# Descrição semântica de experiência de aprendizagem baseada na especificação xAPI

**Thomas Rabelo[1], Manuel Lama[2], Ricardo Amorim[1,3], Juan C. Vidal[2]**

[1]Faculdade de Ciências Aplicadas e Sociais de Petrolina (FACAPE) - Petrolina, PE – Brasil

[2]Centro de Investigación en Tecnoloxías da Información (CITIUS) - Santiago de Compostela, Espanha

[3]Departamento de Educação
Universidade do Estado da Bahia (UNEB) – Senhor do Bonfim, BA– Brasil

thomas.rabelo@facape.br, manuel.lama@usc.es, amorim.ricardo@gmail.com, juan.vidal@usc.es

***Abstract.*** *In this paper, we present an ontology that formally represents both the data model and the restrictions of the xAPI specification, facilitating the conformance testing of the data coming from the learning management system to guarantee the compliance with this specification. In this work, three main aspects were used to describe this ontology: formalization, reasoning and quering. Finally we demonstrate an real world application of the ontology.*

***Resumo.*** *Neste artigo, apresentamos uma ontologia que representa formalmente o modelo de dados e as restrições da especificação xAPI, facilitando a ensaios de conformidade dos dados provenientes do sistema de gestão de aprendizagem para garantir o cumprimento dessa especificação. Neste trabalho, três aspectos principais foram usados para descrever esta ontologia: formalização, raciocínio e consulta. Finalmente demonstramos uma aplicação "mundo real" da ontologia.*

## 1. Introdução

Em Ambientes Virtuais de Aprendizagem (AVA), diversos aspectos da interação entre alunos e serviços podem ser considerados na avaliação de resultados. Por exemplo: a quantidade de posts propondo tópicos e/ou respondendo a estes na participação em fóruns de discussão, o tempo gasto em leituras de material, a quantidade de acertos e o tempo gasto ao responder a questionários de múltipla escolha, download/upload de materiais, e, também, estratégias de estudo utilizadas pelos alunos tais como as consultas a materiais online fora do AVA, formação de equipes, dentre outras.

No entanto, para isso é necessário que o AVA disponibilize serviços não somente para a captura de dados quanto para a análise dos mesmos, considerando que as interações possíveis entre alunos e ambiente virtual são inúmeras e pode gerar uma imensa quantidade de dados. Para resolver esta situação, surgiu, no início de 2010, o conceito de *Learning Analytics* (LA), entendido como a medição, coleta e análise de dados sobre os estudantes e seu contexto, com o objetivo de compreender e facilitar o

DOI: 10.5753/cbie.wcbie.2015.538

ensino/aprendizagem em AVA [IMS 2013]. Esses serviços fornecem informações valiosas para professores e instrutores, tais como prever o desempenho do aluno, descobrir os caminhos reais de aprendizagem, extrair padrões de comportamento do aluno e assim por diante. Para tal, um dos componentes-chave das arquiteturas software orientadas à *Learning Analytics* [IMS 2013], [Rayon et al, 2014] são sensores para a captura de dados. Esta captura ocorre durante a interação entre alunos e serviços em um AVA, e o respectivo armazenamento de dados em um sistema de registro de aprendizagem, em inglês, *Learning Record System* (LRS) para ser consumido pelos serviços de análise de aprendizagem.

No geral, os AVA se constituem em plataformas que integram componentes com aspectos de funcionalidade distintas onde à interoperabilidade software é crucial. Neste sentido, surgiram, nos últimos anos, várias propostas [del Blanco et al, 2013] relacionadas com interoperabilidade software no contexto de LA. Dentre estas, a especificação API Experience (xAPI) [ADL 2013], anteriormente Tin Can API, emergiu como o padrão de fato em função de possuir um modelo de dados simples e de ter sido adotado por um número significativo de plataformas AVA tais como Moodle, Blackboard, Sakai e outras. A xAPI consiste na descrição de um conjunto de serviços para LRS, a partir de um modelo baseado em declarativas RDF. No entanto, grande parte dos requisitos referentes a tais serviços estão descritos em linguagem natural. Isto dificulta bastante a implementação software, em função de questões diversas, dentre as quais, a possibilidade de interpretações distintas entre programadores é uma das principais. Para este tipo de problema, ontologias têm provado ser muito útil para verificar a consistência do modelo de dados quando os requisitos estão formalmente representados.

