Data collection and processing for a multimodal Learning Analytic System

Emanuele Ruffaldi, Giacomo Dabisias, Lorenzo Landolfi
Scuola Superiore Sant'Anna
Pisa,Italy
Email: n.lastname@sssup.it

Daniel Spikol
Malmo University
Place
Email: daniel.spikol@mah.se

***Abstract*—Learning Analytic (LA) systems are aimed at supporting teachers in understanding the learning process by analyzing the information and the interaction of students with computer systems. In the case of a project-based learning process there is a need of introducing measure the student' activity as acquired via multiple modalities and then processed. The acquisition and processing needs to take into account the specificities of the learning context and deployment at schools, in particular in terms of system architecture. The paper proposes an architecture for the acquisition and processing of data for project-based LA designed to be interoperable and scalable. System design, details of the solutions and brief examples of acquired data are presented.**

***Keywords*—Learning Analytics; learning; teacher support; data processing; learning modalities**

I. Introduction

The field of Learning Analytics (LA) has been described by Ferguson [1] as the convergence of different technological and research domains. LA takes advantage of the digitalization of the learning process, which is inherently subject to instrumentation and measure. LA involves solutions from the domains of business intelligence, web analytics, educational data mining and recommendation systems. Chatti et al. [2] tried to identify a reference model of the LA for structuring the various research and development efforts in the domain. In particular they identify the LA process as a cyclic process involving three major steps: data collection plus pre-processing, analytics plus action, and post-processing. Note that the post-processing step of the LA process aims at refining the whole process by selecting new data sources, new analysis methods, and new metrics. This step is necessary due to the large number of variables involved in the analysis and in the peculiarities of each learning settings.

This paper provides a description of a software/hardware framework architecture for supporting LA in project-based learning. The core idea of the system is to process the raw data, acquired in the learning environment with a variety of sensors that a server will collect, to produce learning traces which, based on the identified metrics, can be employed, by the teacher, to better understand the learning process and, by the students, to document their work. The system can be used also in a "smart" way to recognize activities performed by students during their experiments using machine learning techniques on the acquired data [3]. This will highlight important events in the experiments and could give important insights, to teachers, about the learning process using custom visualizations like in [4]. Previous work, like [5], shows the importance and affordability of such a system. The target learning context is based on Arduino electronics projects at different levels, from high-school to graduate students. The different setups will be addressed by varying sensors and measured features.

The contribution of this paper are: 1) presenting multimodal capturing activities during project-based learning, with the support for networkless-environments and 2) describing a framework for data collection and processing oriented at LA.

The paper is structured as follows: the following section presents related works; section 3 introduces the general design of the system and work flow. Section 4 deals with the collection components on the student side and section 5 deals with the server side. Section 6 presents examples of acquired data, right before conclusions.

II. Background

Social Networks Adapting Pedagogical Practice (SNAPP) tools perform real-time social network analysis and visualization of discussion forum activities within popular commercial and open source LMS. LOCO-Analyst [6] is an educational tool aimed at providing teachers with feedback on the relevant aspects of the learning process taking place in a web-based learning environment. LOCO-Analyst provides feedback on the activities performed by students, the usage of the material and social interactions. LA e-Rubric [7] is a plugin for MOODLE [8] that realizes a grading method used for criteria-based assessment. A more general LA can be found in GLASS [9] that tackles the different aspects of LA, spanning from data source management and processing to visualization.

The domain of project-lead learning requires to go beyond the traditional LA domain and investigate the statistics and pattern methods related to action and behavior understanding when interacting in a real environment with real objects. The present paper specifically contributes with a new structure for supporting project-led activities and the processing for extracting learning traces.

