Zenaminer: driving the SCORM standard towards the Web of Data

Ernesto Mudu[1], Luca Schiatti[1], Giuseppe Rizzo[1,2], and Antonio Servetti[1]

[1] Dipartimento di Automatica e Informatica, Politecnico di Torino
Corso Duca degli Abruzzi, 24, 10129 Torino, Italy
ernesto.mudu@studenti.polito.it, luca.schiatti@studenti.polito.it,
giuseppe.rizzo@polito.it, antonio.servetti@polito.it
[2] Department of Multimedia Communication, EURECOM
2229, Route des Crêtes, 06560, Sophia Antipolis, France
giuseppe.rizzo@eurecom.fr

Abstract. In this paper we present Zenaminer, a software architecture for linking the SCORM standard to the Linked Open Data cloud. The main idea is to overcome some limitations of the SCORM Content Packaging standard, the most used standard for sharing e-learning contents, in order to migrate this material into the Web of Data. We analyze the design of a RESTful Web Service for SCORM where each single SCO (Sharable Content Object) is exposed on the web as raw data that can be created, read, updated, and deleted through the HTTP protocol. In addition, borrowing the idea from the concept of web mashup, we propose to build the Learning Management System interface as a mashup that uses Zenaminer web API to access SCO data and that customizes the presentation and interaction with the help of additional CSS and Javascript. To achieve this goal, for the creation of the HTML material, we suggest to use a "light formalism" defined by the W3C with the name Slidy so that the presentation remains separated from the content. Finally, Zenaminer allow users to enrich the SCORM resources with comments. Comments can be contents themselves that extend and improve the original material of the teacher. Unstructured comments are automatically annotated with Spotlight linking them to the LOD cloud through DBpedia.

Keywords: SCORM, e-learning, LOD, DBpedia, web service, REST, learning object

1 Introduction

The evolution of digital communications and the more and more relevant presence of the Web in our society lead to significant changes also in the e-learning scenario. A new term is being used to define this change, "e-learning 2.0" [6]. In relation to the same shift happened in the Web, among the various innovations that characterize the shift from the 1.0 to the 2.0 system, in this work we focus on the new paradigm defined as "Web of Data". That is, the old Web of mostly

human readable documents has to change in a Web of micro contents that can be easily processed by machines [1] and by the users. The big advantage is in the ability to manipulate every single part of a documents on-the-fly with the possibility to aggregate and remix the data in order to generate new documents of enhanced value. Micro contents are the building blocks on which Linked Data can operate, they represent contents that convey one idea and that can be accessed through a single URL in the space of the Web. Examples of micro contents can be the arrival and departure times for an airplane flight, the abstract from a long publication, or other similar. As a consequence of this transformation, the Web is changing from a read-only platform to a read-write platform where users, starting from data available on-line, can also share, remix and create new and original material [12]. Web users are becoming both consumers and producers, i.e., prosumers.

These ideas are not new at all in the e-learning scenario. Here, teaching material is defined in the form of micro contents named Sharable Content Objects (SCO). SCOs form the basis of usability, interoperability, and adaptation. They are the building blocks that can be independently produced, stored, indexed, composed and evaluated. SCOs can be considered a compromise between raw data (e.g., image, video, audio, text) and a whole lesson: they have a context, but small, so they can flexibly arranged to form new lessons or presented in a different shape changing the appearance of their elements (e.g., logos, titles, etc.). For example, in this work we have defined a SCO as a group of slides on the same topic inside a lesson. Thus, the production of teaching material implies the production of a number of SCOs that can be collected to create an archive of sharable micro contents. Nevertheless, no services have been defined to migrate this information into the Web of Data. SCOs are shared as documents, ZIP archives, that a user need to download and extract before being able to use them in a new project.

The paradigm of the "Web as a Platform" [10] requires instead these contents to be exposed on the Web through an Application Programming Interface (API) that allows a set of operations on them, i.e., create, read, update, delete. From a Linked Data point of view, a Learning Object Repository (LOR) can be seen as a Web Service (WS) where, briefly, each LO is a resource (and so identified by URI) that can be processed using the HTTP methods (GET, PUT, POST, and DELETE) of the Representational State Transfer (REST) [7]. In this paper, we propose a novel architecture to expose on the Web, as a Linked Data resource, the teaching material of a Shareable Content Object Reference Model (SCORM) packet, the de-facto standard in the production of e-learning content. Our objective is twofold, the design of a new Web interface for SCORMs and the migration of LOs into the Linked Data Cloud where the "consumer" can infer, connect and aggregate data from different repositories in order to create new value added resources. The added value is not in the data itself, but in how contents are combined in new ways, how they are presented with new interfaces, in how they are enriched far beyond they original content defined by the teacher.

