AN IMS-LEARNING DESIGN PLAYER BASED ON COPPERCORE ENGINE

José R. Hilera, José Escribano, Roberto Barchino, José M. Gutiérrez, Salvador Otón,
José J. Martínez, José A. Gutiérrez, Luis De Marcos
Department of Computer Science
University of Alcala
Alcala de Henares (Madrid) - Spain

ABSTRACT

In this paper we present a simple open source system programmed in Java that aims to analyze the basic services that offers the Coppercore engine from the Open Universiteit Nederland. Coppercore services can be used in any e-learning system to ensure conformance and compliance with the basic requirements of the IMS Global Consortium Learning Designs Specification. The *player* developed can facilitate understanding of the possibilities offered by this IMS-LD engine, and it can be extended in order to create more complex environments.

KEYWORDS

Learning Design, Unit of Learning, Coppercore engine.

1. INTRODUCTION

IMS-Learning Design (LD) is an IMS Global Consortium specification, whose goal is to provide a containment framework of elements that can describe any design of a teaching-learning process in a formal way (IMS, 2003a). In order to support the description of individualized learning designs, learner Properties, Conditions, and Notifications are needed. The designs can be described by a meta-language, based on EML (Hermans et al., 2003), and they might involve a single user or multiple users; the learning and instructional designers and providers might take a behaviorist, cognitivist, constructivist, or some other approach; they might require learners to work separately or collaboratively. These could all be captured in terms of a *Method* containing *Roles*, *Activity*-structures, *Environments* and other related concepts (Koper, 2001).

IMS-LD is a pedagogically neutral specification, that supports mixed mode delivery ('blended learning'), enabling traditional approaches such as face-to-face teaching, the use of books and journals, lab work, and field trips to be also specified as learning activities and combined with ICT (Information and Communication Technologies) supported learning. What it brings to mixed mode teaching is the ability to specify both kinds of learning in a Unit of Learning (UoL) that is itself in digital form. The primary use of IMS Learning Design is to model units of learning by including an IMS Learning Design in a content package, preferably but not necessarily, an IMS Content Package (IMS, 2004). An IMS Content Package is a zip file, and describes their contents in an XML document called the 'package manifest', integrated in the file itself. To create a UoL, IMS Learning Design is integrated with an IMS Content Package by including in the package manifest the learning design element as another kind of organization within the <organizations> element.

```
<manifest>
  <metadata/>
    <organizations>
      <learning-design>
         [learning design elements here]
      </learning-design>
    </organizations>
    <resources/>
</manifest>
```

Learning Design specifies three levels of implementation and compliance. Learning Design Level A includes everything described so far. It thus contains all the core vocabulary needed to support pedagogical

diversity. Levels B and C add three additional concepts and their associated capabilities in order to support more sophisticated behaviors. Learning Design Level B adds Properties and Conditions to level A, which enable personalization and more elaborate sequencing and interactions based on learner portfolios. It can be used to direct the learning activities as well as record outcomes. The separation of Properties and Conditions into a separate Schema also enable it to be used independently of the rest of the Learning Design Specification, typically as an enhancement to IMS Simple Sequencing (IMS, 2003b). Learning Design Level C adds Notification to level B, which, although a fairly small addition to the specification, adds significantly to the capability, but potentially also to the implementation task where something similar is not already in place (IMS, 2003a).

As it can be observed in figure 1, the design of the learning ("Design Time" section), can be made by means of the description of elements like roles, activities, frameworks, methods, properties, conditions and notifications; using, for it, a publisher of learning units, like ReLoad or CopperAuthor. Next, a run time environment is needed to control the execution of the learning process ("Run Time" section), by means of the activation of a software denominated generically Learning Design (LD) Player. One of the most used is ReLoad Player (*http://www.reload.ac.uk*), that is based on a Web engine named CopperCore (*http://coppercore.sourceforge.net*), developed by the Open Universiteit Nederland (OUNL) ("Run time" section).

