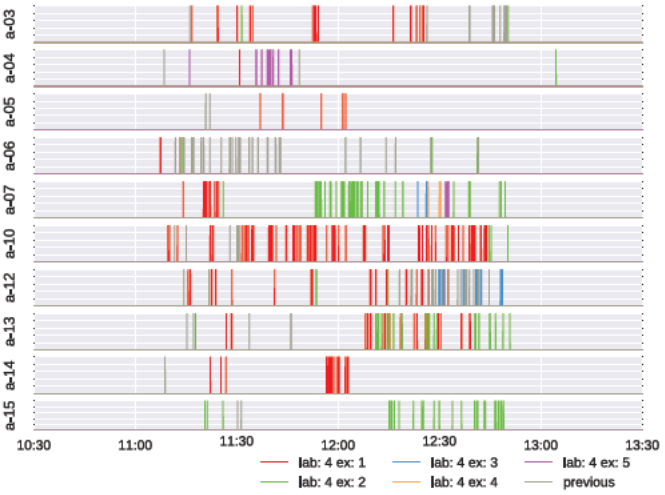

Fig. 6. Group a: events by exercise from 30 min before the 4th lab session until 30 min after. Each line represents an event.

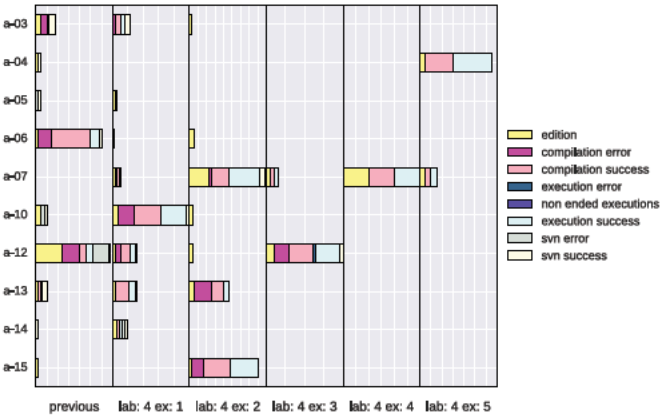

Fig. 7. Group a: number of commands by exercise during the whole 4th lab session. Colors represent the type of command.

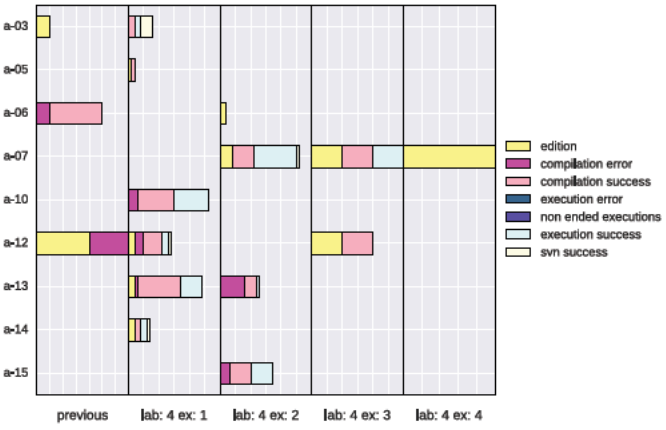

Fig. 8. Group a: number of commands by exercise performed by each team from 12:00 to 12:30.

too much time and generated too many events related to that exercise. That is consistent with the fact that they also needed to work too much time during the week before, as Table 1 shows.

Olivia wants now to get more information about the events gathered during the 4th lab session. However, the previous plots only showed when events happen and which exercise they are related to. Figs. 7 and 8 show her how many events of each type each team generated, grouped by exercise. The event types shown are: launching a text

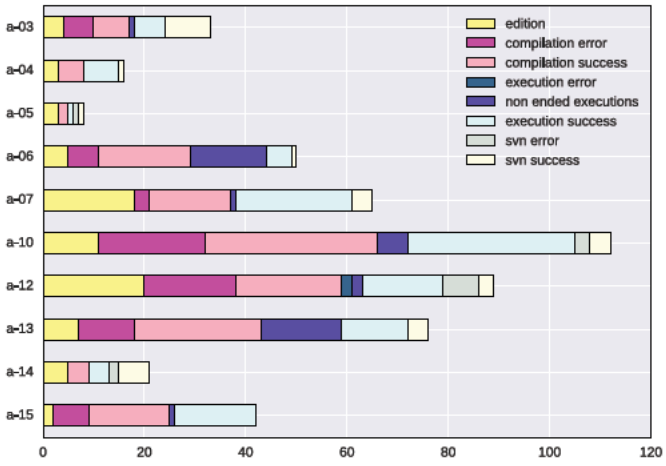

Fig. 9. Group a: number of commands performed by each team during the 4th lab session.

editor, compiling, running their code and running the source control system. It also separates successful actions from erroneous ones (e.g., they run the compiler but it detects errors in their source code.)