The remainder of this paper is organized as follows. A review of the current state of the art is presented in Section 2. The key ideas of our approach, with some details on the issues of exporting SCORMs by means of a REST web service, are described in Section 3. Then, in Section 4, we discuss the concept of separation of concerns, which is at the basis of our tool. In Section 5 we describe the architecture of the web service and, finally, in Section 6 the use case used to test our work. Conclusions follow in Section 7 where we stress on the benefits introduced by the tool into the e-learning community.

2 Related Work

E-learning systems has assumed an important role in the Web community for the important value of the contents which they shared. In this context, contents are named sharable content objects (SCO) and they hold information about lectures and metadata useful to describe better the inferred contents. Based on the assumption that the Web provides the best opportunity to maximize both access to and the reuse of learning contents, the overall objective of these systems are to make sharable content objects usable, reusable and interoperable. In order to address this, our goals are to define and to implement a model to export teaching materials adhering the Web of Data.

Many efforts has been spent to make SCORM environment interoperable[3] within the Web cloud. Many of them focused to make interoperable LMSs in a cloud infrastructure [5] e [4] other, instead, to perform interoperability sharing learning objects. Although the idea of making a network of services which may handle teaching materials is interesting for what concerns managing the author information, security and network reliability, on the other hand it requires other formalism to make interoperable the atomic information. In addition, it is far from the idea of making the Web of Data linkable, which is one of the best practice of the Linked Data.

Vossen and Westerkamp [14] introduced the need to share SCO by means of service. They proposed a service oriented architecture (SOA), but any specific architecture or implementation was described. In [3], Redondo et al. discussed typical problems of SCORM-related standards and proposed a web service-oriented approach as a solution. They introduced the concept of Everything as a Service (Eaas) and they mapped each single SCO as a service. A WSDL exposed information about services available to gather information from the each SCO. Gonzalez et al [8] extended Moodle[4] to the SOA paradigm. Modules and courses were available from the Web and sharable from a different kind of applications, making it a portable tool. To sum up, these approaches try to expose features of these LMSs, using the service-oriented architecture. Our work, instead, exposes data from each single SCO in a raw way, in order to make this data linkable and navigable from the Web applications.

[3] According to the Open Knowledge Initiative (OKI) definition, interoperability is about making the integration as simple and cost effective as technologically possible.

[4] http://moodle.org/

3 A web service for SCORM

SCORM, i.e., the Sharable Content Object Reference Model, is a reference model for the creation of web-based e-learning material with the objective to ensure interoperability, reusability, and accessibility. The SCORM content packaging section specifies how the course material should be packaged, usually in a ZIP file, and described. The course is defined as a collection of Sharable Content Objects (SCOs) that can be associated to a lesson or a part of it. A Content Object must be a web-deliverable learning unit, that is usually designed as an HTML page with CSS and Javascript so that it can be launched in a web browser.

SCORM packages are managed by a Learning Management System (LMS) that, among other important functions such as administration, tracking, and reporting, provides also a mean to display this information to the users. In order to correctly deliver the contents, the LMS parses the package metadata to understand the course structure, known as the "activity tree", and to know how to launch each SCO. Since most of times the material takes the form of slides, "traditional" LMS use a fixed HTML frameset structure, as shown in Fig. 3, that includes a list of the available lessons, a set of navigation buttons, and a panel where the slides are shown.

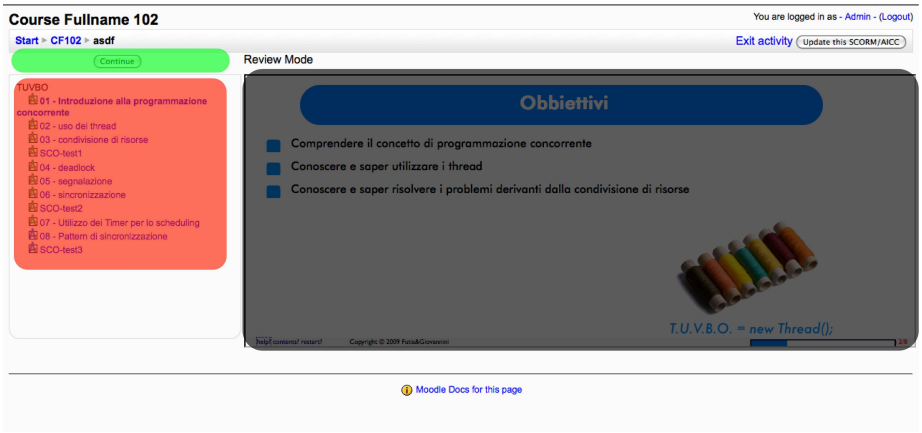

Fig. 1. Screenshot of the web interface of the Moodle Learning Management System.

