An architecture for integrating learning object repository resources into web videoconference services

Aldo Gordillo, Enrique Barra, Daniel Gallego and Juan Quemada
Escuela Técnica Superior de Ingenieros de Telecomunicación
Universidad Politécnica de Madrid
Avenida Complutense 30, 28040, Madrid, Spain
{agordillo, ebarra, dgallego, jquemada}@dit.upm.es

***Abstract*— Reusing Learning Objects saves time and reduce development costs. Hence, achieving their interoperability in multiple contexts is essential when creating a Learning Object Repository. On the other hand, novel web videoconference services are available due to technological advancements. Several benefits can be gained by integrating Learning Objects into these services. For instance, they can allow sharing, co-viewing and synchronized co-browsing of these resources at the same time that provide real time communication. However, several efforts need to be undertaken to achieve the interoperability with these systems. In this paper, we propose a model to integrate the resources of the Learning Object Repositories into web videoconference services. The experience of applying this model in a real e-Learning scenario achieving interoperability with two different web videoconference services is also described.**

***Keywords*— online learning; videoconference; learning objects; interoperability; repository**

I. Introduction

Digital Learning Objects are the building blocks of e-Learning activities. A Digital Learning Object (hereafter just LO) may be as simple as a paragraph of text or an image, or can be a more complex object like a quiz or an educational game. Due to the immensity of learning resources on the Internet, Learning Object Repositories (LOR) play a key role by facilitating their discovery and reuse. Reusing LOs in multiple contexts is essential to enable cost-effective development and foster collaboration between organizations. Hence, achieving their interoperability is one of the main challenges when creating a LOR.

On the other hand, the adoption of novel HTML5 [1] technologies and the growth of cloud computing services are enabling the development of innovative web videoconference services. These new technology solutions provide high quality service at low cost as well as unprecedented easy access since users just need a web browser to establish a real time communication.

These new services can offer huge benefits for distance and blended learning. They allow the sharing, co-viewing and synchronized co-browsing of educational resources at the same time that provide real time communication. However, to take advantage of these benefits, the LOs have to be integrated with the videoconference services.

Basic LOs like a simple image are very easy to integrate, but more complex ones, which can have a high degree of interactivity, need more sophisticated solutions. There are several interoperability standards for LORs, but they are usually more focused on achieving interoperability among them than in increasing the ability of their resources to be reused. On the other hand, some standards have been defined to integrate LOs in external environments such as Learning Management Systems (LMS), but they were not designed with web videoconference services in mind. Therefore it is not possible to take full advantage of the power of these tools using existing standard solutions.

In this paper we propose a novel model to integrate the LOs of a LOR into web videoconference services. We also present the experience of implementing this model in a real e-Learning scenario, in which we achieve the interoperability with two different videoconference services: MashMeTV (http://mashme.tv) and Lynckia (http://lynckia.com).

The rest of the paper is organized as follows. The next section reviews related work of LORs focused on interoperability. Section 3 explains the model. Section 4 presents the scenario in which it has been applied as well as the two use cases. The last section finishes with some concluding remarks together with an outlook on future work.

II. Related Work

The Learning Object Metadata (LOM) standard [2] defines a Learning Object as “any entity, digital or non digital that may be used for learning, education or training”. However, in e-Learning this term only covers digital LOs. The main aim of providing educational resources as LOs is to facilitate reuse [3]. Reusing LOs instead of repeatedly authoring them can lead to several benefits such as savings in time, reducing development cost and even enhancing the quality of digital LOs [4]. A key factor in improving reusability and interoperability of LOs is metadata, where the information of the LO is specified including its structure, properties and educational characteristics.

Usually, LOs are stored, searched and accessed using LORs. A LOR stores both LOs and their metadata, either by storing them physically together or by presenting a combined repository to the outside world [5]. LORs also allow users to search and retrieve LOs. To provide the most possible resources, they take LOs from other LORs using different methods such as federated searches or harvesting. Metadata plays an important role in facilitating the retrieval process since LORs typically support advanced searches based on LOM properties (e.g. subject, target age). The most widespread standard to define the metadata scheme is LOM, although some LORs have implemented its own metadata scheme. LORs also provide several services to users beyond searching and storing such as favorites, recommendations of LOs based on user's history, or private repositories.

