IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 12, 2020, accepted February 18, 2020, date of publication February 20, 2020, date of current version March 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2975384

A Customizable and Incremental Processing Approach for Learning Analytics

DANIEL PÉREZ-BERENGUER[1], MATHIEU KESSLER[1], AND JESÚS GARCÍA-MOLINA[2]
[1]Centro de Producción de Contenidos Digitales, Universidad Politécnica de Cartagena, 30202 Murcia, Spain
[2]Departamento de Informática y Sistemas, Universidad de Murcia, 30003 Murcia, Spain

Corresponding author: Daniel Pérez-Berenguer (daniel.perez@upct.es)

This work was supported by the European Union (Erasmus+ Programme) under Grant 2018-1-ES01-KA201-050924.

ABSTRACT The ability of learning analytics to improve the learning/teaching processes is widely recognized. In this paper, the learning analytics architecture developed at the Digital Content Production Center of the Technical University of Cartagena (Spain) is presented. This architecture contributes to the field of learning analytics in two aspects: it allows for dashboard customization and improves the efficiency of the analysis of learners' interaction data. Events resulting from learners' interaction are captured and stored in Caliper standard format, to be further processed incrementally to allow dashboards to be shown without delay to teachers. Customization is considered a mandatory requirement for learning analytics tools, however, although some proposals have recently been made, a greater research effort in this topic is necessary. In the present work, this requirement is addressed by defining a domain-specific language (DSL) that allows teachers to customize dashboards. This language allows to express indicators (logical expressions) that classify students into different groups depending on their performance level. The paper also shows how our learning analytics approach was evaluated with a course that applies a flipped classroom method, and how it compares to the most relevant related works that have been published.

INDEX TERMS Learning analytics, DSL, model-driven development, custom dashboard, incremental event processing, R language, Caliper.

I. INTRODUCTION

Higher education institutions are tackling the challenge of taking advantage of new online educational methods (e.g. flipped classroom) and technologies (e.g. authoring tools) with the purpose of improving their teaching and learning processes. Nevertheless, this task is quite demanding for teachers, who should be helped and encouraged through software tools, training, and technical and methodological guidance. For that purpose, the Technical University of Cartagena, Spain, - UPCT hereafter - created the Digital Content Production Center (DCPC) in 2013.

The work of this center has been mainly aimed at developing an online content creation platform named INDIeOpen, which, as of today, consists of an *infrastructure*, named UPCTforma, that offers basic services, and an *authoring tool*, named INDIeAuthor, built on such an infrastructure. UPCTforma is based on the interoperability-based architecture presented in [1], and INDIeAuthor provides a family of textual languages to develop courses, as described in [2]. Building INDIeOpen is a strategic decision of the university with two main purposes: (i) having an extendable and interoperable solution which provides the desired functionality for its virtual campus; and (ii) investigating and innovating in the educational technology field. In this paper, our focus is on how learning analytics (hereafter LA) is currently supported by the platform.

Almost a decade ago, LA emerged as an area of data science focused on the learning data analysis. LA is commonly 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'' [3]. The potential of LA to enhance success students is widely recognized [4], and a great effort has been devoted by researchers to propose LA approaches, techniques and tools. However, the adoption of LA is still very limited and new research directions have recently been proposed [5].

A learning analytics software architecture is normally designed to implement an iterative 4-stages workflow [6], [7]:

The associate editor coordinating the review of this manuscript and approving it for publication was Laxmisha Rai.

(i) *capturing* interaction events, (ii) *collecting* data to be analyzed from raw events, (iii) *performing* data analysis to obtain indicators, and (iv) *visualizing* indicators on dashboards that help teachers to gather insights on the learning process and adjust it for its improvement. In [1], an initial LA architecture for UPCTforma was described, and a gamification case study illustrated its application. Here, we will present how this architecture has evolved to satisfy two new requirements: to provide teachers an instrument to customize dashboards, and to improve the efficiency in the visualization of those dashboards when a very large number of events must be analyzed. These requirements are motivated below.

Providing teachers with customization capabilities have been pointed out as a must for the adoption of LA [8], and some proposals have recently been presented [9]–[12]. In addition to a predefined analysis and visualization, LA tools should allow teachers to personalize these LA workflow stages. For this aim, we have explored how a textual domain-specific language (DSL) [13], [14] could be useful to specify which indicators should be displayed in dashboards. In particular, a metamodel-based textual DSL named `CustomLA` has been created and integrated in the DSL family of the INDIeAuthor tool. As described in [2], INDIeAuthor consists of a family of four DSLs tailored to the task of creating educational courses by defining their content, evaluation, course sequencing, and gamification. The `CustomLA` DSL allows teachers to define indicators that determine which students, at a given time, are satisfying one or more conditions in terms of study time or achievements. Indicators can be specified for whole units or for individual learning activities (i.e. a drag-and-drop) of a course.

Three kinds of LA solutions can be identified depending on the frequency at which the course's results are requested [15]: after a course finishes, periodically while the course is taught, (e.g. a few times a week), or while a course's learning activity is being performed by students (i.e. real-time processing). The level of efficiency required for the data analysis is higher for real-time processing, and lower for the complete course feedback where a batch processing is feasible. In the case of UPCTforma, the infrastructure is expected to provide support for a variety of online educational tools which will be used by teachers with different needs, and the three types of mentioned feedback loops would therefore be necessary.

In the previous LA architecture of UPCTforma, events were captured and collected in a relational database. At any time, stored learning data could be analyzed and the results shown on dashboards. But each analysis was performed from scratch by processing all the events. This solution works well in courses and activities in which the analysis is performed after their completion, but it is inefficient for periodic or real-time analysis. Teachers who were following a flipped classroom method in several courses using UPCTforma reported a negative user experience in visualizing dashboards on the student's individual work. They accessed the LA panel a few times a week, and the dashboards' generation took a long time (about 2 minutes) when the course had progressed and a very large number of events had to be processed. This illustrates the limitations of processing the whole set of registered events for a course each time a dashboard's visualization is requested. As a consequence, the data analysis component of UPCTforma has been completely refactored to implement an *incremental processing* strategy Two requirements of the applied incremental approach were (i) the visualization of dashboards at any moment during a course should be efficient enough to ensure a satisfactory user experience and (ii) the dashboards' customization should be available for the teachers.

The research contributions of our work are as follows. Firstly, a textual DSL was created that enables teachers to customize the LA dashboards. Other works have recently presented proposal to customize LA solutions, but the mechanisms provided to teachers are more limited, as discussed in detail in Section VI: wizards to choose predefined indicators [10], [12], [16] or DSLs that offer a more simple expression language to define indicators [11], [17], or DSLs targeted to developers instead of teachers [18].

Secondly, an incremental approach to efficient visualization of dashboards has been designed and implemented, which moreover integrates the customization of learning indicators. Applying some kind of incremental strategy is common to avoid delays in analytics processing, such as the progressive visual analysis technique [19] or those proposed for graph algorithms and machine learning in [20]. The main novelty of our proposal is to apply an incremental procedure when the indicators to be calculated are not previously fixed, and can be changed in execution time. The approach was tested with a case study for a course at the UPCT in which a flipped-classroom method is being used, based on UPCTforma online content.

Thirdly, by integrating the `CustomLA` DSL into INDIeAuthor, we have added capabilities of dashboards customization to this authoring tool. To the best of our knowledge, such a feature is not supported by existing authoring tools.

The present paper is organized as follows. An overview of the LA solution that was designed and implemented is first presented. Next, the two main architectural aspects are described: on the one hand, the elements of `CustomLA` DSL - metamodel of the abstract syntax, concrete syntax or notation, and semantics-; on the second hand, the incremental processing strategy. The evaluation performed on the case study is then reported. Finally, related work is commented upon, and some conclusions and further work are exposed.

II. LEARNING ANALYTICS IN INDIeOpen

In this section, the LA process and architecture defined for the INDIeOpen platform are presented, after introducing two main elements of INDIeOpen: the UPCTforma infrastructure and the INDIeAuthor authoring tool. In the following two sections, the CustomLA DSL and the incremental processing are explained in detail.