Finally, Fig. 9 shows the total number of events during the session, without grouping them by exercise. Olivia can identify in this plot teams such as team `a 10` that are prone to running too many commands, many of them resulting in errors. This, combined with the fact that according to Fig. 7 all those events are related to one single exercise, probably indicates that the team had serious problems with the exercise and followed a trial and error approach. Olivia takes note to follow this team closer. On the other hand, team `a 07` committed much fewer errors despite having worked on all the exercises of the session. Olivia sees that they are on the right track.

5. Results

This section presents the main results obtained from the application of the platform to two editions of the course presented in the case study.

5.1. Recorded Events

The platform captured a total of 384, 702 events during its deployment in the Fall of 2015 and 2016. The events were processed in real time as soon as they reached the analysis infrastructure. The team annotation service accepted 375, 927 events (97.72% of the total) as being part of the course. Fig. 10 shows their distribution day by day,

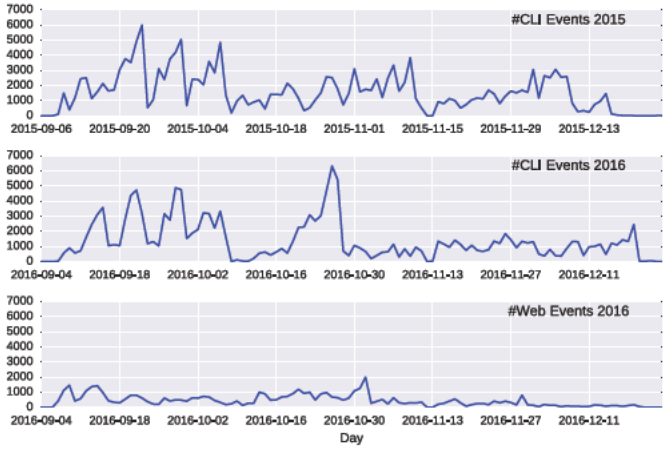

Fig. 10. Gathered course events per day (`ltg collector` data stream).

Table 3
Course events by event type per edition (Fall 2015 and Fall 2016).

Type	Event Type	Number	
		Ed. 2015	Ed. 2016
Edition	emacs/start	14, 985	12, 729
	emacs/end	12, 386	10, 579
	kate/start	1, 281	3, 526
	kate/end	943	2, 986
	Total	29, 595	29, 820
Compilation	javac	24, 187	19, 187
	gcc	35, 942	41, 740
	Total	60, 129	60, 927
Execution	java/start	18, 400	15, 415
	java/end	13, 108	11, 448
	Total	31, 508	26, 863
Debugging	gdb/start	2, 685	2, 587
	gdb/end	2, 445	2, 253
	valgrind/start	13, 643	5, 735
	valgrind/end	12, 556	4, 895
	Total	31, 329	15, 470
Version Control	svn	21, 924	19, 612
Web	web		48, 750
Total		174, 485	201, 442

separating CLI (Command line Interface) events from the virtual machines and lab accounts of the students and Web events, as the Lostrego web event collector was only available for the 2016 edition.

The first stage of the exercise annotation service successfully matched exercise and lab session for 279, 370 events (74.31% of the total). The working session annotation service identified a total of 15, 504 working sessions. Only 8, 541 events (2.27%) were left without a working session, the reason being they reached the publish subscribe infrastructure with an excessive delay. Finally, the second stage of the exercise annotation service successfully assigned the exercise and lab session to 44, 610 more events. Thus, the two stages of this service combined were able to annotate in real time 323, 980 events out of 375, 927 (86.18%).

The instructors observed that most students choose to bring their own device to the classroom (with a virtual machine installed on it). Thus, from the 327, 177 `ltg final` CLI events, 308, 106 (from 233 students) were gathered from the virtual machines of the students and 19, 071 (from 37 students) from the laboratory computers. It is important to notice that all students used the virtual machine at home, except one that withdrew the course at the beginning. The CLI events came from 371 different sources, where each copy of the virtual machine and each lab account are considered a separate data source. All Fall 2016 students (125 students) generated web events. The number of events by event type is shown in Table 3.

5.2. Relationship with individual academic performance

This section explores the potential of the processed data as a predictor of a students' individual academic achievement. A series of indicators were computed for each student from the gathered data: the count of invocations to editors, compilers, the GDB debugger, the Valgrind profiler and the Subversion version control system, the number of visits to course web pages, the ratio of successful compila tions (those that finished without errors), the count of working sessions and the total working time.