Some different interpretations exist for the term interoperability. It is often considered as the capability of one system to communicate and interact with others. But in this context, interoperability can be also defined as the ability of "enabling information that originates in one context to be used in another in ways that are as highly automated as possible" [6]. In this paper, we always refer to interoperability according to the second definition. Then, a crucial factor of a LOR is to provide interoperable LOs. Or in other words, to provide LOs that can be integrated automatically with LMSs and third party services. Several approaches exist to achieve interoperability with different systems. For instance, many LORs export the resources to standard e-Learning formats such as SCORM [7] or AICC (Aviation Industry Computer-Based Training Committee) to reuse LOs in LMSs. Lastly, we must take into account that not all integrations are equally strong. Some approaches (e.g. SCORM) allow communication between the LOs and the context in which they are running while others do not. Furthermore, in the context of videoconference services, regarding the achieved interoperability, the LOs can be *shared* (i.e. participants see their own isolated instance of the LO), *co-viewed* (i.e. all participants see the same LO) or *co-browsed* (i.e. all participants can also realize actions over the co-viewed LO synchronously). For instance, the Bridgit video conferencing tool [8] achieves LO interoperability by using computer desktop sharing. Therefore the LO can be co-viewed since all participants are seeing the same LO at the same time, but co-browsing is not possible since the only participant who can interact with the LO is the one who is currently sharing the computer desktop.

III. MODEL

This model has been designed with Web Videoconferences Services (hereafter WVSs) in mind. Its main aim is to satisfy the main uses of a scenario where a WVS is used for distance and blended learning: sharing, co-viewing and synchronous co-browsing of LOs.

The designed solution is based on delivering all LOs as web contents attaching a lightweight component that provides an API (Application Programming Interface). Enabling this way the communication between the context in which the LOs will be used and themselves. Fig. 1 illustrates this idea that will be explained in detail along the next sections.

A. *Delivering interoperable Learning Objects*

The LOR provides a main function: Request/Deliver LO. This function is in charge of attending the requests of LOs and delivering them enabling their reuse in the way that the interoperability can be achieved as highly automated as possible. Several steps are carried out to attend a LO request:

1. If the LO is not in a web-ready format it should be adapted. This adaption can be done when the LO is stored in the LOR or on the fly in a less efficient way. Hence, if an image is requested, the LOR will deliver a simple HTML page with the image embedded in it.

2. Besides the LO, a lightweight JavaScript API (hereafter LO API) will be attached to the HTML page in order to detect the events triggered as a response of the user's actions. This step makes no sense for non-interactive resources (e.g. an image) and then in those cases can be avoided. However, if an interactive LO is requested like an HTML5 video, the API allows detecting when a user plays or stops the video.

3. Finally, the HTML page will be delivered to the application or website that requested the LO. This application will integrate the provided LO using an HTML element called iframe.

B. *Introducing the iframe element*

Before continuing with the model explanation, we present in this section a brief summary of the iframe element. An iframe is an HTML element that is used to insert or integrate other HTML documents or resources inside a website. For instance, a YouTube video can be embedded in a website using an iframe. The iframe element was standardized by the W3C (World Wide Web Consortium) and is supported in all modern browsers. Other kind of HTML elements can be used to embed web content in a website (e.g. embed or object), but the iframe is the only one that allows bidirectional communication between the website and the embedded object in a standardized way due to HTML5 capabilities. Following with the previous example, by using the YouTube API is possible to detect when the user clicks on the play button or play the video automatically without the user intervention.

C. *Delivering non-standard web resources*

In section 3A we saw that by using an API it is possible to detect the events of standardized elements such as an HTML5 video. However, this approach is not valid for non-standard web resources since they have its own particular actions. For example, the actions that a user can perform in a slideshow (e.g. advance slides) are quite different that the actions he/she can do in an educational game (e.g. jump or shoot). As an example, Slideshare (http://www.slideshare.net) provides an API to interact with their presentations that specify six actions: *jumpTo*, *next*, *previous*, *first*, *last* and *getCurrentSlide*.