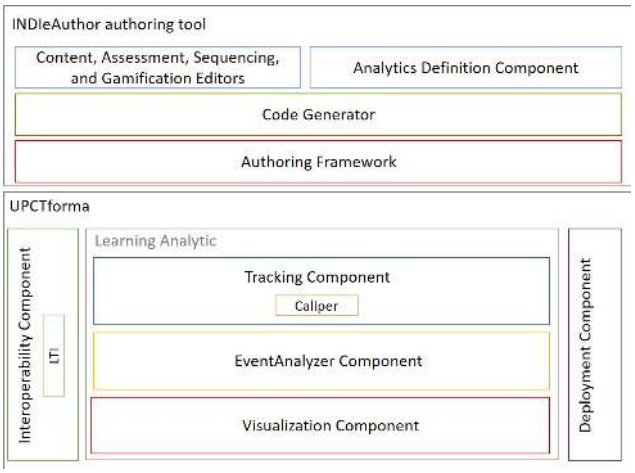

FIGURE 1. UPCTforma learning analytics architecture.

UPCTforma has been supporting LA since early stages [1]. Figure 1 shows the UPCTforma components related to LA, as well as the two basic components of interoperability and deployment. The LA components provide the services that are part of a LA architecture, namely, event tracking, event analysis, and learning outcome visualization. The *Interoperability* component uses the IMS LTI standard [21] to allow learning units to be linked from any LTI-compliant learning platform (e.g. Moodle or Sakai). The *Deployment* component deploys the content (e.g. INDIeAuthor learning units). The IMS Caliper standard is used in the *Tracking* component to capture and record events that are produced when users interact with INDIeOpen learning units (e.g. sessions login, interaction with web elements or the Multimedia component of UPCTforma) or external tools. The *EventAnalyzer* component processes raw events in order to produce the summary data required by the *Visualization* component to generate dashboards. As motivated in Section I, this LA architecture has been modified to support efficient event processing. In particular, a new EventAnalyzer component has been developed, which implements an incremental strategy to calculate partial results. The Visualization component retrieves those pre-computed partial results, avoiding, as a result, delays which would be caused by the processing of all the events. In Section IV, the incremental event processing procedure is described.

INDIeAuthor is an *authoring tool* built on UPCTforma [2]. The upper part of Figure 1 shows the three constituting elements of INDIeAuthor, shortly described below. *Editors* are provided to create educational content with the four defined DSLs. Using these languages/editors teachers can define (i) the course content, (ii) the course assessment, (iii) the course's units sequencing, and (iv) gamification activities. A *code generator* integrates the four textual DSL engines and creates a learning unit (course or activity) by instantiating an authoring framework developed for content creation. As a result, the code that implements a unit (HTML, CSS, JavaScript, JSON and PHP files) is automatically generated from DSL scripts. A textual [22] and graphical [23] version of the DSLs are available.

INDIeAuthor provides a set of widgets to create visual elements within learning units as well as learning activities (e.g., a drag and drop activity or a pair matching activity). Examples of the available widgets can be found in [24]. For each widget, the set of events that can be produced is predefined. For example, in a drag and drop widget, all the associations established by students are recorded, even those selected and removed before the student saves their answer into the system.

In its new version, the EventAnalyzer component receives as input the specification of the learning indicators that teachers want to obtain in order to monitor the students' progress in a course or activity. To enable this feature the *Analytics Definition* component for INDIeAuthor was created. This component is based on a new textual DSL, named *CustomLA*, tailored to allow teachers to write scripts which define analytics for units and activities of a course. An analytics definition consists of one or more indicators composed of four types of conditions: *Completion*, *Dedication*, *Attempts* and *Grade*. In next section, the CustomLA DSL is explained in detail. The values taken by the indicators are calculated as part of the incremental processing and the results are shown in a dashboard whenever the teacher requests it.

Figure 2 shows how UPCTforma supports learning analytics in the case of the INDIeAuthor tool. By using this authoring tool, teachers create and publish the learning units of a course. These units are deployed by means of the UPCTforma Deployment component (step 1). In addition to units, teachers can write CustomLA scripts to customize LA dashboards. The Analytics Definition component transforms these scripts into JSON documents, which are stored in a MongoDB database (step 2), as explained in the next section. Once a learning unit has been published, the student can access it using the LTI link provided by the Interoperability component (steps 3.1 and 3.2). The user's interactions with the learning units are the input to the 4-stages LA architecture of UPCTforma. Next, we will show how this architecture works by describing each stage of the LA processing.

A. CAPTURING AND COLLECTING STAGE

For each INDIeAuthor learning unit, a Caliper sensor is implemented to capture and collect events. As shown in Figure 2, the units are hosted in the Deployment component. A Caliper sensor captures the events generated through the student's interaction, which are first labeled and subsequently sent to the Tracking component (step 3.3). This component has a REST service to receive events and stores them temporarily into a message queue. A continuous execution script is then in charge of asynchronously removing elements from the queue and storing them into a MongoDB database.

B. DATA ANALYSIS STAGE

The EventAnalyzer component performs an incremental processing of the Caliper events received from the capturing

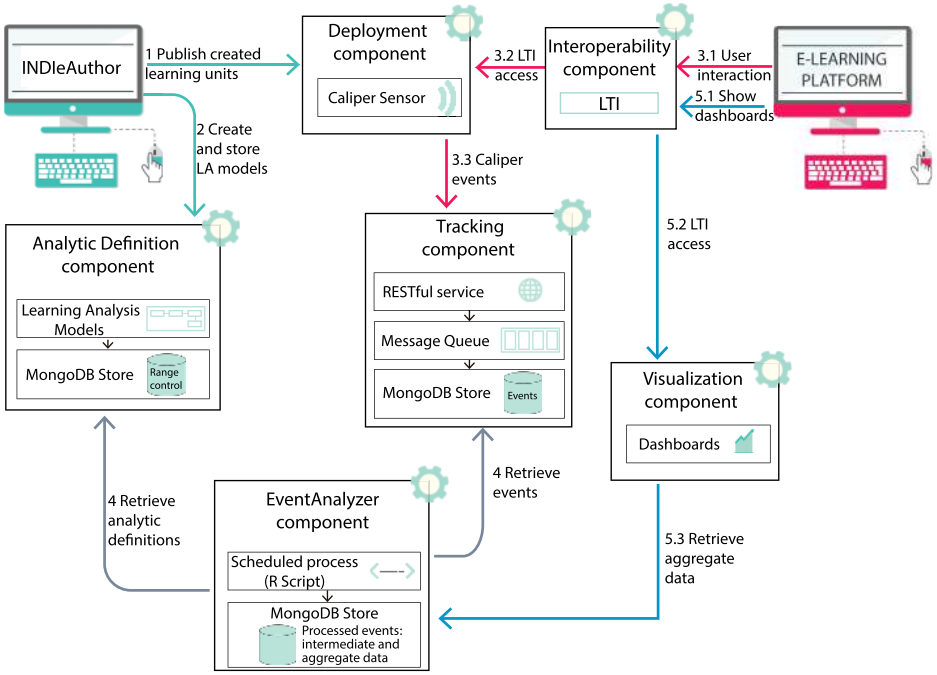

FIGURE 2. UPCTforma learning analytics process.

and collecting stage. A scheduled process is executed on the EventAnalyzer component in order to collect the incoming events along with the LA definitions stored in JSON format (step 4), and pass them to an R script [25] for processing (cleaning, transforming and summarizing). Whenever an incremental batch of events is processed, the summary data that is ultimately required to plot the dashboards is updated and stored into a MongoDB database. When a teacher accesses the dashboards panel, these minimal summary data are instantly retrieved and the plots are constructed, but no additional transformation or manipulation of the data are needed, consequently improving the loading time. This updating process is carried out based on two pieces of information: some intermediate data calculated from previously processed events are updated using the batch of new events, and subsequently combined with the LA definitions to produce the updated summary (aggregate) data. LA definitions can therefore be created or modified at any time, as they are periodically processed. Section IV will describe in more detail how the EventAnalyzer component performs the incremental processing.