In order to understand the presence of a linear relation between each one of those indicators and the final marks of the students, the Pearson correlation between them was computed. Table 4 reports, separately for the Fall 2015 and Fall 2016 editions of the course, the correlation coefficient (r), the coefficient of determination (r^2) and the significance of the correlation (p). A statistically significant and moderate to strong correlation is appreciated in both editions for the number of invocations to the compiler, profiler and version control system, number of working sessions and total working time, being it stronger for the number of working sessions.

Table 4
Correlation analysis between processed events and final marks per edition (Fall 2015, $N = 119$, and Fall 2016, $N = 126$).

	Ed. 2015			Ed. 2016		
Variable	r	r^2	p	r	r^2	p
`Editors`	0.189	0.0357	0.04	0.35	0.122	<0.0001
`Compilation`	0.501	0.251	<0.0001	0.588	0.346	<0.0001
`Ratio successful compilations`	0.304	0.093	0.0007	0.178	0.032	0.04
`Debugging`	0.277	0.077	0.002	0.368	0.135	<0.0001
`Profiling`	0.523	0.273	<0.0001	0.525	0.276	<0.0001
`Subversion`	0.671	0.451	<0.0001	0.562	0.316	<0.0001
`Web pages`				0.342	0.117	<0.0001
`Working sessions`	0.724	0.524	<0.0001	0.716	0.512	<0.0001
`Working time`	0.659	0.431	<0.0001	0.616	0.38	<0.0001

Table 5
Fall 2015 multiple regression analysis summary (N = 119).

Variable	Estimate	Standard Error	β	t value	$Pr(>\|t\|)$
(Intercept)	0.810541	0.325043		2.494	0.0141
Profiling	0.002748	0.000908	0.21	3.026	0.0031
Debugging	0.005374	0.003488	0.10	1.541	0.1261
Editors	0.002772	0.000967	0.20	2.866	0.0049
Working sessions	0.060035	0.006592	0.75	9.107	3.33e 15

$r^2 = 0.5908$. $p < 2.2e - 16$

For the Fall 2015 dataset the linear model produced by the four indicators is shown in Table 5. As expected from Pearson correlations, the number of working sessions has the biggest weight in the linear model, with a standardized coefficient (β) of 0.75, followed by the number of invocations to the profiler and editors. This linear model explains 59.08% (r^2) of the variance of the final marks.

For the Fall 2016 dataset the linear model produced also four indicators, shown in Table 6. The number of working sessions has still a strong weight in the model, and the access to the course web pages, which are only available in the 2016 dataset, replace the number of invocations to editors. This model explains 58.87% of the variance of the final marks.

It is interesting to notice that the number of working sessions and profiling have positive coefficients in both linear models. However, the number of invocations to debuggers, editors and the number of web pages appear with a negative coefficient, suggesting that, between the students that present more working sessions and use more the profiler, those who less frequently need to resort to visiting the course web pages, invoking editors or invoking the debugger achieve greater marks.

Table 6
Fall 2016 multiple regression analysis summary (N = 126).

Variable	Estimate	Standard Error	β	t value	$Pr(>\|t\|)$
(Intercept)	0.502594	0.372705		1.349	0.1800
Profiling	0.005169	0.002102	0.20	2.459	0.0153
Debugging	0.009855	0.004438	0.18	2.222	0.0281
Web pages	0.003071	0.000980	0.27	3.135	0.0022
Working sessions	0.062368	0.007332	0.89	8.506	5.63e 14

$r^2 = 0.5887$. $p < 2.2e - 16$

6. Comparison with other approaches

In order to compare Lostrego with previous approaches, the deployment of the case study presented in Section 4 during the Fall 2015 edition of the course included the PLA monitoring tools of Romero Zaldivar et al. (2012) as well. PLA relays on a version control system for gathering the data. Every action tracked by the system in the student's environment is stored into a hidden directory within the student's local copy of her Subversion course repository. Then, when the student commits code, e.g. her submission for a given exercise, the data about those actions gets automatically sent to the server as part of the commit.

By deploying both systems together, we were able to compare PLA and Lostrego from both a qualitative and quantitative point of view. From a qualitative point of view, their main differences are:

1. Assuming network connectivity between data sources and the publish subscribe infrastructure, events in Lostrego reach the ana lysis infrastructure in a few seconds at most normally, since they are sent as soon as the student's action is detected. On the other hand, data in PLA is not sent to the infrastructure until the next code commit of the student.
2. As a consequence of the point above, events in Lostrego can be analyzed, filtered, aggregated and displayed on the fly as new events reach the infrastructure. For example, a live analysis can be performed at any moment during a laboratory session as shown in Section 4.4. Applications can choose whether to store any data at all. On the contrary, the potentially high delays of PLA prevent performing on the fly analyses, i.e. data can only be analyzed later in batch mode. In addition, data is necessarily stored, at least inside the Subversion repositories where they are received.
3. Since the only prerequisite for data sources in Lostrego is being able to send HTTP POST requests, data sources can easily be integrated in any environment or learning management system. On the contrary, PLA can only track events happening inside the student's virtual machine because of relying on a version control system. Using it in other environments would be far from trivial.
4. The high flexibility of the underlying publish subscribe system makes it possible to easily replicate any component of the Lostrego infrastructure, making it more fault tolerant and scalable. On the other hand, PLA suffers from a single point of failure in the Subversion server.