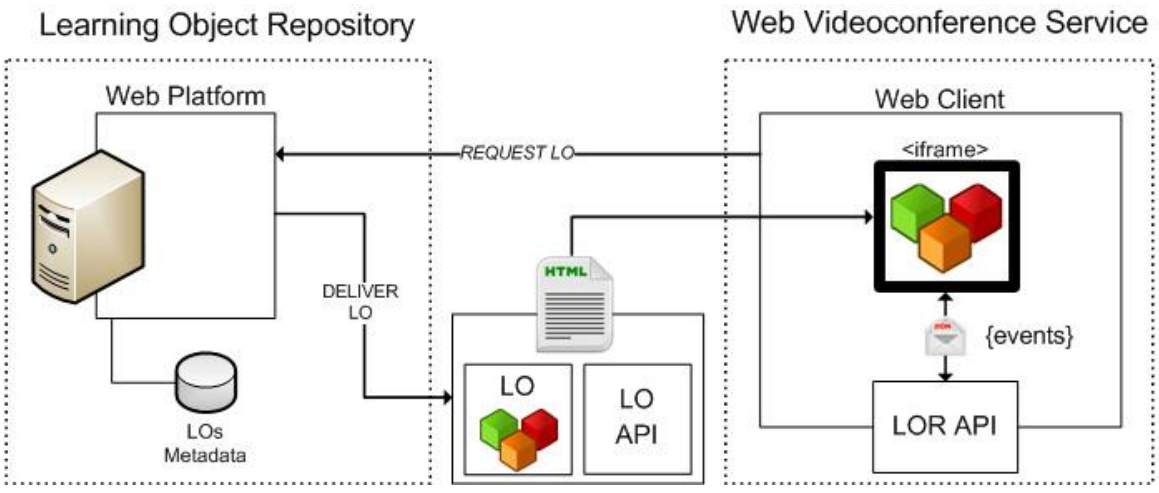

Fig. 1. LOR architecture to achieve interoperability with web videoconferences services

Our solution proposes to define each action and its corresponding triggered event in the LO metadata. Moreover, the LO needs to implement a JavaScript file that extends the LO API adding the handlers for this actions and the listeners for the events. Hence, if we want to make interoperable a custom slideshow following this model, we may define the action "jumpTo" in order to go to a specific slide, and the event "onSlideEnter" triggered when the user enters in a new slide. This way, if a user goes to the slide 5, the onSlideEnter event will be triggered with the parameter 5. On the other hand, if the jumpTo method is invoked with the parameter 5 the slideshow will advance to the slide 5. The key idea relies on the fact that in both cases, the final state of the slideshow is the same. Hence, this way non-standard LOs can extend the LO API including its particular actions and events, allowing this way their integration into WVS.

D. *Delivering Learning Object events*

In the previous sections we have explained how to detect the user actions for standard and non-standard web resources as well as how to replicate these actions based on the events triggered. In this section, we are going to explain how these actions are delivered in a WVS environment. First off, the WVS should include the API of the LOR (hereafter LOR API) the same way as it includes third-party APIs from external services like YouTube or Slideshare to integrate their resources (i.e. videos and slideshows). Fig. 1 also illustrates this fact. This API includes two functions: *onMessage* to receive the messages from the LO API and *sendMessage* to send them. These methods enable communication between the LOR API and the iframe in which the LO is inserted. This way a bidirectional communication channel is built between the LO and the web videoconference client (hereafter just client) through the LOR API.

Fig. 2 represents through an example the different actions that are carried out to deliver the LO events. In this example, Alice and Bob are participating in a videoconference session talking and co-browsing a slideshow presentation. They are commenting the sixth slide. In a certain moment, Alice clicks on a button to go to the seventh slide.

1. The Alice's slideshow (i.e. the slideshow that Alice is seeing) advance to slide 7. As a response an event "onSlideEnter" (previously defined in the LO) is triggered and handled by the LO API. The LO API will compose a message including all relevant information about the event. In other words, all the information needed to replicate the event. At least, the *eventType* should be specified. In this case, *eventType* is *onSlideEnter* and a "slideNumber" parameter with a value of 7 is also included. Lastly, the LO API notifies the LOR API.

2. The LOR API receives the message through its *onMessage* method and delivers it to the client, which will send it to the Messaging Server of the WVS.

3. The Messaging Server do not have to read or understand the message generated by the LO, it just have to broadcast the message to the rest of the participants (in this case just Bob) through the network using its usual data channel.

4. After the broadcasting, the Bob's client receives the message and sends it to the LO API using the *sendMessage* method provided by the LOR API.

5. When the LO receives the message, it composes the original event and replies the action. In this case, the LO will replied the Alice's action by calling the "jumpTo" method passing the *slideNumber* 7 as a parameter. As a consequence, the final state of Alice's and Bob's slideshows is the same.

