An Ontology-enabled Context-aware Learning Record Store Compatible with the Experience API

Jonas Anseeuw[1], Stijn Verstichel[1], Femke Ongenae[1], Ruben Lagatie[2], Sylvie Venant[2] and Filip De Turck[1]

[1]*Department of Information Technology, Ghent University - iMinds, Technologiepark-Zwijnaarde 15, 9052 Ghent, Belgium*

[2]*Televic NV, Leo Bekaertlaan 1, 8870 Izegem, Belgium*

Keywords: Learning Analytics, Ontologies, Experience API, Linked Data, Learning Management System, Learning Record Store, Context-aware.

Abstract: In education, learners no longer perform learning activities in a well-defined and static environment like a physical classroom. Digital learning environments promote learners anytime, anywhere and anyhow learning. As such, the context in which learners undertake these learning activities can be very diverse. To optimize learning and the environment in which it occurs, learning analytics measure data about learners and their context. Unfortunately, current state of the art standards and systems are limited in capturing the context of the learner. In this paper we present a Learning Record Store (LRS), compatible with the Experience API, that is able to capture the learners' context, more concretely his location and used device. We use ontologies to model the xAPI and context information. The data is stored in a RDF triple store to give access to different services. The services will show the advantages of capturing context information. We tested our system by sending statements from 100 learners completing 20 questions to the LRS.

1 INTRODUCTION

In education there is a shift from traditional teacher-led learning within single classroom boundaries, to a digital learning environment where learners can learn anytime (during class, training, preparing examination, examination,...), anywhere (classroom, at home, library, train & bus,...) and anyhow (PC, tablet, smartphone,...) (Fiedler and Väljataga, 2010). As a result, the context in which the learners perform activities such as exercises, assessments and examinations is no longer well-defined and static, but happens in very diverse contexts. Context can be defined as any information that can be used to characterize the situation of an entity (Abowd et al., 1999), e.g., a combination of time, location and used device.

For purposes of understanding and optimizing learning and the environments in which it occurs, the Society for Learning Analytics Research has coined the concept of learning analytics as the measurement, collection and analysis of data about learners and their contexts[1].

Currently, the Experience API (xAPI) specification (Experience API Working Group, 2013), formerly known as Tin Can API, and the successor to SCORM[2], is the de facto standard to log learners their learning experiences. It captures these learning experiences in the form of *I did this and it resulted in that* statements.

The specification already provides a *context* field to add some contextual information. However, in spite of the effort, there are no details about how information such as device and location should be captured. To add more contextual information the specification offers an *extensions* field allowing arbitrary data to be attached as context. This extensions field is merely intended to provide a natural way to extend the context field.

In this paper we present a system that is able to also capture the environment in which learning experiences happened. Consequently, our system stores more information about the learner than current state of the art systems. To make sure that statements sent to our system are still backwards compatible to other systems that support the xAPI specification, we extend the xAPI specification using the extensions

[1]http://www.solaresearch.org/mission/about/

[2]http://adlnet.gov/adl-research/scorm/

Anseeuw, J., Verstichel, S., Ongenae, F., Lagatie, R., Venant, S. and Turck, F.
An Ontology-enabled Context-aware Learning Record Store Compatible with the Experience API.
DOI: 10.5220/0006049000880095
In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 2: KEOD*, pages 88-95
ISBN: 978-989-758-203-5

mechanism.

Ontologies are used extensively to model context (Gruber, 1993). Describing the xAPI specification as an ontology facilitates the integration with existing context ontologies describing context information. Also, the xAPI specification is built on top of the Activity Streams specification, which is already compatible with JSON-LD (J. Snell, M. Atkins, W. Norris, C. Messina, M. Wilkinson, and R. Dolin, 2015). The object properties in JSON-LD documents map to concepts in an ontology.

Initial research has been done to map the xAPI to an ontology model (De Nies et al., 2015; Vidal et al., 2015). We extend on this work by creating our own context ontology and import this with the xAPI ontology.