C. VISUALIZATION STAGE

Finally, when a student or a teacher wants to visualize the LA outcomes in dashboards, they access the Visualization component by means of an LTI link through the Interoperability component (steps 5.1 and 5.2). The Visualization component can therefore be accessed from any LTI-compliant learning platform. Given the user and course information, this retrieves the corresponding summary data (step 5.3), and draws the LA dashboards. In the case of student's access, the dashboards only show the data related to their learning progress, and some general information about the group of students of the course, e.g. a student can compare their learning status with that of other students. A teacher has access to the detailed information on all their students. It is worth noting that student dashboards are predefined, and a set of predefined dashboards are also available for teachers. How dashboards can be customized is explained in Section IV. It should be noted that when a visualization is requested, no processing is performed, since the last calculated aggregate data is retrieved.

It is convenient to note that other learning tools (e.g. a educational game), could also require to define a specific DSL to express learning indicators, for which the CustomDSL could be reused in some cases.

III. A DSL FOR LEARNING ANALYTICS

In this section, the CustomLA DSL, which was created to express learning analytics in INDIeAuthor, is presented. In the context of Model-Driven Software Engineering (MDSE), a DSL consists of three elements [13]: (i) an abstract syntax that defines the DSL concepts and their relationships; (ii) a concrete syntax that defines the notation; and (iii) a semantic that establishes the meaning of the DSL program or script. A metamodel is used to express the abstract syntax, DSL workbenches are used to specify the concrete syntax, and a translation to existing software languages, normally programming languages, is implemented to establish the semantic. In our case, an Ecore metamodel [26] was created; the notation was defined with the Xtext workbench [27]; and a code generator was implemented to transform CustomLA scripts into JSON documents. These three elements are now described.

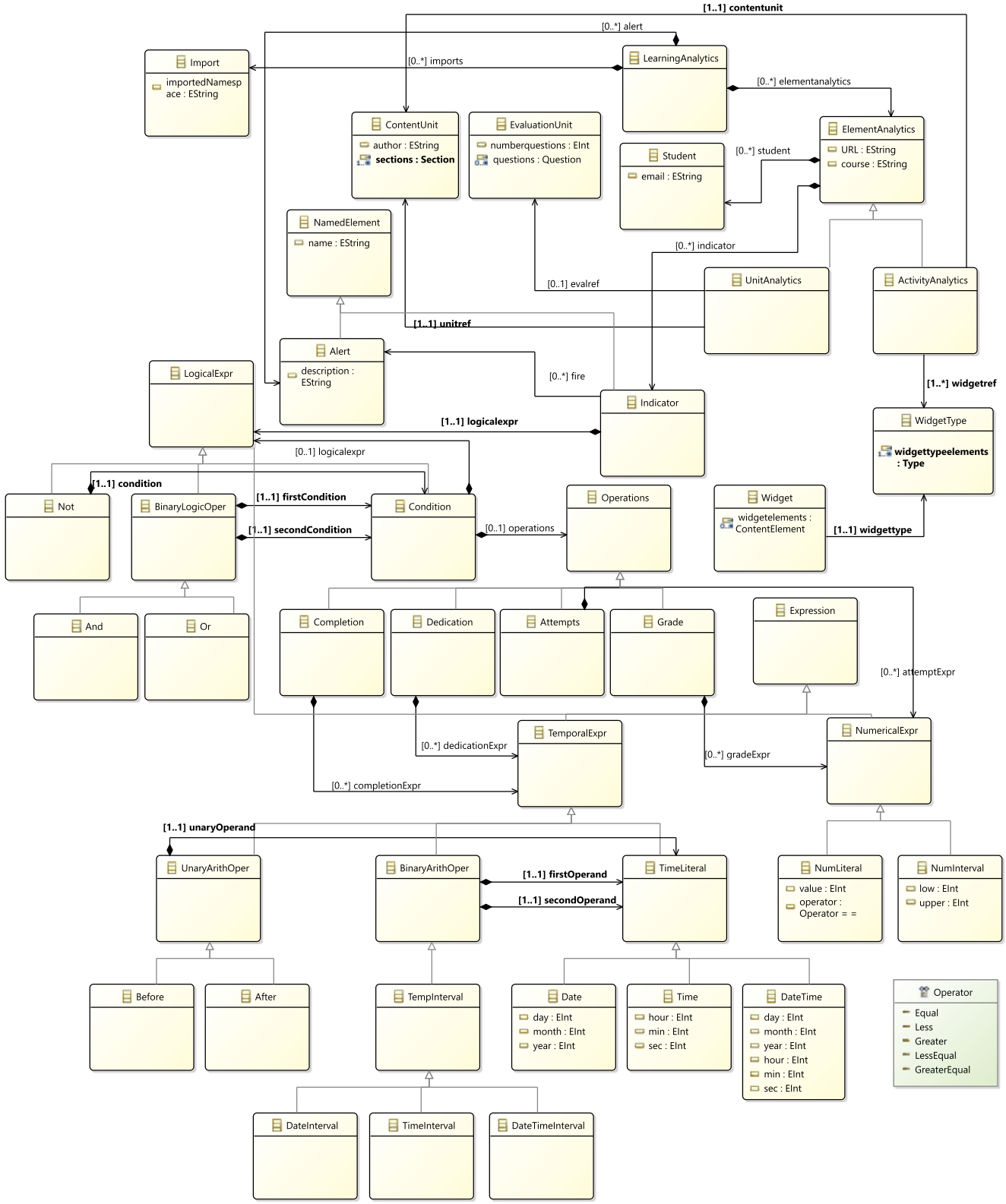

FIGURE 3. Learning Analytics DSL metamodel.

Figure 3 shows the metamodel defined for the *CustomLA*. A learning analytics definition (class *LearningAnalytics*) aggregates a set of analytics defined on learning units and activities of an INDIeAuthor course (*ElementAnalytics hierarchy* with subclasses *UnitAnalytics* and *ActivityAnalytics*). Therefore, some classes of the metamodels defined for

the INDIeAuthor languages family are referenced from the CustomLA metamodel: *ContentUnit*, and *WidgetType* of the Content metamodel, and *EvaluationUnit* of the Assessment metamodel. This requires to *import* both metamodels in the CustomLA metamodel, and CustomLA models will be linked to Content and Assessment models.

Note that a URL and a course identifier are given to each ElementAnalytics in order to reference the course on which learning analytics are applied. In INDIeAuthor, a unit can be linked to different courses. Therefore, different *ElementAnalytics* can be specified for the same unit. In addition, learning analytics can be applied to zero or more students.

Zero or more indicators (*Indicator*) can be defined for each ElementAnalytics. Each indicator consists of a logical expression (*LogicalExpr*) that applies logical operators (And, Or, and Not) to conditions (*Condition*). Conditions are operations of four different types (*Operations hierarchy*) corresponding to the four different variables under consideration: (i) the date on which students begin or complete learning units or activities (*Completion*), (ii) the student learning time (*Dedication*), (iii) the grade obtained (*Grade*), and (iv) the number of attempts needed by the students to complete activities (*Attempts*).

Operations are defined using temporal (*TemporalExpr*) and numerical (*NumericalExpr*) expressions. Three types of temporal expressions have been defined: unary, binary, and literal. A unary expression contains an operand (*TimeLiteral hierarchy*) and a unary operator (*Before* or *After*). The TimeLiteral hierarchy defines the three types of time literals: date, time, and datetime. A binary expression contains two operands and an interval operator: date, time, and datetime intervals (*TempInterval hierarchy*). A numerical expression can be formed by a numerical operand and a relational operator (*NumLiteral*), or either a numerical interval (*NumInterval*).

Figures 4 and 5 show CustomLA scripts which express indicators suggested by teachers participating in the evaluation explained in Section V. Three indicators are defined in the script of Figure 4 (lines 5 to 19): `CorrectLearning`, `WarningLearning` and `ProblemLearning`. These indicators specify when completion and dedication are considered to be correct, whether a warning should be issued or a problem is detected. An OR expression is applied in `CorrectLearning`, while an AND expression is applied in the other two indicators. Students' progress will fall into one of these three categories depending on their dedication and completion values.