Assim, com base nestas questões de interoperabilidade e com o intuito de facilitar a implementação de serviços em plataformas software orientadas a analíticos de aprendizagem, apresentamos, neste artigo, uma proposta de ontologia que representa semanticamente e de forma ampla a especificação xAPI.

O restante do artigo está organizado da seguinte forma: Seção 2 descreve a especificação xAPI, sua estrutura e limitações; Seção 3 descreve a ontologia xAPI proposta, seus modelos semânticos e axiomática. Seção 4 descreve o caso de uso da ontologia para checagem de conformidade dentro da Plataforma de Analítica de Aprendizagem SmartLAK. Finalmente, a Seção 5 descreve as conclusões e trabalhos futuros.

## 2. Especificação xAPI

A Experience API (xAPI), também conhecida como Tin Can API, é uma especificação de metadados orientada à recolha de dados de aprendizagem, de forma consistente, a partir de muitas tecnologias educacionais, tanto online quanto *offline*. O vocabulário simples da xAPI possibilita uma comunicação e compartilhamento de fluxo de atividades de forma segura, entre muitos sistemas diferentes. A xAPI baseia-se no estilo arquitetural REST [ADL 2013], que descreve um conjunto de serviços para interagir com os LRS, e em um modelo baseado em RDF que representa os dados gerados nos AVA através de triplas sujeito-predicado-objeto conhecidas como declarativas.

### 2.1 Estrutura e funcionamento

As interações entre pessoas, conteúdos e outras tecnologias, que ocorrem independente de local e hora, podem ser registradas. Isto é feito a partir de declarativas seguras seguindo um formato típico de declarativas RDF, com "Nome, verbo e objeto" (ou Sujeito, Predicado, Objeto). Quando o registro de uma atividade é necessário, a aplicação software envia uma mensagem, tal e qual, "Eu fiz isto" a um LRS (interno ou independente do AVA) que armazena todos os registros. A XAPI baseia-se em princípios de liberdades: (i) de declarativas sobre qualquer coisa com a estrutura nome, verbo e objeto. Isto permite o registro de pelo menos uma atividade; (ii) histórico, LRSs diferentes podem comunicar-se para compartilhar dados e transcrever um sobre o outro e assim a experiência de um usuário pode seguir de um LRS para outro; (iii) os alunos podem ter seu próprio "arquivo fechado com chave" com suas informações pessoais dentro; (iv) independência de dispositivo. Qualquer dispositivo pode enviar uma declarativa xAPI (celular, game, aplicativo), sem que a uma conexão permanente seja necessária; (v) fluxo livre. O rastreio de um evento de aprendizagem não necessita ter inicio e fim. Pode começar a partir de qualquer ponto ou dispositivo que usuário escolha utilizar.

### 2.2 Limitações

A xAPI também inclui um conjunto de requisitos (ou restrições) que limitam a forma como os dados devem ser representados do ponto de vista dos fornecedores que geram dados do aluno, bem como os LRS onde eles estão armazenados. Os requisitos são descritos em linguagem natural e para representar o modelo de dados, define um formato JSON. Uma das limitações que se destacam na xAPI, é o fato de emissor e receptor terem que implementá-la de forma equivalente para garantir uma comunicação livre de falhas. A linguagem natural utilizada na representação dos requisitos dificulta a implementação da xAPI por desenvolvedores diferentes de forma que sejam equivalentes entre aplicações software distintas (interoperáveis). A mesma é bastante suscetível a erros de interpretação por parte de programadores diferentes. Este é um fator chave que tem dificultado a adoção desta especificação por parte de desenvolvedores software.

Portanto, a busca por formas que simplifiquem a implementação desta especificação e que garantam equivalência entre aplicações distintas, é necessária para melhorar a aceitação da mesma. Neste sentido, o uso de ontologias tem demonstrado ser muito útil para verificar a consistência do modelo de dados quando os requisitos estão formalmente representados [Taamallah and Khemaja 2014].