The use of ontologies offers additional advantages:

- Validation: the concepts and axioms of an ontology provide a sound foundation for the validation of the xAPI statements.
- Reasoning: ontologies defined in OWL 2, support reasoning mechanisms provided by description logics.
- Query: xAPI statements defined in RDF can be stored in a triple store and queried through a SPARQL endpoint.

The remainder of this paper is structured as follows. First, we motivate this research with concrete use case scenarios in Section 2. We present the state of the art with regards to capturing learner's data in Section 3. In Section 4 we present the xAPI ontology and our extended context model. In Sections 5 and 6, we show the architecture and implementation of our system. Finally, we give an evaluation in Section 7 and give our conclusions and outlook to future work in Section 8.

2 MOTIVATION

Learning analytics have already been proven beneficial in supporting learners and instructors in education. While it can be sufficient to log which activities a learner performs and use this data to discover the real learning process followed by the learners (Vzquez-Barreiros et al., 2015; Kinnebrew et al., 2013), predict learner's performance (Romero et al., 2013; Fernndez-Delgado et al., 2014) and recommend learning resources (Verbert et al., 2012).

Taking into account also contextual dimensions such as location and device information can further optimize the learning environment for learners and give more insights to instructors.

- **Learners** can be recommended course material based on their location and device. For example, when accessing course material with a smartphone on a train during commuting where there is limited network connectivity, it is not recommended to stream high-quality video lectures. When making exercises in a cafeteria with high noise levels, it is more appropriate to rehearse content instead of introducing new content.
- **Instructors** can monitor students during an examination. Fraud detection services can take into account the location of where a student is sitting, because it is more likely that students sitting next to each other commit fraud. When a student is struggling on an exercise, it can help the instructor to not only provide the name of the student, but also provide information about where exactly the student is located in the room.

To realize these use cases, it is key to capture not only the user their activities, but also in which context these activities were performed.

3 STATE OF THE ART

The first step in capturing and storing learner's data, is to properly model and store the interactions the learner did with respect to the learning objects, e.g., completing an exercise or passing an exam.

A high-level overview of how current learning systems work is depicted in Figure 1. A Learning Management Systems (LMS) serves learning objects to the learner. A learning object is a collection of content items and assessment items that are combined based on a single learning objective. The interactions that the learner has with respect to the learning object are tracked and stored in a Learning Record Store (LRS).

Since 2001, the Shareable Content Object Reference Model (SCORM) (Bohl et al., 2002) has been

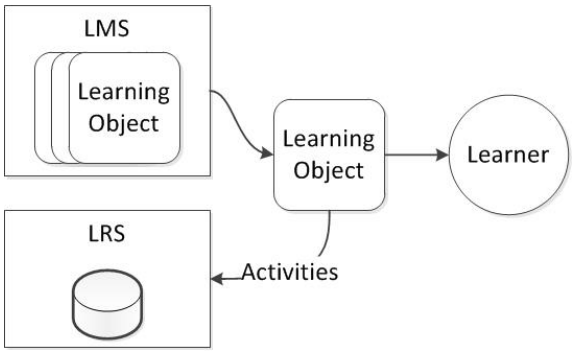

Figure 1: Overview of how current e-learning systems capture learner's activities (De Meester et al., 2015).

the most used learning object standard by the Advanced Distributed Learning initiative (ADL)[3]. Unfortunately, SCORM is limited to storing simple data points such as a final score or that a course has been started or completed. Other standards such as the IEEE standard 1484.11.1/2 (IEEE Std, 2005), Activity Streams (J. Snell, M. Atkins, W. Norris, C. Messina, M. Wilkinson, and R. Dolin, 2015), the Experience API (xAPI) (Experience API Working Group, 2013) and the *Caliper* framework by IMS (IMS Global Learning Consortium et at., 2013), were extensively reviewed by Corbi & Burgos (Corbi and Burgos, 2014). We can conclude that the xAPI standard and IMS Caliper framework already have a notion of context. However this context is related to the course material, e.g., to which curriculum a course belongs or who the instructor of the course is. It does not capture the environment where the user performed the learning or which device he used. The xAPI specification is open source and can be extended.

