Journal of Universal Computer Science, vol. 21, no. 5 (2015), 656-678
submitted: 15/12/14, accepted: 10/3/15, appeared: 1/5/15 © J.UCS

Cloud Interoperability Service Architecture for Education Environments

Rocael Hernández Rizzardini
(Galileo University, Guatemala, Guatemala
roc@galileo.edu)

Abstract: MOOC adoption is growing, and several challenges are presented with it. One of them is the use of innovative tools for learning, with a special emphasis in having learners to represent their acquired knowledge in creative forms; therefore, some experiences in that regard will be introduced. Thus, orchestrating the learning experience with cloud-based external tools (realized as Web 2.0 tools) brings interoperability issues such as automated management of tools and interoperability scalability. This paper presents a new version of an architecture that is capable of interoperability with external tools by defining a semantic description of the tools' Web API using linked data. This creates the next generation of tool interoperability for educational environments. Furthermore, it makes machine discovery of the Web API possible; therefore, it does not require custom system interfaces to interoperate. It simplifies the plugging in of new external tools and maintenance of integrated services. Additionally, the architecture makes it possible to automate simple and complex tasks to be performed with the external tools, such as creating thousands of tool instances to be used by MOOC learners. The results are very promising and demonstrate that this approach is innovative, scalable and highly accurate. Currently, no standard, specification or framework has the same type of flexibility, integration simplicity and robust management for external tools.

Keywords: MOOC, Scalability, Hydra, JSON-LD, Semantic Web, Interoperability, Vocabularies, Cloud Education Environments, Cloud-based tools.
Categories: L.1.3, L.1.4, L.3.0, L.3.6

1 Introduction

Massive open online courses (MOOCs) are a major trend and have been analyzed elsewhere [Siemens 2012] [Hernández et al. 2014a]. The concept brings new challenges and opportunities. One of the characteristics of the MOOCs, and possibly one of the more attractive for the educational and research community, is that they are—as their name implies—massive. Thousands of learners subscribe to them, although high dropout rates have been seen [Chamberlin and Parish 2011] [Guetl et al. 2014] for many reasons. Still, they are massive; compared to a traditional classroom that usually has less than a hundred learners, the MOOCs have thousands of participants. Thus, enabling scalable educational environments is one of the key factors for MOOCs. Cloud computing becomes essential in the technical infrastructure level due to its proven scalability [Armbrust et al. 2010], but innovation in the educational context is also a big challenge in MOOCs. Authors in [Hernández et al. 2014b] have already experimented using learning activities that encourage learners to use mediating artifacts [Conole 2007] as an enhancement to the educational process. Thus the use of innovative tools comes to the scene. Tools such as Web 2.0 tools, also called *Cloud-*

Based Tools (CBTs), can bring further innovation by exposing new and innovative tools that also have user-friendly interfaces. CBTs serve different purposes and have a variety of ways to express acquired knowledge. CBTs for storytelling; visual representation (boards, trees, maps, others); use of audio and multimedia, etc., are already available on the Internet, and many of them have Web APIs available as well, thus opening the path for further interoperability. The CBTs considered in this work run in a cloud-computing environment.

This work presents an innovative architecture for the interoperability of educational systems, such as virtual learning environments (VLEs) and CBTs, to create a truly cloud education environment (CEE) [Hernández et al. 2013b]. The layered architecture takes a semantic approach toward the inclusion of CBTs. This permits it to interoperate with machine-processable CBT Web APIs without custom-made interfaces. The architecture, thanks to its cloud computing nature, is capable of handling thousands of learners.

The rest of the paper is structured as follows: in section 2, the main use case for interoperability is introduced. Also a brief overview of current standards and architectures is given that focuses on the relevant ones for this research and elaborates their current capabilities and limitations. For section 3, a semantic approach for interoperability is presented, and how this approach overcomes the limitations exposed in the previous section is described. In section 4, a cloud-based tool example is elaborated using the sematic approach. The main contribution of this work, the cloud interoperability service layered architecture, is in section 5. Section 6 presents the cloud learning activities' orchestration environment and its architecture. Then, the use of that environment within MOOCs is covered in section 7, while the discussion, early results from the evaluation prototype, and main advantages and possible drawbacks are presented in section 8. In section 9, the conclusions and future work are discussed.

2 System Interoperability for Educational Settings

The interoperability use case that this research is presented below. For this use case were selected two sample *Cloud-Based Tools (CBTs)*, which are mind maps and online document editor, although there are many other type of CBTs that could be used, ranging from storytelling, spreadsheets, content publication tools, and many other specialized tools for specific knowledge domains such as project management, wire frames, strategic planning tools, etc. The selected CBTs for this use case aims to typify a simple yet elaborated scenario, and later the focus is on automatically manage the CBTs through their interoperability Web API. The use case is as follows: the teacher can manually or automatically manipulate several CBTs when building a learning activity and during its deployment. For example, the teacher designs a learning activity in which she may want to create a document in Google Drive [Google Drive 2014]; the document is a template to be completed by the learners on a certain course topic. Next, the teacher asks the learners to create a mind map in MindMeister [MindMeister 2014] that represents the knowledge compiled in the document they just completed. The teacher may even choose to use CBTs in general (rather than a specific brand) for the documents and mind maps; this way, the VLE may have more than one CBT available for documents and mind maps, and the learner chooses which one to use. In a MOOC environment, where thousands of

learners participate, having a CEE with CBTs that are scalable through cloud computing is essential. In the examples, were automatically created thousands of Google Drive documents and MindMeister mind maps, share write permissions; and, once the deadline is over, change that permission to read-only for the learners. This scenario will enable many enhancements for orchestrating learning activities.

Currently there are many standards or specifications for educational systems, many of them within the IMS Global Learning Consortium [IMS 2014]. Some of the most used [Olmedilla et. al 2006], [Aroyo 2006] are: learning object interoperability framework (LORI), content object repository discovery and resolution architecture (CORDRA), Edutella or learning tools interoperability (IMS LTI).

Concerning the use case, IMS LTI v.2 specification [IMS 2010] is quite relevant because it enables tool consumers (TC), such as a VLE, and tool providers (TP), such as any CBT, to exchange information. It uses OAuth for authentication. It basically allows us to launch a TP from a TC environment. Before that happen, an initial negotiated interface contract between both TP and TC must happen. The contract includes information exchange details and the capabilities that both make available. IMS LTI v.2 uses JSON-LD [JSONLD 2014] for payload and expects REST-based requests. As identified in [Hernández et al. 2014d], the defined use case could not be supported by IMS LTI v.2 because it requires defining operations (e.g., CRUD operations) over resources (e.g., maps, ideas, users, documents). Thus the need to provide a more sophisticated interoperability means is clear.

On the other hand, there are architectures that support the integration of many tools, such as CBT. A good example is GLUE! [Alario-Hoyos et al. 2013] , an architecture for the integration of external tools in VLE. It provides an architecture that is capable of creating, configuring and assigning external tool instances; using the external tool instances; updating the users who share control over the external tool instances; and, finally, deleting the external tool instances. Additionally, the referenced GLUE! implementation of the architecture defines that there is a GLUE! core that handles all communication back and forth between the VLE and the external tool and processes the integration contracts. Those contracts are represented and materialized as adapters for both the VLE and the external tools. By using GLUE! to interoperate a VLE with an external tool, first an API interface, known as a VLE adapter, has to be created and programmed to connect the VLE to the GLUE!. Then, each new external tool that is integrated has to be programmed an new tool adapter that communicates between the external tool API and the GLUE!. This is a clean approach to interoperating VLE with external tools that allows a single intermediary to handle all communication logic in a standardized way. However, it requires manually developing and maintaining the tool adapters, which involves custom programming. Also, GLUE! does not support operations (e.g., CRUD), nor does it have the notion of resources and related properties. As pointed out, current specifications and architectures that aim to solve interoperability in an educational environment, lack the possibility to clearly define resources, operations, properties, etc. Furthermore, they do not use current semantic technologies that are capable of enabling machine-processable definitions of Web APIs, which will simplify interoperability efforts.