Figure 5 contains a CustomLA script for a *RectangleDragAndDrop* widget. This script includes three indicators (lines 6 to 23): HighLevel, MediumLevel, and TooManyTimes. These indicators classify students in three categories depending on the numbers of attempts and the amount of time devoted to complete the activity: (i) one attempt and less than five minutes (*HighLevel* indicator), (ii) activity completed before 05/30, two or three attempts, and a dedication between five and ten minutes (*MediumLevel* indicator), and (iii) more

```
1  UnitAnalytic{
2   contentunit 'Demo Unit 1'
3   URL 'https://serverURL/unidad1/index.php'
4   course '15'
5   Indicator 'CorrectLearning'{
6    And((completion (before (datetime 05/30/2019 12:00:00))),
7    (dedication (after (time 02:30:00))))
8   },
9   Indicator 'WarningLearning'{
10   Or(
11   (completion (days (between date 06/01/2019 and
12    date 06/15/2019))),
13   (dedication (time (between time 01:30:00 and
14    time 02:29:00))))
15  },
16  Indicator 'ProblemLearning'{
17   Or((completion (after (date 06/16/2019))),
18   (dedication (before (time 01:29:00))))
19  }}
```

FIGURE 4. Example of learning analytics definition for units.

```
1   ActivityAnalytics{
2    contentunit 'Demo Unit 1'
3    URL 'https://serverURL/unidad1/index.php'
4    course '15'
5    widget RectangleDragAndDrop
6     Indicator 'HighLevel'{
7      And((attempts (number = 1)),
8      (dedication (before (time 00:05:00))))
9     },
10    Indicator 'MediumLevel'{
11     And(
12      (And(
13       (completion (before (date 05/30/2019))),
14       (attempts (between 2 and 3)))
15      ),
16      (dedication (time (between time 00:05:01 and
17       time 00:10:00)))
18     )
19    },
20    Indicator 'TooManyTimes' {
21     (attempts (number > 3))
22     fire 'FailedComprehension'
23  }}
```

FIGURE 5. Example of learning analytics definition for widgets.

than three attempts (*TooManyTimes* indicator). The *FailedComprehension* alert is fired for each student classified in the *TooManyTimes* indicator. For each alert triggered, the EventAnalyzer component sends a notification to the Motivation component which generates and sends motivation and information messages to students and teachers, respectively. The motivation aspect is nonethelesss out of the scope of this paper.

As commented in Section II, the Learning Analytics DSL had to be integrated into the INDIeAuthor DSL family. In particular, Content and Assessment models must be imported into Learning Analytics models, as indicated above. The CustomLA semantics (i.e. the CustomLA engine) was implemented by means of a model-to-text (m2t) transformation that converts a CustomLA input model into a JSON document. Figure 6 shows the JSON file generated for the *MediumLevel* indicator defined in Figure 5. Each indicator aggregates

```
1  {"ActivityAnalytics":[
2   {"URL":"https://serverURL/unidad1/index.php",
3    "course":"15",
4    "indicators":[
5     {"name":"MediumLevel",
6      "Expressions":[
7       {"type":"and",
8        "condition1":
9         {"type":"and",
10         "condition1":
11          {"type":"completion",
12           "expression":[
13            {"operator":"before",
14             "operand":{"type":"date","value":"05/30/2019"}
15            }]
16          },
17          "condition2":
18          {"type":"attempts",
19            "expression":[
20             {"operator":"intervalNum",
21              "operand1":{"type":"number","value":"2"},
22              "operand2":{"type":"number","value":"3"}
23             }]}
24        },
25        "condition2":
26         {"type":"dedication",
27          "expression":[
28           {"operator":"intervalTime",
29            "operand1":{"type":"time","value":"00:05:01"},
30            "operand2":{"type":"time","value":"00:10:00"}
31          }]}
32      }]}
33    ]}]}
```

FIGURE 6. JSON document generated for MediumLevel indicator in Figure 5.

its conditions, and each condition, in turn, aggregates its expressions. In this case, an AND expression (i.e. condition) is nested in another AND expression (lines 7-24). Figure 6 illustrates how CustomLA offers a more usable (ease, concise and legible) notation than the JSON language to specify the logical expression used as indicators.

IV. INCREMENTAL PROCESSING OF CALIPER EVENTS

The creation of a dashboard normally requires to manipulate and summarize a very large number of events. Treating all captured events in each request of LA visualization is far from efficient as the number of registered events may grow considerably as the course progresses. This strategy was used in a first version of UPCTforma [1], and some teachers reported waiting times above two minutes in the dashboard visualization. The design of a more efficient approach to compute the summary data was therefore tackled. Incremental computation algorithms are commonly used to achieve high performance in processing data streams. In this section, the incremental approach defined for UPCTforma is described. The strategy was designed to satisfy two requirements: (i) to achieve the efficiency required by LA solutions in which dashboards are requested at any moment during the course; and (ii) to support the dashboard customization based on the metamodel described in Section III. It is worth noting that sophisticated Big Data technologies are not used.

As a first step, all summary data which are required to build the LA dashboards (predefined or personalized indicators) were identified; they are referred to as ''aggregate data'', as usual in incremental processing. For each aggregate data, it is also necessary to identify the minimal ''intermediate data'' which is required to update it. This data satisfies two properties: (i) it can be incrementally updated, using only new events and previous versions of the intermediate data; and (ii) they can be transformed into aggregate data. A very simple example of an aggregate object would be the mean of a quantity, which cannot be incrementally updated, but for which the corresponding intermediate data consist of the sum and the number of cases.

Once intermediate data (and its updating function) is identified for each summary data, it is possible to implement the function that calculates such an summary data. It is remarkable that the identification of intermediate data was the major difficulty in defining our incremental strategy. Moreover, it should be emphasized that the definition of learning analytics for other UPCTforma tools would require the identification of new intermediate data.

A very convenient achieved feature is that, in the case of of dashboards for indicators specified in CustomLA scripts, the JSON files generated from CustomLA scripts (see Section III) are also input to the summary functions that update aggregate data, as shown in Figure 2. This allows, in particular, to easily update the aggregate data and consequently the corresponding dashboard in the case when the teacher decides to modify his indicators along the way, while teaching.

As indicated in Section II, a scheduled process retrieves batches of new events and learning analytics definitions periodically. This process is responsible for updating intermediate data, and transforming it into aggregate data. The frequency of execution is configurable (the default value is 60 minutes). It should be adjusted based on the number of users and the expected frequency of teachers' dashboards visualizations. The shorter the execution interval, the smaller is the number of processed events at each execution of the scheduled process. Moreover, the case study of the following section illustrates that the number of events in each batch should also be considered a configuration parameter, as explained below.

The calculated intermediate and aggregate data are stored as JSON documents into a MongoDB database, see Figure 2 (step 3.4).

In the case of INDIeAuthor, some examples of intermediate data are:

1) `user_unit_objective`: for each learning unit, each user, and each percentage of achievement associated to objectives within the unit, it registers the time needed to achieve the objective, the time spent in the unit, and the date of achievement.
2) `last_event`: for each learning unit and user, it registers the date/time of the last registered event, and the time spent at the date of the last registered event.

From `user_unit_objective`, an aggregate data that contains the number of visitors and finishers is easily obtained, which is required to build the plot in Figure 8, or the summary of achievements and time as displayed in Figure 10. These are examples of predefined dashboards that are shown for the case study presented in the next section.

Another intermediate data is `last_event`. It is essential in the updating of other intermediate data like `user_unit_objective`. Indeed, a given session for a user can span over multiple batches of events, and, consequently, the process of updating the time spent on a given unit requires summing up the time from the last registered event from the previous batch.

In order to cope with the range of possible indicators' specifications by the teachers, the visualization was chosen to be flexible enough: a bubble chart is displayed, and the user can choose the X-axis and Y-axis variables as well as the variable that sets the size of the bubbles, among the four kinds of conditions used to express indicators. The kind of visualization is the same for whole units and learning activities (e.g. a drag-and-drop widget). For the latter, the number of attempts is a natural variable to choose from when selecting, for example, the size of the bubbles. Figure 11 shows an example of a visualization corresponding to a whole unit of the case study.