III. General Design

The system is composed by a single central server and a variety of remote clients, with scalability aspects discussed later in section V-F. Each client is composed by a single computing machine and a series of sensors which interact with it, corresponding to a student (single or group) desk. A sensor can be a physical interface like a camera or a piece of

software, gathering data from a remote resource, in any case related to the activity performed on the desk. The system is built in such a way that the array of acquisition sensors can be easily upgraded with new sensors depending on teachers needs. The acquired information is meant to be initially processed locally for the purpose of reducing the bandwidth requirement and for removing any privacy-related issues, mainly associated the camera systems. The data that is acquired is then sent to the remote server, unless the connection is absent, in which case it is cached locally and uploaded when the connection is available by intervention of the teacher. The remote server is in charge of storing the data, extract statistics, process new data from the raw acquisitions, extracts metrics for the LA and support visualization.

Given that the system can run at the same time a variety of client applications, each execution has to be identified by a unique value, called session id. A session identifies the execution of a client machine from a starting point to an end. All the gathered data is associated to that session along with the information of the creating user and starting and finishing time. Session duration depends on the teaching context, spanning from half an hour to few hours.

A. Workflow

The general interaction of the teacher and students with the system is composed by the following steps:

1) The system is booted and, after logging in, the Arduino interface is started along with the client system which runs in background.
2) The system requests the server to open a new session and, if authorized, a new session is created on the server. The system will display a QR code to be used with the mobile application in order to synchronize the sessions.
3) The system starts to capture data from the different sensors and stores everything locally and remotely, if connection is available.
4) Students, teachers and researchers can upload multimedia content from a mobile app on their phone.
5) Students and teachers can get feedback from the system on how experiments are progressing.
6) The system is closed. A request to close the session is sent to the remote server.
7) The teacher logs out and the system is ready for a new session.

In the aftermaths of the session the teacher can look back at the session studying the progress of the data by visualization supported by LA processing.

B. Data Management

Data management on the server is based on the concept of graphs of *data streams*. Each data stream is a time-index sequence of items of the same nature. The content of the streams can span from entities obtained from computer vision system (like hand positions) to logs of the Arduino IDE. Streams are grouped in sessions, and sessions are grouped in namespaces corresponding to the different teaching locations.

The possible computations take streams and produce new streams, with algorithms spanning from basic statistics computation to more sophisticated machine learning based algorithms. In this way the streams create a flow graph that has the input data received from the client.

The graph structure depends much on the types of analytics computed, and due to the fact that this is also a research platform, it is grounded on the principles of reproducible research. This means that at any time the system can clean up all the generated streams and restart from the input data applying a new set of operations. As happens in event-based management systems [10], computations can be applied when new data arrives, or offline as a batch process.

Finally each stream should be externally accessible for the purpose of creating Web visualizations, available both to students and teachers.

IV. Collection Service

This section discusses the features of the student-side component called Collector, which is in charge of capturing data from sensors, pre-process them and send them to the processing server. For the multimodal LA system being investigated, the Collector deals with three types of information produced by students: the interaction in the physical world, the activity on the computer used for controlling Arduino electronics, and the self-documentation activity that is performed via mobile applications.

A. Physical Worlds

The events in physical world comprise information captured from cameras (both Kinect2 and webcam), physical buttons and audio events. A computer vision module extracts different indicators of activity from the cameras, never sending out the full images. In particular, in the current prototype, the OpenCV library is used to identify the presence of faces in the scene for the purpose of understanding the number of students active at the desk. By asking students to wear a fiducial marker on the wrists, it is possible to identify the motion of the hands in the scene to measures activity. Finally object tracking is possible by using a RGB-D camera such as Kinect2 and model-based algorithms such as LineMod [11].

The pose of the faces and hands is transformed from the camera reference frame to a table reference frame (e.g. with origin in a corner of the table with Z up) by means of a calibration computed when the cameras are installed. This way the presence and activity location information can be related to the table (e.g. moving away, working on something inside the table). This approach is not found in activity recognition and is specifically added to improve the understanding of students' actions.

Additional information about the activity at the desk can be provided by physical buttons, like asking the teacher for help or giving feedback for the learning activity.

Finally audio is being analyzed only for identifying if the students are talking during the project and if there is some activity. This is obtained by performing FFT over microphone data, filtering over interesting bands, and comparing against the background.