The quantitative analysis focuses on comparing two indicators: events successfully received at the server infrastructure of each system and their delivery delay, i.e. the time passed since the event was detected and it reached the server. Since PLA only works on the course virtual machines, events coming from other sources (the laboratory computers) were not taken into account for this comparison.

Table 7 summarizes the most relevant data regarding those two indicators. Both approaches were able to collect a high number of events, with a median of around 1000 events per student. They also

Table 7
Comparison between PLA and Lostrego approaches (only events from the virtual appliances are taken into account).

		Approach	
		PLA	Lostrego
Students enrolled in the course		118	
Students participating in the activities		115	
Students with recorded events		111	113
# of recorded events		157, 293	167, 536
# of recorded events per Student	Median	965.0	1, 005.5
Delay	Median (s)	6, 364.00	1.00

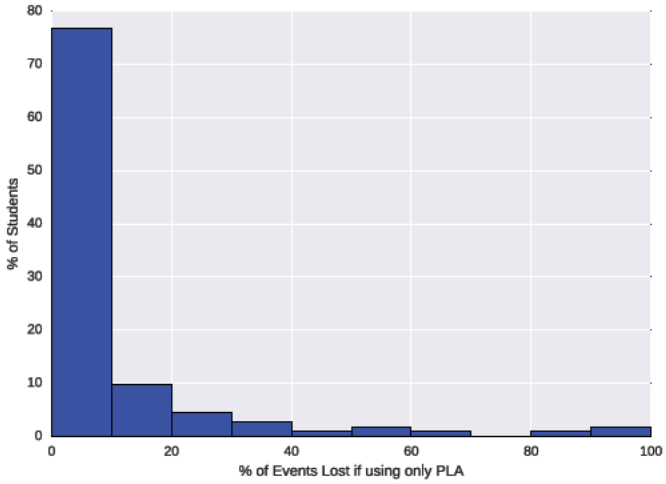

Fig. 11. Histogram of the percentage of events lost per student if PLA were the only monitoring mechanism.

gathered data from most of the students that participated in the course activities. This means that most of them did not opt out of the tracking features installed in the virtual machines. Every single event gathered by PLA was also gathered by Lostrego. However, 6.11% of the events gathered by Lostrego from the course virtual machines (10, 243 events) were not gathered by PLA, the reason being that those events were stored in the local repository within the virtual machine, but the student did no further Subversion commit after. Fig. 11 shows that for most students (76.79%) PLA lost less than the 10% of their generated events. The median percentage of events lost per student was 0%, with a mean value of 8.80% and a standard deviation of 18.86%. It can be concluded that PLA gathered less events, but loss rates seem to be acceptable.

Regarding event delivery delay, there was a significant difference between the two approaches. Once an event is detected and needs to be sent to the server infrastructure, the most important sources of delay in the two systems are:

- In Lostrego: loss of network connectivity between the virtual machine and the publish subscribe server when an event is created, due normally to the virtual machine not having Internet access or a failure at the server side. In that case, events are stored by the tracker, which tries to send them again when the next event happens. For example, the server was actually down during the case study for a whole weekend due to works on the electrical system of the building. During the following days the delayed events reached the server gradually as students started new working sessions on their virtual machines.
- In PLA: amount of time until students do the next Subversion commit and loss of network connectivity between the virtual machine and the Subversion server when attempting to do the commit operation, due normally to the virtual machine not having Internet access or a failure at the server side. Events are stored by the tracker until a commit operation initiated by the student succeeds.

The box plot of Fig. 12 shows that Lostrego clearly outperforms PLA in this indicator. Events needed just a median of 1 s to reach the Lostrego publish subscribe server since the instant they were created within the virtual machine. However, they needed a median of 106 min to reach the server side of the Subversion repositories in PLA. A more detailed analysis based on the cumulative distribution functions presented Fig. 13 shows that:

- A total of 67.61% of the events reached the Lostrego server in at most 1 s, versus more than 15 h in PLA.