V. EVALUATION

This section illustrates how the incremental processing developed for UPCTforma and the dashboard personalization DSL have been evaluated. Both the usability of the CustomLA DSL for teachers and the efficiency to render dashboards are evaluated through a case study based on a UPCT course.

An experiment was conducted with the ''Human Resource Management'' (HRM) course, a subject in the Business Management and Administration degree at the UPCT [28]. The learning units for the course were produced with INDIeAuthor. One hundred and twenty-three users participated: 119 students and 4 teachers. The students were divided into four groups, depending on which language (English or Spanish) the course was taught in and the class timetable.

A three hours training session was delivered to the involved teachers. In the first part of the session, the attendees received training in the tool to create analytic definitions. The examples shown in Figures 4 and 5 were implemented following their suggestions. The participants were asked to create analytic definition examples with the tool. Each teacher had a computer with the tool (CustomLA editor and engine). By the end of the session, all the attendees ended up being able to write the examples provided. The dashboards' panel was explained in the second part of the session. The four teachers indicated that the tool was easy to use and that the dashboards were easy to understand. The students' training on the interpretation of dashboards was provided by the teachers themselves.

The incremental event processing was tested during the teaching of the subject, from January 2019 to July 2019.

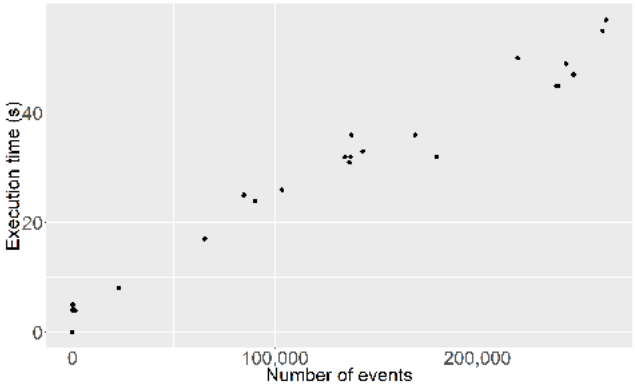

FIGURE 7. Execution times of the events' processing algorithm for different sizes of batches of events.

A total of 263,495 events, overall homogeneously distributed, were registered and processed. The types of events were: 8,455 session's opening or closing, 4,890 ''keep alive'' events (to check active sessions), 9,399 evaluation events (e.g, questions, assessments, and grades) and, finally, 240,751 events related to learning activities (e.g, drag and drop, and test activities). The frequency of the scheduled process which collects new events and updates intermediate and aggregate data, as described in Section IV, was set to 60 minutes, which led to short execution times. The longest execution time was 6 seconds for three batches of about 10,000 events. Moreover, this setting implies that when a teacher accesses the dashboards' panel, the data are, at most, 60 minutes old, which was considered as sufficient by the participating teachers

On the other hand, it was decided to take advantage of the registered events in this real scenario to test the behavior of the events' processing algorithm. Concretely, a number of batches of events of different sizes were prepared from the whole set of events and sent to the processing script. Figure 7 displays the execution times versus the number of events contained within the batch. For batches of close to 20,000 events, the processing takes less than 10 seconds. Even when the full set of 263,000 events is sent to the script, the execution time does not exceed one minute.

As a result of the simulation experiment, the execution procedure of the events' processing algorithm was modified and improved. Instead of only scheduling it to launch on a given time interval, an additional triggering criterion is put into place: if, before the scheduled time, the number of collected events in the batch reaches a threshold (for instance, 20,000), they are directly sent to the incremental processing script, and the events' retrieving process is reset. If the limit value is not reached in the established period, the algorithm is launched as scheduled. The execution time associated to the events processing can therefore be ensured to stay below a value.

Finally, figures 8, 9, and 10 show three examples of dashboards from one of the student groups. Figure 11 shows an example, for a given learning unit, of the customizable dashboard which displays the indicators suggested and created by the teachers through the analytic definition tool. In this case, the user chose to display the date of achievement

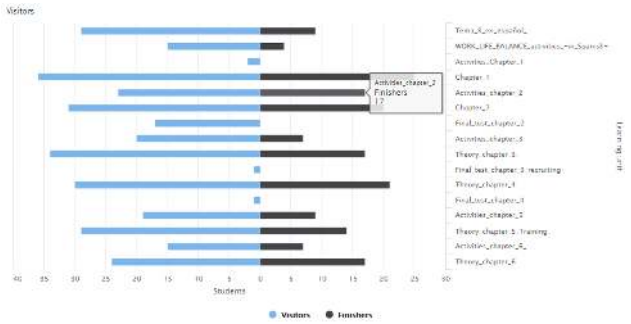

FIGURE 8. Visitors vs Finishers students by learning units.

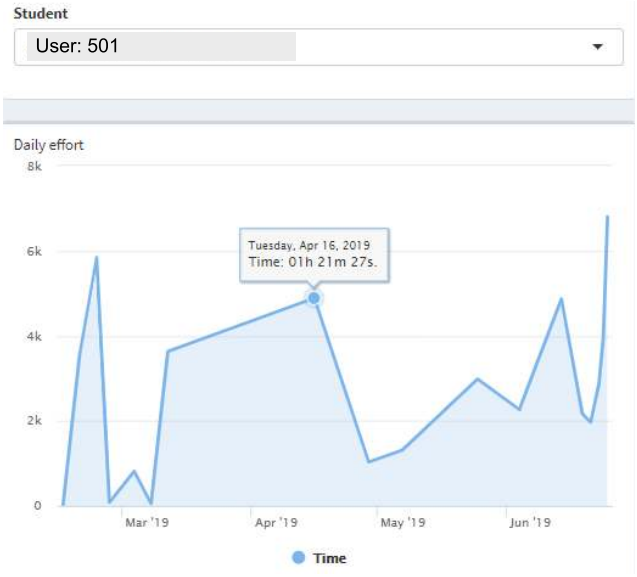

FIGURE 9. Daily effort for a selected student.

Results
Show 10 entries
Search:

current_title	Percentage	Time
Tema_3_en_español_	100	02h 24m 51s.
WORK_LIFE_BALANCE_activities_-in_Spanish-	0	00h 00m 21s.
Chapter_1	100	03h 57m 43s.
Chapter_2	100	00h 13m 36s.
Theory_chapter_3_	100	02h 15m 19s.
Theory_chapter_4	100	01h 41m 42s.
Theory_chapter_5_Training_	100	01h 45m 49s.
Theory_chapter_6	100	01h 20m 16s.

Showing 1 to 8 of 8 entries Previous 1 Next

FIGURE 10. Summary of achievements and time for a selected student.

in the X-axis, the degree of achievement (over 100%) in the Y-axis, and selected, for the size of the bubble, the time spent for achievement.

Unlike the first, non incremental, version of the LA architecture of UPCTforma, no visualization problems associated to loading time were reported during the teaching of the subject, despite the large number of registered events.

VI. RELATED WORK

In this section the three research contributions of our work are contrasted with some relevant and related works of the literature. The comparison is organized in three parts: (i) LA customization approaches; (ii) Efficiency in LA architectures; and (iii) LA support integrated into the authoring tool.

A. LA CUSTOMIZATION APPROACHES

Several works and surveys have identified the availability of tools providing teachers customization capabilities as a must for the adoption of LA [8], [9], [11], [12], [29]. Instead of predetermined analysis and visualization, such tools should allow teachers to configure the stages of a LA workflow. The Student Relationship Engagement System (SRES) is a LA tool devised to provide teachers several ways of ''customizing analysis to the needs of their students and courses'' [9], [11]. When using SRES ,[1] teachers can decide which learning and teaching data to be collected, curate data with spreadsheets, indicate conditions to identify particular student groups, and select data to be analyzed. The data selected are imported from SIS and LMS systems and are analyzed using machine learning algorithms As for the conditions characterizing student groups, they are very simple and expressed through a query form which allows to select a column id, a relational operator, a value and its data type (e.g. task_time <=100 as number). In contrast, the LA solution described in the present work provides a DSL that allows more complex queries to be expressed in order to characterize student groups through indicators. Moreover, efficiency issues were not considered in SRES. In our case, both custom and predefined dashboards display frequently retrieved data, which allows the teachers to monitor the students' progress during a learning activity of a course. Moreover, the live modification of an indicator does not require the processing of all the student's registered events from scratch since, as mentioned in Section IV, the aggregate data are updated directly from the intermediate data and the indicators' analytic definitions.