The xAPI specification, formerly known as Tin Can API, has emerged as a successor of SCORM and captures much richer activity data compared to SCORM. The specification is built on top of the Activity Streams specification, used in social networks to capture user activity in the form of *I - did - this* statements. The xAPI specification added extra functionality to the statements specifically for storing and transferring records of learning experiences. A *result* object is added to better collect outcomes of learning and a *context* object is added to record learning done in context. However, the specification lacks the definition of an extensive context model and merely provides an extendable *extensions* field. With over 160 adopters[4], such as Moodle[5], Blackboard[6] and Sakai[7], it is fair to say that this is now the de facto standard.

(Verbert et al., 2012) already illustrated in a review paper that including context aspects in existing standards, such as SCORM and xAPI, or map standardized representation of context to these standards, is a challenging future line of research to enable data compliant to these standards and specification to be exchanged and reused. Therefore, we extend the xAPI specification with a formal context model using ontologies.

Some initial research has successfully mapped the xAPI specification to an ontology (De Nies et al., 2015; Vidal et al., 2015). The SmartLAK architecture, a big data architecture for supporting learning analytics services (Rabelo et al., 2015) uses an ontology, based on the xAPI specification. Their system is however limited to the xAPI ontology model and only uses the model for data conformance validation. The system presented in this paper uses an extended version of the xAPI ontology model with a context ontology and not only uses the ontologies for validation purposes, but also for reasoning.

4 ONTOLOGY DEFINITION

At the moment of writing, the xAPI specification is only available in a GitHub repository[8]. The specification is not available in a machine-interpretable version and thus not suitable to be used as an OWL ontology or RDF Schema. Lately, we see that the xAPI specification is adopting Semantic Web Technologies. The xAPI Vocabulary Working Group provided IRIs for the Verb (e.g. completed, answered,...) and Activity (assessment, course,...) terms online at https://w3id.org/xapi/adl to improve semantic interoperability. This vocabulary is available as RDFa and can also be retrieved as machine-readable JSON-LD. In the future it is expected that more vocabularies will become available as the ADL working group recently published an xAPI companion guide and vocabulary primer (xAPI Vocabulary Working Group, 2016) for publishing vocabulary datasets as Linked Data (xAPI Vocabulary Working Group, 2015).

To allow us to extend the xAPI context extensions with concepts from a context ontology, an ontology of the xAPI specification is needed.

The statements of xAPI are a good candidate to be described in the Resource Description Framework (RDF) because it is built on top of Activity Streams and the second version of this specification even requires that the statements must be serialized conform JSON-LD (M. Sporny, G. Kellogg, M. Lanthaler (Eds.), and W3C RDF Working Group, 2014). There is a JSON-LD @context definition[9] and an OWL ontology[10] of the specification available online.

Since there is no ontology available from xAPI specification itself, we extend from an ontology of the xAPI described by other researchers (De Nies et al., 2015; Vidal et al., 2015). In section 4.1 we describe this xAPI ontology. We describe how we extended this ontology with our context ontology in section 4.2.

[3]https://www.adlnet.gov/

[4]http://experienceapi.com/adopters

[5]https://www.moodle.org

[6]http://www.blackboard.com

[7]https://sakaiproject.org

[8]http://GitHub.com/adlnet/xAPI-Spec

[9]https://www.w3.org/TR/activitystreams-core/activitystreams2-context.jsonld

[10]https://www.w3.org/TR/activitystreams-vocabulary/activitystreams2.owl

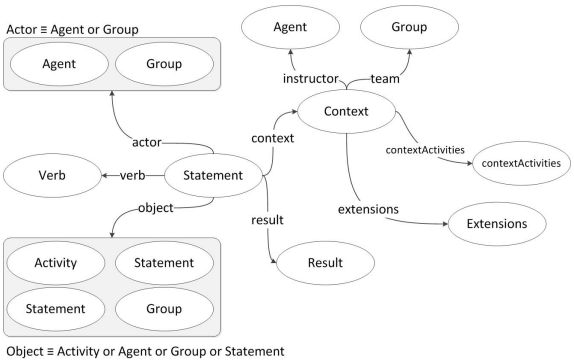

Figure 2: The xAPI ontology.