This is an example of co-viewing, but considering that Bob can also perform actions in the slideshow in the same way Alice does, co-browsing is also provided.

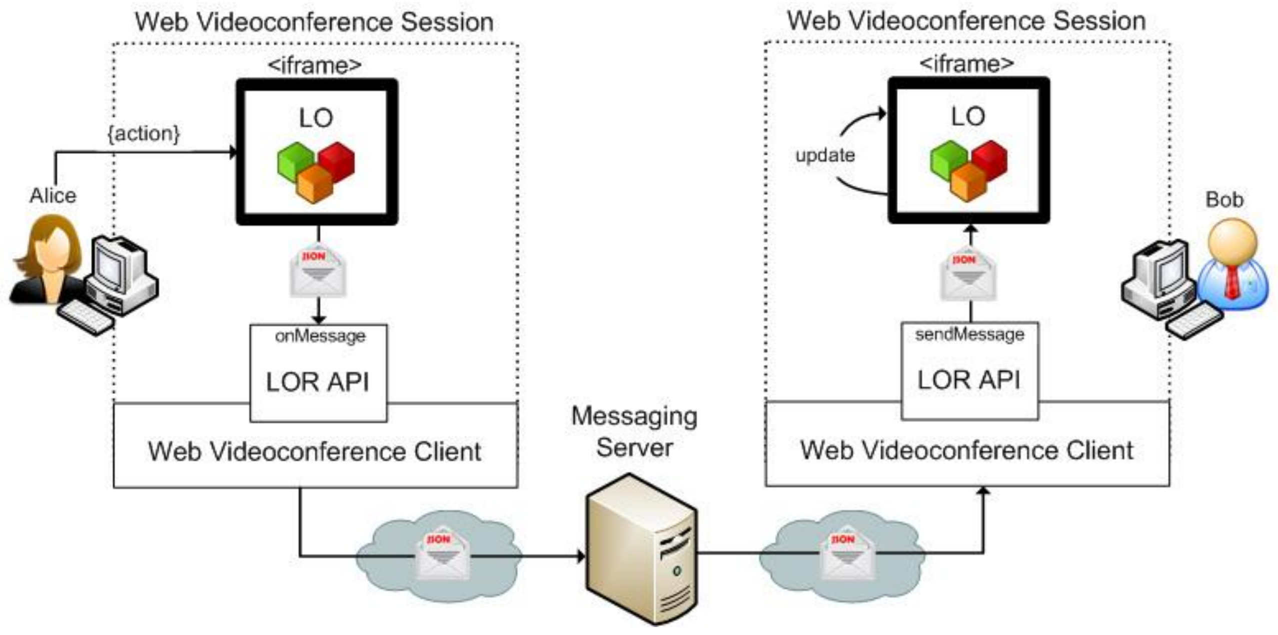
Web Videoconference Session
<iframe>
LO
{action}
Alice
JSON
onMessage
LOR API
Web Videoconference Client
Messaging
Server
Web Videoconference Session
<iframe>
LO
update
Bob
JSON
sendMessage
LOR API
Web Videoconference Client

Fig. 2. Delivering Learning Object events

E. *Achieving synchronous co-browsing*

Previous section shows an example of co-browsing, however it is not synchronous. Considering again the initial situation of the previous example where the slideshow is in the sixth slide. If Alice clicks on the next button and at the same time Bob clicks on the previous button, Alice notifies an *onSlideEnter 7* event while Bob notifies an *onSlideEnter 5* event. This way, the final state will be different: Alice will be in the slide 5 since she receives the Bob's event, while Bob will be in the slide 7 due to the Alice's event. This problem is called "state synchronization" and is one of the biggest concerns in the development of real time applications that requires synchronization such as multiplayer games. In the field of games, it has been defined as "the problem of maintaining the same game state information on each of the players' instance of the game and generating the actions of each player belonging to the same game instance [9]. This definition is perfectly valid in our context, replacing game state for application state and players for participants. After all, a LO can be an educational game.