Issues like as hierarchy and control problems, roles management, authority over resources created, etc., still remain in the previously presented solutions (GLUE,

ROLE, IMS LTI v.2, others). Along with the maintenance costs associated with standard Web contracts that require to manually write code for each new CBT to be integrated.

In the following sections, it is presented the CIS architecture for interoperability that tackles both issues; i.e., semantic definitions for machine-to-machine processing and dealing with self described Web APIs.

3 A Semantic Approach Using Linked Data For Interoperability

To enable interoperability between systems that are capable of performing operations (e.g., CRUD operations), it is necessary to create customized interfaces for each tool to be integrated. That is the current approach in most interoperability systems that use traditional Web services, REST or others. In the use case, each new CBT to be incorporated in the VLE requires a custom API interface, which requires a significant programing effort, as well as a maintenance effort when changes and updates—which happen quite frequently—take place in the CBT Web API. A new enhanced solution has been presented in [Hernández et al. 2014c], [Hernández et al 2014d], which proposes a CBT Web API that uses JSON-LD [JSONLD 2014] and Hydra [HYDRA 2014]. This will allow exposure of the operations and properties of a given CBT; then, an educational environment, such as VLE, will be able to consume and manage those operations and properties.

Traditional REST services lack of the use of sematic Web technologies for interaction, which hinders the possibility of having Web APIs that have a machine-processable syntax. Hypermedia-Driven Web APIs (Hydra), a W3C community workgroup [W3C 2012], defined an innovative solution to this issue [Lanthaler 2013]: "a definition of a number of concepts, such as collections, commonly used in Web APIs and provides a vocabulary that can be used to describe the domain application protocol of services." Hydra uses JSON-LD to describe a Web API. This enables the CBT to publish its Web API using Hydra, and it can represent any operations and properties that the CBT may have and need to expose. For the use case, this may include operations such as creating, updating, deleting and publishing a map; creating, updating and deleting an idea; giving permissions over singular resources, such as a map for other users; etc.

Using Hydra also reduces the interoperability effort through the use of a generic description of an application domain type (e.g., mind map tool, online document editor tool, etc.). This is achieved by representing common operations and properties in a generic vocabulary. Following the use case, some of the operations in a mind map tool's generic vocabulary will include creating, updating, deleting and sharing a map, as well as using the same operations for an idea. Also to be included in the generic vocabulary is each resource (e.g., map, idea, user in a mind map tool) and its set of properties. The generic vocabulary avoids duplicity of definitions and enables the TC, such as the VLE, to have a predefined description of the application domain type, which is independent of a TP. This becomes useful in order to administer tools and to which functionalities to enable for learners. For example, the VLE administrator may choose which actions and properties will be available for the user, independent of the real CBT to be used. Then, each TP will use the generic vocabulary and extend it by defining an API documentation, which is also built using Hydra. This permits the

CBT TP to define what it can support, and it may extend to all additional operations and properties that it supports.

The CBT Web API (e.g., MindMeister mind map tool) will create its own API documentation, which will use the generic vocabulary and extend it to describe the specifics of the Web API, including naming conventions, input and ouut data types, and supported operations and properties. This makes it possible to design an engine to inspect the CBT Hydra Web API (i.e., a Web API that uses Hydra vocabulary to describe its resources, properties, operations, etc.) and discover all the available operations at run-time. This single fact provides a way to create a generic engine that will be capable of discovering the Web API without any custom programming. Thus, adding a new CBT to CEE will require no further coding. Once the engine is in place, the next step is to connect it to the VLE and interoperate with the CBT.

JSON-LD and Hydra are selected to support a semantic interoperability approach. JSON itself is a widely used payload format thus is quite straightforward to further adoption of the proposed solution. Many TP already have JSON payload, then those TP are able to adopt JSON-LD without too many changes over their existing Web API. Furthermore RESTful services are highly common in Web APIs, thus supporting them correctly is necessary. Hydra is a small but powerful addition to the current Web APIs to describe CBTs semantically using Linked Data and using a robust REST implementation. This is a powerful mechanism that enables discovery at run time of the Web API. Hydra provides the possibility to create generic descriptions of CBTs. In addition, Hydra also provides de flexibility for the TP extend and support their own set of objects, operations and properties.

Using Hydra vocabulary it is possible to create self-descriptive, hypermedia-driven Web APIs. They can fully use Linked Data expressivity with REST's major benefits, such as scalability, evolution, and loose coupling. Hydra models represent resources, properties, and operations, among other useful classes for describing Web APIs.

As will be detailed in the next sections, the use of JSON-LD and Hydra, presents access to all the features offered by a Web API, thus enables learning orchestration mechanisms that can serve to create enhanced learning activities.

4 Cloud-Based Tools Using Hydra: a Mind Map Example

As described in the presented use case, a mind map tool will be used. Mindmapping is a very useful technique for visualization of ideas and its connections, "a well-known technique used in note taking and is known to encourage learning and studying, mainly used for representing knowledge, concepts and ideas" [Sarker et al. 2008]. A recent work [Hernández et al. 2014d] introduced an analysis of multiple mind map CBTs, selected those with the most mature interoperability features, and developed an ontology that will support a semantic description of the mind map CBT with interoperability as a target that especially focuses on supporting operations as well as representation of its resources, dependencies, properties, etc. Finally, a Hydra generic vocabulary for mind map tools was introduced. For the use case introduced in section 2, it is covered in this publication only the integration of a mind map CBT; thus, the Google Drive document editor is left out of the scope of the publication and has been added to elaborate the learning activity described in the use case.

This generic vocabulary served to develop API documentation for mind map CBTs, which are MindMeister and Cacoo [Cacoo 2014]. Each one will have its own API documentation that uses the generic vocabulary as a base for describing its own Web API. As depicted in Listing 1 and 2, the MindMeister mind map CBT Hydra API documentation is as follows:

```
{
  "@id":"vocab:Idea",
  "@type":"hydra:Class",
  "rdfs:subClassOf":{
     "@id":"mm:Idea"
  },
  "label":"Idea",
  "description":"The final node of a graph 'map'",
  "supportedOperation":[
     {
        "@id":"_:editIdea",
        "@type":"mm:EditIdeaOperation",
        "method":"POST",
        "label":"edit-idea",
        "description":"Edit idea",
        "expects":"vocab:Idea",
        "returns":"vocab:Idea"
     }
  ],
  "supportedProperty":[
     {
        "property":{
           "@id":"vocab:Idea/id",
           "@type":"rdf:Property",
           "domain":"vocab:Idea",
           "range":"http://www.w3.org/2001/XMLSchema#integer",
           "label":"idea-id",
           "description":"Idea identifer"
        },
        "readonly":true,
        "writeonly":false
     },
     {
        "property":{
           "@id":"vocab:Idea/title",
           "@type":"rdf:Property",
           "domain":"vocab:Idea",
           "range":"http://www.w3.org/2001/XMLSchema#string",
           "label":"idea-title",
           "description":"The text that identify an idea."
        },
        "readonly":false,
        "writeonly":true
     },
...
```

Listing 1: MindMeister API Documentation (part of it)

```
    {
       "property":{
          "@id":"vocab:Idea/modifiedAt",
          "@type":"rdf:Property",
          "domain":"vocab:Idea",
          "range":"http://www.w3.org/2001/XMLSchema#datetime",
          "label":"modified-at",
          "description":"Modified at"
       },
       "readonly":true,
       "writeonly":false
    },
...
    {
       "property":{
          "@id":"vocab:Idea/position",
          "@type":"rdf:Property",
          "domain":"voca:Idea",
          "range":"vocab:Position",
          "label":"idea-pos",
          "description":"Idea position"
       },
       "readonly":false,
       "writeonly":true
    },
...
  ]
}
```

Listing 2: MindMeister API Documentation (continue from Listing 1)

5 The Cloud Interoperability Service Architecture

The following Subsections describe the architecture of the CIS created. Including a general overview of the whole architecture and its different layers, selected implementation details are presented.

