# Browser-Based Collaborative Modeling in Near Real-Time

Petru Nicolaescu, Michael Derntl and Ralf Klamma
RWTH Aachen University, Advanced Community Information Systems (ACIS)
Informatik 5, Ahornstr. 55, 52056 Aachen, Germany
Email: {nicolaescu, derntl, klamma}@dbis.rwth-aachen.de

***Abstract*—Collaboration on digital products—for instance in science, design or production—is typically being practiced using cumbersome means like sending document drafts back and forth among collaborators. Recent advances in Web technologies allow collaborators to synchronously edit artifacts. From an engineering perspective, adding (near) real-time, multi-user collaboration to single-user applications is a challenging task as it requires the implementation of features such as conflict resolution as well as propagation and visualization of updates in near real-time. In this paper, we present SyncLD, a collaborative system that was built for a community of practice on 'learning design' to support Web-based, synchronous collaborative editing of learning design models. The system was built on widget technology and implements propagation of edits using inter-widget communication based on the Extensible Messaging and Presence Protocol (XMPP) as well as synchronization of edits using Operational Transformation algorithms. A system evaluation shows that the near real-time collaboration features work as intended, and an end-user evaluation demonstrates the usefulness and usability perceived by practitioners. The core near real-time features are bundled in an open-source library that can be reused for building applications for similar use cases, hopefully propelling the future availability and adoption of near real-time collaboration as a standard feature in Web applications.**

***Keywords—Near Real-Time Collaboration, Operational Transformation, IMS Learning Design, XMPP, Web Applications.***

## I. Introduction

Recent advances in Web technology have a massive contribution in enabling collaboration on the Web. Protocols such as Real-time Transport Protocol (RTP)[1], Extensible Messaging and Presence Protocol (XMPP)[2] or the Web-based Real-time Communication (Web RTC)[3] propel the adoption of near real-time (NRT) collaboration in a variety of domains such as education, medicine, transport, gaming, to name a few. Remote or on-site collaborators can achieve better results with less effort and a higher productivity by means of computer supported collaboration. Moreover, NRT collaboration systems revolutionized team work and eased the large-scale adoption of such systems, e.g. Google Drive[4] or CoWord [1]. These systems profited from new possibilities offered by new communication protocols and from the usage of Operational Transformation (OT) [2] techniques for conflict resolution in shared editing, an active research area for more than two decades now.

With the paradigm shift from desktop applications to Web-based and mobile systems, research and development on supporting collaboration in this rather new setting is in an early phase. Virtual communities of practice, formed by groups of people who share common interests and interact regularly in order to improve their shared practice [3] represent a good example of environments which can greatly benefit from NRT collaboration beyond Google Docs.

In some communities, such collaboration practices may be very common (e.g. among scientists), while other communities that also rely strongly on collaboration struggle with the adoption of collaboration technology. Learning designers are an example of a community that has shown great resistance to tool support [4] using modern Web technology although the creation of learning design models is typically a collaborative process involving teachers, students, instructional designers, education managers, and other stakeholders. Past research [5], [6] found that teachers or instructional designers rely on interactions with each other such as discussions and brainstorming as a main means to develop and decide instructional strategies.

One active area of research on such learning design processes concerns the authoring of formal models that are expressed in a machine readable way and can be reused across different virtual learning environments. The only comprehensive formal specification for learning design that is currently available is IMS Learning Design (IMS LD) [7]. While browser-based authoring tools for IMS LD (e.g. Web-Collage [8]) and desktop apps that allow import and export of the models exist (e.g. Reload [9]), there are currently no tools available for NRT collaboration on learning design models. In this paper, we therefore address the usage of distributed client-side NRT collaboration for creating IMS LD conformant learning design models. We achieve a lightweight, reusable and non-obtrusive solution for enabling NRT collaboration through a widget-based Web application. The solution is based on open source projects, namely on the ROLE SDK[5] for the NRT collaboration components and the provided widget-hosting environment and on the Javascript OT engine adopted from the OpenCoweb project [10] for management of conflicts during concurrent editing.

The rest of the paper is structured as follows. The next Section contains a review of related research and existing software systems in the areas of NRT collaboration, shared editing and IMS LD authoring. Section III describes the conceptual approach and design considerations behind the

[1]http://tools.ietf.org/html/rfc3550
[2]http://xmpp.org/
[3]http://www.webrtc.org/
[4]https://drive.google.com
[5]http://sourceforge.net/projects/role-project/files/role-m10-sdk/

COLLABORATECOM 2013, October 20-23, Austin, United States
Copyright © 2013 ICST
DOI 10.4108/icst.collaboratecom.2013.254111

SyncLD prototype, while the implementation details are presented in Section IV. Section V presents an end-user evaluation and a technical evaluation of SyncLD, respectively. Finally, Section VI wraps the paper up by presenting the main contributions and providing an overview of current work on extending the approach.

## II. Related Work

### A. Near Real-Time Collaborative Editing

NRT collaborative editing (CE) systems are applications that allow multiple users to work together to simultaneously produce a shared output using a set of defined operations from multiple sites connected by communication networks [2], [11]. A considerable number of previous studies deal with solving conflicts in CE settings. Coined during 1980s, well known techniques for concurrency control [11], [12] include locking, transactions, single active participation (participation is decided based on a token), dependency detection (based on timestamps for conflict detection), reversible execution, OT [13], differential synchronization (client-server asynchronous approach) [14], and three-way merge (client-server approach, used in classic versioning systems).

Consistency control is one of the major requirements when developing a system for NRT multi-user collaboration, i.e whenever team members work on a single document, all copies of the document have to be identical at all sites after executing a set of generated operations in a deterministic order. Extensive research has been conducted in the past on developing a consistency model for CE systems. Studies reported in [1], [2], [11] outline three consistency requirements that have to be met by any groupware editing system: **C**onvergence, **C**ausality-preservation and **I**ntention-preservation (CCI). Convergence implies that upon concurrent changes coming from several users, a shared document will have the same content after all the changes have been applied. Causality-preservation refers to the order of performed operations, which has to respect the cause-effect principles (e.g. ensure the same order for changes generated at different time points). Finally, the intention preservation implies that the intention of an operation is still valid after its execution.

Besides data integrity, another important requirement of NRT CE systems is a high responsiveness [11]. As a practice for achieving a better performance whenever a user generates an operation, it should be immediately executed at the local site. This has the advantage of giving the user the feeling of working in a single-user application.

Operational Transformation (OT) [2] is a technique which allows collaborative editing for multiple users without imposing restrictions such as locking. It has been used in applications like Google Wave, now an Apache project[6] and Google Docs [7]. An OT system contains two major components: OT control algorithms and OT transformation functions. The former determine which operations have to be transformed against others according to their concurrency/context relations. The latter determine how to transform a pair of operations according to their type, position and other parameters. The responsibilities between these two components are defined by a set of transformation properties and conditions.