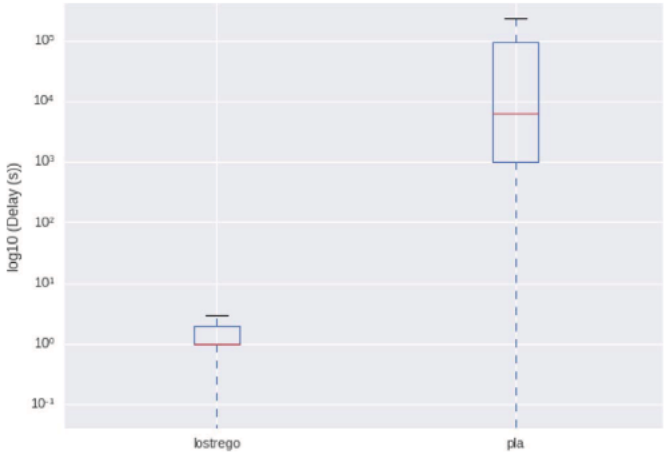

Fig. 12. Comparison between the delay of Lostrego and PLA (y-axis in logarithmic scale) for those events gathered through the two approaches.

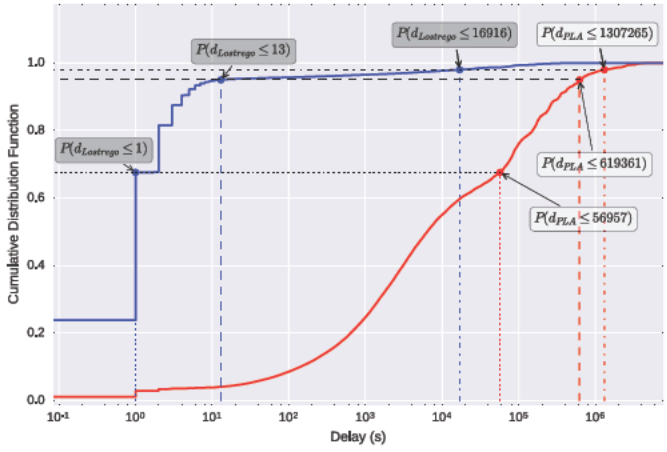

Fig. 13. Cumulative Distribution Function of the Lostrego and PLA delay (x-axis in logarithmic scale).

- A total of 95% of the events reached the Lostrego server in at most 13 *s*. versus more than 7 days in PLA.
- A total of 98% of events reached the Lostrego server in less than 4.7 *h*, versus more than 15 days in PLA.

7. Discussion

This section discusses, from the experience and results obtained in the application of Lostrego to the case study, the main advantages and limitations of our proposal. We identify the following main advantages:

- Ability to gather and analyze the data in real time: As showed in Section 5, the vast majority of the events were ready to be processed at the analysis infrastructure within a few seconds after they happened. This delay, much smaller than with other previous approaches, allows the system to produce, for example, visualizations that present instructors with an overview of the progress of their class at a certain point in the middle of a laboratory session, such as those shown in Figs. 5 and 8, or to raise an alarm when a student gets stuck in an exercise for too much time, so that instructors or connected automatic tutoring systems can give her a hint on how to progress.
- Ability to process events on the fly: Lostrego applications are stream based, so, inherently, events can be processed, filtered and analyzed on the fly without the need of being stored within a database or repository. This allows to deploy responsive applications that react almost immediately to a change on the data. Other approaches that claim to be real time are based on polling a repository (Kitto et al., 2016) or their data come from a real time environment although the analysis framework is based on polling repositories (Dodero et al., 2017).
- Extensibility: Because communications are conducted through the HTTP protocol, which means that every event sent through HTTP is susceptible of being injected into Lostrego infrastructure, new data sources, analysis modules and end user applications for instructors and students can easily be plugged into the system, according to the needs of a course or educational institution, instead of being tied to a specific LMS or workspace as other previous approaches (Gómez Aguilar et al., 2015; Petkovic et al., 2014; Romero Zaldivar et al., 2012).
- Flexibility and scalability of the platform: The platform underlying Lostrego is flexible in the sense that many alternative network layouts are possible according to the needs of every course or institution. In addition, the gathering of the data as well as their processing can be distributed across several nodes in high data volume scenarios. Ztreamy has been proved to cope with the gathering of more than 25,000 events per minute from a single server setup in a smart city setting (Fisteus et al., 2016).
- Data interoperability: As Lostrego is agnostic regarding the internal event body data structure, it naturally allows the coexistence of applications that use different internal event models. Furthermore, if the application to be developed on top of the Lostrego infra structure deals only with data streams that internally use the same data specification (for example, xAPI (Experience, 2014)), the development of data adapters is not needed.

In addition, the results presented in Section 5.2 show that some parameters derived from the data gathered in our case study, and especially those due to the working sessions detection service, present a moderate to strong correlation with the final marks of the students, and that those correlation coefficients are fairly consistent between the two editions of the course. A combination of several of those indicators can also be used as predictors of final marks. This fact contributes to the evidence that statistical models derived from previous editions of a course could potentially be used to predict the performance of students and help the instructors to early identify students that need help. This and other ways to exploit the gathered data will be explored in our future work.