5.1 Cloud Interoperability Service (CIS), a Multi-Layered Architecture

The original CIS, which was presented in [Hernández et al., 2014b] used CIS service modules, which are built upon specific CBT Web APIs to interoperate with learning activities. Each module is a Symfony2 [Symfony 2014] framework *bundle* that enables the CBT Web API to communicate. The *bundle* uses the original CIS services for authentication, communication, and business logic. This approach has proven to be capable of enabling interoperability, but it has major drawbacks in light of semantic technologies. First, it scales linearly, meaning that for each new CBT to interoperate, it requires a new custom-made bundle; this implies software programming. Another issue is maintenance of *bundles*, which requires human intervention; this happens when the CBT Web API gets an upgrade. Furthermore, it takes time to make the upgrades, which may cause service interruption until the *bundle* is updated. The whole original CIS architecture has been revamped to a semantic technology Hydra approach and is presented in this section, see Figure 1. This new CIS approach eliminates the aforementioned drawbacks of custom

programming and custom maintenance. Thus, integrations of new CBTs to CIS can be fully automatized, this implies no custom effort to make use of CBT.

The new CIS is built upon the Symfony2 framework, which is comprised of a layered architecture. The core layer is the business logic layer, which is capable of processing the Hydra Web API. This layer provides the structure for operations, properties, data types, etc., as well as an embeddable JavaScript API to the template layer, enabling a user interface to the Web API. It uses the authentication layer to manage token-based identification, CBT Web API identity management and others. Finally, the communication layer processes all requests and responses between the CIS and the CBT Hydra Web API. In the following subsections, the layers are introduced.

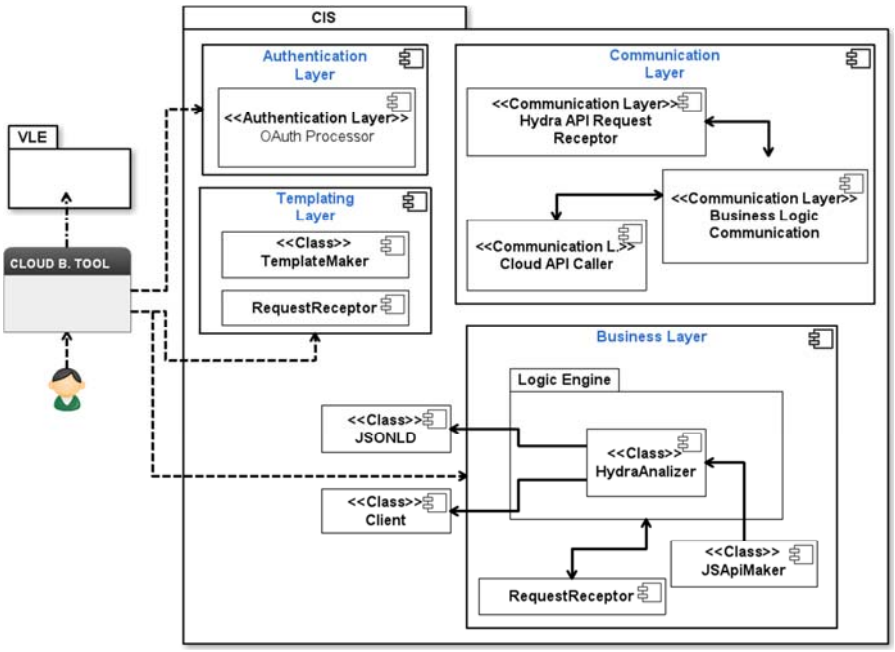

Figure 1: The CIS architecture.

5.2 Business Logic Layer

This is a core layer for the CIS, wherein the Web API sematic definition is processed. Here, the Hydra generic vocabulary and API documentation are processed to automatically discover all operations, properties, etc., of a Hydra Web API. It has three main components: a) JSON-LD Processor; b) Hydra Client; c) Business Logic Engine; d) Embeddable API, as depicted in Figure 2.

The *JSON-LD Processor* [JSONLD 2014] is used to deserialize a valid JSON-LD and then create PHP objects. These objects will contain all information related to that particular resource; for instance, a map or an idea resource, with its corresponding operations, properties, etc. This processor has been released and is available [Lanthaler 2014], [JSONLD 2014]; thus, it is used as a third-party library.

The Hydra client is also available as a Symfony class. It is used to process the Hydra Web API; for example, it is able to get the API documentation URI of a Hydra Web API response. A set of enhancements has recently been done to support the latest Hydra vocabulary (as of June 2014).

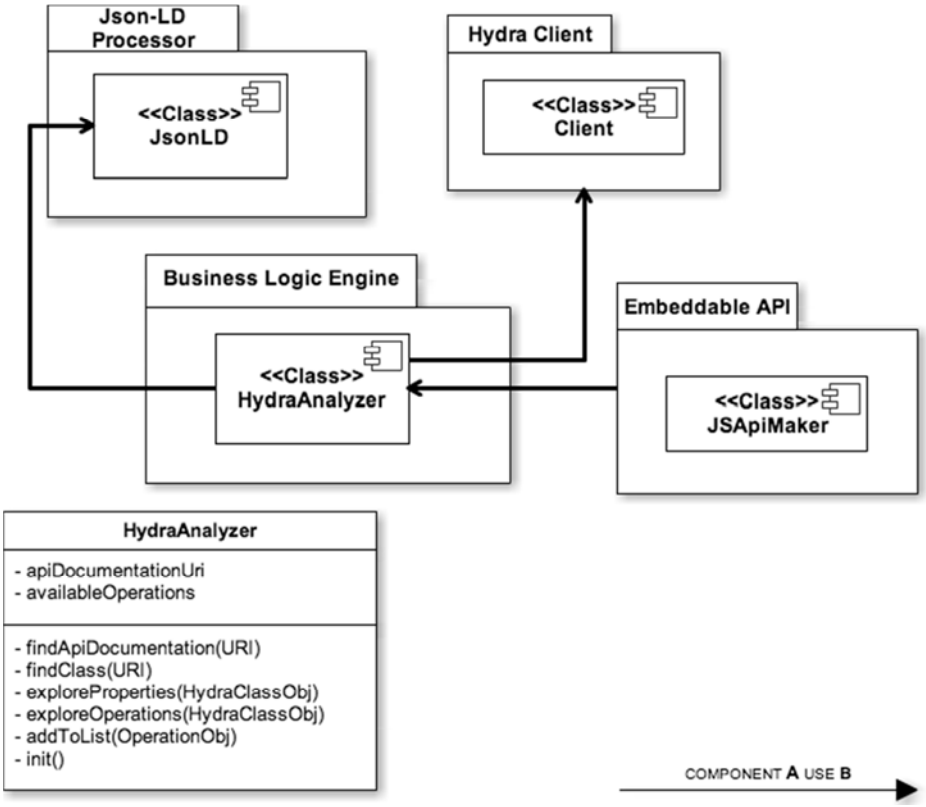

Figure 2: Business logic layer components

Next is the *Business Logic Engine (BLE)*, which is capable of discovering all operations and properties of a CBT Hydra Web API. The main purpose of the BLE is to build an operations list for the CBT Web API. It all starts with the entry point of the Hydra Web API. The BLE will make a request to that entry point, which gives a response in JSON-LD using Hydra. This will contain a directive @context, which will lead the BLE to the API documentation, which contains all the descriptions of the Hydra based Web API. This document is a JSON-LD document as well, and it is used the JSON-LD processor to convert the description into PHP objects; which, in this case, will be available within CIS. The created objects have all the descriptions of the Web API. The JSON-LD processor has built-in methods that allow searches within a JSON-LD document, which enables the BLE to use the results to further explore the document (using its own class, named HydraAnalyzer). At this point, the BLE begins a recursive search of the API documentation classes. It keeps discovering classes, as

well as their properties and related operations (if there are any) until it finds primitive types (such as xsd:integer, xsd:date, etc.). Each newly discovered class will be stored for future reference and use. This process is depicted in Figure 3.