B. Computer Activities

During projectual work with Arduino electronics, the students interact with the computer by working in the Integrated Development Enviroment (IDE) with the classic compile-run-debug cycles or with visual programming interfaces. Such integration is collected via instrumentation of the IDE and sent to the Collector software for assembling the analytics.

C. Self-Documentation

The last source of information for the analytics is self-documentation, which is the capacity of the students to document their activity. This can be provided in several ways: 1) taking a snapshot of the environment around the table, 2) taking a snapshot of the computer, screen 3) taking pictures and videos via smartphone of the students. In any case this is a voluntary action that allows the students to show the progress of their activity.

D. Implementation

The Collector is implemented via a program running on a computer that is connected with the different information sources. This component uses sessions as the unit of processing, associating all the acquired data to it, even if there is no connection to the remote server. The Collector manages a variety of sensors (physical or virtual), processes the data coming from these sensors, and produces entities associated to different streams. These entities are then sent to the server for storage and analytics, if connected, otherwise stored locally for later upload.

In addition to device-based sources, the Collector supports various network sources for accommodating different services: UDP, push HTTP requests, pull HTTP requests and WebSockets.

The nature of the processing and the connection with the server is mainly message based, and among the many message-based frameworks, this project has chosen to use one based on Web standards. Each stream item is a JSON entity, management exchanges (like session) are based on REST interfaces and data streaming occurs via WebSockets which provide, over HTTP, a message based channel. Figure 1 shows the overall structure with the data sources.

Internally the Collector uses multiple threads for addressing the sources, employing the OpenCV and Aruco [12] library for the computer vision tasks, with the help of GPU for fast face identification. In particular, the Collector is implemented in C++ and runs under Linux taking advantage of the ASIO library for asynchronous network communication. This choice is motivated by performance reasons and by the precise memory management possible with native languages.

The deployment of the Collector to the teacher locations (currently in three countries) is based on Docker containers [13]. Docker is a type of operating system virtualization on Linux that allows to isolate the application from the system without incurring in the penalties of virtualization. With proper configuration it allows the Collector to access both Kinect2 (USB3 device) and NVidia GPU directly.

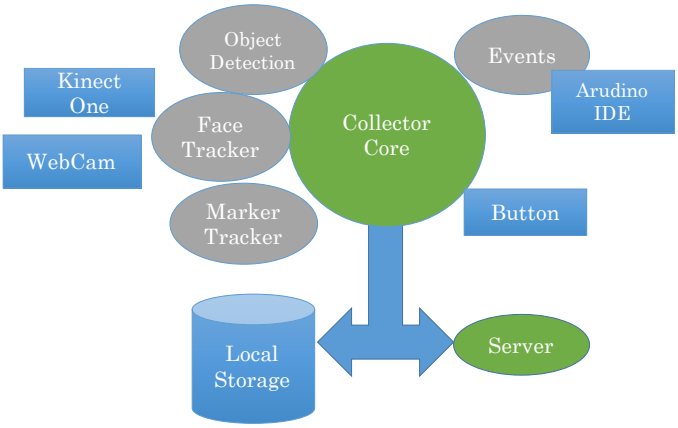

Fig. 1: Structure of the Collector with the connection to the Server. Rectangles are data sources, ellipses computing elements.

V. Analytics Server

This section presents the characteristics of the LA server as discussed in the general design section. The specific implementation relies on the Java server frameworks, mainly due to the availability of libraries, tools and the strong typing of the language. Other options, like node.js are possible while keeping the external interface.