[6]http://incubator.apache.org/wave/

[7]https://drive.google.com

### B. Architectures and Tools for Collaborative Editing

From an engineering view, there are three types of CE system architectures: client-server, peer-to-peer (or replicated) and hybrid.

In a centralized architecture [15] clients connect to a server in a collaboration session. All changes done on the client machines are sent to the server, which ensures that all other clients can get the newest state. For a shared document editing, the central server is responsible for managing the concurrent updates coming from the participants, for storing the document and for maintaining its consistency. Each participant (client) either holds a synchronized copy (thick client) or a view of the main document (thin client). Such an architecture may also have a number of drawbacks such as low responsiveness, single point of failure, high bandwidth requirements and the like.

In a peer-to-peer (or replicated) architecture each participant holds a replica of the shared data and is responsible for processing all the changes to the replica that it holds. The changes are first applied locally and then broadcast to the other participants, without the need for a central server for integrity management. Each site acts as both a client that interfaces with the user and a server that manages the actual collaboration. The replicated architecture has several advantages—e.g., faster response to the user input as the local site is updated immediately before the changes are sent and applied at the remote sites, and it avoids the central server as a single point of failure. Among the disadvantages of a replicated architecture are that it increases storage requirements at the client side, and it requires each client to manage the various aspects of the collaboration, such as consistency management.

There are several existing NRT CE systems available. For example, one of the first attempts, GRoup Outline Viewing Editor (GROVE), pioneered by Ellis and Gibbs [16], was a collaborative text editing system. Among more recent techniques, GoogleWave (now taken over by the Apache Wave project [17]) started as Google's attempt to create an integrated messaging system allowing NRT sharing of messages between selective groups of people. It is based on NRT simultaneous editing of documents via an OT mechanism. Several further developer support libraries for shared editing on the Web are presented in the work of Koren et al. [18].

The OpenCoweb project [10] offers an open-source framework for concurrent NRT interactions between multiple users and external data sources. OpenCoweb implements client-side OT algorithms in Javascript and thus provides cross-browser collaboration support. The framework also contains a Javascript API for framework-specific event handling and server-side components developed in various technologies (e.g. Java and Python). In our implementation, we make use of the self-contained Javascript OT engine adopted from the OpenCoweb project [8], which we modify to suit a pure client-to-client communication setting built on top of the ROLE SDK.

[8]https://github.com/opencoweb/coweb-jsoe

## C. *IMS Learning Design – Specification and Tools*

IMS Learning Design (LD) [7] is a specification used to formally describe the learning design of a unit of learning [19], [20], which we refer to in this paper as the *learning design model*. A learning design model is a description of any teaching-learning process and can, for instance, be the model of a course, a seminar unit, or a self-study unit.

**Design-time vs. run-time.** IMS LD differentiates between the design-time, when the learning design models are being authored, and the run-time, when these models are instantiated and executed. This enables the transfer of designs between different learning design authoring tools and the reuse of designs and materials in any IMS LD compliant run-time engines [21] such as CopperCore[9]. There are three implementation levels defined for IMS LD, with level A being the basic and most important one, since it offers the core concepts such as roles, activities, activity structures, and other concepts that are needed to describe a process. Levels B and C add additional concepts to describe more complex models. For the scope of this work we have focused on IMS LD level A in order to demonstrate the use of collaborative NRT LD authoring. Of course, our prototype can easily be extended to support authoring of levels B and C, respectively, without requiring modifications to our underlying CE engine.

**Packaging.** IMS LD makes use of, or is extensible with other specifications. One of them is IMS Content Packaging (CP) [7], [22], which is used to package and describe a learning design model in a way that is transferable between different IMS LD systems. The packages contain an XML manifest file named “imsmanifest”. This manifest file can be seen as the entry point to the package, which describes the other resources in the package (e.g. text files, binary files, etc.) and also contains an IMS LD representation of the learning design that uses those resources.

**Level A Metamodel.** To give an impression of the structure of the IMS LD metamodel which is supported by SyncLD, the conceptual model of IMS LD level A elements is presented in Fig. 1, adapted from [7]. A learning design model contains learning activities (e.g. ‘read a book’) or support activities (e.g. ‘supervise learners’) to achieve particular learning outcomes using learning environments. An environment is a container that can contain learning objects (e.g. a text document, a video) and services (e.g. conferencing, mail). The activities are performed by roles, which can either be learner roles (e.g. ‘student’) or staff roles (e.g. ‘tutor’). At runtime, the roles are assumed by particular persons—that is, user accounts in the run-time environment. Activity structures can be used to organize activities as a sequence or a selection (e.g., when learners may choose between ‘read a book’ and ‘watch a video’). The metamodel was built with the metaphor of a theater play in mind. Therefore the assignment of roles to activities is made with role-parts (e.g. role ‘student’ performs activity ‘read a book’). Several role-parts are subsumed under an act. The begin and end of an act are used as synchronization points for the role-parts. A play can contain several acts, which are executed in a strict sequence. The learning design model is captured in a method, which contains at least one play.

[9]http://coppercore.sourceforge.net

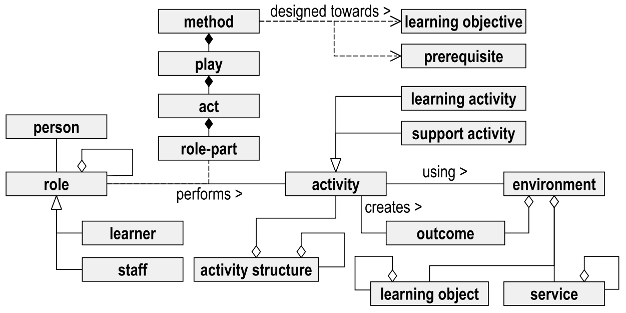
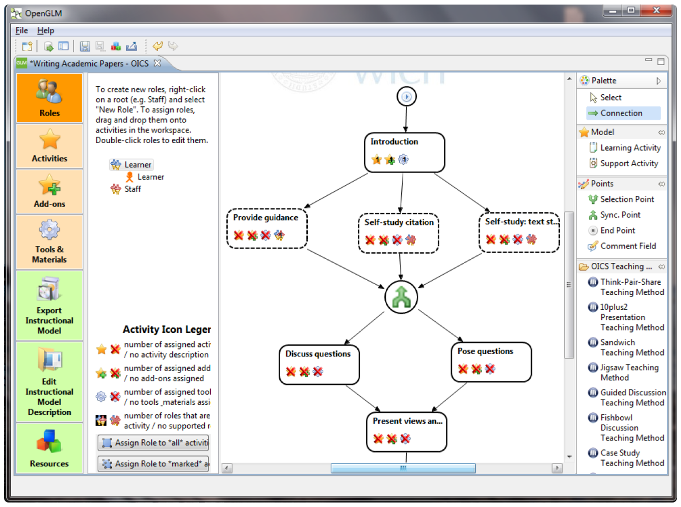


Figure 1: IMS LD level A metamodel



To create new roles, right-click on a root (e.g. Staff) and select "New Role". To assign roles, drag and drop them onto activities in the workspace. Double-click roles to edit them.