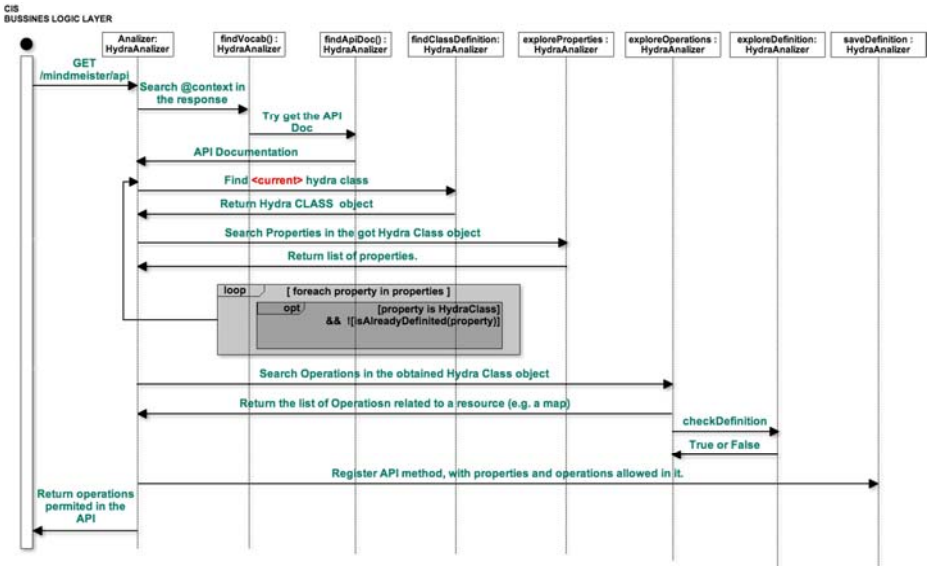

Figure 3: Business logic engine sequence diagram

The *Embeddable API* (see Figure 4) component is used to generate a JavaScript API that represents the Hydra API. It uses JavaScript because it will be able to run this API in a browser, which opens the possibility to embed it. Embedding the API becomes highly relevant in supporting *Functional Widget Interface* [Soylu 2012] and opening the path to embed other systems such as Personal Learning Environments [Friedrich 2011]. With the detailed description of the Hydra Web API that is provided by the BLE, the *Embeddable API* component (Figure 4) builds JavaScript-signed AJAX functions. These functions will enact processes with the communication layer and manage the *expect* and *return* definitions. Ways have also been conceived to add extra pieces of code that also can be embedded for other purposes, such as analytics code that collects relevant behavior information.

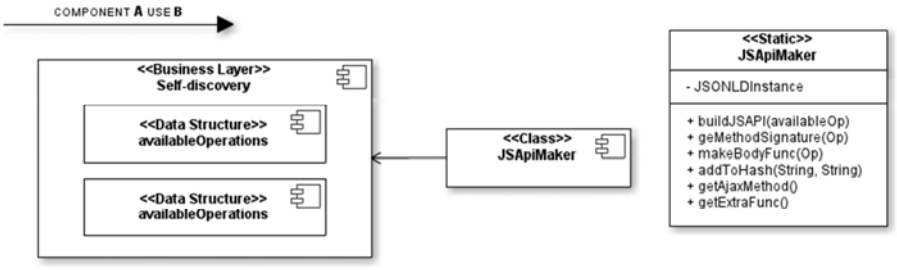

Figure 4: Embeddable API components

The *Embeddable API* module creates embeddable JavaScript that represents the Web API. As an example Listing 1 and 2 depicts part of the *idea* Hydra class. It has an `editIdea` operation, along with some properties such as `id, title, modifiedAt, position`. Notice that `position` is a Hydra class. Second, the *Embeddable API* that is generated is described in Listing 3. It defines a namespace, then it contains the related operations, such as `mindmeister.createIdea`, which uses the properties required by that action; and, when referring to another resource, it will use a JSON object for further description. Then, *`doAjax`* will send related requests to the communication layer.

```
(function (mindmeister, undefined) {
   mindmeister.doAjax = function(hydraUrl, method, hydraClass, params) {
        ...
   };
   ...
   mindmeister.createIdea = function (mapId, title, ... , position) {
        /* mapID and title are primitive type, integer y string
respectively
           position is a JSON object. */
       mindmeister.doAjax(mindmeister.hydraUrl,"POST",mindmeister.apiDoc
.idea,{"mapId": mapId, "title": title, "position": {"x": position.x,
"y": position.y}});
      ...
   }
}) (window.mindmeister= window.mindmeister || {});
```

Listing 3: The generated JavaScript from Embedable API module based on Hydra Web API depicted in Listing 1 and 2.

In Figure 5, the browser client will have the user interface and a HTML form that will use the generated JavaScript, which will call the proper JavaScript functions for each operation and then send requests to the CBT through the communication layer.

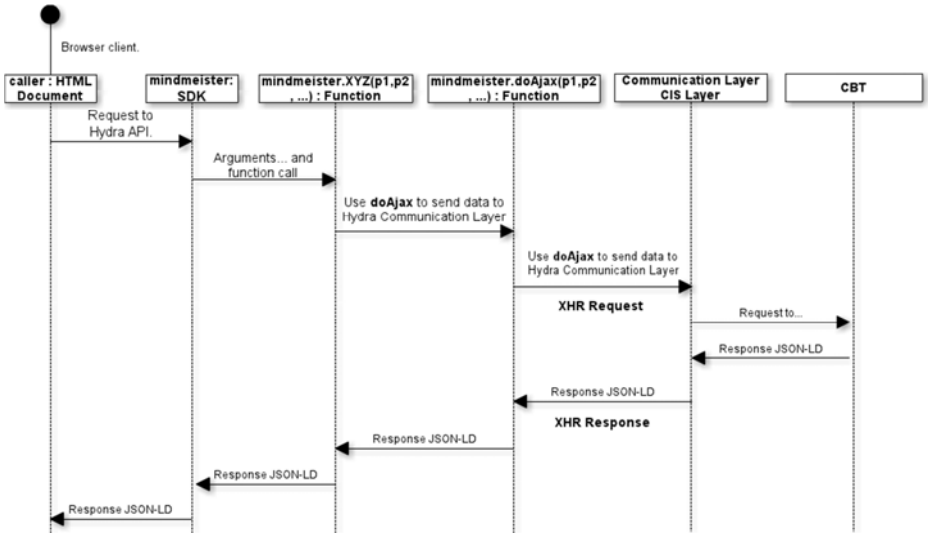

Figure 5: Sequence diagram from the client request to the CBT and the response.