4.1 xAPI Ontology

The most important classes and relationships of the xAPI ontology are shown in Figure 2. The statement class represents the statements of the form *I did this*, where *I*, *did* and *this*, are the Actor, Verb and Object classes respectively. The statement can also contain the outcome or result and the context associated with the event by using the Result and Context classes.

The Context class has relations to concepts , such as the instructor for an experience, if the experience happened as part of a team activity, or how an experience fits into some broader activity. We can for example state that a learner took some course, under the instruction of a specific instructor, as part of a specific curriculum, in a specific school.

To add more contextual information, the Context class has an extensions relationship that allows relevant domain-specific context. For example, in a flight simulator altitude, airspeed, wind, GPS coordinates might all be relevant.

4.2 Context Ontology

The context its extensions are described by our own context ontology. Figure 3 shows some classes. As it is depicted, our ctx:Extensions class is a subclass of the xAPI Extensions. Two important properties of our context are *location* and *device*. The *location* can be a *Library*, *Home* and *Classroom*. The *Device* can be a *Smartphone*, *PC* or *Tablet*.

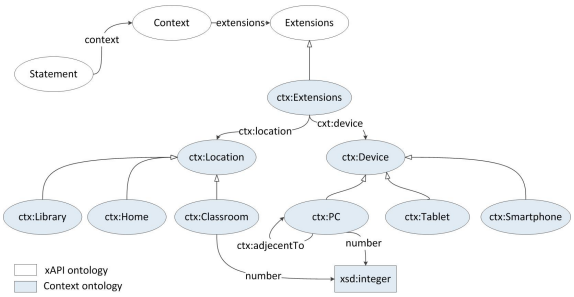

Figure 3: Part of the context ontology.

An illustrative example of how these concepts can be used for the progress and fraud detection services (Section 5 and Section 6) is as follows: the concept of a *Classroom*, *PC* and relationship *number* which numbers all the *PC* and *Room* individuals can be used to filter learners by examination room and detect at which device the slacking student is located. The *PC* has a property *adjecentTo* to model which devices are located next to each other, this can help the algorithm to detect fraud.

5 ARCHITECTURE

Once the ontology has been designed, the architecture can be built to process data, map it to our ontology and define different services to support learners and instructors. Figure 4 depicts an overview of the architecture of our system.

The **Learning Record Store** contains 4 components:

- **Data Collection & Transformation** component is responsible for capturing the xAPI JSON statements that are sent to our system through the API. The component then processes these JSON statements, transforms the statements to RDF and finally forwards the transformed RDF statements to the Storage In component.
- **Storage In** efficiently stores the xAPI RDF statements coming from the Data Collection & Transformation component.
- **Triple Store** is responsible for storing the OWL 2 ontologies and RDF data. The triple store supports reasoning to infer knowledge not explicitly stated in the data.
- **Storage Out** forms the link between the triple store and the different services. The component facilitates access to the data by query answering.

The **Services** are independent components that process the LRS data to provide valuable information to learners and instructors. Each service has an endpoint that allows the service to be consumed by applications or other services. The current version of our architecture contains services that are specially designed to support instructors during class room assignments or examinations. These services are the following:

- **Progress Service** measures the progress from learners during an examination or exercises. Capturing the location where the learner is sitting in a room aids instructors identifying where struggling learners are sitting and help them more proactively.