This model has been designed with this concern in mind. To deal with it, a new operation mode is provided. If a WVS wants to use this mode, it has to initialize the LOR API including a special option called *preventDefaults*. In this mode, each time a user tries to perform an action, instead of allowing it to do the action and then notify it to the rest of the participants, the action will be prevented but the notification will be sent. Then, the WVS has to broadcast the event, not only to the rest of the participants but also to the user who performs the broadcasted event. This way, all of the participants will receive the event approximately at the same time depending on their latencies. Latency it is the most critical network aspect related to the performance of online multi-user applications (e.g. multiplayer games) [10]. In the previous example where Alice and Bob click a button at the same time, with this mode, the final state of the slideshow will be the same or not regarding the latencies. Hence, this is a first important step but is not enough. Fortunately, game developers have been invented several mechanisms to try to avoid latency such as dead reckoning [10] and lockstep synchronization [11] that can be used in this context.

In our case, by applying a particular implementation of a lockstep event-locking mechanism we can achieve synchronous co-browsing.

In a lockstep mechanism, the server distributes the application status periodically. During each period or time slot, the clients can send their information to the server, but the messages that arrive too late will be considered lost. Hence, the state is updated every period. A particular implementation of this mechanism is called event-locking: it specifies that a client can advance to the next time slot only when all other clients are also ready to advance. This is a state synchronization method since involves that each client send the state of its instance to the rest. However, we are using an input synchronization method, where each client sends all their events.

The model proposes to use the lockstep event-locking mechanism adapted to input synchronization: send and broadcast the events instead of send the full state of the LO, which is much more complex. This way, the server just has to deal with the "event collisions": when two incompatible events are received in the same time slot. A straightforward approach may consists of allow just one event per time slot and in case of collision choose one randomly.

This implementation has some limitations, for example, new participants (that enter in the videoconference session when several actions have been performed over the LO) will not be able to compose the full LO state. So, a last improvement can be also be made. The server can store all of the actions performed over the LO, and deliver them to the new participants that join later.

Finally, back again to the example where Alice and Bob click a button at the same time but considering now these new improvements, the process will occur in the following way:

1. The Alice's and Bob's slideshows remain static but the events "onSlideEnter" are triggered. One message is generated per event.
2. The two messages are delivered to their corresponding clients and sent to the Messaging Server.
3. As the two messages will arrive in the same time slot, the Messaging Server choose one randomly, for example, the Alice's message, and broadcast it to all participants (Alice and Bob).
4. When the message arrives, each client sends it to their corresponding LOs.
5. The Alice's action is performed in both slideshows. As a consequence, the final state of Alice's and Bob's slideshows is the same again.

Without synchronous mode, the Messaging Server is only in charge of the message broadcasting. It does not have to parse or understand the messages. In some occasions, a WVS may be interested in encrypting messages, but is not mandatory. With the synchronized co-browsed approach, some logic needs to be implemented in the Messaging Server and hence it must be done by the WVS. The LOR can provide facilities but a high level of interoperability requires cooperation to be achieved.

IV. ViSH: A Case Study

The GLOBAL excursion (Extended Curriculum for Science Infrastructure Online) project [12] is a European project which main aim is to enrich science teaching in European schools. Via a central web portal, called the Virtual Science Hub (ViSH), GLOBAL excursion provides scientists, teachers and their pupils a package of activities, materials and tools for enabling the integration of e-Infrastructures into school curricula. The ViSH platform includes a social network, an e-Learning authoring tool and a LOR that stores all the LOs created and uploaded by the users and institutions.

The authoring tool allows ViSH users to create a novel LO called Virtual Excursion [13], [14]. Virtual Excursions are presented as rich interactive slideshows. They can contain diverse resources: multimedia files, e-Infrastructure resources such as a webcam o a remote pendulum, web games or flashcards among others. Flashcards are resources presented as an background image with several "hot zones" identified by arrows where the user can touch and see additional contents that the teacher has previously tagged [15]. The e-Infrastructure resources are provided by scientific institutions and allow students to virtually visit different infrastructures such as research laboratories or natural parks. However, sometimes the guidance of an expert is required to enjoy these resources. For example, a scientist is required to control a microscope and explain the concepts to the students. The solution selected in the GLOBAL excursion project to connect scientist and classrooms was to share the Virtual Excursions in a web videoconference service by applying this model to the ViSH LOR.

A. *MashMeTV*

In this service people can share many resources such as Slideshare or PDF presentations, blackboards, maps and YouTube videos. All of the elements are synchronized with the room (i.e. conference session) ensuring that all participants see the same at the same time. For this reason, MashMeTV was selected as the most appropriate WVS to co-browse synchronously the ViSH LOR resources. A new button was added in all of the excursions of the ViSH platform to init a MashMeTV videoconference session sharing this resource. This way, all participants can explore synchronously the Virtual Excursions and at the same time communicate in real time among them. As an example, Fig.3 shows a flashcard shared in a MashMeTV session.