5.3 Authentication Layer

As with the previous version of the CIS architecture, the authentication layer will use a token-based authentication mechanism using OAuth 2.0. The OAuth 2.0 authorization framework "enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf" [IETF 2012]. It has been chosen as the basic authentication because it is a widely used mechanism, and many of the CBTs to be used already support it. OAuth 2.0 has 4 authorization grant types; for the interoperability requirements, the *Authorization Code Grant (ACG)* and the *Implicit Grant (IG)* types are the most suitable for the CIS architecture (the others are for desktop or mobile scenarios). Both will request and get an "access token" that is used to send authenticated requests to the Web API. The access tokens are defined as "credentials used to access protected resources. An access token is a string representing an authorization issued to the client… Tokens represent specific scopes and durations of access, granted by the resource owner, and enforced by the resource server and authorization server" [IETF 2012].

The ACG process starts with a user accessing the CLAO, which will tell the user to login to the CBT (e.g., MindMeister, Google Drive, etc.). Then the CIS will send its `client ID` to the authorization server. After the user is successfully logged in, the user is asked to grant access to the CIS, then is directed to the CIS again, and then the authorization server send an authorization code, to which the CIS responds by sending back the code with a *client secret* and it receives the access token. IG is quite similar to ACG, except that the access token is given back to the CIS once the user has completed the authorization. The main difference between IG and ACG is that IG directly receives the access token and ACG requires an extra step; first, it will get an

authorization code that will be used in a second step to get the access token. Both will ultimately return an access token, which then will be injected into the headers of each REST-based request.

To enable either the ACG or IG, the CIS manages the following data structure: a `hydra_api` that contains the cloud-based tool Hydra API identifier. Then it will define the authorization grant type (e.g., ACG or IG). Regarding the `authorization_uri`, for IG it will return the access token; and, for ACG, it will return an authorization code that will then be used to request the access token. The CIS is modeled to support sending dynamic parameters in this request. The *code_name* is the parameter used for the authorization code for ACG to further request the access token. The *token_uri* that only serves the ACG is the URI to get the access token. The *return_uri* contains the URI to which the user will be returned once the initial authorization has been completed. Finally, the *token_name* has the parameter name that will contain the access token. The CIS, ACG and IG data structure is depicted in Listing 4.

```
hydra_api: 01235,
authorization_type: IG | ACG [a],
authorization_uri: {
    uri: http[s]://www.mindmeister.com/api/auth,
    method: POST | GET,
    params: {
           secret_id: {
               optional: false,
               value: 0123456
           },
           client_id: {
               optional: false,
               value: ABCDEF
           },
           sig: "<?>"
       },
       dynamic_params: {
           "sig": {
               "calculateBy":
        "md5(secret_id${secret_id}client_id${client_id})"
           }
       }
}
code_name: "<code_param_name>",
token_uri: {
   uri: http[s]://www.mindmeister.com/api/auth?method=mm.auth.getToken,
   method: GET,
   params: {
       "code": hydra_api.code_name
   }
},
return_uri: http[s]://miapp.hydra.com/api/auth,
token_name: "<token_param_name>"
```

Listing 4: The CIS ACG and IG JSON data structure for authentication

With this structure, the authentication mechanisms have the following process: first, the user, through the Web interface generated by the template layer, will request the URL to get the authorization using OAuth 2.0. This request is channeled through the communication layer, which sends a request to the cloud-based Hydra API, which will return the URL that will be used for the OAuth 2.0 process. With this URL, it sends a request that is ultimately performed by the authentication layer. This layer,

which builds all the negotiation parameters, is required for the authentication. It will perform the necessary steps for the ACG or IG authentication process; and, once authenticated will register the access token for further request to the Hydra Web API that is done by the CIS. The whole authentication process is depicted in Figure 6.

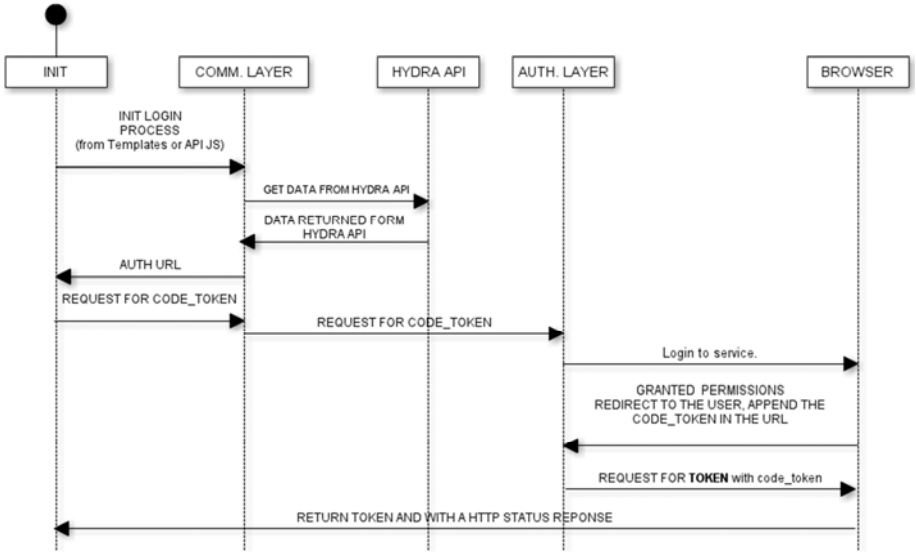

Figure 6: The authentication process for requesting and getting the access token.

5.4 Communication Layer

This layer will act as the communication channel between the CIS architecture and the CBT Hydra Web API. The client will send a request to the Hydra Web API, which only needs to send the URI, the operation and its parameters, and the HTTP method. All this is obtained from the *Embeddable API* layer. This communication layer also helps generate a single request point to the CBT Web API, instead of direct requests from clients, which are something that is likely to be rejected by CBT security policies. Also, this single request channel serves to eliminate the *Same Origin Policy* restriction [W3C Same Origin 2012]. This restriction does not permit a JavaScript to make requests to webpages that are allocated to other domains. Although cross-origin resource sharing allows overcoming that issue, it is still not widely supported by all browsers. JavaScript requests to other servers prove to be useful if it is planned to use the *Embeddable API* in a widget, as has been done in previous versions of the architecture [Hernández et al. 2013b].

In Figure 7 is shown the 3 components of this layer. The first layer is the *Hydra API Request Receptor*, which is in charge of receiving a request from the client (as either the template layer or the business logic layer), and it verifies that the request comes with all required data. Then it is sent to the *Request Constructor and Serializer*, which builds the real Hydra Web API serialized request. The *Cloud API Caller* injects the authentication data provided by the authentication layer, sends the final request to the cloud service and processes its response. The response comes in a JSON format, with the HTTP status code indicating request status.

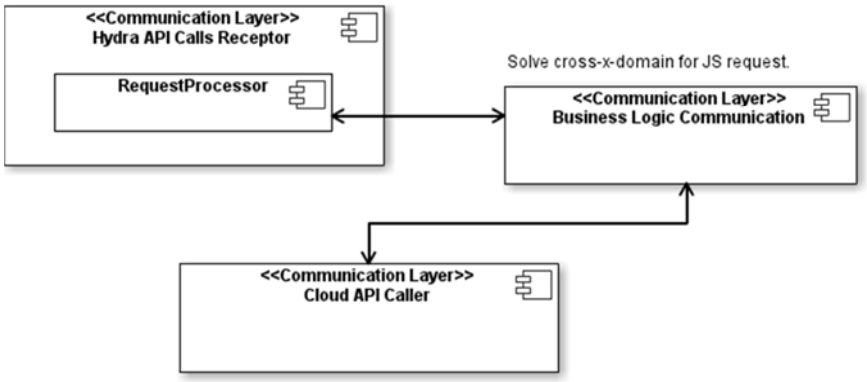

Figure 7: Communication layer modules.