## 3. Ontologia xAPI

Para este trabalho, três aspectos chave de uma ontologia serviram de base para a descrição semântica, de forma ampla, da especificação xAPI: (i) a formalização de termos e restrições descritas em linguagem natural na xAPI permite garantir a conformidade do modelo de dados de uma aplicação em relação à esta especificação, de

forma que aplicações diferentes, desenvolvidas por programadores diferentes, sejam interoperáveis; (ii) possibilidade de raciocínio sobre experiência de aprendizagem proporcionada pelo formalismo lógico; (iii) declarativas de experiência de aprendizagem definidas em OWL podem ser facilmente armazenadas em um repositório semântico com base em RDF. Com isso, usuários poderiam acessar e consultar dados utilizando a linguagem SPARQL.

No desenvolvimento da ontologia, inicialmente foi estruturado um dicionário de dados e árvore de classificação de conceitos com o intuito de facilitar a definição de uma taxonomia de classes, subclasses e definição de suas propriedades. Os principais conceitos identificados no dicionário foram convertidos facilmente em classes através do conhecimento adquirido a partir da documentação da xAPI. Assim, foi definida uma taxonomia com base nos conceitos de *Statement, Agents, Verbs, Activities*, conforme descrito a partir das figuras 1, 2, 3 e 4 a seguir:

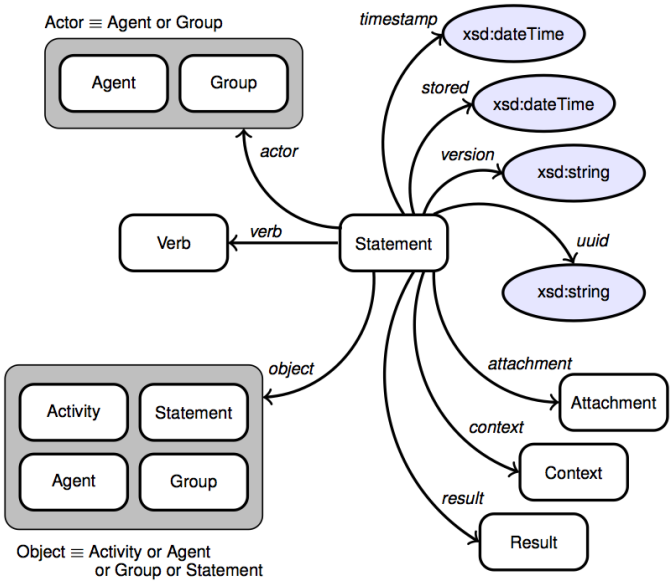


Figura 1. A classe *Statement.*

A classe *Statement* representa o núcleo da especificação xAPI e é utilizada para representar os eventos de aprendizagem em um AVA na forma de declarativas. As declarativas são estruturadas na forma de sujeito-objeto-predicado, conforme as declarativas RDF. Por exemplo: “Eu falei com Mike”; “John escreveu um ensaio sobre surf”. Desta forma, as propriedades da classe Statement são: *Actor*, representa o sujeito que no exemplo é representado por “Eu” e “John”; *Verb*, representa o predicado, “falei com” e “escreveu”; e, *Object*, que pode ser um agente, um grupo, uma atividade ou outra declarativa. No exemplo, “Mike” representa um agente e “um ensaio sobre surfing” é uma atividade. Conforme a Figura 1, as propriedades *Result* e *Context* permitem detalhar os resultados e condições nas quais o evento foi executado. A propriedade *Attachment* foi pensada como forma de permitir a inclusão de artefatos digitais anexados à declarativa e a propriedade *Authority* para identificar o agente grupo afirmando ser verdade a declarativa. As quatro propriedades restantes, UUID, timestamp, stored e version, são propriedades de dados que descrevem o identificador

UUID atribuído à declaração, o momento em que o evento ocorreu, o momento em que foi armazenada nos LRS, e a versão da xAPI, respectivamente.