Activity Icon Leger

number of assigned acti / no activity description

number of assigned add / no add-ons assigned

number of assigned tool / no tools _materials assi

number of roles that are activity / no supported r

Figure 2: OpenGLM graphical user interface

**Authoring tools.** There are several available visual IMS LD tools that simplify the authoring task by providing an easy to use visual modeling environment that does not require the user to write XML documents. The visual modeling metaphors of these tools typically strive to conceal the strict hierarchy and the Web of element references of the XML binding in the graphical user interface. The tools allow to export the native design models as IMS LD packages that could be deployed in any IMS LD compliant run-time environment such as CopperCore.

The available authoring tools have different user interface metaphors. The Reload (Reusable E-Learning Object Authoring and Delivery) tool [9] is a tree based authoring tool, essentially putting a graphical interface on top of the XML tree, and is regarded as a reference implementation of a learning design authoring tool. The ReCourse tool [23] contains a slightly modified structure of the learning design with more intuitive naming and use. OpenGLM [24] is a more recent implementation providing a diagram-based visual modeling environment in an attempt to hide the complexity of the underlying XML representation from the user by providing a modeling metaphor similar to that of UML activity diagrams. A screenshot of the tool is presented in Fig. 2. OpenGLM also allows the user to import from and export to a remote learning design repository.

In contrast to the previously mentioned tools, which are all desktop applications, WebCollage (Web COLlaborative LeArning desiGn Editor) [8] is a Web-based graphical learning design authoring tool based on collaborative learning flow and assessment patterns. It allows practitioners to produce IMS LD compliant learning designs, instantiate them and eventually deploy them in a virtual learning environment.

Most IMS LD authoring tools offer import and export features for IMS LD packages, which enables asynchronous collaboration during authoring of learning design models. There is currently no IMS LD authoring tool available that supports NRT collaboration of multiple users on a shared model.

## III. SyncLD Design Considerations

To support our design decisions we first briefly define Web widgets and describe the reasons behind choosing this technology for the SyncLD prototype. As described in [25], Web widgets are small self-contained applications which fulfill specific functionalities. They have limited display sizes and can be embedded in a variety of Web applications. Widgets can be repositioned in the browser window and—if supported by the widget container—they can be distributed across several devices to achieve a multi-device personal computing environment [25]. Furthermore, widgets can be easily used on both desktop computers and mobile device screens. Due to their nature, Web widgets can provide a good solution for interacting with data as they constitute front-ends for complex Web Services.

In SyncLD, widgets are used for creating a dynamic authoring environment where the collaborative authoring session can be augmented with other tools (widgets), such as video conferencing, multi-user chat, shared text editing, Google Docs, and so forth. Moreover, we use the layers available on top of the widget container, i.e. widget spaces and inter-widget communication, as communication infrastructure for propagating changes. A widget space [26] is the working context for users and widgets. Multiple users can collaboratively manage and interact with various widgets located in the same space. Also, the space offers the core functions for managing users' presence, multi-user chat, adding-removing widgets, etc. Another basic widget-specific feature used by SyncLD is inter-widget communication (IWC) [27]. This enables the communication between widgets, both local (i.e. in the same browser instance) and remote (i.e. between multiple browser instances and users). With IWC widgets are able to send, receive, and interpret messages. Through these exchanges, multi-user and/or multi-widget applications can be built. SyncLD uses the remote IWC for propagating the authoring operations, updates and synchronization messages, in a multi-user NRT collaboration scenario.

The solution we propose for solving the lack of collaborative LD authoring is a Web-based framework, which can support the NRT message exchange between multiple clients and conflict resolution through the usage of a peer-to-peer OT engine. Our approach can be reusable for other applications which should contain NRT collaboration features and can profit from the various components which we integrate in this work. Concerning the visualization aspects of SyncLD, the interactions with the IMS model are realized through visual modeling of IMS LD compliant units of learning. In a space, users can concurrently work on a shared learning design model and everyone can view all the modifications on their editor instantly. Fig. 3 presents the abstract architecture of SyncLD.

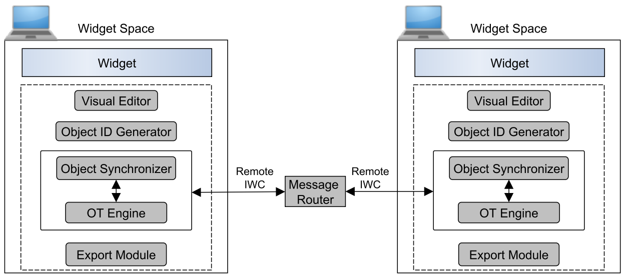


Figure 3: SyncLD architecture

SyncLD uses several inter-operable client-side modules. The "Object Synchronizer" and the "OT Engine" ensure that possible conflicts are resolved and that each client operates on the same version of the learning design model. Because IMS LD is an XML-based specification, we assign to each element a unique identifier. This feature is offered by the "Object ID Generator" module. The "Visual Editor" deals with the modeling aspects at the interface level and the "Export Module" permits an IMS LD compatible export of an XML document containing the authored model, meant to be persisted or used in other authoring applications. Below we briefly describe the modules.

The **OT Engine** builds on the OpenCoweb OT library for maintaining document consistency during concurrent edits by multiple users. The local operations on the shared model are directly reflected, thus providing a short response time. When users join an editing session, the shared authoring state is replicated at all collaborating client sites. When a user makes a modification, a message containing the necessary details is sent to all remote sites via the remote inter-widget communication offered by the ROLE SDK. Users can alter any part of the model during authoring and can see the modifications coming from the remote sites in NRT. For this purpose, the publish subscribe infrastructure provides a way to push messages and notifications to clients without having to poll a request each time.

The objects in the prototype have a certain number of properties. Some of them are editable by users, while some just provide predefined options. An example for the properties of an instance of a learning activity are presented in Table I. Concurrent operations on any of the properties listed in the table may lead to conflicts or inconsistencies in the clients' model representation. For solving concurrent editing problems, the event handling is done through the OT Engine component, which applies OT algorithms for transforming the incoming remote events against the local client operations history and returns the new (transformed) operations. Among the actions that need the OT engine and that may lead to conflicts one can enumerate activity renaming, creation and deletion of activity connections, value selections and text editing. An illustration of such a case can be seen in Fig. 4.