The integration was performed successfully, with a little difference with respect to the presented model. MashMeTV provides its own API to allow third party web applications to use functionalities like message delivering. In order to take advantage of this API, we modified slightly the model introducing an extra step in the Request/Deliver LO function described in Section 3A. Before deliver the web page with the embedded LO, the LOR (i.e. ViSH) embeds the entire HTML page in an iframe which will act as a gateway between MashMeTV and the LO. We call this method the "Iframe Gateway". Its operation is represented in Fig.4. The LO stills communicate with the LOR API. But in this occasion, the LOR API will deliver the messages to the iframeGateway,

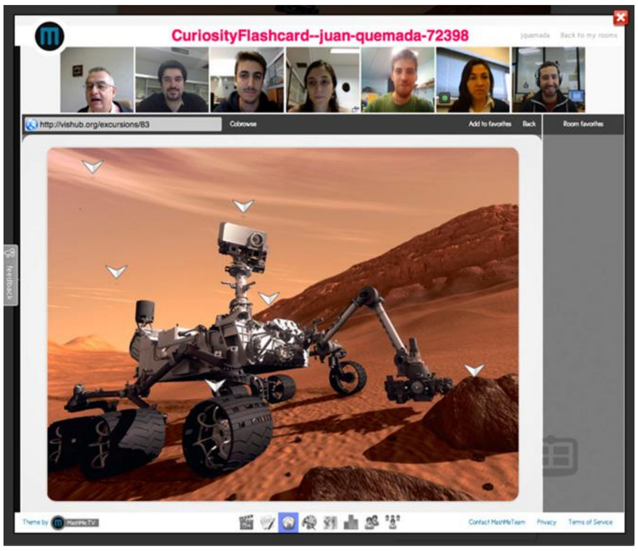
CuriosityFlashcard--juan-quemada-72398
http://vishub.org/excursions/83
Cobrowse
Add to favorites
Back
Room favorites

Fig. 3. Flashcard shared through MashMeTV

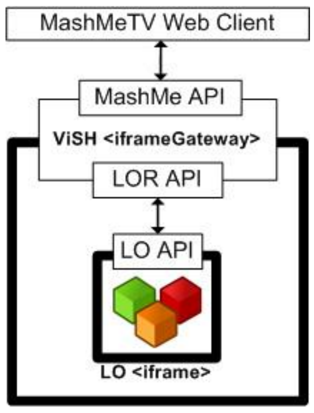

Fig. 4. Iframe Gateway Method

which will communicate with the MashMeTV client through the official API of MashMeTV. This is a good strategy since keeps intact both MashMeTV and ViSH APIs and can be implemented with little work. Since this integration, MashMeTV has been used regularly to hold meetings with scientists, students and teachers, sharing and co-browsing a lot of Virtual Excursions.

B. Lynckia

The experiences of co-browsing Virtual Excursions through MashMeTV were a success. However, new LOs are being developed in the ViSH platform such as educational multiplayer games which have higher latency requirements. Latency plays a key role to provide good game experiences in multiplayer online games. As we mention before, several latency avoidance mechanisms can be implemented to deal with this problem. Nevertheless, many of them (including the proposed version of the lockstep event-locking mechanism) require implementing logic in the messaging server. MashMeTV uses its singular messaging server that resolves problems like the event collision, allowing this way synchronized co-browsing. However, it uses techniques that are not designed with applications like games in mind, where the latency is a critical factor. Co-browsing a Virtual Excursion with a relatively high latency may be unpleasant, but playing a multiplayer game under these conditions may result impossible. Taken all this into account, we decided to develop our own WVS in order to implement techniques in the messaging server tailored to our needs.

For this purpose, we decided to use Lynckia, an open source communications platform that anyone can used as a base to build his/her own videoconference service. Lynckia also provides Videoconference as a Service, but this way we would have no control over the server. So, we implemented our videoconference service built on top of Lynckia providing video, audio and data communication. A first algorithm was implemented allowing synchronized co-browsing of the LOs using the data channel. This allows ViSH users to co-browse synchronously the Virtual Excursions without using third party services. Also, a prototype of a second algorithm implementing the event-locking mechanism described has been designed and partially implementing at the time of writing. This first approach solves the event collision problem and deal in some respects with the latency, but the techniques need to be enhanced.