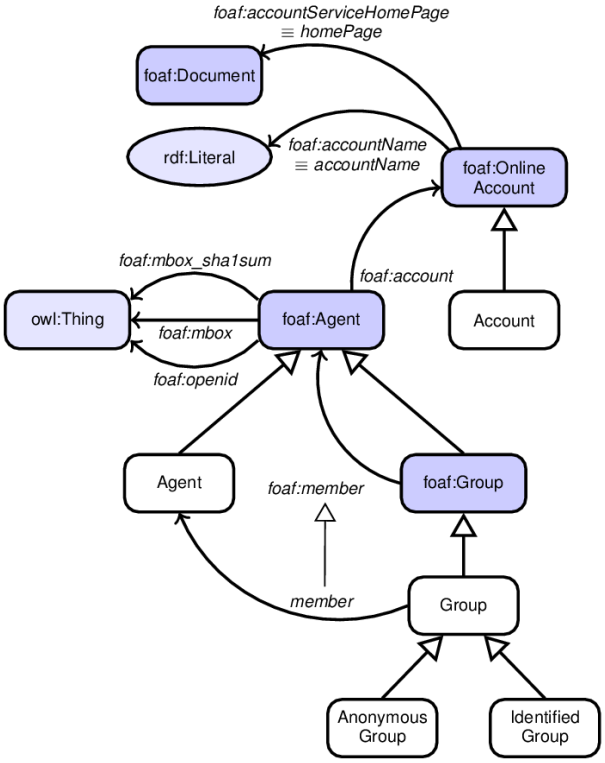


Figura 2. A classe *Agent.*

A Figura 2 mostra uma hierarquia de classes utilizados para definir agentes e grupos. Para descrever esta parte do modelo foi selecionado a ontologia FOAF (um acrônimo de "amigo de um amigo") [Brickley and Miller 2010], que é uma ontologia legível por máquina para descrever pessoas, suas atividades e suas relações com outras pessoas e objetos. Conforme a figura, um agente é uma subclasse de foaf: Agent que é definida em FOAF como uma pessoa, grupo, software ou artefato físico, enquanto que um grupo é uma subclasse de foaf: Group. Embora esta definição pode ser válida para a maioria das aplicações, os agentes xAPI não podem ser grupos, nem os grupos podem ser agentes. Estas duas classes foram definidas como mutuamente disjuntas para evitar esta inconsistência. Os grupos são organizados em duas subclasses, AnonymousGroup e IdentifiedGroup, que também são disjuntas. AnonymousGroup é usado para descrever um grupo de pessoas em que não há identificador para esse conjunto. No sentido contrário, IdentifiedGroup (ex. agentes) é identificado por um identificador de inversa funcional, em inglês, Inverse Functional Identifier (IFI). Especificamente, foaf: mbox, foaf: mbox sha1sum, foaf: OpenID, e foaf:Account são usados para garantir essa identificação.

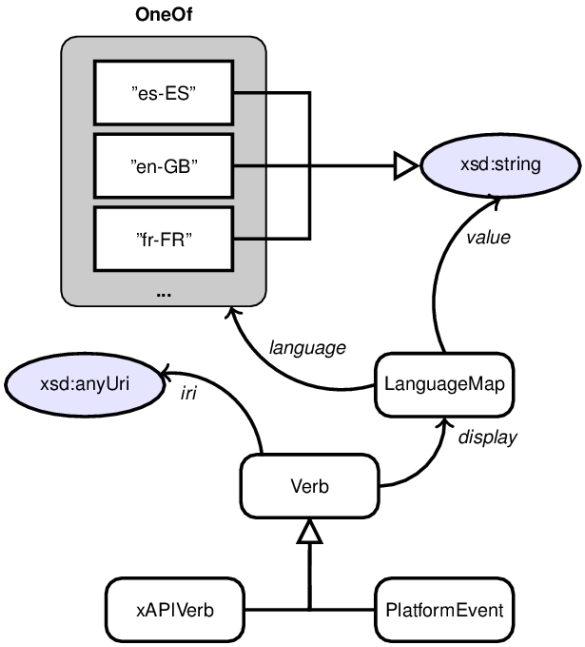


Figura 3. A classe *Verb.*

Verbos representam a ação executada durante a experiência de aprendizagem. O nó raiz dessa hierarquia é a classe *Verb* (Figura 3) que é descrita por dois atributos. Por um lado, todos os verbos têm um único identificador internacionalizado de recursos, em inglês, Internationalized Resource Identifier (IRI) (propriedade iri). Cabe destacar a flexibilidade da propriedade iri. Embora o tipo de dados xsd: anyURI pode ser confuso, suporta o protocolo IRI. Por outro lado, o significado é descrito na propriedade *Display* que contém as representações humano legíveis do verbo em diferentes línguas (LanguageMap) de modo que cada descrição tem uma etiqueta (propriedade de idioma) e o seu valor associado no idioma correspondente. Como está representado na parte superior da Figura 3, o rótulo é restrito a um valor local, tal como, "es-ES" "fr-FR", ou "pt-PT".