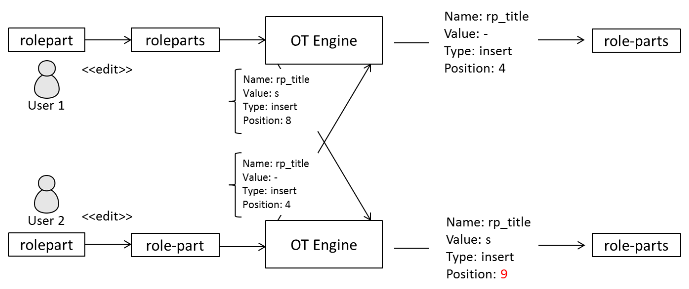


Figure 4: Resolving a text editing conflict using OT

TABLE I: Example of conflict-prone object properties

| Property name | Property format | HTML representation |
|---|---|---|
| Activity description | Editable text | Textarea |
| Prerequisite | Editable text | Textarea |
| Learning objectives | Editable text | Textarea |
| Visibility | True of false (single selection) | Checkbox |
| Resources | Option (single selection) | Select box |
| Environment | Option (multiple selection) | Multiple checkboxes |

The **Object Synchronizer** manages the non-conflicting part of the collaboration on the learning design models, involving object manipulation. The objects can have multiple types, such as "learning activity", "support activity", "role", etc. The synchronizer deals with operations such as "create" or "delete", which can be performed on these objects. Because we consider such operations to create unique objects, the module does not use the OT engine; instead it allows users to collaboratively create and delete such elements.

The **Object ID Generator** assigns an unique identifier for each object in the learning design document. The given identifier is unique across all participating sites. As such, we fulfill the IMS LD specification requirements, which state that each entity (element) should have an unique identifier distinguishing it from all other elements. This also enables referencing of other elements. Furthermore, during a collaborative session, when a user performs an operation on one of the objects, a message is sent to all other participating sites specifying the object the operation should be applied to. The unique string generation across all sites is implemented using a hierarchical nested approach. Every node on a given tree structure is assigned an *id* of type string, made by concatenating the *id* of its parent node, a text that indicates the node type (e.g. "Resource", "Environment", "Learner", etc.) and a number (a unique integer, generated for all the nodes in a total order). The root node's string is concatenated from the *userId*, the node type and the unique number. The *userId* is a unique identifier obtained from the ROLE framework when joining a space, assigned for every member of the space. Incorporating the *userId* in the pattern is crucial as it guarantees the uniqueness of objects in the case when multiple users create the same type of objects. In such cases, no two patterns will be the same, as the *userId* is unique across all participating client sites. The unique number (index) is an integer generated by a function which analyzes the sibling nodes' ids and generates the next maximum number.

The **Visual Editor** module is the module for the SyncLD user interface and handles the management of the various learning design authoring steps which involve the creation of objects (activities, environments, roles, etc.), deletion of objects and property value modifications. These operations can emerge both from the local and remote users. The various UI elements in the SyncLD widget are dynamically updated as a result of executing operations.

The objective of the **Export Module** is to convert the current learning design model into an XML document and resource package that is conformant with the IMS LD specification. The exported file can be imported in IMS LD run-time systems or in other IMS LD authoring applications.

## IV. Implementation

The implementation of SyncLD builds upon the open source Java-based ROLE SDK[10], which provides a platform for responsive and open learning environments. The SDK provides the technical means for rendering and managing Web widgets, using the OpenSocial [28] standard container Apache Shindig[11].

The ROLE SDK infrastructure provides a set of technologies relevant for the design decisions of SyncLD. Namely, the platform implements on top of the Shindig container services for user management and a collaborative widget space management with NRT enabled features. The space concept (see Section III) is currently being standardized under the OpenSocial 3.0 specification. The NRT component is based on the Extensible Messaging and Presence Protocol (XMPP) [29] and uses the XMPP publish-subscribe extension (XEP-0060) [30] for realizing the inter-widget communication and the multi-user chat extension (XEP-045) [31] for the communication and presence information within the space. In collaboration with user and space management services, the platform NRT service manages one dedicated publish-subscribe channel per space for IWC including whitelist-based access control. The *IWC proxy* routes outgoing IWC messages to the affiliated XMPP server via the Strophe-based XMPP connection and incoming messages to all widgets in the space via HTML5 Web Messaging [32]. Strophe[12] is a collection of JavaScript libraries for implementing the XMPP protocol. Widgets can be equipped with IWC support by simply importing the *IWC client* library of the ROLE SDK and implementing functions for publishing and processing IWC messages. In SyncLD we use IWC over Web Sockets as means of propagating the edit operations among all clients in NRT. The IWC permits the remote communication between widgets across browsers, for the users which are members in the same space. Furthermore, the platform supports secure authentication and authorization using OpenID and OAuth, allowing for unique representation of the users from a certain space and therefore for identifying the users which are collaborating.

Concerning the OT module, the OpenCoweb project used is a standalone Javascript OT library, which guarantees document convergence, given that all the local operations are performed and sent to the remote clients and that all the remote incoming operations are honored. Each participating site has its own instance of the OpenCoweb OT engine. The local application

[10]http://sourceforge.net/projects/role-project/
[11]http://shindig.apache.org/
[12]http://strophe.im/

```
ote := OTEngine(siteId);
activityId := "l-activity-00";
propertyId := "description";
objToModify := "l-activity-00-description";
op := ote.createOp(objToModify,"character",
      "insert","position");
opToSend := ote.localEvent(op);
```

Listing 1: Creating a local operation

```
constructPropertyIntent = function(objectId,
    propertyId, value) {
  var key = objectId + "_" + propertyId;
  var op = ote.createOp(key, value, "update",
      "0");
  op = ote.localEvent(op);
  var intent = {
    "component" : "",
    "action" : "ACTION_CHANGE",
    "data" : "",
    "dataType" : "text/plain",
    "flags" : [ "PUBLISH_GLOBAL" ],
    "extras": {
       "objectId" : objectId,
       "elementId" : propertyId,
       "operation" : op } };
  updateObjectProperty(objectId, propertyId,
      value);
  sendIntent(intent);
}
```

Listing 2: Example of intent propagation

has to call the `OTEngine.localEvent` method for all local changes and send the object returned from this call to the remote peers unchanged. When remote operations are received, the engine passes them to an `OTEngine.remoteEvent` method. The transformed operation returned by this method is then applied to the local document. The OpenCoweb OT engine can be used in one application for several collaborating objects. For example, an application can have a chat and a text editor. Each one of these parts are treated separately by the engine. We consider each editable field of a given object as a separate collaborating object. Thus, whenever an edit event occurs on a particular text field, the OT engine is called by passing the combinations of the *id* of the object and the *id* of the HTML field as an argument. To illustrate this, consider the following example: a user selects a given learning activity that has the *id* attribute of `l-activity-00` and starts to edit the activity's description represented by the corresponding HTML textarea element. Upon typing in the HTML textarea elements, the application generates key press events, executes the necessary updates locally and creates operations that are sent to remote sites as shown in Listing 1.