However, our proposal also presents some limitations that could prevent its use in some courses:

- Privacy: The system needs to do an extensive tracking of the students' activity while they work on the activities of the course. In order to protect their privacy, they should be able to opt out of the tracking (Pardo and Siemens, 2014), but if too many students did so, the system would become useless. The problem may be alleviated by: (1) limiting the tracking to the working environment of the course (e.g. in the case study the course specific virtual machines and the laboratory account) and avoiding tracking from devices the students use for personal matters and activities not related to the course; (2) limiting the tracking to just the most relevant actions (e.g. avoid tracking every program they execute, their browsing history outside course materials, etc.); (3) use the gathered data to provide useful and personalized feedback to students, which should increase their willingness to be tracked. In the deployment of our case study we applied measures (1) and (2) and results show that in general our students accepted the use of tracking tools.
- Need to instrument learning environments with monitoring agents: The usefulness of our proposal depends on the ability to track the students' actions as they work on course activities. In this area, we identify several potential limitations. First, learning environments are heterogeneous and therefore ad hoc monitoring agents need to

be developed for them. Our experience with the case study is, however, that those agents were not costly to develop, being around 100 lines each and easily reusable for tracking other commands or the web materials of other courses. The use of HTTP and JSON, which are conveniently supported in most application platforms and programming languages, also alleviates the problem. Second, some environments such as institution wide LMSs and MOOC platforms require institutional support in order to install the agents. Where that is not possible, developing the agent as a browser extension, and relying on the students to install it in their browsers and give it permission to track their interaction with the LMS or MOOC platform might be a solution. As explained before, bundling some value added services for students in those extensions might make them more willing to collaborate. Third, while our proposal is quite appropriate for courses with frequent interaction with computer tools such as learning management systems, computer based laboratories (e.g. simulators, design tools, programming tools, mathematics and statistical tools, educational games, etc.) and MOOC platforms, it will not be useful in courses in which the majority of the activities do not happen within a computerized environment.

- Need of qualified technical support and IT infrastructure: The system requires IT infrastructure to run its server side, and technical support to deploying and maintaining it, as well as programming applications tailored to the needs of every course. We believe that, if the initiative raises enough interest, some companies in the area of educational technologies could be willing to provide it as a cloud service, and some universities and other educational institutions might provide it as another service to their instructors the same way they currently provide other services such as learning management systems like Moodle. Their deployment of the system might include a portfolio of useful analysis services and applications covering the needs of most courses.

8. Conclusions and future work

This paper presented the Lostrego infrastructure, which allows the automatic and real time collection and analysis of events from heterogeneous learning environments. Its main difference with respect to other approaches in the state of the art is its real time and scalable delivery and processing of the gathered data, which opens the door to the development of innovative analysis tools in which immediacy is a requirement. As a proof of concept, the Lostrego infrastructure was validated in a second year computer programming course. This case study shows the usefulness of those kinds of analysis tools and the suitability of the proposed infrastructure to gather and analyze learning events in a timely fashion.

The infrastructure has been designed to be generic, modular and flexible. New monitoring agents and analysis modules can be developed and plugged into it. A set of core reusable modules have already been integrated into the infrastructure, such as the working session detection service and the team annotation service. However, other analysis modules can be developed and plugged according to the needs of each educational institution or specific course.

Once the Lostrego infrastructure has been developed, deployed and tested, several research lines arise:

- Other STEM courses from the same institution are using Lostrego to gather and analyze learning data from students. We plan to analyze the completeness of the gathered data in these settings.
- Different tools are being developed within the Lostrego ecosystem. Some are general and suitable for different courses, such as a visual learning analytics dashboard, while others are tailored to the case study presented in this paper, such as an error annotation system that identifies compilation errors (from the output of the gcc and javac compilers) and memory errors (from the output of Valgrind) of a given event. We will provide these tools to different teachers and we plan to measure if the tools effectively increment teacher awareness during lectures.
- Since the analysis in Section 5.2 suggests that the gathered data could be used as a predictor of a student's performance, we plan to explore other statistical learning techniques in order to integrate such predictors into the platform.
- We are also working on the identification of the patterns followed by students when learning using process mining (Bannert et al., 2014).
- We plan to analyze the gathered data as a time series, as proposed by González Nespereira et al. (2015), in order to try to predict students' results.
- Finally, we plan to develop a visual web application composer, suitable for teachers with less knowledge in computer technologies, where they can select and ensemble the Lostrego services to be used in an application (including the visualizations from the student data).

Acknowledgments

This work was partially funded by: the Spanish Competitiveness and Economy Ministry through projects "*RESET UC3M: Reformulando Ecosistemas Escalables Educativos*" (TIN 2014 53199 C3 1 R) and "*Hermes Smartdriver. Conducción eficiente y procesamiento semántico de la información*" (TIN2013 46801 C4 2 R); and by the Community of Madrid through its regional project "*eMadrid*" (S2013/ICE 2715).