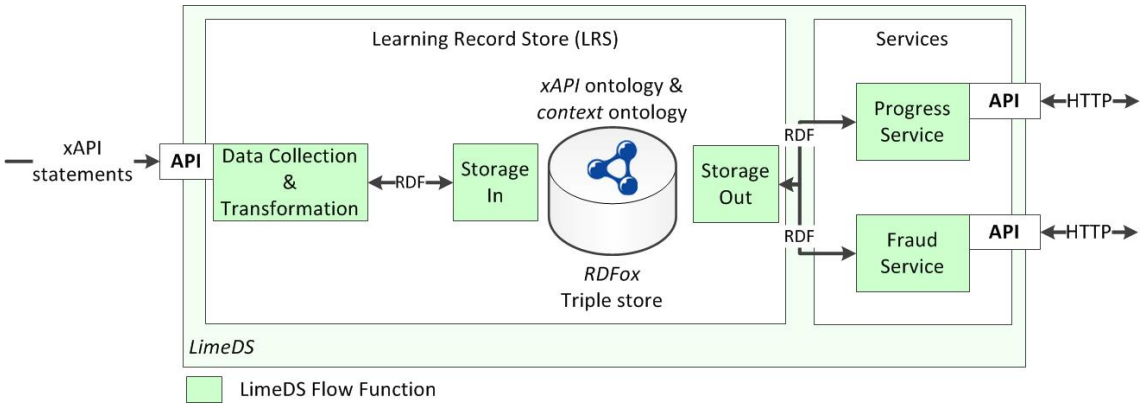

Figure 4: The architecture of our system.

- **Fraud service** detects anomalous behavior from learners during an examination. Taking into account context information such as where learners are located in the room can help detecting anomalous behavior.

6 IMPLEMENTATION

Our architecture is built on top of LimeDS[11], an OSGi-based framework for building REST/JSON-based server applications (Verstichel et al., 2015). LimeDS allows us to adapt our architecture more efficiently for future demands, e.g. adding additional services or plug in different triple store implementations. LimeDS also provides support for load-balancing and data caching so that our system is scalable.

LimeDS facilitates development by leveraging on the Data Flow abstraction. The Data Flow abstraction defines how data flows through the different components in the architecture. This Data Flow system revolves around two conceptual component types:

- **Flow Functions** consume an optional JSON argument and can optionally produce processed JSON data
- **Flow Processes** that execute a specific set of actions when triggered (e.g. time-based).

By combining instances of these two conceptual types, complex data-oriented services can be built. In our current architecture implementation, only Flow Functions were used. The different Flow Functions are depicted in Figure 4.

[11]http://limeds.intec.ugent.be

6.1 Data Collection & Transformation

The data in our LRS must be in a format that maps to the xAPI and context OWL 2 ontologies. Therefore, we have to transform our data to RDF data. Since xAPI statements are JSON documents, converting these document to JSON-LD is the most straightforward approach. Additionally, the JSON-LD documents can then in the future also be stored in other data stores that are not aware of Linked Data, e.g., MongoDB.

Converting a JSON document to JSON-LD is done using a JSON-LD context document[12], which maps all the terms that may occur in the xAPI JSON statement to IRIs in the ontology. This is illustrated in Example 1. A *@context* entry referencing this document is added to the root of the JSON, an *@type* entry with value *xapi:Statement*, as well as the following snipped to every *:verb* and *:object* property: *"@context: { "id": "@id"}*, because the *id* is reserved for the URI. Because the context document introduced in other research (De Nies et al., 2015) does not include any context extensions, we extended this document to map context extensions to IRIs from our own context ontology.