The intent used for the exchange of messages—using JavaScript Object Notation (JSON) format—is presented in Listing 2. JSON is also used for the representation of internal objects.

The sequence diagram shown in Fig. 5 presents the methods called for an edit operation that represents a simple renaming of an activity, in the collaboration setting. The operation is

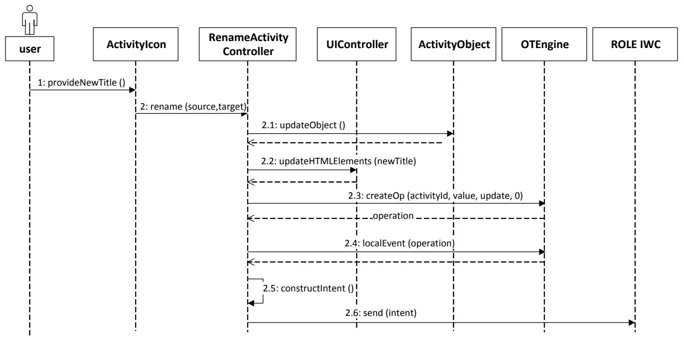

user
ActivityIcon
RenameActivity Controller
UIController
ActivityObject
OTEngine
ROLE IWC
1: provideNewTitle ()
2: rename (source,target)
2.1: updateObject ()
2.2: updateHTMLElements (newTitle)
2.3: createOp (activityId, value, update, 0)
operation
2.4: localEvent (operation)
2.5: constructIntent ()
2.6: send (intent)

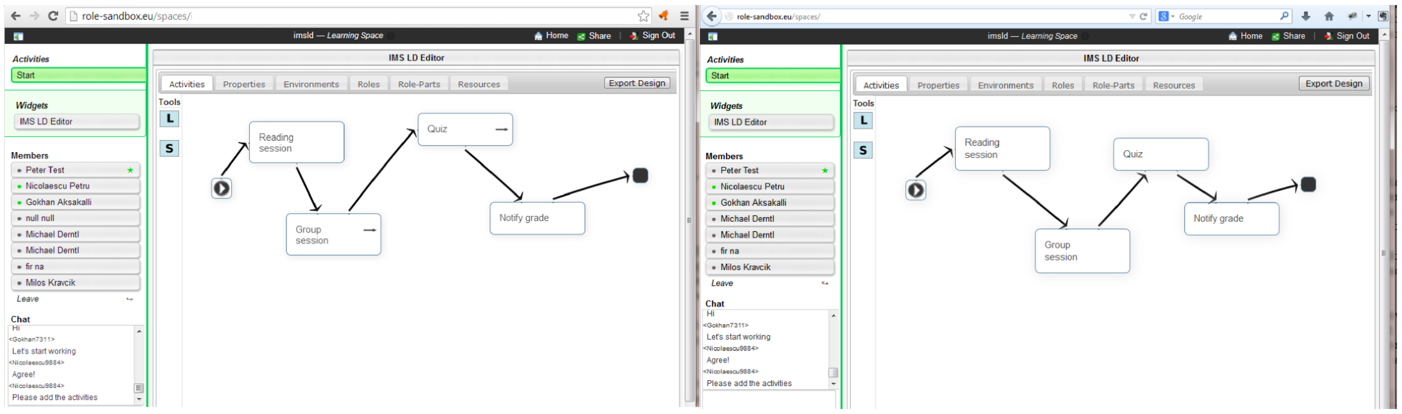
Figure 5: Sequence diagram for an activity renaming operation

displayed and then applied to the local OT engine. The engine saves internally the operation and creates the actual operation JSON object, containing information about the unique element where it should be applied, the value to be changed, etc. This can be passed to remote peer's `OTEngine.remoteEvent` using the IWC intent. From this point on, all clients that receive the IWC intent compare the operation with their local operation history, transform the operation if necessary and apply it locally to their copy of the model.

At the interface level, the visual modeling is realized in two ways: a tree structure for detailed properties of elements and a UML-like activity diagram approach for producing the sequence of learning and support activities. The activities are created by dragging and dropping the `L` (learning activity) or `S` (support activity) icons from the left-hand toolbar onto the drawing canvas. The flow sequence between them is built by connecting the activities via a directed arrow. A screenshot depicting four connected activities, edited in two browser instances is presented in Fig. 6. Activities also have a tree structure representation that is used to provide a convenient navigation for editing the various properties of the activities. The rest of the entities, such as environments, roles, role-parts and resources are presented as a collaborative tree structure. Each entity type has a separate tree-like visualization. To modify a property a user selects a specific node from the tree.

Finally, for the export functionality, the conversion process is divided into series of tasks, such as analyzing the JSON objects in the visual learning design. This involves differentiating between attributes and child elements, eliminating empty elements from appearing in the XML document and extracting the relevant data. For example a learning activity JSON object contains data related with it's position on the canvas, yet, this data should not appear in the XML document since it will violate the specification. Another task is creating a new text file for every property that is editable (e.g. a learning activity's description, prerequisite, learning objective, etc.), which is required by the IMS LD specification. The files are represented as a separate XML element which can be referenced by other elements.

Once all objects have a valid structure and are glued together accordingly, they are converted to XML elements using the function "json2xml", that takes a JSON object as an argument and returns the equivalent XML representation. This function—an efficient and precise algorithm—is reused in our prototype with a slight modification to include names-



Hi
<Gokhan7311>
Let's start working
<Nicolaescu9884>
Agree!
<Nicolaescu9884>
Please add the activities

Figure 6: Sequence of activities in the SyncLD user interface showing synchronized model instances in two side-by-side browsers

paces [33]. The resulting XML is stored as the imsmanifest file mentioned previously, and zipped along with the referenced file resources.

## V. Evaluation

### A. End-User Evaluation

The innovation on the user interface layer of SyncLD is that the application puts the combined "goodies" of existing IMS LD authoring interfaces to work in the Web browser in a NRT collaboration environment. The activity modeling metaphors are similar to those in OpenGLM and WebCollage. The editing of detailed element properties is achieved using a familiar tree structure navigation where editing forms for the elements can be accessed. This is a common user interface metaphor in most IMS LD authoring tools. The key novelty is NRT synchronization of edits by different users.

IMS LD authoring can be approached very flexibly. There are only few restrictions on the order of creation of elements, which relate to situations where an element references another element. Logically, the referenced element must be created before referencing it (e.g. an activity referencing an environment). Other than that authors are free to choose in which order and level of detail they want to proceed during authoring. This is one of the reasons why we chose IMS LD authoring as a pilot application for our NRT collaboration technology.