V. CONCLUSIONS AND FUTURE WORK

We have presented a model to integrate the resources of a LOR into WVSs. Achieving interoperability with WVSs can lead to several benefits such as co-viewing and synchronized co-browsing of the LOs. However, several efforts need to be undertaken both from LORs and WVSs to allow co-browsing in a synchronized way. Moreover, if a LO has strict latency requirements (e.g. an educational game), the WVS should provide latency avoidance mechanisms. Finally, the experience of applying the model in a real e-Learning scenario achieving interoperability with two different WVSs has been also presented.

New learning opportunities to increase student's motivation and engagement arises when integrating LOs into videoconference systems as a consequence of the combination of real time communication and synchronized co-browsing. A good example of this is the GLOBAL excursion project, in which the integration of e-Infrastructures resources (e.g. a microscope) into a videoconference service is used to connect scientists and classrooms.

The next step in this research consists of improving the videoconference service built on top of Lynckia with the main aim of integrating new resources like cooperative multiplayer games that take advantage of videoconference functionalities.

Finally, based on the lessons learned from the MashMeTV experience, we plan to adapt this model to build a LO Gateway with the main aim of achieving the interoperability of standardized e-Learning resources (e.g. SCORM packages) with web videoconference services.

VI. ACKNOWLEDGMENT

We wish to acknowledge our gratitude and appreciation to all the GLOBAL excursion project partners, and each one of the project team members, for their contribution during the development of various ideas and concepts presented in this paper. This work is financially supported by the European Union under FP7, Infrastructures.

REFERENCES

[1] W3C, "HTML5 specification." [Online]. Available: http://www.w3.org/html/wg/drafts/html/master/Overview.html.

[2] IEEE LTSC, "Draft Standard for Learning Object Metadata" 2002.

[3] D. A. Wiley, "Learning Object Design and Sequencing Theory" Brigham Young University, 2000.

[4] P. Mohan and C. Brooks, "Learning Objects on the Semantic Web" in Proceedings of the 3rd IEEE International Conference on Advanced Technologies, 2003, pp. 195–199.

[5] F. Neven and E. Duval, "Reusable Learning Objects: a Survey of LOM-Based Repositories," vol. 68, no. 4, pp. 291–294, 2002.

[6] G. Rust and M. Bide, "The indecs metadata framework: Principles, model and data dictionary," 2000.

[7] "Advanced Distributed Learning (ADL), SCORM 2004 4th Edition," 2004. [Online]. Available: http://www.adlnet.gov/capabilities/scorm/scorm-2004-4th.

[8] “Using learning objects with Bridgit, video conferencing and interactive whiteboards to connect classrooms” 2008.

[9] A. Spurling, “QoS Issues for Multiplayer Gaming” [Online]. Available: http://users.cs.cf.ac.uk/O.F.Rana/data-comms/gaming.pdf.

[10] C. Westermark, “Mobile Multiplayer Gaming” KTH Information and Communication Technology, 2007.

[11] N. E. Baughman and B. Neil, “Cheat-Proof Playout for Centralized and Distributed Online Games” in Proceedings of the IEEE INFOCOM 2001 Conference, 2001, pp. 1–11.

[12] T. Holocher-ertl, B. Kieslinger, and C. M. Fabian, “Linking schools with science: How innovative tools can increase the effectiveness of science teaching in the classroom” 2012.

[13] A. Gordillo, E. Barra, and J. Quemada, “Enhancing K-12 science education through a multi-device web tool to facilitate content integration and e-Infrastructure access” in Proceedings of the 7th International Technology, Education and Development Conference (INTED 2013), 2013.

[14] B. Kieslinger, T. Holocher, C. M. Fabian, D. Gallego, S. Aguirre, E. Barra, and G. Mihai, “Virtual Excursions: a New Way to Explore Science in Class” in Proceddings of the 2th International Conference on New Perspectives in Science Education (NPSE 2013), 2013.

[15] E. Barra, D. Gallego, S. Aguirre, and J. Quemada, “Facilitating the creation of K-12 interactive learning objects using a multi device web tool” in Proceedings of the 2012 Frontiers in Education Conference (FIE 2012), 2012.