References

Andrews, J., Higson, H., 2008. Graduate employability, 'soft skills' versus 'hard business' knowledge: A European study. High. Educ. Eur. 33, 411 422.

Babu, S., Widom, J., 2001. Continuous queries over data streams. ACM Sigmod Rec. 30, 109 120.

Bannert, M., Reimann, P., Sonnenberg, C., 2014. Process mining techniques for analysing patterns and strategies in students' self-regulated learning. Metacognition Learn. 9, 161 185.

Boletín Oficial del Estado, 1999. Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M.J., Hellerstein, J.M., Hong, W., Krishnamurthy, S., Madden, S.R., Reiss, F., Shah, M.A., 2003. Telegraphcq: continuous dataflow processing, In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of data, ACM. pp. 668 668.

Del Blanco, Á., Serrano, Á., Freire, M., Martínez-Ortiz, I., Fernández-Manjón, B., 2013. E-learning standards and learning analytics. can data collection be improved by using standard data models? In: Global Engineering Education Conference (EDUCON), 2013 IEEE, IEEE. pp. 1255 1261.

Dodero, J.M., González-Conejero, E.J., Gutiérrez-Herrera, G., Peinado, S., Tocino, J.T., Ruiz-Rube, I., 2017. Trade-off between interoperability and data collection performance when designing an architecture for learning analytics. Future Gener. Comput. Syst. 68, 31 37, (URL ⟨doi:10.1016/j.future.2016.06.040⟩.).

Estévez-Ayres, I., Alario-Hoyos, C., Pérez-Sanagustín, M., Pardo, A., Crespo-García, R.M., Leony, D., Parada, G., H.A., Delgado-Kloos, C., 2015. A methodology for improving active learning engineering courses with a large number of students and teachers through feedback gathering and iterative refinement. Int. J. Technol. Des. Educ. 25, 387 408. http://dx.doi.org/10.1007/s10798-014-9288-6.

Experience API Working Group, 2014. Experience API, version 1.0.2. Advanced Distributed Learning (ADL).URL ⟨https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md⟩.

Ferguson, R., Brasher, A., Clow, D., Cooper, A.and Hillaire, G., Mittelmeier, J., Rienties, B., Ullmann, T., Vuorikari, R., 2016. Research Evidence on the Use of Learning Analytics - Implications for Education Policy. EUR - Scientific and Technical Research Reports. Joint Research Centre Science for Policy Report. http://dx.doi.org/10.2791/955210. eUR 28294 EN.

Fisteus, J.A., Fernandez, L.S., Magana, V.C., Organero, M.M., Fernandez, J.Y., Alvarez-Garcia, J.A., 2016. A scalable data streaming infrastructure for smart cities, In: Falomir, Z., Ortega, J.A., (Eds.), In: Proceedings of the XVIII Workshop on Qualitative Systems and Applications in Diagnosis, Robotics and Ambient Intelligence (JARCA 2016), Aachen. pp. 7 13. URL ⟨http://ceur-ws.org/vol.1812/JARCA16-paper-2.pdf⟩.

Fisteus, J.A., Garcia, N.F., Fernandez, L.S., Fuentes-Lorenzo, D., 2014. Ztreamy: a middleware for publishing semantic streams on the web. Web Semant.: Sci., Serv. Agents World Wide Web 25, 16 23. http://dx.doi.org/10.1016/j.websem.2013.11.002.

Fox, A., 2013. From MOOCs to SPOCs. Commun. ACM 56, 38 40.

Ghate, P.V., Pati, H.K., 2016. Collaborative distributed communication in heterogeneous environments: a comprehensive survey. J. Netw. Comput. Appl. 61, 1 20.

Golab, L., Özsu, M.T., 2003. Issues in data stream management. ACM Sigmod Rec. 32, 5 14.

Gómez-Aguilar, D.A., Hernández-García, Á., García-Peñalvo, F.J., Therón, R., 2015. Tap into visual analysis of customization of grouping of activities in e-learning. Comput. Human. Behav. 47, 60 67. http://dx.doi.org/10.1016/j.chb.2014.11.001.

González Nespereira, C., Fernández Vilas, A., Díaz Redondo, R.P., 2015. Am I failing this course?: risk prediction using e-learning data, In: Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality, ACM. pp. 271 276.

Haag, V., Millar, M., Nayak, P., Vento, C., Whyte, A., Sinha, V.,2015. Caliper Analytics v1 Final Specication. IMS Global Learning Consortium. URL: ⟨https://www.imsglobal.org/activity//caliperram⟩.).

Hwang, G.J., 2014. Definition, framework and research issues of smart learning environments-a context-aware ubiquitous learning perspective. Smart Learn. Environ. 1, 4.