The end-user evaluation was performed by providing a predefined authoring scenario, which was enacted in each evaluation session by three concurrent users with varying levels of IMS LD expertise under the guidance of a moderator, who was an IMS LD expert. The scenario consisted of four steps as listed in Table II. The three authors were instructed at the beginning of each step and synchronized at the end of each step by the moderator. During the session each participant had as a backup a sheet with a description of his/her duties during the authoring session as listed in the table. The scenario was designed to allow users to experience the key features of synchronized collaborative IMS LD authoring while increasing the chance of concurrent edits. At the end of the session the users were asked to fill an online questionnaire that surveyed their perceptions of usability and usefulness of the tool in seven multiple-choice questions and open questions on aspects they liked and disliked the most about the tool, respectively.

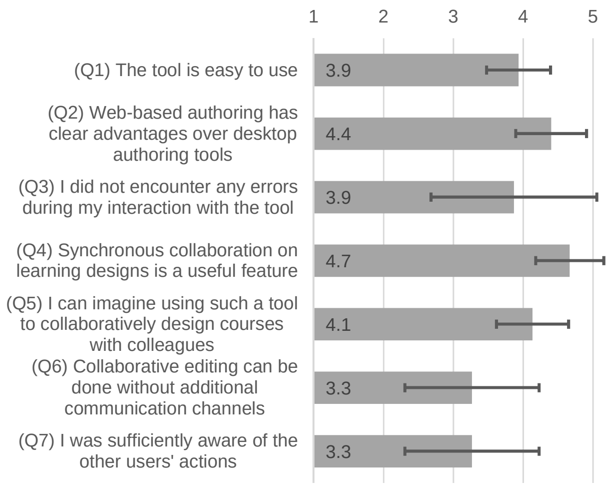


Figure 7: Evaluation survey results indicating average rating and standard deviation for each question [1 = fully disagree … 5 = fully agree]

We had five evaluation sessions with three participants each, making a total of 15 participants. The participants were asked to rate their IMS LD expertise on a Likert scale ranging from novice (1) to expert (5). The mean rating was 2.6 (±1.3), slightly below the middle point on the scale.

The rating results of the multiple-choice questions are displayed in Fig. 7. All but two participants agreed or strongly agreed that the tool was easy to use (mean $= 3.9$ on $Q_1$). In the open-ended questions some negative comments related to usability were about not being familiar with IMS LD which complicates the use of the tool, which indicates that the tool would need some in-place assistance ("I am not accustomed to IMS LD. The UI is quite complicated." — "It is a bit complex." — "…requires a deep knowledge of IMS LD. Maybe some definitions could be helpful, e.g. meaning of environment"). In current work we are addressing this issue by providing users with *nudges* on what to do next at each stage. In the second question ($Q_3$) users indicated that they encountered some errors during use (mean $= 3.9$).

The tool did not have any visual awareness indicators to

TABLE II: End-user evaluation session script with three users collaboratively creating a simple learning design model.

| Step | Moderator | User 1 | User 2 | User 3 |
|---|---|---|---|---|
| 1 | Tell the three users to perform step 1. | In the Resources tab, add a new resource '*Learning materials*'. Add a random local file from your computer to this resource and title it '*Tech-writing book*'. | In the Resources tab, add a new resource '*Conference materials*'. Add a random local file from your computer to this resource and title it '*Conference slides*'. | In the Resources tab, add a new resource '*Quiz materials*'. Add '*Online material*' to this resource, with the title '*Online quiz*' and the url '*www.example.org*'. |
| 2 | Create environment '*Study materials*'. Tell the three users to perform step 2. While they do so, create environment '*Grade report*'. | Wait for the moderator to create the '*Study materials*' environment. Then add a learning object '*Tech-writing book*' to this environment. Select learning object type: '*knowledge-object*' and resource: '*Learning materials*'. | Wait for the moderator to create the '*Study materials*' environment. Then add a learning object '*Quiz*' to this environment. Select learning object type: '*test-object*' and resource: '*Quiz materials*' | Wait for the moderator to create the '*Study materials*' environment. Then add a '*New Conference*' service to this environment with the title '*Tech-writing conference*'. Associate the '*Conference materials*' resource to this service. In the same view, please select the student and the tutor as participants, and the tutor as manager and moderator. |
| 3 | When user 3 complains that he cannot add roles to the conference service, tell user 1 and 2 to first complete step 2. Then tell user 1 and 2 that they should proceed with step 3, i.e. user 1 to add student role and user 2 to add tutor role. Ask user 3 whether he can see the roles popping up now. | Add learner role with title '*Student*'. | Add staff role with title '*Tutor*'. | Wait until roles from step 2 appear in your user interface. |
| 4 | Tell users to create learning activities in step 4 by dragging the '*L*' on the canvas. In the meantime add support activity '*Notify grade*', with environment '*Grade report*' and supported role '*Student*'. When all users ready, tell them to go to '*Properties*', activity '*Reading session*'. Modify the description to '*Read the provided materials*'. Then tell them to go to the '*Activities tab*' and wait. Tell them that you are now connecting the activities. Do it. | Create learning activity '*Reading session*'. Put this in the description: '*Read the materials*'. Select environment '*Study materials*'. | Create learning activity '*Group Discussion*'. Put this in the description: '*Technical writing*'. Select environment '*Study materials*'. | Create learning activity '*Quiz*'. Put this in the description: '*Publish quiz*'. Select environment '*Study materials*' and completion-rule '*time limit*'. |

indicate to a user what part of the learning design model the other users are currently editing. However, since the scenario made them edit the same portions of the model in each step, there was some built-in awareness. This is also indicated by the responses to $Q_7$ which averaged a rating of 3.3. In the open questions users emphasized the need for awareness mechanisms (e.g., "it is not so clear what the others are currently doing").

We also asked whether the users felt that there was no need for an additional communication channel (e.g. audio, chat) while collaboratively editing the learning design ($Q_6$). The average rating of 3.3 along with a high standard deviation shows that the opinions differed considerably. Clearly, being able to speak to collaborators is an advantage, yet obviously several users were confident that they could use the tool collaboratively without additional communication channels.

All participants agreed or strongly agreed with the statement that web-based authoring has clear advantages over the desktop authoring tools ($Q_2$; mean $= 4.4$). In addition, all of them agreed that the synchronous collaboration feature was useful ($Q_4$; mean $= 4.7$), which provide strong support for the objectives and outcome of this research. Some of comments about what they liked about the NRT editing include: "co-designing and sharing a common space" — "to see other users' changes almost in real-time" — "everything is automatically synchronized/stored (like Google Docs)" — "work can be done faster if everyone works in a different area" — "to see things done by collaborators popping up."

Last but not least, almost all participants agreed that they could imagine using such a tool to collaboratively design courses with their colleagues (mean $= 4.1$ for $Q_5$), which is an encouraging result when thinking about productive use of this or similar tools by learning designers for real tasks.