This kind of interface presents some limitations:

- The SCO designer can completely define the content and presentation of each single SCO, but it can not control at all the interface used to present and navigate it (that is part of the LMS functions);
- The SCO designer's work area is limited by the dimension of the HTML frame defined by the LMS, thus it is difficult to integrate different media and materials, i.e., a video of the teacher, slideshows, comments;

- SCOs are included in the LMS with the graphical style defined by their own SCO designer, style that can be unhomogeneous if an LMS integrates different SCO sources. If content and presentation where independent, it would be instead easy to redefine each SCO in order to present an homogeneous presentation;
- A fixed interface and the impossibility to separate presentation from content make hard to adapt the interface to different displays, i.e., smartphones, tablets, netbooks, etc.

To overcome these limitations, we present *Zenaminer*, a web service to export SCORM packages through a RESTful web API, that is a web service implemented using HTTP and the principles of REST (REpresentational State Transfer). An architecture where a SCORM package is not shared as a single object (the ZIP packet), but as a collection of smaller objects, the SCOs, each of them accessible independently from the others. It is the implementation of the new paradigm of the "Web of Data", SCOs are publicly available as read/write raw data that can be retrieved or updated using a REST interface as described in the next sections.

The key factors that enable Zenaminer to provide such a functionality are two:

- the knowledge of the SCORM content packaging standard, to import the ZIP files;
- the knowledge of the content "internals", i.e., how the slide information is defined, to separate the content from the presentation.

Zenaminer, for the creation of HTML documents, suggests the introduction of a "light formalism" proposed by the W3C with the name Slidy [11]. Slidy defines simple keywords for the class attribute of HTML tags that can be used to identify HTML elements as slides, titles, sidebar, incremental lists, etc. – each of them with a particular presentation style defined by CSS or behavior implemented in Javascript.

In short, SCORM contents are available as resources of the web service given a specific URI. The SCO designer can freely define the interface as he likes it, then contents will be dynamically retrieved from Zenaminer using AJAX (Asynchronous Javascript and XML) calls. As illustrated in Section 5, web API give access to each single slide of the course as well as to the table of contents for the navigation. In addition Zenaminer extends the SCORM metadata to include also references to additional information such as the video recording of a lesson, synchronization information between video and slides, etc.

Taking into consideration another characteristic of what is defined as e-learning 2.0, that is collaborative learning, Zenaminer enables students to enrich the SCORM resources with comments on the course, the lesson, or the slide. Comments can be contents themselves that extend and improve the original material provided by the teacher. Furthermore, unstructured comments are au-

tomatically annotated with DBpedia Spotlight[5] linking them to the Linked Open Data cloud through DBpedia [1][2].

4 Content presentation

In order to structure the presentation of contents we adopted the Slidy formalism, which uses HTML for the description of its items. The choice to adopt HTML becomes important in our approach because it opens to all devices able to connect and visualize Web pages and, more important, it intrinsically performs the separation of contents from the presentation (also called view). According to that, we may define data able to exchange information and how this data may be used to build the presentation: this is obtained with the use of the Cascading Style Sheet (CSS) file. By means of it, a view maker may define models to render specific items or group of them. These models are named class and are referred to an HTML page. The interaction between HTML page and CSS is performed by means of selectors, which are able to point to specific items or class of an HTML page. In this context, the Javascript, a "scripting" language, performs the possibility to select an item or a set of them and to change the behaviour of the view, previously defined. In addition, it works as a tool to enrich dynamically the presentation, e.g. making table of contents (TOC), suggesting the value of an acronyms, changing font or window dimensions. Then, it performs an important role to dynamically access structured data without any layout details, or raw data, which are coming from different archives spread within the Web cloud. The technology that allows to gather data from external sources is the Asynchronous Javascript and XML (AJAX).

Our work exploits the Separation of Concerns (SoC) principle, introduced above, in the context of a Learning management system. The difference between data and view allows to navigate through resources, making customizable views which better respond to the need of who provide contents (author) and who use, reuse and redistribute them (user). Data is presented without layout details, raw, but it is rich of semantics: data is stored with the information about inference to others and are exposed to users by means of REST APIs. Our approach exploits the MVC (Model-View-Controller) paradigm, in which the view is the set of presentation rules for the data, the Model is the amount of data raw available for a generic topic, while Controller is the set of methods which are needed to create the communication channel from the Model and the View. Our data Model is deployed in the system which it holds SCORM packages, while the View is created on all user machines whenever a user require it. The Controller, instead, is located on the user machine where the view is created and on the LMS where SCORMs are available.