Jadud, M.C., 2006. Methods and tools for exploring novice compilation behaviour, In: Proceedings of the Second International Workshop on Computing Education Research, ACM. pp. 73 84.

Kim, M., Karenos, K., Ye, F., Reason, J., Lei, H., Shagin, K., 2010. Efficacy of techniques for responsiveness in a wide-area publish/subscribe system, In: Proceedings of the 11th International Middleware Conference Industrial Track, ACM. pp. 40 45.

Kinshuk, Chen, N.S., Cheng, I.L., Chew, S.W., 2016. Evolution is not enough: revolutionizing current learning environments to smart learning environments. Int. J. Artif. Intell. Educ. 26, 561 581.

Kitto, K., Bakharia, A., Lupton, M., Mallet, D., Banks, J., Bruza, P., Pardo, A., Shum, S.B., Dawson, S., Gašević, D., et al., 2016. The connected learning analytics toolkit. In: Proceedings of the Sixth International Conference on Learning Analytics & Knowledge, ACM. pp. 548 549.

Kovanovic, V., Gašević, D., Dawson, S., Joksimovic, S., Baker, R., 2016. Does time-on-task estimation matter? Implications on validity of learning analytics findings. J. Learn. Anal. 2, 81 110.

Lewkow, N., Feild, J., Zimmerman, N., Riedesel, M., Essa, A., Boulanger, D., Seanosky, J., Kumar, V., Kinshuk, Kode, S., 2016. A scalable learning analytics platform for automated writing feedback. In: Proceedings of the Third (2016) ACM Conference on Learning @ Scale, ACM, New York, NY, USA. pp. 109 112. URL doi:10.1145/2876034.2893380.

Pardo, A., Kloos, C.D., 2011. Subcollaboration: large-scale group management in collaborative learning. Softw.: Pract. Exp. 41, 449 465. http://dx.doi.org/10.1002/spe.1023.

Pardo, A., Siemens, G., 2014. Ethical and privacy principles for learning analytics. Br. J. Educ. Technol. 45, 438 450.

Pérez-Sanagustín, M., Ramirez-Gonzalez, G., Hernández-Leo, D., Muñoz-Organero, M., Santos, P., Blat, J., Kloos, C.D., 2012. Discovering the campus together: a mobile and computer-based learning experience. J. Netw. Comput. Appl. 35, 176 188.

Petkovic, D., Sosnick-Pérez, M., Huang, S., Todtenhoefer, R., Okada, K., Arora, S., Sreenivasen, R., Flores, L., Dubey, S., 2014. Setap: Software engineering teamwork assessment and prediction using machine learning. In: Proceedings of Frontiers in Education Conference (FIE), IEEE, pp. 1 8.

Petkovic, D., Sosnick-Perez, M., Okada, K., Todtenhoefer, R., Huang, S., Miglani, N., Vigil, A., 2016. Using the random forest classifier to assess and predict student learning of software engineering teamwork. In: Proceedings of Frontiers in Education Conference (FIE), IEEE, pp. 1 7.

Rodrigo, M.M.T., Tabanao, E., Lahoz, M.B.E., Jadud, M.C., 2009. Analyzing online protocols to characterize novice java programmers. Philipp. J. Sci. 138, 177 190.

Romero, C., Ventura, S., 2013. Data mining in education. Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 3, 12 27.

Romero-Zaldivar, V.A., Pardo, A., Burgos, D., Kloos, C.D., 2012. Monitoring student progress using virtual appliances: a case study. Comput. Educ. 58, 1058 1067. http://dx.doi.org/10.1016/j.compedu.2011.12.003.

Sahami, M., Danyluk, A., Fincher, S., Fisher, K., Grossman, D., Hawthorne, E., Katz, R., LeBlanc, R., Reed, D., Roach, S., et al., 2013. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. Technical Report. The Association for Computing Machinery (ACM) and IEEE Computer Society Joint Task Force on Computing Curricula. Available at ⟨https://www.acm.org/education/CS2013-final-report.pdf⟩.

Siemens, G., Gasevic, D., 2012. Guest editorial-learning and knowledge analytics. Educ. Technol. Soc. 15, 1 2.

Suppes, P., 1968. Computer technology and the future of education. Creat. Educ. Mater..

Tatbul, N., 2010. Streaming data integration: Challenges and opportunities. In: Proceedings of the 26th International Conference on Data Engineering Workshops (ICDEW), IEEE. pp. 155 158.

Watson, C., Li, F.W., Godwin, J.L., 2013. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In: Proceedings of the 13th International Conference on Advanced Learning Technologies (ICALT), IEEE. pp. 319 323.

Willman, S., Lindén, R., Kaila, E., Rajala, T., Laakso, M.J., Salakoski, T., 2015. On study habits on an introductory course on programming. Comput. Sci. Educ. 25, 276 291.