5.5 Template Layer

This layer's objective is to create the HTML-based interface that corresponds to each of the available operations and provides the correct input types. This enables the user (teacher or learner) to manipulate or control these operations from the CLAO. Following the use case of section 2, the teacher might need to give an independent map for each learner; and, through the CLAO, finds an interface to automatically create thousands of maps and assign specific permission for each learner to his or her own map instance. Browser interfaces to do this administrative task on behalf of the teacher will be automatically generated and provided by the template layer.

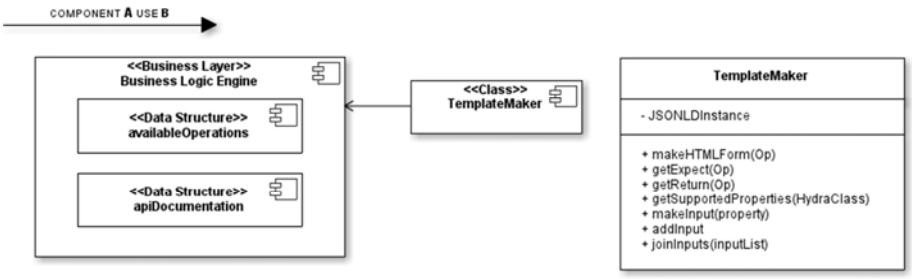

Figure 8: Template layer components and use of business logic layer

The template layer, depicted in Figure 8, uses the business logic layer; more specifically, the BLE, as a base to automatically create Web interfaces for each of the operations with its correspondent properties and input types if they are available. The avid reader may notice that a property that is `readonly` will, therefore, be presented as non-editable. An HTML form will be constructed by each operation, which contains all the related properties needed for that operation. Following Listing 1 and 2, the HTML result is depicted in Listing 5. The HTML form will include both editable and non-editable properties that are configured accordingly. The form generation also

takes into account the use of the specified Web form method (POST or GET) for the operation and uses a form action to call the *Embeddable API* (generated at the business logic layer), which will then send the request to the communication layer.

```
<form method="<[hydra:Operation]method>" action="<[hydra:Url]>">
   <label>Title</label><input name="map.title"></input><br/>
...
   <div>
       <span>POSITION</span>
       <label>X</label><input name="map.position.x"
type="number"></input><br/>
       <label>Y</label><input name="map.position.y"
type="number"></input><br/>
   </div>
</form>
```

Listing 5: HTML automatically generated by the template layer from the CBT Hydra Web API

6 The Cloud Learning Activities Orchestration

The following Section introduces a system that uses the presented CIS. This system is available for teachers and learners, therefore it allows to perform evaluations.

Coordinating interventions from learners across multiple learning activities to enhance productivity is a process known as learning orchestration [Hernández et al 2014b]. The teacher will design learning activities, assessments and rubrics; monitor group or individual activities; distribute workload and set deadlines; etc. As an example, the teacher will design a set of steps to perform a learning activity that might require using a mediating artifact. A mediating artifact definition is "*different tools and resources can provide support and guidance on the context of a learning activity, the choice of pedagogy, the creation of associated learner tasks or any combination of these. They range from contextually rich illustrative examples of good practice (case studies, guidelines, narratives, etc.) to more abstract forms of representation that distil out the 'essences' of good practice (models or patterns)*" [Conole 2007]. In the presented use case the mediating artifact is the document and mind map produced by the learners while they are doing the learning activity.

To create a CEE that is capable to orchestrate learning activities using CBT, a *Cloud Learning Activities Orchestration (CLAO)* has been created [Hernández et al 2014a], which is a system that allows us to perform the aforementioned Learning Orchestration scenarios using CBT. The CLAO provides a unified workspace environment that enables teachers and learners to use CBTs; in addition, it handles communication with CBT, as well as authenticates and provides opportunities to create and orchestrate learning activities. The CLAO consists of the following layered architecture: 1) Learning Activities Orchestrator (LAO), which constitutes the user interaction layer (interface and interaction). 2) Cloud Interoperability System (CIS), which is the core component of the interoperability architecture. It does the interoperability with each of the CBTs used in CLAO. The enhanced version of architecture that is presented in section 5, now using a Hydra approach for interoperability. 3) Learning Environment Connector (LEC), which enables interoperability between the LMS and the CLAO. Further details of the CLAO

architecture are described in [Hernández et al., 2014d]. Finally, it is relevant to note that CLAO uses a cloud-computing infrastructure that utilizes the cloud infrastructure of Amazon's Elastic Compute Cloud (EC2) [Barr 2006].

7 MOOC Experiences Using CLAO

The following Section presents an initial evaluation of the use of the CLAO to demonstrate the CIS usage in a MOOC experience. The CLAO has been used in several MOOC courses given at the Telescope project [Telescope 2014] (an initiative like Coursera or EdX) at Galileo University, Guatemala. The Telescope project is managed by the Galileo Educational System (GES) Department, which does educational technology R&D at the university.

A publication [Hernández et al 2014b] about CLAO and MOOCs describes three courses, with over 6,000 learners enrolled, from more than 15 countries. Courses are taught in Spanish, using the xMOOC pedagogical approach (cognitive-behavioral teaching model). Using CLAO implies using the presented CIS architecture and system in Section 5. The courses use CLAO to deliver about the half of its learning activities and uses CBT, such as the Google Drive Document editor and MindMeister mind map tool. The courses consist of four learning units (one unit per week); each unit includes a common MOOC course content and organization such as short videos, reading content, forums for QA and several learning activities. Other CBT and MOOC experiences at Telescope have been also published [Hernández et al 2014a].

For this evaluation is selected a MOOC that used CLAO, its title is "*Community Manager"*, which gives an introduction about the basic skills needed to perform such a role in social networks. With over nine thousand learners, it had the same structure as mentioned, with the important addition of an induction week at the beginning to get used to a learning environment that includes the use of CLAO and therefore the CIS. It had four learning activities, one for each learning unit. The learning activity required learners to individually use a mediating artifact, which had the following main steps, similar to the presented use case: 1) Read about a the current unit topic with the material provided; 2) create a summary in Google Drive editor; 3) extract the main concepts by representing them in a mind map using MindMeister. The emotional and usability aspects for the CLAO were evaluated with the Computer Emotions Scale (CES) [Kay and Loverock 2008] and the System Usability Scale (SUS), respectively. Regarding the CES, the 12 items of the Computer Emotion Scale describe four different emotions: happiness, sadness, anxiety and anger, as shown in Table 1. A total of 272 participants that successfully finished the course completed the online survey. The findings revealed that MOOC participants perceived low anger and sadness as well as significantly higher happiness while performing learning activities using the cloud-based tools.

Emotion	Explanation	Value
Happiness	When I used the tool, I felt satisfied/excited/curious.	1.79 (σ=0,84) (60%)
Sadness	When I used the tool, I felt disheartened/dispirited.	0.56 (σ=0,84) (19%)
Anxiety	When I used the tool, I felt anxious/insecure/helpless/nervous.	0.59 (σ=0,70) (20%)
Anger	When I used the tool, I felt irritable/frustrated/angry.	0.43 (σ=0,65) (14%)

Table 1: MOOC Computer Emotions Scale 4-point Likert scale (0 to 3) [Kay and Loverock 2008]

Regarding SUS, it shows acceptable results with M=65.35 over 100 (σ=18.58), with a broad range of opinion from 35 to 100. As negative aspects learners mentioned "didn't know how to perform the learning activities, until I learned the platform [CLAO]", "Had troubles sending the assignments"; as positive comments: "I liked the interactive, practical and mastering the [cloud-based] tools", "to get to know the CLAO platform", "I liked to make mind maps, it helps to understand better the concepts", "the platform [CLAO] to make the assignments, gain grades by doing the its learning activities".