To provide the maximum interoperability and reusability, contents indexed and managed by our LMS are slides, lectures and courses. All of these are Web resources available by URIs time-invariant. This feature is important because we

[5] http://dbpedia.org/spotlight

need to performs durability and reusability to each Web resource. Slide contents are described by means of HTML Slidy formalism; each slide is composed of title, pictures, vertical and horizontal scroll bars, interactive items (buttons, check box, etc.). Lectures are organized as follows: slides and video fragments. A fragment is a part of video which is related to a particular lecture. To map lecture and fragments, we introduced a synchronization file. A video is available and accessible by means of a URI with the above requirements. Course, lecture and slide exploit the SoC principle, separating the view from the data model. The video, instead, holds the information about its presentation and it is shared without changing.

5 Architecture

The architecture is structured in four building blocks that are summarized in Figure 2:

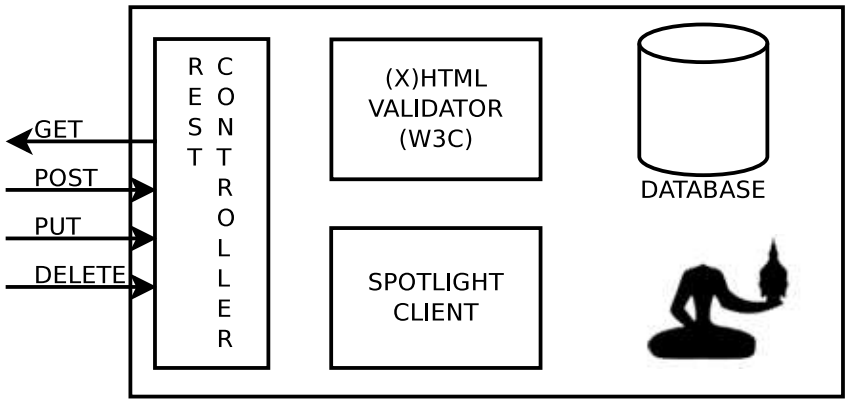

Fig. 2. Zenaminer architecture. The REST controller is the interface between Zenaminer and the Internet; the W3C validator is used to check that imported (X)HTML files are well-formed; the Spotlight client provides automatic annotation; the database stores imported SCORMs together with comments and annotations.

1. The REST controller is the interface between Zenaminer and the Internet. Its features are grouped in three sets (as shown in Figure 3): user management and authentication (login), get and post of comments (comment) and management of SCORM packages (scorms, item, outline, sync and slide). Each of these features is implemented using the REST architecture, thus resources are available through URIs in the Internet. Access to REST calls that provide access to (GET) or modify (POST, PUT, DELETE) existing contents can be limited using the existing facility for user management and authentication.

2. A local validator is used to verify that (X)HTML files are valid and well-formed before they are parsed. When a new SCORM package is imported each HTML file that contains slides is validated using the remote interface offered by the official W3C validator[6].
3. When a comment (an enrichment) is received it is passed to the Spotlight client. The Spotlight client sends a request to DBpedia Spotlight service and receives an annotated version of the comment.
4. The SQL Database is used to store imported SCORM packages together with comments and annotations.

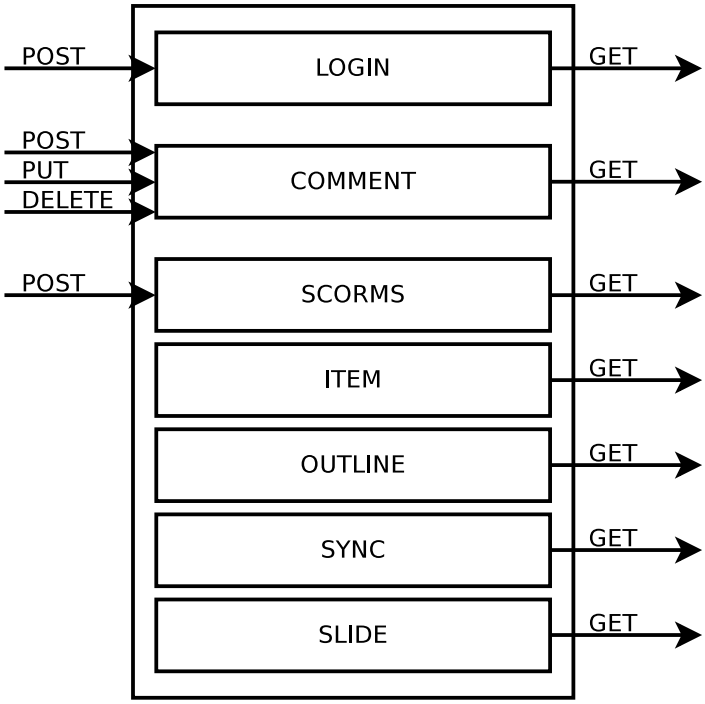

Fig. 3. Detail of the REST controller, its three main features are: user management, comment management and SCORM management.