A. Core structure

The server software infrastructure is basically divided in two main parts: the *WebSocket collector* and the *servlets*.
The WebSocket collector is a web application waiting for data to be sent in a *websocket*, which is a web transmission protocol requiring a connection between two partners. Websockets [14] have been chosen as the protocol to exchange data from the client application to the server because they allow small overhead in case data exchange is perpetrated over a long period of time.
Instead, servlets are a collection of operations directly exposed to any web user through a specific URL. Since such operations are *Request-response* operations, they are best supported by the connectionless and widespread HTTP protocol. More detailed information about servlet will be given in section V-D.
Data storage is managed via MySQL, the most popular open source relational database management system; Hibernate, which maps Java objects into MySQL entities, has been used to maintain a higher level of abstraction from the actual database, which can be easily substituted with another one. Figure **??** shows the structure of the server environment and outlines both the interaction with the client applications and with the database.
The LA web server also supports fixed and custom operations to manipulate data stored in the database, mainly streams, which are acquired during the learning sessions.
Streams are intended as a sequence of data in time, acquired through the sensors or through the mobile application. *Jobs* are executed over the streams and produce as result new streams or single value results.
The system is designed in order to execute multiple tasks

simultaneously using asynchronous requests. The structure is explained in more detail in section V-C.

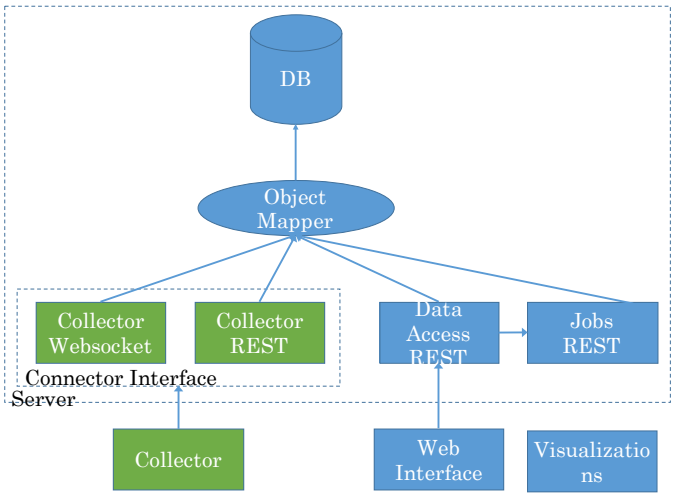

Fig. 2: Structure of the collector-server LA system with the interfaces

B. Data Management and Operations

As already anticipated in the section V-A, LA data management may be divided in the following parts:

1) **Database**: handled by MySQL.
2) **Mapping**: handled by Hibernate.
3) **Data manipulation**: handled by the collector and by the REST APIs.

Concerning point 3, it is necessary to specify that data obtained from the learning sessions are stored in the database by the Collector, while they can be read through a specific REST servlet by the client. Data not directly acquired by the sensors (e.g. users and sessions descriptions, multimedia contents) are managed with REST operations.

C. Data stream manipulation

The LA server provides the facility of transforming the raw data acquired by the sensors during the learning sessions in streams which are more suitable for data analysis.
Such data manipulation is reachable to users or web client applications through two REST endpoints, that we will call for brevity `status` and `result`. A client may submit a *job* capable of manipulating data by performing an HTTP PUT(`status`) request, which returns to the client a message containing the identification number i of the submitted job. The status of the very same job can be queried at any time performing an HTTP GET(`status`/i) request. Such status can be one of the following:

- `executing`: if job i is still in execution
- `terminated`: if job i has terminated and its result has been properly stored
- `queued`: if job i has been submitted to the system but it has not yet started its execution
- `failed`: if job i has terminated its execution with an exception.

In any case the answer to the GET(`status`/i) request will contain the date in which job i was submitted, while only in case of `terminated` status, it will contain also the execution time in milliseconds.
An HTTP GET request on endpoint `result` is needed to retrieve the results of a given job. The answer to the GET(`result`/i) request depends on the job i type: it can be a single value result (if job i outputs a single value or a *constant* number of values) or a *stream*.

For instance if job i asks to compute the average intensity value of the audio samples captured during session j, the result will be a single number, else if job i is asking all the audio samples in session j having intensity I such that $I > C$, then the result will be a stream (a subset of the collected data).
In order to get the result of each job, the relational database provides the table `Job_table` containing entities describing the status of each job and its' result in case of single value result. There is also a table containing the results of the jobs (called `Stream_table`), in case such results are streams.
Each entity of `Stream_table` contains the identifier of a data sample stored by the collector and the identifier of the job it relates to.