One issue found with the use of CLAO in this MOOC is that, for many learners, it was their first time using the MindMeister tool, which currently requires registration. The main concern argued by many learners was their belief that the tool provider required a credit card to create an account. This misled many learners, since the provider gives a free account without requiring a credit card, but it is not communicated as clearly as the paid accounts.

There are two main reasons for using CBT in an educational environment: first, from an educational point of view, the CBT should deliver some kind of innovation in two areas: to present learning content, or to enable the construction of mediating artifacts used by the learner that enhance knowledge acquisition. Second, from a technological point of view, CBT allows: a) scalability, since CBT uses a highly scalable cloud-computing environment; b) vendor and technology innovation, which means that the educational environment can include the best breed of educational tools available, those tools built atop different technologies. This results in vendor and technology independence. These two reasons are magnified in a MOOC environment; from the educational side, these results have already been demonstrated, as previous MOOC experiences presented and referenced in this publication indicate, but also due to the increasing need to include innovation in MOOCs in general in order to advance beyond the standard of xMOOC tools. From the technological side, this is also relevant because system scalability is rather crucial.

8 Discussion

The CIS architecture is able to handle all the requirements for interoperability with a CBT. The CIS uses Hydra to improve the integration of CBTs. That approach does not

requires custom coding for each CBT, since the CBT Hydra Web APIs are self-described, and the business logic layer will be able to understand and utilize all their services and use the communication layer to further interact with the CBT. The authentication issue is solved as well by the CIS, and the template layer of the CIS automatically generates administrative interfaces for the CBT operations.

To enable this scenario, a communication process handler (CPH) [Hernández et al. 2014c] has been created and established to deal with the services that are not currently exposing their Web API using Hydra. In fact, since Hydra is still is a W3C working group, it is not in use by any of these CBT. The CPH allows the conversion communication back and forth between the CIS and the CBT, using Hydra, so it acts as a communication-processing platform to register Web APIs that use JSON or XML payload and return a Hydra response. On the way back, any Hydra request will be transformed properly into the Web API's accepted request types.

The subject use case is fully covered using the CIS architecture. The early prototype has been tested for scalability by creating a set of 3 learning activities that represent that use case, with slight variations between them. In other words, it was simulated that the teacher assigned to the learners a map instance they have to work with. Each map instance will contain a few initial ideas that will be created automatically through the CIS by a definition that was given by the teacher. Once the learners completed the work with the map, they will submit it to the CLAO, and the map will be in read-only mode for the learner to prevent further changes. It was conceived to create more than 10,000 individual resources (map instances), one instance for each learner, since the learner will do each learning activity independently. This means that, for the given MOOC test environment, has been simulated that 10,000 learners will perform the learning activity. At the time they need to begin the activity, the correspondent resource will be automatically created and set up for the current context. This includes the creation of a mind map using the MindMeister CBT and the correspondent permissions given to the learners. This proved to be highly efficient due to the CBT cloud infrastructure, and had no noticeable delay while performing the process when accessing the learning activity resources. The performance results were within the range of 100 to 1000 milliseconds for the following operations of map instance manipulations: creating map instances, adding ideas to maps, editing permissions rights over a map; this performance range is true for the following scalability tests: first over sequential map instance manipulation, second over blocks of 100 simultaneous map instance manipulations, third over blocks of 1000 simultaneous map instance manipulations. The test results indicated non-noticeable performance impact for system users with the different scalability tests. These tests enable real life scalability scenarios, such as having 1000, 100 or multiple sequential access to the map instances, thus simulating learners doing their learning activities. One of the problems that have arisen while using MindMeister is that it provides only 3 map instances with the free account. To overcome that problem, the Galileo main paid account had unlimited map creation, and the CIS only needs to share the map instance created with the main account to a given learner. This gives control over what has been shared with each learner. The ownership of the map instance resource that belongs to the VLE is highly fundamental, as it has been identified in previous studies [Hernández et al. 2014b]. Therefore, the VLE can manage the access control to it, which is quite

important, because it is important to not lose any of the maps and to make such map instances read-only after the assignment deadline.

One current drawback of the architecture is that it does not manage Hydra Web API versions; therefore, it is not possible to automatically determine if the current request is dealing with an API that has been already identified and structured in previous request. As for now, in the prototype, with every new request, the Hydra Web API is processed and then cached in memory, and the cache is cleaned every day to update it to the latest version of the Hydra Web API. A more effective and efficient approach will be to simply create versions of the CBT Hydra Web API; this will allow the CIS to identify and update the cached Hydra Web API information when needed.

Regarding the CIS template layer, it gives to the CLAO environment a user interface structure that is automatically built upon the service definition. This works well, but still needs a custom organizer of operations, resources, etc., to enable better user interfaces and automated administrative processes.

Finally, the CIS authentication layer does a great job simplifying the authentication mechanism using OAuth 2.0. This authentication approach, as well as how to best use the CIS architecture and IMS-LTI v.2 specification, remains an open question. Additionally, with IMS-LTI v.2, the use of the *context* concept to identify a class, course, etc. will be of high value to the CIS architecture.

9 Conclusions and Outlook

Current MOOC challenges regarding educational system interoperability with external tools (CBTs) range from enabling customized management of these tools to the maintenance and costs associated with integrating new CBTs. This paper has described the CIS architecture, which enables a real CEE using CBTs. This innovative approach, which uses a semantic description of a CBT Web API along with linked data with the Hydra vocabulary for Web API descriptions, yields a robust interoperability process. There is no further need to custom-program API interfaces between systems; instead, using linked data makes the Web API discoverable. With a Hydra Web API, the CIS is capable of processing it at run time. In addition to that, the CIS also brings enhanced control to CBTs by interacting with their available operations (e.g., CRUD operations). Neither Web API discoverability nor operations management is yet supported by interoperability frameworks and international specifications.

The prototype evaluation indicated that this architecture scales well for MOOCs. The CIS performs the discoverability of the Hydra Web API efficiently and effectively. The need for an intermediary between the CIS and the CBT Web API is necessary, because tool providers have not yet enabled their Web API to use Hydra. Some improvements will be useful, such as a better orchestration between CBTs within a learning activity; this includes communication between the CBTs themselves. Also, the automatically generated user interface needs to be configurable to present a simpler and more coherent interface to the user (e.g., hiding properties that are not meant to be seen by users, such as a map instance ID). Further work includes providing a user interface for system administrators to verify which are the available operations and configure which of those operations will have a correspondent user interface

control that will be available for the user (teacher or learner) to manipulate when using the CBT.

MOOC learner's experience will benefit by using the CLAO and CIS, by introducing innovative learning activities that use CBT, opening new educational experiences that were not possible before with a VLE monolithic approach. The whole MOOC technical deployment is easier, the inclusion of new CBT is simpler, and the scalability to use and orchestrate those CBT with hundred of thousands of learners is proven to be successful and manageable.

Finally, two interesting paths still need to be followed. First and foremost is how to use the CIS architecture and Hydra vocabularies with the IMS LTI v.2 to make the most of both and avoid duplicate solutions on certain aspects, such as authentication. Second, this architecture opens the door to gather analytics from the e-Learning ecosystem of the CBT, thus providing rich data to further analyze and enhance the learning process.

Acknowledgements

To the great team in the GES department at Galileo University, which enables and actively participates in this research.