Figure 4 shows the workflow for the upload of a SCORM package into Zenaminer, it is performed in the four steps below:

1. the client sends a POST request to /scorm page, including in the request body the SCORM package;

[6] http://validator.w3.org

2. the REST controller receives the package, unpacks and analyzes the files containing slides and sends them to the W3C validator;
3. the W3C validator validates individual files and reports the result to the REST controller;
4. if all files have been validated correctly the package is imported into Zenaminer. Otherwise if files are not validated Zenaminer sends to the client an error message containing errors generated by the W3C validator.

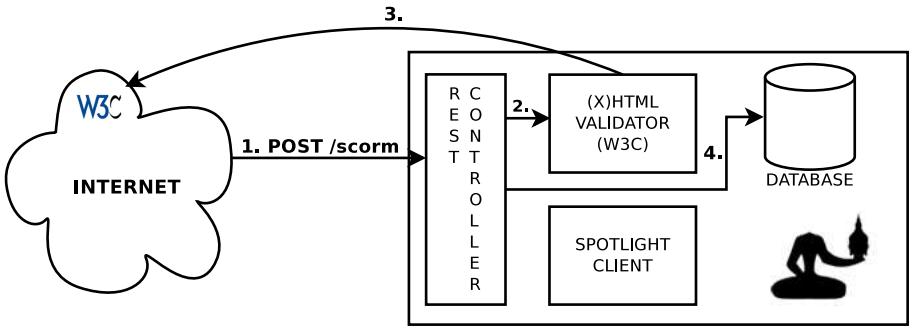

Fig. 4. Workflow describing the upload of a SCORM package in Zenaminer: 1. a SCORM package is received from a client; 2. (X)HTML files are forwarded to the validator; 3. (X)HTML files are validated by sending requests to the offical W3C validator; 4. if all documents are valid, the SCORM package is stored to the database.

Zenaminer gives the possibility to its users to enrich the content of a given content (e.g. a slide) by adding a textual comment to it. Such comments are named enrichments because they are meant to add more information to the content of a SCO, for example an enrichment to a slide could be a proof to a theorem, a correction to its content, an in-depth explanation to a concept, etc.. In order to avoid that content of an enrichment is buried in the database we added automatic annotation in order to enable machines to access to the content, enhancing search search and the possibility to display complementary information.

Figure 5 shows the actions performed by Zenaminer when a comment is received:

1. the client sends a POST request to /comment page, the message body contains the text of the comment;
   An example of enrichment is:
   `President Obama called Wednesday on Congress to extend a tax break for students included in last year's economic stimulus package, arguing that the policy provides more generous assistance.`
2. the text is forwarded to the Spotlight client;

3. the Spotlight client sends a request to DBpedia Spotlight and obtains as response an automatically annotated version of the comment. As an additional parameter to the annotate request we could pass a minimum confidence and support values that will impact on precision and recall of the annotation itself;
   The request URI to annotate the enrichment in the previous example is:
   `GET http://spotlight.dbpedia.org/rest/annotate?`
   `text=President%20Obama%20called%20Wednesday%20on%20Congress%20to`
   `%20extend%20a%20tax%20break%20for%20students%20included%20in%20last`
   `%20year%27s%20economic%20stimulus%20package,%20arguing%20that`
   `%20the%20policy%20provides%20more%20generous%20assistance.`
   `&confidence=0.4`
   `&support=20`
   The response (in JSON format below) contains a reference to the annotated term in the enrichment (@ surfaceForm) and the DBpedia URI relative to the annotated content (@URI):


   ```
   {
   "@URI": "http://dbpedia.org/resource/United_States_Congress",
   "@support": "9321",
   "@types": "Organisation,Legislature",
   "@surfaceForm": "Congress",
   "@offset": "44",
   "@similarityScore": "0.14802740514278412",
   "@percentageOfSecondRank": "0.6257434730652487"
   }
   ```

4. finally, the enrichment and its annotations are stored into the database.

A SCORM package includes a XML file named “manifest”, it describes a course and is used to list all lectures in it, all resources (files) associated to each lecture and some sequencing rules used to define how the user accesses to SCOs. Figure 6 shows in a tree structure how the manifest file is represented into the relational database. In addition, a SCORM package lists several SCOs, each SCO includes an HTML file containing slides, a CSS file to describe the style, a Javascript file to describe the behavior. Additional files can be included in a SCO: videos, subtitles or files describing synchronization between a video and the slides. Files linked into the manifest are then packaged together into a ZIP file. Currently Zenaminer does not support sequencing rules, thus management of such rules is up to the SCO designer.