Atividades são necessárias para representar a parte objeto de declarações tais como "John fez o exame" ou "Mike postou um texto sobre o Egito Antigo" (Figura 4). Esta parte da ontologia é composta de três classes que representam as atividades, o seu modelo de interação, e as respostas corretas para essas atividades. Por um lado, a classe *Activity* representa o conceito de atividade que é identificado por um único IRI. Como está representado na Figura 4, as atividades têm um nome; pode ter várias descrições (Propriedade *descripton*. Esta propriedade é implementada da mesma forma como a propriedade de verbos *display*) por isso é possível, por exemplo, descrevê-la em diferentes idiomas; tem algumas extensões que são basicamente um mapa de propriedades com seus valores correspondentes que complementam a atividade, conforme necessário; e pode ter um link para informações adicionais (propriedade moreInfo) que deve conter uma descrição mais detalhada ou até mesmo uma forma de iniciar a atividade.

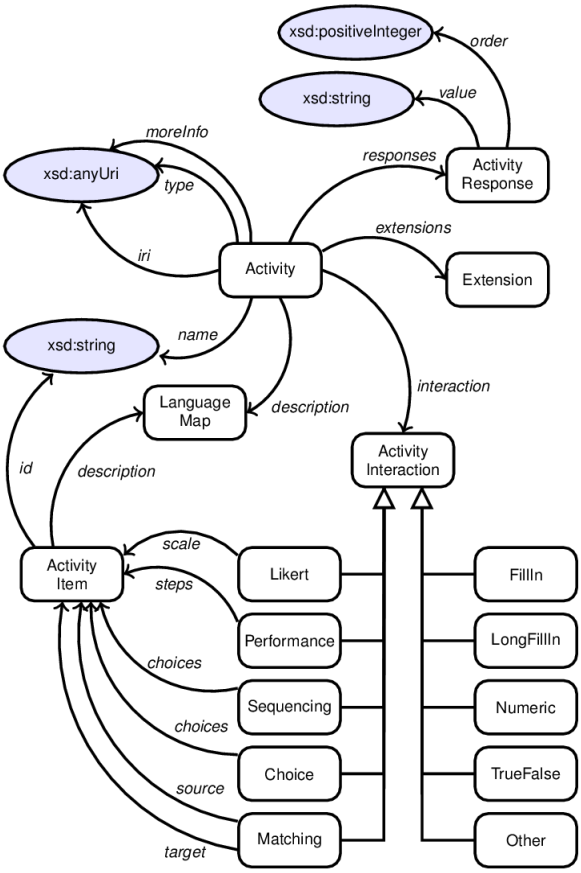


Figura 4. A classe *Activity*.

Uma parte importante da definição de *Activity* vem através de sua propriedade interação que, em certo sentido, capta o comportamento da atividade. Como está representado na parte inferior da Figura 4, a classe *ActivityInteraction* detalha esta parte do modelo que inclui um conjunto limitado de subclasses, cada um representando uma estrutura de interação diferente, as quais, até o momento estão limitadas a dez construtos diferentes definidos a partir do modelo de dados de SCORM 2004. Cada subclasse herda os campos ID e descrição que aponta para um mapa de linguagem. Os identificadores do tipo de interação são emprestados do *runtime* de SCORM 2004 e deve ser exclusivo.

### 3.1 Axiomas

Uma série de axiomas foi definida para representar os requisitos descritos em linguagem natural na xAPI. Por exemplo, há diversos axiomas para verificar a semântica do conceito na declarativa. Os axiomas foram criados a partir de uma reestruturação de partes do texto da xAPI, que foram considerados relevantes, em um modo semiformal, a partir do qual uma codificação em lógica descritiva (SWRL) foi obtida (Figura 5).