References

[Alario-Hoyos et. al 2013] Alario-Hoyos, C., Bote-Lorenzo, M. L., Gómez-Sánchez, E., Asensio-Perez, J. I., Vega-Gorgojo, G., & Ruiz-Calleja, A.: "GLUE!: An architecture for the integration of external tools in Virtual Learning Environments". Computers & Education, 60(1), 122-137.

[Armbrust et al. 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., & Zaharia, M.: "A view of cloud computing". Communications of the ACM, 53(4), 50-58. (2010)

[Aroyo 2006] Aroyo, L., Dolog, P., Houben, G. J., Kravcik, M., Naeve, A., Nilsson, M., & Wild, F.:. "Interoperability in Personalized Adaptive Learning". Journal of Educational Technology & Society, 9(2). (2006)

[Barr 2006] Barr, J., Varia, J., and Wood, M., "Amazon EC2 Beta," Retrieved May 3, 2011, from http://aws.ty- pepad.com/aws/2006/08/amazon_ec2_beta.html, 2006.

[Cacoo 2014] Cacoo website, Online Diagram Software for Flow Chart & UML. Tool available online: https://cacoo.com, Last visit June 2014.

[Chamberlin and Parish 2011] Chamberlin, L., & Parish, T.: "MOOCs: Massive open online courses or massive and often obtuse courses?"; eLearn 2011, 8. DOI=10.1145/2016016.2016017 http://doi.acm.org/10.1145/2016016.2016017 (2011).

[Conole 2007] Conole, G.: "Describing learning activities: tools and resources to guide practice". In H. Beetham & R. Sharpe (Eds.), Rethinking Pedagogy for a Digital Age: Designing and Delivering E-Learning (pp. 81–91). London: RoutledgeFalmer. (2007).

[Friedrich 2011] Friedrich , M., Wolpers , Shen ,M., Ullrich , R., C., Klamma , R., Renzel , D., Richert , A. and von der Heiden , B., "Early Results of Experiments with Responsive Open Learning Environments," Journal of Universal Computer Science, vol. 17, no. 3, pp 451- 471, 2011.

[Google Drive 2014] Google Drive Tool. Available online: http://drive.google.com (Last visit June 2014)

[Guetl et al. 2014] Gütl, C.,, Hernández-Rizzardini, R., Chang, V., & Morales, M.: "MOOC in Latin America: Lessons Learned from Drop-out Learners"; In the 3rd International Workshop on Learning Technology for Education in Cloud (LTEC), Knowledge Management in Organizations, Springer Netherlands, (2014)

[Hernandez 2011] Hernandez R, Guetl C, Amado H, "Facebook for e-moderation: a Latin-American experience", doi: 10.1145/2024288.2024332 S. Proc. 11th I-Know'11, 2011.

[Hernández et al. 2013a] Hernández R., Amado-Salvatierra H. R., Guetl C.: "Cloud-based Learning Environments: Investigating learning activities experiences from Motivation, Usability and Emotional Perspective". In: CSEDU 2013 Proceedings, SciTePress (2013a)

[Hernández et al. 2013b] Hernández Rizzardini, R.., Linares, B. H., Mikroyannidis, A., & Schmitz, H. C. "Cloud services, interoperability and analytics within a ROLE-enabled personal learning environment". Journal of Universal Computer Science, 19(14), 2054-2074. (2013b)

[Hernández et al. 2014a] Hernández Rizzardini, R., Gütl, C., Chang, V., & Morales, M.: "MOOC in Latin America: Implementation and Lessons Learned"; In the 2nd International Workshop on Learning Technology for Education in Cloud (LTEC), Knowledge Management in Organizations, Springer Netherlands, pp. 147-158. (2014a)

[Hernández et al. 2014b] Hernández R, Gütl C & Amado-Salvatierra H.: "Cloud Learning Activities Orchestration for MOOC Environments"; In the 3rd International Workshop on Learning Technology for Education in Cloud (LTEC), Knowledge Management in Organizations, Springer Netherlands, (2014b)

[Hernández et al. 2014c] Hernández R, Gütl C & Amado-Salvatierra H.: "Using JSON-LD and Hydra for Cloud-based tool Interoperability: A working prototype based on a Vocabulary and Communication Process Handler for Mind Map Tools"; In Proceedings of the Ninth European Conference on Technology Enhanced Learning EC-TEL 2014, Springer 2014

[Hernández et al. 2014d] Hernández R, Gütl C & Amado-Salvatierra H.: "Interoperability for Cloud-based Applications for Education Settings Based on JSON-LD and Hydra: Ontology and a Generic Vocabulary for Mind Map Tools"; In Proceedings of the 14th i-Know Conference, ACM 2014

[HYDRA 2014] HYDRA Project code Available online https://github.com/lanthaler/HydraClient (2014)

[IETF 2012] Internet Engineering Task Force (IETF) "The OAuth 2.0 Authorization Framework RFC 5849". Internet Engineering Task Force (IETF) http://tools.ietf.org/html/rfc6749 (2012)

[IMS 2010] IMS Global Learning Consortium. (2010). IMS GLC Learning Tools Interoperability Basic LTI Implementation Guide. (2010)

[IMS 2014a] IMS Global Learning Consortium www.ims.org (2014)

[JSONLD 2014] JSON-LD Project code JSON-LD processor. Last retrieved April 2014. Available online https://github.com/lanthaler/JsonLD (2014)

[Kay & Loverock 2008] Kay, R.H., & Loverock, S. (2008). Assessing emotions related to learning new software: The computer emotion scale. Computers in Human Behavior. 24, 1605-1623.

[Lanthaler and Gütl 2013] Lanthaler, M., & Gütl, C.: "Hydra: A Vocabulary for Hypermedia-Driven Web APIs." In Proceedings of the 6th Workshop on Linked Data on the Web (LDOW2013) at the 22nd International World Wide Web Conference (WWW2013). (2013)

[Lanthaler 2013] Lanthaler, M. "Creating 3rd generation web APIs with hydra". In Proceedings of the 22nd international conference on World Wide Web companion (pp. 35-38). International World Wide Web Conferences Steering Committee. (2013).

[Lanthaler 2014] Lanthaler, M. JSON-HydraClient project. Available online: http://www.markus-lanthaler.com/ (Last visit Jun, 2014)

[MindMeister 2014] MindMeister, Tool available online: http://www.mindmeister.com, Last visit June 2014.

[Olmedilla et al. 2006] Olmedilla, D. Saito, N., & Simon, B. "Interoperability of Educational Systems" – Editorial of Special Issue. Educational Technology & Society, 9(2), 1-3. (2006)

[Sarker et al. 2008] Sarker, B. K., Wallace, P., & Gill, W. "Some observations on mind map and ontology building tools for knowledge management". Ubiquity, 2008(March), 2. (2008)

[Siemens 2012] Siemens, G.: "MOOCs are really a platform"; Elearnspace, last edited July 25, 2012. Accessed 15 Feb 2013, from http://www.elearnspace.org/blog/2012/07/25/moocs-are-really-a-platform/ (2012)

[Soylu 2012] Soylu, A., Mödritscher, F., Wild, F., De Causmaecker, P., and Desmet, P. : Mashups by Orchestration and Widget-based Personal Environments: Key Challenges, Solution Strategies and an Application. Journal paper, Emerald. (2012)

[Symfony 2014] Symfony: High Performance PHP Framework for Web App. SensioLabsNetwork 2014 Available online in: http:// symfony.com (2014)

[Telescope 2014] Telescope, MOOC's. Website Galileo University. Available in http://telescopio.galileo.edu (Last visit Jun, 2014)

[W3C Same Origin 2012] W3C Same Origin Policy, http://www.w3.org/Security/wiki/Same_Origin_Policy, Last visit 11-08-2012.]