6 Use case

The validation phase of the Web-service was conducted during the course “Multimedia Environments” (Academic Year 2010/2011) for the Master of Science in Cinema and Media Engineering at the Politecnico di Torino. Zenaminer was thought as a service in order to design e-learning projects (SCORM Packages)

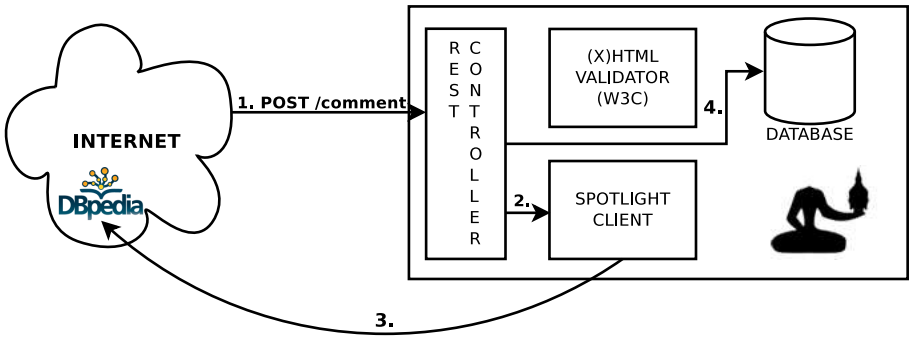

Fig. 5. Workflow describing the reception of a comment in Zenaminer: 1. a comment is received from a client; 2. the content of the comment is forwarded to the Spotlight client; 3. the Spotlight clients contacts DBpedia Spotlight to obtain an annotated version of the comment; 4. the comment together with annotations are stored to the database.

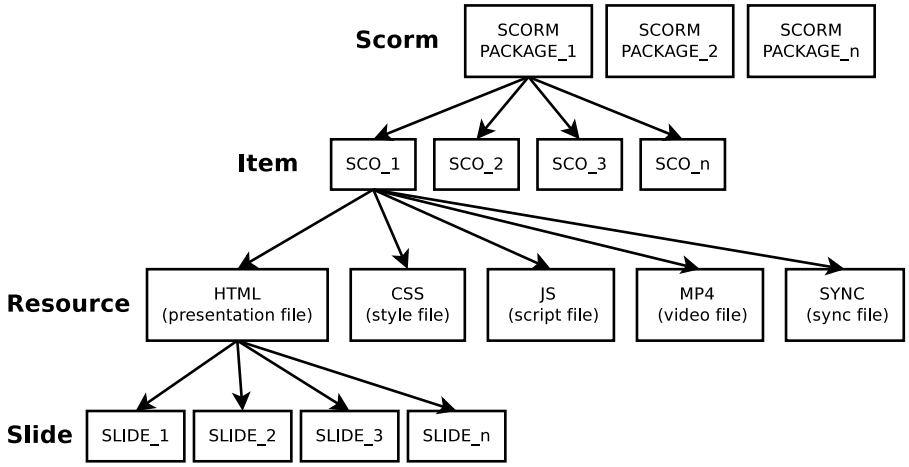

Fig. 6. A tree representation of a SCORM package in Zenaminer. A SCORM package is a set of SCOs, each SCO represents a lesson in a course, each SCO contains several files like HTML, CSS, JS or videos. HTML files containing a presentation are parsed and slides are extracted.

for the course "Multimedia Environments". The objective for the students was to create a SCORM package defining both content and presentation. Students were divided in 20 teams, each team defined a personal learning environment building different interfaces (using the SoC concept). Students acted as SCO designers, the projects were the use case for Zenaminer and we verified the potentiality of the separation of content from presentation. The demo is available at http://eridano.polito.it:8080. Figure 7 and 8 show two different views of the same raw data done by two different groups of students. Such interfaces are able to show same contents in two different ways. The controller of each interface collects the list of SCORM packages stored into Zenaminer. Depending on the SCORM package selected by the user, the controller gathers the entire set of lectures for that package and displays the related Table of Contents (ToC). The user is, then, free to navigate the lectures (SCOs) following the ToC, when a lecture is selected the controller requests all files associated to that lecture and the view is updated accordingly. For example, in both figures the selected SCO had a set of slides and a video associated to it. Thanks to the sync file, the view is able to synchronize the video with the slides.