In [12], a rule-based indicator definition tool (RIDT), to customize LA is presented. Users (e.g. teachers) express a Goal/Question/Indicator (GQI) triple by means of an editor. A generator transforms these rules into Drools rules.[2] For example, a goal for a teacher could be ''to know which students are active in her Software Technology class''. If no indicator exists for this goal then she/he should use the tool to create a new question ''how active are my students in the Software Technology class'', and she/he should use RIDT to create a new indicator by using available wizard to indicate: the indicator name, the indicator type (e.g. statistics), the data source (e.g., L^2P or Moodle), the indicator category (e.g. forum discussion), the indicator filter to be applied to obtain required data (in this case, Software Technology class data), and the kind of dashboard (e.g. a chart bar). GQI rules automatically generated are executed by the Drools

[1] https://www.sres.io/
[2] http://www.drools.org/

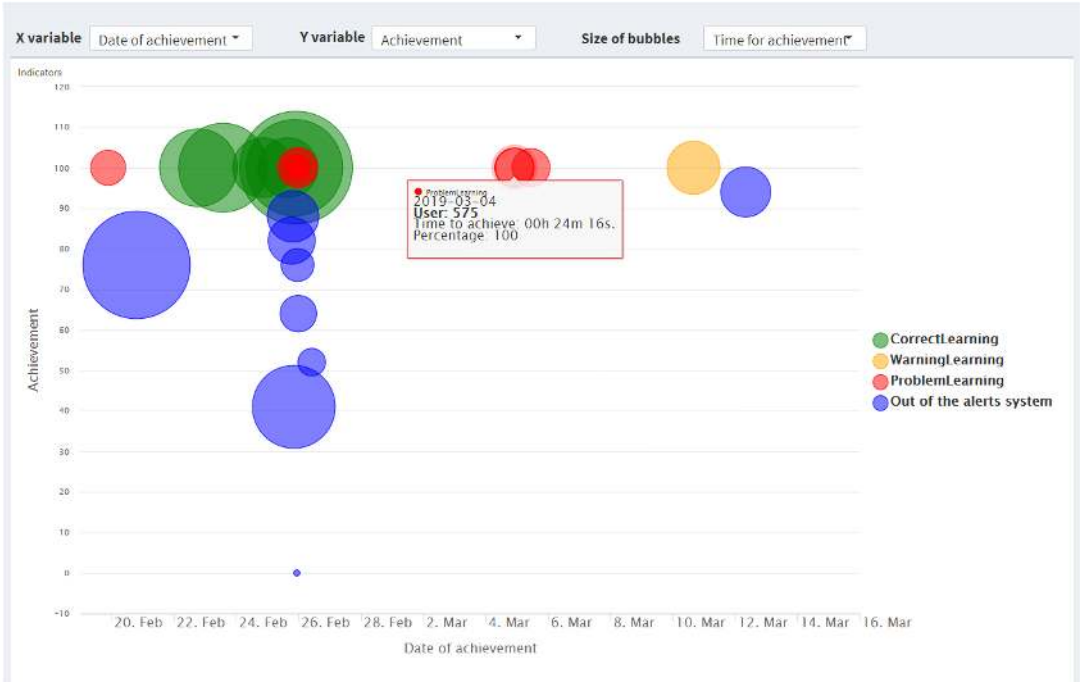

FIGURE 11. Date and time control ranges for the group of students for a selected unit.

rule engine, and the results obtained are then visualized. A database stores LCDM (Learning Context Data Model) learning events from external data sources. In contrast, our proposal uses a DSL to express indicators instead of reusing a set of predefined indicators. Our DSL allows to express one or more indicators for an unit or activity of a course. Logical expressions with temporal and arithmetic operators can be used to express indicators.

A framework to develop adaptive VLE (Virtual Learning Environment) is proposed in [17]. A DSL is provided to teachers to configure data collecting and adaptation stages, and the Weka workbench is applied to process data. The DSL is not rigorously described and code examples are not visible in [17] or available in other sources. The authors claim that the DSL is intended to ''express weekly content (resources), and information related to LA and adaptation''. However, they do not indicate what kind of information teachers must provide. In our proposal, the adaptation of learning processes is not adressed, but the DSL is aimed at expressing indicators that give teachers insight into the students' progress in a course or activity, and these indicators are shown in a very flexible and general-purpose dashboard.

EvalCourse is a DSL proposed to enable teachers to choose indicators to evaluate learning activities [16]. These activities take place on LMS platforms and log files are used to collect information on learner interactions. The EvalCourse engine generates Pentaho-based ETL scripts from the choices expressed with the DSL. These ETL scripts generate reports from learning data stored into LMS databases. Like EvalCourse, CustomLA is also intended for teachers to define learning indicators However, it presents substantial differences. First, CustomLA is integrated into a LA architecture which is based on standards to achieve platform-independence, while EvalCourse is a Moodle-specific solution (the DSL engine should be changed to be applicable in other LMS); Second, EvalCourse is an example where a wizard is an appropriate solution instead of creating a DSL, because the language only allows to choose a predefined value for a few parameters: milestones (e.g. participation or evaluation), assignments (forum, campus, or workshop) and date range. In contraste, with CustomLA, teachers can write logical, numerical, and temporal expressions. Finally, in the current proposal, a LA dashboard is made available for the teachers to monitor their indicators, while EvalCourse only provides a predefined dashboard for each kind of predefined indicator.

A recent work has presented the EngAge engine which separates the assessment from the educational game itself [10], [18]. A DSL is offered to developers to configure the assessment of any educational game available in the engine, and a set of web services is in charge of performing the assessment. This DSL allows to express very simple conditions about scores, and also information as feedback messages and player profiles. When students play with games their interactions are captured and collected in a proprietary format. Once a game is over, EngAge obtains the player's assessment, and offers to educators a very simple editor to modify the assessment. EngAge also includes some LA dashboards that display indicators, such as learning curves between gameplays, and learning curves within a game. In our case, the LA architecture is activity-independent. It could be applied to any course or game that integrates a Caliper sensor to collect events. In addition, our DSL is targeted to teachers, which can define expressions to calculate

TABLE 1. Comparison of the approaches.

	[11]	[12]	[17]	[16]	[10] [18]	CustomLA
Purpose	LA Customization	LA Customization	Adaptive VLE	Course assessment	Integrating LA in games (developer) Modify and visualize assessment (educator)	LA Customization
Configurable LA stages	Capturing/ Collecting, Analysis, Visualization	Analysis, Visualization	Collecting and Adaption	Visualization	None	Analysis, Visualization
Learning Indicators	Students classified in groups	Predefined Indicators	Weekly content Information on LA and adaptation	Predefined indicators	Scores in educational games	Predefined and teacher-defined Indicators
Definition (or choice) Mechanism of indicators	DSL (Parser Creation)	Wizard (to choose predefined indicators)	DSL (Metamodel-based workbench: MPS)	DSL	DSL (For Developers) (Parser Creation) GUI (For Educators)	DSL (Metamodel-based workbench: Xtext)
Kind of expressions	Very simple query	N/A	Unknow	Indicators are not defined but chosen	Conditional Expression for badges, ...	Conditional Expression for 4 measures
Target User	Teachers	Teachers	Teachers	Teachers	Developers (DSL) Educators (GUI)	Teachers
Standards	No	LCDM	No	No	No	LTI/Caliper

learning indicators to be shown in LA dashboards. As indicated in Section III, the CustomLA DSL described in this paper allows to define indicators on dedication, attempts, grade and completion, while in EngAge only scores are considered. Moreover, our DSL supports logical, numerical and temporal expressions, and only one-term numerical expressions are supported in EngAge.