Any operation that can be provided by the server to manipulate the collected data is defined in a specific Java class implementing a common interface. Such a class exposes a `run()` method that must be implemented by any sub-class of it along with some basic information describing the job. The same abstract class exposes also the `store_result()` method, which describes how to store the results of the `run()` computation in the database and the `extract()` method that describe how to fetch the input of the `run()` from the database.
The actual scheduling of the requested jobs is managed internally by a *scheduler* which is not visible to the users.
The Scheduler class contains two data structures: a thread pool (of standard Java threads) and a FIFO queue of jobs which have to be executed.
The thread pool is created and initialized before the server is deployed, and it is kept always alive.
The threads belonging to the thread pool are called *workers* and the scheduler FIFO queue is shared by all of them. Each worker stays idle until a job is available in the job queue. As soon as a job is popped in reference `op`, the worker updates its state to `executing` and then calls the following line.

```
op.store_result(op.run(op.extract()))
```

If an exception is thrown during the execution of the above code, the worker sets the status of the entity corresponding to the popped job to `failed` and stores the reason of the failure in the same entity.
If the computation terminates properly, the worker simply sets the status to `terminated`; the results of the computation are correctly stored according to the implementation of the `store_result()` method.

The framework provides multiple operations to compute various statistical values; it also provides a special operation, called `Pipeline`, which allows to execute a sequence of operations one after the other avoiding to submit an HTTP PUT on `status` for each operation.

D. REST API

The LA server exposes a REST interface for supporting data visualization, external processing and interoperability. First of all we point out that any user willing to access the server's resources has to register and authenticate himself.
Registration must be done via the web server interface. The authentication is token based and each servlet accepts a request either if such a token is transmitted as a cookie or as an HTTP parameter.
We call the permission to *write* on resource r the possibility of performing PUT, DELETE and POST on endpoint `r`, whilst we call the permission to *read* on r the possibility of performing GET on endpoint `r`.
Each registered user is associated to a role (embedded and encrypted in the access token) that determines his *read* and *write* permission, stored in the server. For instance, an administrator can perform a DELETE on endpoint `session` whilst a researcher may only perform a GET on the very same endpoint.
In general each servlet endpoint responds to any HTTP request in JSON format, but it is also possible to visualize data in a more human readable form (in tables) through the web interface.

E. Higher-level processing

The data processing scheme discussed above is the starting point for the effective high-level learning analytics processing that can give insights about the progress of students in projectual work. Higher level actions and activity recognition, and identification of patterns are possible in medium-level timescale, while longer-time scales allow for user profiling based on Bayesian methods [15].

F. Scalability

Scalability is an important aspect for any data collection and processing system, in particular when aimed at being replicated across many schools and teaching sites. The natural partitioning of the namespaces across schools and classes allows for horizontal-scalability of the server systems, and correspondingly of the data flow from the collectors to the associated servers. For the computation inside each namespace, many operations are limited to local resources requiring vertical-scalability, but there are possibilities of distributed machine learning operations for analyzing the collected data [16].

VI. Acquired Data

This section presents an example of session acquired using the system of the paper and involving different types of data: hands motion, faces, logging of Arduino operations and multimedia contents. This single session has been divided in working phases by the teachers.

Type	Count
Hand	2282
Face	1444
Arduino	193
Button Press	16
Multimedia	94
Phases	27
Duration	3257s

TABLE I: Data types present in the example session with the corresponding items

Type	Count
Text	57
Image	37

TABLE II: Multimedia data content

Table I shows a data example extracted from a real test scenario. Data types represent the following information:

- Hand: represents a hand in the work space with a unique id and a 3D space position relative to a fixed point on the table.
- Face: represents a face looking toward the camera (Arduino screen) along with its 3D position in space relative to the camera.
- Arduino logger: represents all the events which happened while programming with the Arduino IDE.
- Button presses: represents the moments when a special event happened.
- Work phases: represents the beginning of a new phase of the experiment, added by the instructor.
- Multimedia content: represents text, images, videos. In general it can represent any multimedia content.
- Audio content: represents the moment when a certain audio intensity level is exceeded. During this trial audio was disable.