## B. *Technical Evaluation*

Parallel with the user evaluation, we considered also the evaluation of the technical realization of SyncLD, the collaboration and NRT features of the prototype and the underlying Javascript library integration. Two approaches were used: the first approach was to perform functional tests during the development of the prototype. Moreover, in the ending phase of the development cycle the performance of the XMPP-based NRT messaging infrastructure working together with the OT engine was tested. The second approach involved engaging multiple users to collaboratively author a learning design using the visual component of the prototype.

The performance evaluation of the framework was conducted using the same NRT shared editing setting with the one used by SyncLD. The analysis was performed using three collaborating browser instances with three different users logged in with their accounts in the same ROLE space. The clients accessed a wireless network different from the one containing the XMPP server. Each clients collaborated using a text editing widget (containing a text area HTML element connected to the OT library and the ROLE IWC for event handling). Events were keyboard-generated using the text area element. All underlying editing infrastructure messages were logged, persisting information such as the exact time of sending and receiving each message together with operation type and the ids of the sender and receiver. The experiment consisted of several sessions, in order to test the different possible IWC exchange scenarios between the clients. Two shared text editing sessions at usual writing speed were considered and one with continuous character input. Because part of SyncLD collaboration involved single operations (e.g. activity creation, checkboxes, etc.), single events (created using single character input at a time interval greater than one second) were also analyzed. In each session one hundred messages were generated. Finally, the message delay between different clients was analyzed, using one sender and two receivers and measuring the time difference between the message receive time at the receivers. The results are presented in Table III.

TABLE III: Technical evaluation of messaging infrastructure and OT Engine

| **Test Performed** | **Average Value (ms)** | **Standard Deviation (ms)** |
|---|---|---|
| Continuous character input | 778 | 1237 |
| Writing (normal speed) | 457 | 856 |
| Single operations | 99 | 58 |
| Time difference at receivers | 28 | 24 |

The performance evaluation results demonstrate the feasibility of our methods for enabling NRT collaboration for Web applications. As such, the small receive time delay between the clients outputs a real-time perception for remote collaborators. The increased time difference obtained for the test cases containing multiple characters input is due to the buffering of characters and the bundling of a certain number of such events under a single IWC intent, in order to minimize the network traffic.

Concerning the visual component—that is, the visual modeling of the learning design—this feature has been tested by iteratively creating and erasing model elements from the canvas. For the collaborative features, the focus was in measuring the convergence (consistency of the model at the participating clients) of the learning design when multiple users are collaboratively authoring. This was carried out as an extension of the user evaluation, where we tried to gain knowledge of the technology usability and reliability. In this sense, we used part of the users presented in the user study, in order to assess the consistency of the client replicas. After the users collaboratively designed the UOL, they were asked to export the UOL model using the tool. We collected from each user the downloaded UOL package and inspected each file in the package for convergence. Each package contained one XML ("imsmanifest") file and four text files. A comparison of the content of the files were carried out manually since, except the XML file, the content of the text files was very short (predefined by the guideline). The comparison of the XML files were done by inspecting the elements. With respect to this test, it was observed that the design at each of the three sites was convergent, meaning, the "imsmanifest" XML file as well as the text files generated by each user had the same content. However, we also note that there are factors which can have an impact on this result such as the overall design being guided by predefined steps, number of participants, etc.

Parts of the SyncLD requirements (such as the space presence signaling, the ability to share and join a space, etc.) are supported by the underlying ROLE SDK infrastructure and further performance tests were not considered as being within the scope of the present work. However, the IWC messaging and the usage of the widget technology is indirectly included in our evaluation.

## VI. Conclusions and Future Work

In this paper we reported pioneering work on providing support for NRT collaborative editing of learning design models. Technically we used a decentralized approach for managing the collaboration operations and OT algorithms for ensuring data integrity and consistency on all participating sites. The resulting application called SyncLD (**Sync**hronous **L**earning **D**esign) has several distinct features.

First, it allows *concurrent editing* by multiple users connected via a network to work on a shared model at the same time without producing inconsistencies and versioning issues. This solves the problem of the otherwise asynchronous process of sharing a document for editing which can create inconsistencies and can be time-consuming, e.g. when a manual merge is required. From the practitioner perspective the end-user evaluation showed that learning designers perceive such a feature as highly useful for their work.

Second, the application uses a *NRT communication infrastructure* based on XMPP—an open and extensible protocol—for synchronizing concurrent user inputs and displaying the effects in a Web based user interface. This is necessary so that changes made by one of the participating users be reflected on the replicas of the shared document on the other users' workstations. The development of the communication infrastructure was fully based on open source projects, including the ROLE SDK for hosting widget-based applications using an inter-widget communication API, and the OpenCoweb framework for handling the OT that eventually ensures synchronization of user actions at all participating ends. By integrating the two projects and tailoring the OT engine to support the learning design structure and the ROLE SDK as a collaboration platform, we have achieved a powerful setting for enabling NRT collaboration for Web widgets.

The end-user evaluation has demonstrated that learning design practitioners considered the NRT collaboration features offered by SyncLD as very useful and also usable.Almost all participants agreed that web-based authoring has advantages over desktop based authoring and that they can imagine using a tool like SyncLD in their real design work. The technical evaluation has shown that after the end-user evaluation sessions the model replicas at the client machines were congruent.

Current work on extending SyncLD follows several threads. For one, we are working on improving the user interface to offer awareness about collaborators' actions. This was an issue mentioned by several users in the end-user evaluation. It will be offered in a separate widget that can be added to the SyncLD widget space and that displays hyperlinked information on what each collaborator is currently working on, e.g. it might say "User X is editing the description of activity 'Read a book'. User Y is currently assigning the tutor role to support activities."

On a more substantial thread we are working on ways to provide *nudges* to users on what actions they could perform next to proceed in collaboratively completing the model. To achieve this we will use the awareness information in combination with a model of the dependencies and rules in effect during the IMS LD modeling process to provide those nudges in a widget.

Finally, we are working on an abstraction of the NRT collaboration framework from the IMS LD context to supporting collaborative creation of arbitrary models using flexible modeling processes. The source code of all components of SyncLD is and will continue to be released with free and open source licenses. This is one of the major differences to projects like the Realtime API in Google Drive, which offers the synchronization infrastructure as a black box, thus obtaining control over the functionality, content and eventually the 'fate' of tools which are developed using such black box APIs. In contrast, by releasing open frameworks for NRT communication, application developers will have full control over the communication infrastructure and can contribute to making the Web a place where NRT collaboration is offered in all applications.

## Acknowledgment

This research was supported by the European Commission in the Lifelong Learning Programme multilateral project METIS (531262-LLP-2012-ES-KA3-KA3MP) as well as in the 7th Framework Programme large-scale integrated projects LAYERS (grant no. 318209) and ROLE (grant no. 231396). The authors would like to thank Bezunesh Alemu Terkik for the prototype implementation.