Next to some additional necessary conventions to have a smooth conversion, mentioned in (De Nies et al., 2015), the XML Schema duration (xsd:duration) datatype used in xAPI statements is discouraged in RDF and OWL[13]. The XQuery and XSLT Working Groups have proposed an alternative in the form of two derived datatypes: *xdt:yearMonthDuration* and *xdt:dayTimeDuration*. The derived datatypes restrict the lexical representation to contain only year and month components

[12]http://www.w3.org/TR/json-ld/#the-context

[13]https://www.w3.org/TR/swbp-xsch-datatypes/#section-duration

or days, hours, minutes and seconds components. Since learning experiences will generally not take longer than days, we used the xdt:dayTimeDuration datatype.

6.2 Triple Store

The Learning Record store is composed out of two LimeDS Flow Functions: *Data Collection & Transformation* and *Storage Out* and a triple store that stores the xAPI ontology, the context ontology and the processed xAPI statement triples.

We choose RDFox as our triple store implementation. RDFox is a highly scalable RDF store that supports materialisation-based parallel datalog reasoning and SPARQL query answering (Nenov et al., 2015).

6.3 Storage Out

Storage Out provides a HTTP REST API that accepts SPARQL queries through HTTP POST. The service returns an array of RDF triples. An example of a JSON message is given below.

```
{
 "query":"SELECT ?x WHERE{?x rdf:type :Result}"
}
```

```
{
  "@context": "http://http://users.ugent.be
/~joanseeu/xapi.jsonld",
  "@type": "http://users.ugent.be
/~joanseeu/Statement",
  "actor": {
    "name": "John Doe",
    "objectType": "Agent"
  },
  "verb": {
     "@context": { "id" : "@id" },
     "id": "xapi-verbs:completed",
     "display": { "en": "completed" }
  },
  "object": {
     "@context": { "id" : "@id" },
     "id": "http://example.org/exercise1",
     "objectType": "Activity",
     "definition": {
       "name": { "en": "Example Activity" }
     }
   },
 "result": {
    "score": { scaled: 0.7 },
    "duration": "PT50S"
  },
  "context" {
    "extensions": {
      "room": { "number": 209 },
      "device": { "number": 5 }
    }
  }
}
```

Figure 5: An example xAPI JSON-LD document.

6.4 Services

Different services have been developed, mainly to provide teachers added value from the data available in the LRS. Each service is contained in a LimeDS Flow Function. The following services are implemented:

Progress Service

Progress service returns a list of places in the room where learners take longer than average to complete their exercises/assessments are, enabling teachers to help students more proactively. The captured device identifier as context can be used to locate where the student is sitting in a room (e.g. the PC number). An illustrative example of a query that this service uses is shown below. This service sends the query to the Storage Out component.

```
SELECT ?pcnumber
WHERE {
  ?result rdf:type :Result.
  ?statement :result ?result.
  ?statement :context ?context.
  ?context ctx:device ?pc.
  ?pc ctx:number ?pcnumber.
  ?result :duration ?d FILTER(?d > x)
}
```

Fraud Service

The fraud service queries on the progress learners make and at which device they are located. This is similar to the query that the progress service uses, but without filtering. The service infers potential cheating behavior when learners have similar progress and sit at adjacent devices. A simplified version of the used query is shown below.

```
SELECT ?name1 ?name2
WHERE{
  { SELECT ?name1 ?result1 ?duration1 ?device1 }
  { SELECT ?name2 ?result2 ?duration2 ?device2 }
  ?device1 ctx:adjacentTo ?device2
  FILTER(?result1 = ?result2 &&
    fn:numeric-abs(?duration1-?duration2) < x)
}
```

The list of adjacent devices, described as triples, is loaded in the triple store as background knowledge.

```
@prefix : <http://www.example.org/> .
:device1 :adjacentTo :device2 .
:device2 :adjacentTo :device3 .
```

7 EVALUATION

Since our LRS transforms the xAPI statements to Linked Data, all information in the xAPI statements must also be present as Linked Data. As long as the information in the statements is also defined in the xAPI and context ontologies and the corresponding mapping, we observed no loss in information when transforming the xAPI statements to Linked Data.