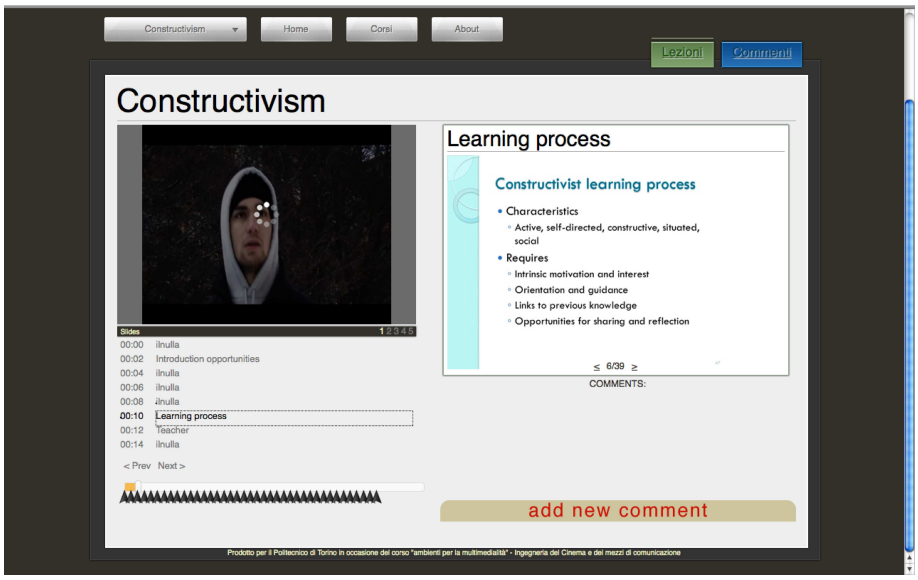

Fig. 7. One of the user interfaces designed by students for the course Multimedia Environments.

The Web Service was developed with the framework Pylons 1.0[7]. The source code[8] is released under the GNU GPLv3. In order to better balance and dis-

[7] http://pylonshq.com
[8] https://sourceforge.net/projects/zenaminer/

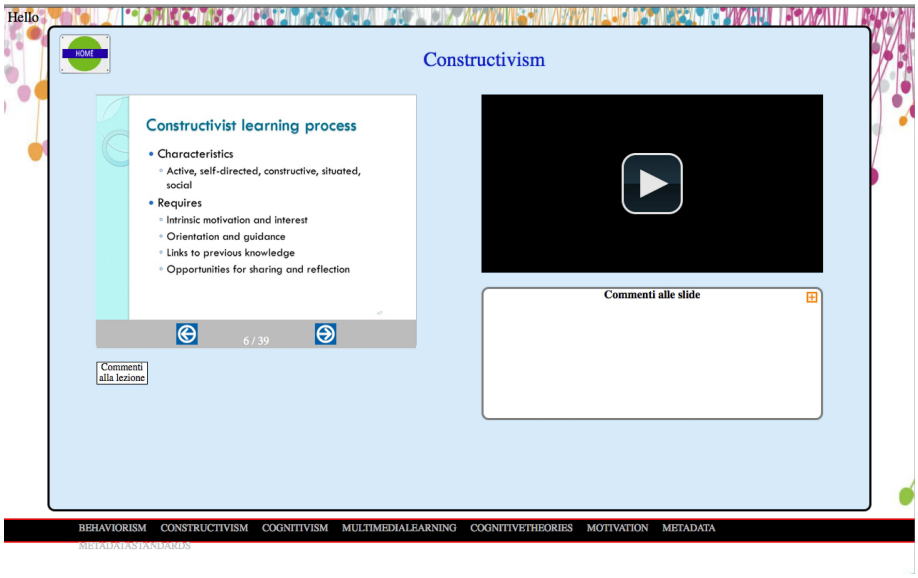

Fig. 8. Another example of user interfaces designed by students for the course Multimedia Environments.

tribute the load we decided to use Apache 2.2 HTTP Server[9] as interface between the requests towards the Web Service. As relational database we used PostgresSQL 9[10]; it stores data extracted from the SCORM by the web-service. Raw data stored in the database are available through REST calls and serialized into JSON format. The W3C Markup Validation Service is used to validate the HTML of the lectures that must be conform to W3C standards. Finally we used Spotlight to automatically annotate the enrichments of learners.

7 Conclusions

In this paper we proposed and argued a Web Service Architecture for e-learning. Zenaminer goes beyond the limitations of the SCORM and allows a complete separation of content and presentation. Content creators are free to define learning environments without constraints of traditional LMSs. Zenaminer does not limit the definition of the interfaces, the graphic design and the learner interaction with a SCO. In the field of e-learning the presentation (the interface) does not only involve the aesthetic appeal and the usability but also affects the learning process. In fact the interface design [13] and the adaptation of interfaces to the learner [9] are two key factors for e-learning success.

[9] http://httpd.apache.org
[10] http://www.postgresql.org/

The "Separation of Concerns" (SoC) allows SCO designers to improve e-learning acceptance. For example a SCO designer could create different environments for the same course according to the learners' computer literacy or could improve the interoperability of SCOs designing different interfaces for different devices (e.g. Smart Phones). Not secondary is the possibility to integrate and manage multimedia contents as separate entities from the learning content. In traditional LMSs a multimedia content must be integrated in a SCO while, in Zenaminer, content-types other than text (e.g. videos) can be managed and integrated regardless of SCOs. This feature does not compromise the SCORM compatibility with traditional LMSs.