## References

[1] C. Sun, S. Xia, D. Sun, D. Chen, H. Shen, and W. Cai, "Transparent Adaptation of Single-User Applications for Multi-User Real-Time Collaboration," *ACM Transactions on Computer-Human Interaction*, vol. 13, no. 4, p. 531582, 2006.

[2] C. A. Ellis and S. J. Gibbs, "Concurrency Control in Groupware Systems," *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, vol. 18, pp. 399–407, 1989.

[3] E. Wenger and W. Snyder, "Learning in Communities," *LiNE Zine*, no. 1, 2000.

[4] M. Derntl, S. Neumann, and P. Oberhuemer, "Opportunities and challenges of formal instructional modeling for web-based learning," in *Proceedings of ICWL 2011, LNCS vol. 7048*. Springer, 2011.

[5] T. K. Christensen and R. T. Osguthorpe, "How Do Instructional-Design Practitioners Make Instructional-Strategy Decisions?" *Performance Improvement Quarterly*, vol. 17, no. 3, pp. 45–65, 2004.

[6] P. Kirschner, C. Carr, J. Merrinboer, and P. Sloep, "How Expert Designers Design," *Performance Improvement Quarterly*, vol. 15, no. 4, pp. 86–104, 2002.

[7] IMSGlobal Learning Consortium. (2003) Learning Design Specification. [Online]. Available: http://www.imsglobal.org/learningdesign/

[8] D. Herndez-Leo, Villasclaras-Fernndez E. D., Asensio-Prez J. I., Dimitriadis Y., Jorrn-Abelln I. M., and Ruiz-Requies I., "COLLAGE, a Collaborative Learning Design Editor Based on Patterns," *Educational Technology and Society*, vol. 9, no. 1, p. 5871, 2006.

[9] Reusable eLearning Object Authoring & Delivery. (2004) RELOAD Editor Introductory Manual. [Online]. Available: http://www.reload.ac.uk/ex/editor_v1_3_manual.pdf

[10] The Dojo Foundation. Open Cooperative Web Framework: Javascript Enablement of Concurrent Real-Time Interactions. [Online]. Available: http://opencoweb.org/

[11] C. Sun, X. Jia, Y. Zhang, Y. Yang, and D. Chen, "Achieving Convergence, Causality Preservation and Intention Preservation in Real-time Cooperative Editing Systems," *ACM Transactions on Computer-Human Interaction*, vol. 5, no. 1, p. 63108, 1998.

[12] J. Liu, G. Teng, Y. Shao, W. Yao, and S. Dong, "Concurrency Control Strategy in Real-time Collaborative Editing System," in *2010 2nd International Conference on Education Technology and Computer (ICETC)*, p. 222225.

[13] S. Kumawat and A. Khunteta, "A Survey on Operational Transformation Algorithms: Challenges, Issues and Achievements," *International Journal of Computer Applications*, vol. 3, no. 12, p. 3038, 2010.

[14] N. Fraser, "Differential Synchronization," *Proceedings of the 9th ACM Symposium on Document Engineering*, p. 1320, 2009.

[15] C. Sun and D. Chen, "Consistency Maintenance in Real-time Collaborative Graphics Editing Systems," *ACM Transactions on Computer-Human Interaction*, vol. 9, no. 1, pp. 1–41, 2002.

[16] C. A. Ellis, S. J. Gibbs, and G. Rein, "Groupware: Some Issues and Experiences," *Communications of the ACM*, vol. 34, no. 1, p. 3958, 1991.

[17] The Apache Software Foundation. Apache Wave (Incubating). [Online]. Available: http://incubator.apache.org/wave/

[18] I. Koren, A. Guth, and R. Klamma, "Shared Editing on the Web: A Classication of Developer Support Libraries," in *Proceedings of the Ninth International Conference on Collaborative Computing (CollaborateCom 2013)*. IEEE, 2013.

[19] H. Hummel, J. Manderveld, C. Tattersall, and R. Koper, "Educational Modelling Language and Learning Design: New Opportunities for Instructional Reusability and Personalised Learning," *International Journal of Learning Technology*, vol. 1, no. 1, pp. 111–126, 2004.

[20] R. Koper and B. Olivier, "Representing the Learning Design of Units of Learning," *Educational Technology and Society*, vol. 7, no. 3, pp. 97–111, 2004.

[21] W. Westera, F. Brouns, K. Pannekeet, J. Janssen, and M. J., "Achieving E-learning with IMS Learning Design - Workflow Implications at the Open University of the Netherlands," *Educational Technology and Society*, vol. 8, no. 3, p. 216225, 2005.

[22] M. Pullin. (2007) IMS Content Packaging Research Report. [Online]. Available: http://thecblor.unt.edu/files/IMSCP-Report-2.pdf

[23] TEN Competence Foundation. (2010) ReCourse Learning Design Editor. [Online]. Available: http://tencompetence-project.bolton.ac.uk/ldauthor/

[24] M. Derntl, S. Neumann, and P. Oberhuemer, "Propelling standards-based sharing and reuse in instructional modeling communities: The open graphical learning modeler (openglm)," in *Proceedings of IEEE ICALT 2011, Athens, GA*. IEEE, 2011, pp. 431–435.

[25] D. Kovachev, D. Renzel, P. Nicolaescu, and R. Klamma, "DireWolf - Distributing and Migrating User Interfaces for Widget-Based Web Applications," in *Proceedings of ICWE 2013, LNCS 7977*. Springer, 2013, pp. 99–113.

[26] E. Bogdanov, C. Salzmann, and D. Gillet, "Contextual Spaces with Functional Skins as OpenSocial Extension," in *ACHI 2011, The Fourth International Conference on Advances in Computer-Human Interactions*, 2011, pp. 158–163.

[27] S. Govaerts, K. Verbert, D. Dahrendorf, C. Ullrich, M. Schmidt, M. Werkle, A. Chatterjee, A. Nussbaumer, D. Renzel, M. Scheffel, M. Friedrich, J. L. Santos, E. Duval, and E. L.-C. Law, "Towards responsive open learning environments: the ROLE interoperability framework," in *Proceedings of the 6th European conference on Technology enhanced learning: towards ubiquitous learning*, ser. EC-TEL11. Springer-Verlag, 2011, pp. 125–138.

[28] OpenSocial and Gadgets Specification Group. OpenSocial Specification 2.5.0. [Online]. Available: http://opensocial-resources.googlecode.com/svn/spec/2.5/

[29] P. Saint-Andre, "RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core," 2011.

[30] Millard, Peter and Saint-Andre, Peter and Meijer, Ralph, "XEP-0060: Publish-Subscribe Version 1.13, Draft," 2010.

[31] P. Saint-Andre, "XEP-0045: Multi-User Chat," 2008.

[32] W3C, "HTML5 Web Messaging," 2011.

[33] S. Goessner. (2006) Converting between XML and JSON. [Online]. Available: http://goessner.net/download/prj/jsonxml/json2xml.js