Multimedia content is subdivided as shown in table II. Figures 3 and 4 show how the data is distributed in time. All data is acquired with a one second time interval which can be changed according to the experimenters' need.

VII. Conclusions

The paper introduced a framework for distributed data collection and analysis aimed at creating a Learning Analytics system in the context of project based learning. The framework provides a Web compatible interface both for data collection and processing, and aims at supporting a flexible range of LA algorithms. The testing with students of different location is in progress providing insights on the scalability and flexibility of the system itself. At the end of the experimentation the system will be released as Open Source to increasing usage and enhancements.

Future actions are mainly devoted to the creation of LA algorithms starting from the wealth of the collected data, in particular with the aim of creating profiles of the students. One aspect of the analysis is the extraction and evaluation of known

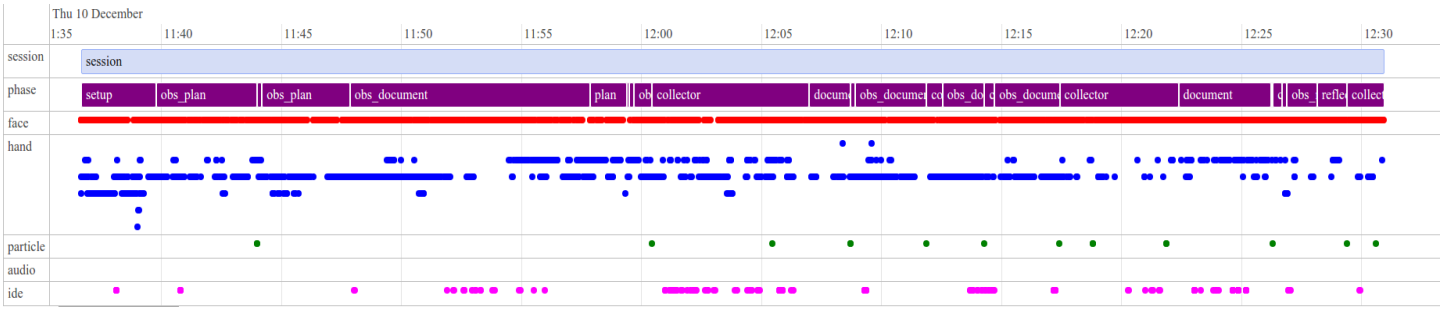

Fig. 3: The figure shows the sampled data on a timeline, gathered during a test session, grouped by item type. The top filled bar corresponds to the overall session, below that the phase decomposition is reported. Faces presence is almost continuous. Hand identification is presented with some dots, with one row for every distinct hand; the particle row corresponds to button presses. The lowest part deals with the Arduino IDE information.

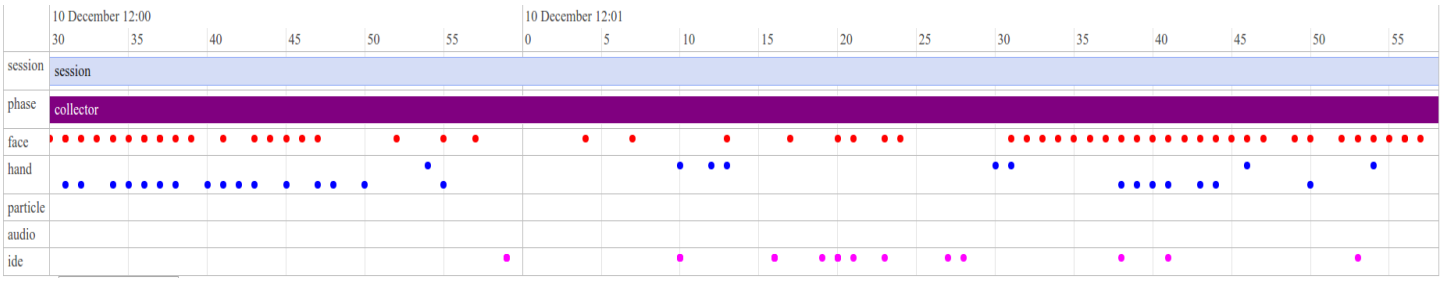

Fig. 4: Zoom of the full session depicted in figure 3.