Finally Zenaminer allows collaborative learning. Learners have an active role in the definition of contents thanks the possibility to enrich lectures with personal contributions. Content creators are free to define different models of collaborative learning; they can implement a Wiki model or they can choose a model with less collaboration. In addition to collaborative learning it is possible to integrate contents thanks to instruments offered by the Semantic Web. The Automatic annotation allows the integration between of content with hypertext link navigation in order to satisfy more effectively the information needs of the learner. Furthermore the SCO designer could decide to allow learners to disambiguate or integrate automatic annotations. The use of ontology-based annotation pushes Zenaminer to be a Linked Data LMS. The architecture proposed in this paper allows Instructional designers to better implement learning theories (e.g. Cognitivism, Constructivism) and teaching strategies. For example they could choose to minimize the content of a course and focus on collaborative learning; this approach satisfies the key aspect of Constructivism that considers fundamental the collaboration between learners [15]. Moreover this work represents a step towards the publishing of LMS related data to the Web of Data and new future works are planned. First of all, to be totally compliant to the Linked Data paradigm we want to investigate about the conversion of LMS related data to RDF, and to expose them using a SPARQL endpoint. Then we want to investigate about the use of disambiguation process through the DBpedia Spotlight. The idea behind is offer to users the possibility to better understand ambiguous concepts by means of the semantic disambiguator. This tool may highlight ambiguous concepts in a SCO and may suggest multiple references per each ambiguous concept, then user may choose the most interesting.

Acknowledgments

Thanks to all students of the course "Multimedia Environments" academic year 2010/2011 of the Politecnico di Torino. They were developers and analysts of our Web Service with a vibrant cooperation spirit, which was fundamental in achieving this important goal. A special thanks to SCO designers Jacopo Berta, Federica Tina Bossa (who developed the SCO view shown in Figure 7) and Lucia Marengo, Alice Ferrari (who developed the SCO view shown in Figure 8).

References

1. Berners-Lee, T.: Linked data. International Journal on Semantic Web and Information Systems 4(2) (2006)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Web Semantics Science Services and Agents on the World Wide Web 7(3), 154–165 (2009)
3. Díaz Redondo, R., Fernández Vilas, A., Pazos Arias, J.: Educateca: A web 2.0 approach to e-learning with SCORM. In: Cellary, W., Estevez, E. (eds.) Software Services for e-World, IFIP Advances in Information and Communication Technology, vol. 341, pp. 118–126. Springer Boston (2010)
4. Dong, B., Zheng, Q., Qiao, M., Shu, J., Yang, J.: BlueSky cloud framework: An e-learning framework embracing cloud computing. Lecture Notes in Computer Science Cloud Computing 5931, 577–582 (2009)
5. Dong, B., Zheng, Q., Yang, J., Li, H., Qiao, M.: An e-learning ecosystem based on cloud computing infrastructure. In: Proceedings of the 2009 Ninth IEEE International Conference on Advanced Learning Technologies. pp. 125–127. ICALT '09, IEEE Computer Society, Washington, DC, USA (2009)
6. Downes, S.: E-learning 2.0. eLearn 2005 (October 2005)
7. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Transactions on Internet Technology 2(2), 115–150 (2002)
8. Gonzalez, M., Penalvo, F., Guerrero, M., Forment, M.: Adapting lms architecture to the soa: An architectural approach. In: Proc. of the Fourth International Conference on Internet and Web Applications and Services, ICIW '09. pp. 322 –327 (May 2009)
9. Mödritscher, F., Barrios, V.M.G., Gütl, C.: Enhancement of SCORM to support adaptive e-learning within the scope of the research project AdeLE. In: Nall, J., Robson, R. (eds.) Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2004. pp. 2499–2505. AACE, Washington, DC, USA (2004)
10. Murugesan, S.: Understanding web 2.0. It Professional 9(4), 34–41 (2007)
11. Raggett, D.: Slidy - a web based alternative to Microsoft PowerPoint. W3C (2006)
12. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media, Inc. (May 2007)
13. Selim, H.M.: Critical success factors for e-learning acceptance: Confirmatory factor models. Computers & Education 49(2), 396–413 (September 2007)
14. Vossen, G., Westerkamp, P.: Why service-orientation could make e-learning standards obsolete. International Journal of Technology Enhanced Learning 1(1), 85–97 (2008)
15. Wilson, B.G.: Constructivist Learning Environments: Case Studies in Instructional Design. Educational Technology Publications, illustrated edn. (1996)