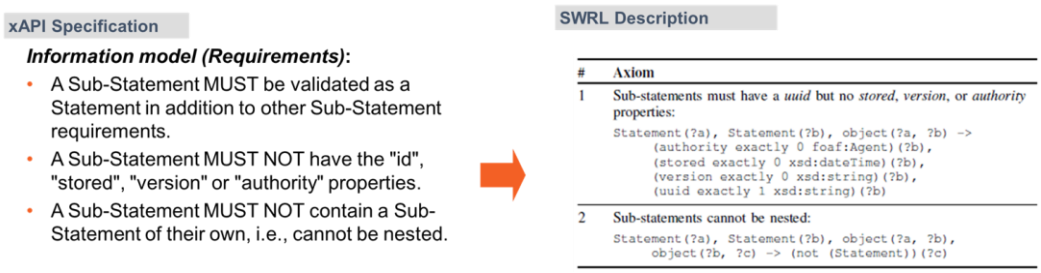


Figura 5. Exemplos de axiomas e o processo de definição.

Conforme a figura, o primeiro axioma lida com o caso de declarativas desempenhando o papel de sub-declarativas. O outro axioma é para verificar que sub-declarativas não podem ser aninhadas, isto é, uma sub-declarativa não pode conter outra declarativa na parte objeto. Isto significa que as declarativas têm apenas um nível de composição.

## 4. Caso de uso: Teste de conformidade xAPI

O teste de conformidade compreendido no âmbito deste trabalho, é um teste para determinar se um produto ou sistema ou apenas um meio, está em conformidade com os requisitos de uma especificação, contrato ou regulamento [Smythe 2006].

O teste tem como objetivo verificar se uma instrução é compatível com a especificação xAPI ou não. Para isto, utiliza-se Pellet [Sirin et al, 2007], um motor de inferência baseado em lógica descritiva, para verificar se uma declaração que vem da camada de serviço xAPI a conformidade de ambos os axiomas e regras da ontologia, significando que as restrições da especificação xAPI sejam cumpridas. Nesse caso, a instrução será armazenada na base de dados LRS, garantindo que os dados são compatíveis com a especificação xAPI. Para o teste, foi utilizado SmartLAK [Rabelo 2015], uma plataforma de ensino online orientada à analíticos de aprendizagem, que utiliza a ontologia (Figura 6).

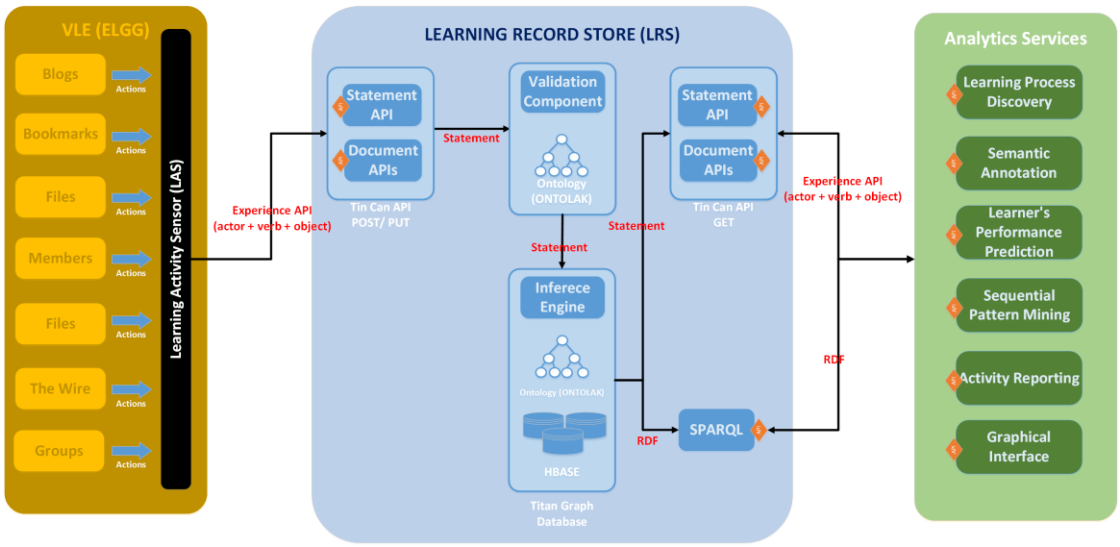


Figura 6. Caso de Uso com a plataforma SmartLAK.