The RDFox triple store implementation used in our LRS supports OWL 2 RL, an OWL 2 profile that trades expressive power for the efficiency of reasoning by placing restrictions on the structure of OWL ontologies. Reasoning and query answering can then be solved in time that is polynomial with respect to the size of the ontology.

Furthermore, the triple store implementation seems to be highly-scalable according to the evaluation of the creators (Nenov et al., 2015). The LimeDS framework used to implement the architecture is scalable due to its load balancing support.

The evaluation was done on an Ubuntu 14.04 server with an Intel Xeon CPU E5645 @ 2.40 GHz with 24 GB of memory. Statements were simulated and sent to the LRS to evaluate. The statements described the activity of 100 learners completing 20 exercises with an average time of 30 seconds and standard deviation of 5 seconds per question. The LRS captured and stored 2000 statements. To evaluate the performance of our system, the time for processing the data and executing the services was measured. Since these services support instructors during class room exercises, the services should respond in an acceptable time. Transforming the xAPI statements takes on average 10ms. Figures 6 and 7 plot service execution times, which are both around 160ms when the LRS stores 2000 statements. We can conclude that the execution time of the services scales linear with the number of statements stored in the triple store. Since there is 1 statement per user and per question answered. The system will also scale linearly with the number of questions and number of users.

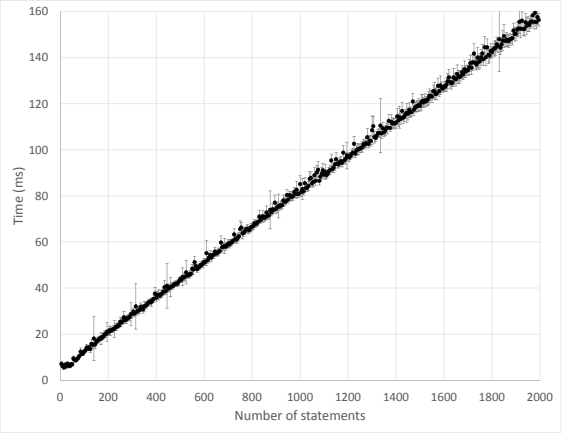

Figure 6: Progress service execution time as a function of the amount of statements stored.

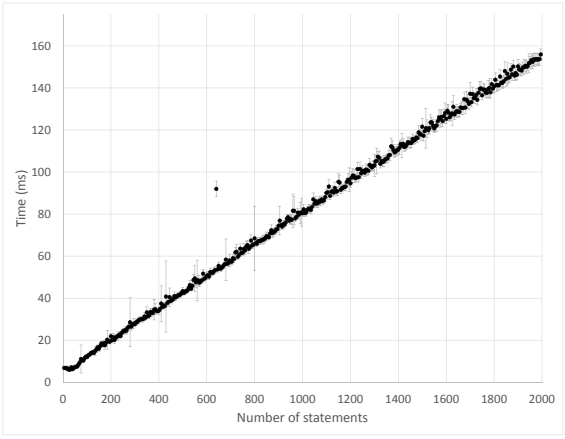

Figure 7: Fraud service execution time as a function of the amount of statements stored.

8 CONCLUSIONS AND NEXT STEPS

In this paper we presented a Learning Record Store that is able to capture more context information compared to current state of the art systems. We extended the xAPI specification using its extensions mechanisms so that learning activity sent to our system is still backward compatible to other systems. We described our extensions using an ontology and used an ontology based on the xAPI specification. We have implemented our system on top of LimeDS and built services that offer learning analytics services that take advantage of context information.

In next steps, our system will be integrated with the learning software of one of the partners in the CAPRADS project. This will allow us to collect real-life data to evaluate how the services behave with regard to false negatives and false positives.

ACKNOWLEDGEMENTS

The iMinds CAPRADS project is co funded by iMinds (Interdisciplinary Institute for Technology), a research institute founded by the Flemish Government with project support of the IWT.

REFERENCES

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer.