In [30], the customization of multimodal LA solutions is considered essential in blended learning scenarios. The authors indicate that teachers should be able to adapt these solutions to a particular blended scenario. Two kinds of customization are considered: (i) teachers can add data sources whenever they analyze learning results, and (ii) teachers can choose indicators from a set of predefined indicators. Tools/languages to define indicators or strategies to customize LA dashboards are not addressed in [30], but a customization process is described and applied to two case studies.

This part of the section is concluded with a work that presents a generative solution to customize dashboards in the context of decision-making [31]. Although the work is not focused on LA visualization, a model-based solution is proposed as is the case in our approach. In [31], a metamodel is defined to represent concepts and relationships in the dashboard domain, and feature models to specify the variability in that domain. Layout and Content of a dashboard are defined by means of an XML-based notation, and visualization feature and restrictions are expressed by means of feature models. The XML document and feature models are used to automatically generate customized dashboards. In contrast to our solution, the authors use XML instead of taking advantage of the dashboard metamodel to create a DSL which could provide syntax constructs adapted to the domain concepts and relationships. Our DSL is aimed at defining learning indicators but dashboard design specification is not considered, while the proposal of [31] allows for the specification of the dashboard structure. In both solutions, dashboards are automatically generated.

Table 1 summarizes the comparison of the different approaches. The work presented in [31] is not included in the table because it does not address LA customization.

B. LA ARCHITECTURES

SmartLAK is a big data architecture for virtual learning environment (VLE) [32]. An event management component is in charge of colecting events in real time. These events are labeled and stored using xAPI (Experience API) [33], and they are processed by means of big-data techniques to provide LA services. Our LA architecture has been designed to be integrated into the UPCTforma infrastructure. Therefore, it could be applied to any educational tool created within that platform. In this paper, its application to the IndieAuthor authoring tool has been described. As indicated in Section II, UPCTforma is based on IMS LTI instead of xAPI, but this specification is planned to be supported also. As new widgets are incorporated into the authoring tool, the associated events recollection has to be defined. No sophisticated big data techniques are used in our solution, which is based on an incremental frequent processing of Caliper events.

In [34], a LA architecture for data acquisition, analysis, and notifications is presented. Events are encoded in the ActivityStream format, and a SQLSpaces shared memory for coordination and communication is used. Scenario-specific analysis agents can be included to send recommendation messages. Ex-post analysis on a data warehouse and concept mapping analysis can be made. Our architecture shows two significant differences in relation to the proposal of Tobias Hecking *et al.* The first one is that events are stored in Caliper format as indicated in Section II. This standard specification can be used since UPCTforma uses the LTI

interoperability standard. Any educational tool that is integrated into UPCTforma could provide learning analytics by implementing a Caliper sensor to collect, label and store events. The second one is that the customization of dashboards by the user is dealt with by providing a DSL. Moreover, an incremental frequent processing of events is used in contrast to ex-post or concept mapping analysis.

In [35], a learning analytics architecture for Khan Academy platform (ALAS-KA) is presented. The architecture proposed is a tightly coupled solution. ALAS-KA processes the Khan Academy data proposing new visualizations based on the Google Charts API. Non-incremental data processing at regular intervals (every 6 hours) is carried out. In contrast, our work presents an interoperable infrastructure for learning analytics with incremental processing.

The Progressive Visual Analytics (PVA) is an incremental processing technique aimed at enabling analysts to inspect and interact with partial results instead of waiting for the full completion of the analysis [19]. PVA is useful when complex analysis are applied on large datasets. The incremental processing presented in the current paper significantly differs from PVA in that: (i) an event stream is processed instead of a large dataset; (ii) our goal is to build dashboards efficiently. In our case, batches of events are processed as they arrive and the dashboards are updated with the new aggregate results.

In [36], a services-based architecture for Ubiquitous LA is presented. A message queue is used to collect learning events from several sources, cleaning and enriching them by means of a pipeline processing. The processed events can be stored in several kind of storage such as ElasticSearch or Postgres. Some metrics are computed and reported for developers and researchers. This architecture does not address issues on LA customization or performance of LA visualization. The authors do not provide a detailed description about the visualization for teachers.

Finally, it should be noted that computational efficiency has not been addressed in any of the LA customization approaches discussed above.

C. LA SUPPORT IN AUTHORING TOOL

In [2] a comparative study of 9 authoring tools was presented. Tables summarizing the results of this study can be found in [37]. Five of these tools provided predefined data analysis and visualization. In contrast, our proposal presented in this paper permits customized and predefined analysis and visualization.

VII. CONCLUSION AND FUTURE WORK

In this paper an approach that allows teachers to customize learning analytics solutions has been presented. For that purpose, a simple textual DSL was developed for teachers to express the indicators they consider relevant to monitor the progress of their students. In addition, an incremental processing strategy was designed to support LA in scenarios in which the results can be requested as frequently as desired. This strategy has integrated the calculation of indicators expressed with our CustomLA DSL. It is remarkable that LA personalization is identified as an essential feature for the adoption of LA tools. Some approaches have recently been presented to go along this path, but a great research effort is still necessary.

The proposed DSL-based personalization and incremental processing approaches were integrated into the INDIeAuthor authoring tool built upon the UPCTforma infraestructure. To the best of our knowledge, no existing authoring tool allows for LA customization, and neither are we aware of any LA tool which integrates an incremental processing procedure combined with customization features.

The LA solution presented in this paper was developed as part of the INDIe project.[3] Our LA approach was evaluated using the Human Resources Management semester course of the UPCT, for which online content had been created using UPCTforma. When a first, non incremental, version of the events processing procedure was used, teachers experienced efficiency problems to visualize dashboards as the number of events grew. With the new incremental approach, these problems have disappeared and dashboards are updated every 60 minutes, which is, to our understanding, sufficient for most teaching scenarios. Along the course, a very large number of events was processed. On the other hand, the teachers participating in the case study were able to write CustomLA scripts for their indicators without any difficulty.

As further work, we are planning to (i) extend the CustomLA metamodel with new definitions, for example to take into account the possible sequencing of learning units, or to allow indicators to be defined for a section within the learning unit; (ii) develop a graphical notation for CustomLA to facilitate the definition of indicators by teachers unfamiliar with coding; (iii) extend the number of customizable dashboards; and (iv) to define a DSL that would enable students to customize their dashboards, a requested feature according to the survey on preferences of students for LA presented in [38].

REFERENCES

[1] D. Pérez-Berenguer and J. García-Molina, "A standard-based architecture to support learning interoperability: A practical experience in gamification," *Softw., Pract. Exper.*, vol. 48, no. 6, pp. 1238–1268, Feb. 2018.

[2] D. Perez-Berenguer and J. Garcia-Molina, "INDIeAuthor: A metamodel-based textual language for authoring educational courses," *IEEE Access*, vol. 7, pp. 51396–51416, 2019.

[3] G. Siemens, D. Gasevic, C. Haythornthwaite, S. Dawson, S. B. Shum, R. Ferguson, E. Duval, K. Verbert, and R. Baker, "Open learning analytics: An integrated & modularized platform," Ph.D. dissertation, Open University Press, London, U.K., 2011.

[4] D. Gašević, S. Dawson, and G. Siemens, "Let's not forget: Learning analytics are about learning," *TechTrends*, vol. 59, no. 1, pp. 64–71, Dec. 2014.

[5] S. Dawson, S. Joksimovic, O. Poquet, and G. Siemens, "Increasing the impact of learning analytics," in *Proc. 9th Int. Conf. Learn. Anal. Knowl. (LAK)*, 2019, pp. 446–455.

[6] A. L. Dyckhoff, D. Zielke, M. Bültmann, M. A. Chatti, and U. Schroeder, "Design and implementation of a learning analytics toolkit for teachers," *J. Educ. Technol. Soc.*, vol. 15, no. 3, pp. 58–76, 2012.

[3]Interactive Digital Content Platform to Share, Reuse and Innovate in the Classroom, a three-year Erasmus+ project(2018-1-ES01-KA201-050924), cofunded by the European Commission.