A figura 6 apresenta a dinâmica de interação dos diversos componentes de SmartLAK. O teste de conformidade realiza-se no contexto das interações em um processo de ensino e aprendizagem com a plataforma, conforme os seguintes passos:

1. Instrumentação e sensoriamento das Atividades de Aprendizagem. Tipicamente os usuários geram dados durante uma interação com o AVA, os quais são capturados em tempo real pelos sensores, e enviados como declarativas xAPI (ator + verbo + objeto).
2. A representação da declarativa xAPI é traduzido para uma linguagem de descrição de ontologias como OWL.
3. Pellet ou outro motor de inferência pode ser usado para checar se a declarativa criada pelo autor está em conformidade com os axiomas lógicos da ontologia.
4. Uma vez que o teste de conformidade da declarativa está completa, uma declarativa válida (Statement) é gravado no Repositório Semântico.
5. Serviços Inteligentes podem consumir estes dados usando xAPI REST endpoints ou mesmo o SPARQL endpoint.

## 5. Conclusões e Trabalhos futuros

Este artigo apresentou uma ontologia baseada na especificação xAPI para representar o fluxo de atividade gerado pelos alunos durante o curso, favorecendo a interoperabilidade entre os componentes da arquitetura. Esta ontologia também é utilizada para verificar a conformidade do fluxo de dados com a especificação xAPI, que garante que os dados nos LRS serão compatíveis com esta especificação. Uma vez que esta conformidade é verificada, os dados são armazenados em um banco de dados de alto desempenho que permitem implementar serviços baseados em técnicas de *Big Data*.

Atualmente, estão sendo realizados experimentos de utilização da ontologia para checagem de conformidade de grandes volumes de dados. O primeiro experimento colheu o conjunto de dados público da Tin Can API [Rustici 2015], que no momento do experimento contava com 37.000 registros de Statements.

Como trabalhos futuros, está prevista a criação de um indicador de nível de conformidade como métrica para dizer quão aderente está a especificação. Também, incrementar as relações semânticas de verbos e atividades com outras ontologias standards para melhorar a interoperabilidade.

## Referências

Advanced Distributed Learning (ADL), (2013) “Experience API. Version1.0.1,” http://www.adlnet.gov/wp-content/uploads/2013/10/xAPI v1.0.1-2013-10-01.pdf.

Brickley, D. and Miller, L., (2010) “FOAF vocabulary specification 0.99,” http://xmlns.com/foaf/spec.

del Blanco, A.; Serrano, A.; Freire, M.; Martinez-Ortiz, I.; Fernandez-Manjon, B., (2013) "E-Learning standards and learning analytics. Can data collection be improved by using standard data models?," in *Global Engineering Education Conference (EDUCON), 2013 IEEE* , vol., no., pp.1255-1261, 13-15.

IMS GLOBAL Learning Consortium (IMS), (2013) "Learning measurement for analytics whitepaper," http://imsglobal.org/IMSLearningAnalyticsWP.pdf.

Rabelo, T. A., Penin, M. L., Amorim, R., (2015) SmartLAK: A Big Data-based Architecture for supporting Learning Analytics Services In: Frontiers in Education, El Paso.

Rayon, A.; Guenaga, M.; Nunez, A., (2014) "Ensuring the integrity and interoperability of educational usage and social data through Caliper framework to support competency-assessment," in *Frontiers in Education Conference (FIE), 2014 IEEE* , vol., no., pp.1-9, 22-25.

Rustici Software, LLC, (2015) Public Tin Can API LRS. http://tincanapi.com/public-lrs/.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y., (2007). Pellet: A practical OWL-DL reasoner. *Web Semant.* 5, 2 (June 2007), 51-53.

Smythe, C., (2006) "Initial Investigations into Interoperability Testing of Web Services from their Specification Using the Unified Modelling." *International Workshop on Web Services–Modeling and Testing (WS-MaTe 2006).*

Taamallah, A. and Khemaja, M., (2014) Designing and eXperiencing smart objects based learning scenarios: an approach combining IMS LD, XAPI and IoT. In *Proceedings of the Second International Conference on Technological Ecosystems for Enhancing Multiculturality* (TEEM '14). ACM, New York, NY, USA, 373-379.