Bohl, O., Scheuhase, J., Sengler, R., and Winand, U. (2002). The sharable content object reference model (scorm) - a critical review. In *Computers in Education, 2002. Proceedings. International Conference on*, pages 950–951 vol.2.

Corbi, A. and Burgos, D. (2014). Review of Current Student-Monitoring Techniques used in eLearning-Focused recommender Systems and Learning analytics. The Experience API & LIME model Case Study. *International Journal of Interactive Multimedia and Artificial Intelligence*, 2(7):44–52.

De Meester, B., Ghaem Sigarchian, H., De Nies, T., Verborgh, R., Salliau, F., Mannens, E., and Van de Walle, R. (2015). SERIF: A Semantic ExeRcise Interchange Format. In *Proceedings of the 1st International Workshop on LINKed EDucation*.

De Nies, T., Salliau, F., Verborgh, R., Mannens, E., and Van de Walle, R. (2015). Tincan2prov: Exposing interoperable provenance of learning processes through experience api logs. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 689–694, New York, NY, USA. ACM.

Experience API Working Group (2013). *Experience API. Version 1.0.1*.

Fernndez-Delgado, M., Mucientes, M., Vzquez-Barreiros, B., and Lama, M. (2014). Learning analytics for the prediction of the educational objectives achievement. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pages 1–4.

Fiedler, S. and Väljataga, T. (2010). Personal learning environments: concept or technology?

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220.

IEEE Std (2005). IEEE Standard for Learning Technology - Data Model for Content to Learning Management System Communication. *IEEE Std 1484.11.1-2004*.

IMS Global Learning Consortium et at. (2013). Caliper Learning Analytics Framework. Technical report.

J. Snell, M. Atkins, W. Norris, C. Messina, M. Wilkinson, and R. Dolin (2015). Activity streams 2.0 w3c working draft. W3C Working Draft, W3C. http://www.w3.org/TR/2015/WD-activitystreams-core/.

Kinnebrew, J. S., Loretz, K. M., and Biswas, G. (2013). A contextualized, differential sequence mining method to derive students' learning behavior patterns. *JEDM-Journal of Educational Data Mining*, 5(1):190–219.

M. Sporny, G. Kellogg, M. Lanthaler (Eds.), and W3C RDF Working Group (2014). JSON-LD 1.0: A JSON-based Serialization for Linked Data. W3C Recommendation, W3C.

Nenov, Y., Piro, R., Motik, B., Horrocks, I., Wu, Z., and Banerjee, J. (2015). Rdfox: A highly-scalable rdf store. pages 3–20.

Rabelo, T., Lama, M., Amorim, R. R., and Vidal, J. C. (2015). Smartlak: A big data architecture for supporting learning analytics services. In *Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE*, pages 1–5.

Romero, C., López, M.-I., Luna, J.-M., and Ventura, S. (2013). Predicting students' final performance from participation in on-line discussion forums. *Comput. Educ.*, 68:458–472.

Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., and Duval, E. (2012). Context-aware recommender systems for learning: A survey and future challenges. *IEEE Trans. Learn. Technol.*, 5(4):318–335.

Verstichel, S., Kerckhove, W., Dupont, T., Volckaert, B., Ongenae, F., De Turck, F., and Demeester, P. (2015). Limeds and the trapist project: a case study. In *7e International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, Proceedings*, volume 2 KEOD, pages 501–508.

Vidal, J. C., Rabelo, T., and Lama, M. (2015). Semantic description of the experience api specification. In *2015 IEEE 15th International Conference on Advanced Learning Technologies*, pages 268–269.

Vzquez-Barreiros, B., Mucientes, M., and Lama, M. (2015). Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm. *Information Sciences*, 294:315 – 333. Innovative Applications of Artificial Neural Networks in Engineering.

xAPI Vocabulary Working Group (2015). *Companion Specification for xAPI Vocabularies*.

xAPI Vocabulary Working Group (2016). *Experience xAPI Vocabulary Primer*.