gestures using techniques such as the ones in [17]. Another aspect is the general extraction of patterns of projectual activity that characterize each of the project phases. The difficulty is mainly in the variability of the projects and the interactions of the students at the desk.

Acknowledgments

We acknowledge the European Commission for funding the PELARS part of the FP7-ICT-2013-11 Objective ICT-2013.8.2 Technology-enhanced Learning.

References

[1] R. Ferguson, "Learning analytics: drivers, developments and challenges," *International Journal of Technology Enhanced Learning*, vol. 4, no. 5-6, pp. 304–317, 2012.

[2] M. A. Chatti, A. L. Dyckhoff, U. Schroeder, and H. Thüs, "A reference model for learning analytics," *International Journal of Technology Enhanced Learning*, vol. 4, no. 5-6, pp. 318–331, 2012.

[3] D. Gasevic, C. Rose, G. Siemens, A. Wolff, and Z. Zdrahal, "Learning analytics and machine learning," in *Proceedings of the 4th International Conference on Learning Analytics And Knowledge*. ACM, 2014, pp. 287–288.

[4] E. Duval, "Attention please!: learning analytics for visualization and recommendation," in *Proceedings of the 1st International Conference on Learning Analytics and Knowledge*. ACM, 2011, pp. 9–17.

[5] G. Siemens and P. Long, "Penetrating the fog: Analytics in learning and education." *EDUCAUSE review*, vol. 46, no. 5, p. 30, 2011.

[6] J. Jovanović, D. Gašević, C. Brooks, V. Devedžić, and M. Hatala, "Loco-analyst: A tool for raising teachers awareness in online learning environments," in *Creating New Learning Experiences on a Global Scale*. Springer, 2007, pp. 112–126.

[7] M. R. Rivasy, M. C. De La Serna, and E. Martínez-Figueira, "Electronic rubrics to assess competences in ict subjects," *European Educational Research Journal*, vol. 13, no. 5, pp. 584–594, 2014.

[8] M. Dougiamas and P. Taylor, "Moodle: Using learning communities to create an open source course management system," in *World conference on educational multimedia, hypermedia and telecommunications*, vol. 2003, no. 1, 2003, pp. 171–178.

[9] D. Leony, A. Pardo, L. de la Fuente Valentín, D. S. de Castro, and C. D. Kloos, "Glass: a learning analytics visualization tool," in *Proceedings of the 2nd international conference on learning analytics and knowledge*. ACM, 2012, pp. 162–163.

[10] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys (CSUR)*, vol. 44, no. 3, p. 15, 2012.

[11] S. Hinterstoisser *et al.*, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Computer Vision (ICCV)*. IEEE, 2011, pp. 858–865.

[12] S. Garrido-Jurado *et al.*, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320314000235

[13] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[14] A. M. Ian Fette, *The WebSocket Protocol*, RFC, 2011. [Online]. Available: https://tools.ietf.org/html/rfc6455

[15] P. García, A. Amandi, S. Schiaffino, and M. Campo, "Evaluating bayesian networks precision for detecting students learning styles," *Computers & Education*, vol. 49, no. 3, pp. 794–808, 2007.

[16] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proc. OSDI*, 2014, pp. 583–598.

[17] Portillo-Rodriguez *et al.*, "Real-time gesture recognition, evaluation and feed-forward correction of a multimodal tai-chi platform," in *HAID*, ser. Lecture Notes in Computer Science, vol. 5270. Springer, 2008, pp. 30–39.