[7] G. Siemens, "Learning analytics: The emergence of a discipline," *Amer. Behav. Scientist*, vol. 57, no. 10, pp. 1380–1400, Aug. 2013.
[8] A. F. Wise and J. Vytasek, "Learning analytics implementation design," *Handbook of Learning Analytics*. Alberta, AB, Canada: Society for Learning Analytics Research, 2017, pp. 151–160.
[9] N. Arthars, M. Dollinger, L. Vigentini, D. Y.-T. Liu, E. Kondo, and D. M. King, "Empowering teachers to personalize learning support," in *Utilizing Learning Analytics to Support Study Success*. Cham, Switzerland: Springer, 2019, pp. 223–248.
[10] Y. Chaudy and T. Connolly, "Specification and evaluation of an assessment engine for educational games: Empowering educators with an assessment editor and a learning analytics dashboard," *Entertainment Comput.*, vol. 27, pp. 209–224, Aug. 2018, doi: 10.1016/j.entcom.2018.07.003.
[11] D. Y. T. Liu, C. E. Taylor, A. J. Bridgeman, K. Bartimote-Aufflick, and A. Pardo, "Empowering instructors through customizable collection and analyses of actionable information," in *Proc. 1st Learn. Anal. Curriculum Program Qual. Improvement Workshop Co-Located With Int. Learn. Anal. Knowl. Conf. (LAK)*, Edinburgh, U.K., Apr. 2016, pp. 3–9.
[12] A. Muslim, M. A. Chatti, T. Mahapatra, and U. Schroeder, "A rule-based indicator definition tool for personalized learning analytics," in *Proc. 6th Int. Conf. Learn. Anal. Knowl. (LAK)*, Edinburgh, U.K., Apr. 2016, pp. 264–273.
[13] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*. San Rafael, CA, USA: Morgan & Claypool Publishers, 2012.
[14] M. Fowler, *Domain-Specific Languages*. Reading, MA, USA: Addison-Wesley, 2010.
[15] A. Shimada, S. Konomi, and H. Ogata, "Real-time learning analytics system for improvement of on-site lectures," *Interact. Technol. Smart Edu.*, vol. 15, no. 4, pp. 314–331, Nov. 2018.
[16] A. Balderas, J. M. Dodero, M. Palomo-Duarte, and I. R. Rube, "A domain specific language for online learning competence assessments," *Int. J. Eng. Edu.*, vol. 31, no. 3, pp. 851–862, 2015.
[17] S. Meacham, D. Nauck, and H. Zhao, "Framework for personalised online education based on learning analytics through the use of domain-specific modelling and data analytics," in *Proc. Conf. Next Gener. Comput. Appl. (NextComp)*, Sep. 2019, pp. 1–7.
[18] Y. Chaudy and T. Connolly, "Specification and evaluation of an assessment engine for educational games: Integrating learning analytics and providing an assessment authoring tool," *Entertainment Comput.*, vol. 30, May 2019, Art. no. 100294, doi: 10.1016/j.entcom.2019.100294.
[19] C. D. Stolper, A. Perer, and D. Gotz, "Progressive visual analytics: user-driven visual exploration of in-progress analytics," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 1653–1662, Dec. 2014, doi: 10.1109/TVCG.2014.2346574.
[20] S. Venkataraman, I. Roy, A. AuYoung, and R. S. Schreiber, "Using R for iterative and incremental processing," USENIX, Boston, MA, USA, Tech. Rep., 2012, pp. 1–6. [Online]. Available: https://www.usenix.org/conference/hotcloud12/using-r-iterative-and-incremental-processing
[21] *Learning Tools Interoperability (LTI)*, Accessed: Jan. 12, 2019. [Online]. Available: https://www.imsglobal.org/activity/learning-tools-interoperability
[22] *Textual version of INDIeAuthor*. [Online]. Available: https://github.com/cpcdupct?tab=repositories
[23] *INDIeAuthor Demo video*. [Online]. Available: http://indie.upct.es/videos/
[24] *Examples of the Implemented Widgets in INDIeAuthor*. [Online]. Available: http://indie.upct.es/widgets/
[25] R. C. Team, *R Foundation for Statistical Computing*. Vienna, Austria. Accessed: 2019. [Online]. Available: https://www.R-project.org/
[26] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley, 2009.
[27] *A Programming Language Framework*. Accessed: Jun. 12, 2019. [Online]. Available: http://www.eclipse.org/Xtext/
[28] *Human Resources Management*. [Online]. Available: http://cpcd.upct.es/documentos/510101005_en.pdf
[29] D. Gašević, S. Dawson, T. Rogers, and D. Gasevic, "Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success," *Internet Higher Edu.*, vol. 28, pp. 68–84, Jan. 2016.
[30] M. J. Rodríguez-Triana, L. P. Prieto, A. Martínez-Monés, J. I. Asensio-Pérez, and Y. Dimitriadis, "The teacher in the loop: Customizing multimodal learning analytics for blended learning," in *Proc. 8th Int. Conf. Learn. Anal. Knowl. (LAK)*, Sydney, NSW, Australia, Mar. 2018, pp. 417–426, doi: 10.1145/3170358.3170364.
[31] A. Vázquez-Ingelmo, F. J. García-Peñalvo, and R. Therón, "Taking advantage of the software product line paradigm to generate customized user interfaces for decision-making processes: A case study on university employability," *PeerJ Comput. Sci.*, vol. 5, p. e203, Jul. 2019, doi: 10.7717/peerj-cs.203.
[32] T. Rabelo, M. Lama, R. R. Amorim, and J. C. Vidal, "SmartLAK: A big data architecture for supporting learning analytics services," in *Proc. IEEE Frontiers Edu. Conf. (FIE)*, Oct. 2015, pp. 1–5.
[33] *xAPI Website*. Accessed: Jun. 12, 2019. [Online]. Available: https://xapi.com/
[34] T. Hecking, S. Manske, L. Bollen, S. Govaerts, A. Vozniuk, and H. U. Hoppe, "A flexible and extendable learning analytics infrastructure," in *Proc. Int. Conf. Web-Based Learn.* Cham, Switzerland: Springer, 2014, pp. 123–132.
[35] J. A. Ruipérez-Valiente, P. J. Muñoz-Merino, and C. D. Kloos, "An architecture for extending the learning analytics support in the khan academy framework," in *Proc. 1st Int. Conf. Technol. Ecosyst. Enhancing Multiculturality (TEEM)*, 2013, pp. 277–284.
[36] T. Rohloff, J. Renz, G. N. Suarez, and C. Meinel, "A ubiquitous learning analytics architecture for a service-oriented mooc platform," in *Eur. MOOCs Stakeholders Summit* Cham, Switzerland: Springer, 2019, pp. 162–171.
[37] D. Pérez-Berenguer and J. García-Molina. *Authoring Tools Comparative Study*. Accessed: Jun. 12, 2019. [Online]. Available: http://cpcd.upct.es/upctforma/study.html
[38] L. D. Roberts, J. A. Howell, and K. Seaman, "Give me a customizable dashboard: Personalized learning analytics dashboards in higher education," *Technol., Knowl. Learn.*, vol. 22, no. 3, pp. 317–333, Jun. 2017.

DANIEL PÉREZ-BERENGUER received the degree in computer science from the University of Murcia, Murcia, Spain, in 2002, where he is currently pursuing the Ph.D. degree in computer science. He is also an Assistant Professor with the Department of Information Technology and Communications, Technical University of Cartagena, where he also leads the Digital Content Production Center. His research interests include educational technology, authoring tools, gamification, and learning analytics.

MATHIEU KESSLER received the Ph.D. degree in statistics for stochastic differential equations from the University of Pierre et Marie Curie, Paris. He is currently a Full Professor of statistics with the Department of Applied Mathematics and Statistics, Universidad Politécnica de Cartagena, Cartagena, Spain. Over the last few years, he has become increasingly interested in applications of statistics, data analysis, and data science, with a particular interest in data visualization and learning analytics.

JESÚS GARCÍA-MOLINA received the Ph.D. degree in chemistry from the University of Murcia, Spain, in 1987.

Since 1991, he has been a Full Professor with the Faculty of Informatics, University of Murcia. He currently leads the Modelum Group and an R&D Group focused on Model-Driven Engineering with a close partnership with industry. His research interests include model-driven development, domain-specific languages, and model-driven modernization. He is the author of two textbooks. His main research contributions are listed in DBLP.