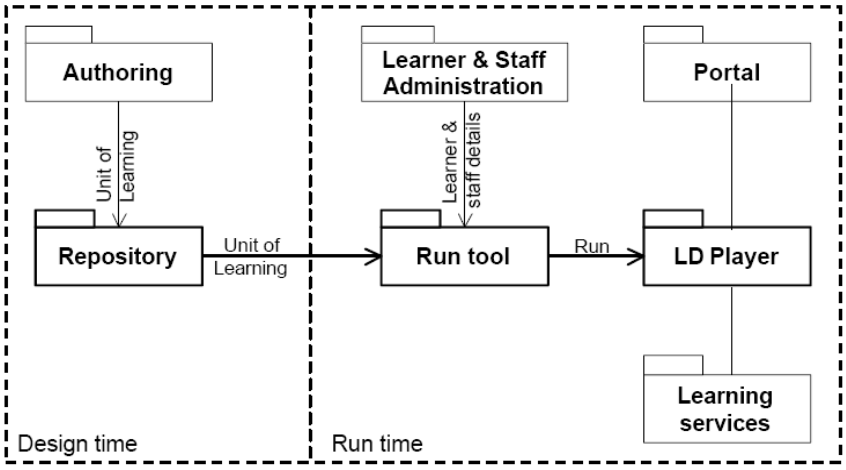

Figure 1. Components involved in the design and execution of a Unit of Learning (IMS)

CopperCore is an open source IMS Learning Design Engine that supports all three levels of IMS Learning Design (A, B and C). IMS Learning Design is a complex, semantically rich specification, so it is not trivial to provide full support for it. IMS Learning Design specifies a template of a synchronized and personalized workflow through a course. In the runtime environment it must be use this template to provide a user with an up-to-date view on his or her learning process. So, for example, when a UoL specifies a group assignment where all learners need to complete a specific activity before they can proceed to the next activity, the runtime environment should check this constraint and it should synchronize access to the second activity by continuously checking to see if all users did already complete their first activity. All this checking, synchronizing and personalizing is called the business logic of Learning Design, and this is exactly what CopperCore handles for the developer. By implementing this business logic, CopperCore hides the developers from these complexities when incorporating the IMS Learning Design specification.

CopperCore is a J2EE (Java Platform Enterprise Edition) runtime engine which can be used to incorporate IMS Learning Design to an e-learning system, providing to developers three API's (Application Programming Interfaces) and a Test Suite. Some characteristics are:

- full support for IMS Learning Design including level A, B and C
- has three API's covering publication, administration and delivery of IMS Learning Design
- exposes J2EE, native Java and SOAP (Simple Object Access Protocol) interfaces
- provides a validation library
- includes a command line interface to most of the API calls

- includes an example of a publication interface
- includes an example of a web delivery interface
- is platform independent
- has built-in support for three relational databases (MS SQL Server/MSDE, PostgreSQL and HSQLDB)
- is ready for use with JBoss application server, but runs on other application servers as well
- is licensed under the GNU GPL

Following sections of this document describe the design and implementation of a simple LD Player for the Level A IMS-LD specification. It is programmed as a Java application which works with the run engine Coppercore, that allows the user: (1) to load a UoL (for example, created with Reload editor), (2) to administer the different executions, users and roles which can interact with the UoL and (3), finally, to visualize the didactic content according to the profile of the user. This work is a first approach to the use of Coppercore engine that can serve as aid to developers that wish to incorporate IMS-LD in an e-learning platform. The source code of the application developed can be downloaded from *ftp://www.cc.uah.es/software/ldplayer1.zip*.

2. IMS-LD PLAYER DESIGN

The architecture of developed software is made up basically of three elements programmed in Java (figure 2):

- *Publisher*: Java Web application with an interface that allows to load, in the Coppercore engine, packed Units of Learning according to specification IMS Content Packaging, created previously using an LD editor (for example ReLoad).
- *Manager*: Java Windows application to select the UoL to visualize among the loaded ones in the Coppercore engine, to create executions and new users to support, and to assign them a role within the available ones for the selected UoL.
- *Player*: HTML application to show the execution of the UoL selected for a user with a certain role. This execution will contain the tree of content of the UoL for that user profile, and will allow navigate the tree to show the content of the UoL. This content corresponds with the course materials that are provided according to the level of knowledge or the interests of the use".

Publication system is an application that is made up of an Web interface where the user introduces the location of the UoL to load. Once it is selected, the request is sent to a class called *Publisher_system*, that contains a method *Post* that loads the UoL in the execution engine; this method uses a private method named *load*, that uses the interface of the Coppercore engine come with its load. Once finalized the load process, a Web page with the result is generated, which contains a report of each step of the load, including errors encountered. For integration with Coppercore engine, the following classes offered through the API of this engine are used:

- *org.coppercore.common.Message*
- *org.coppercore.common.MessageList*
- *org.coppercore.delegate.LDCourseManagerDelegate*
- *org.coppercore.dto.PublicationResult*
- *org.coppercore.dto.ValidationResult*
- *org.coppercore.exceptions.ConfigurationException*
- *org.coppercore.interfaces.LDCourseManagerHome*

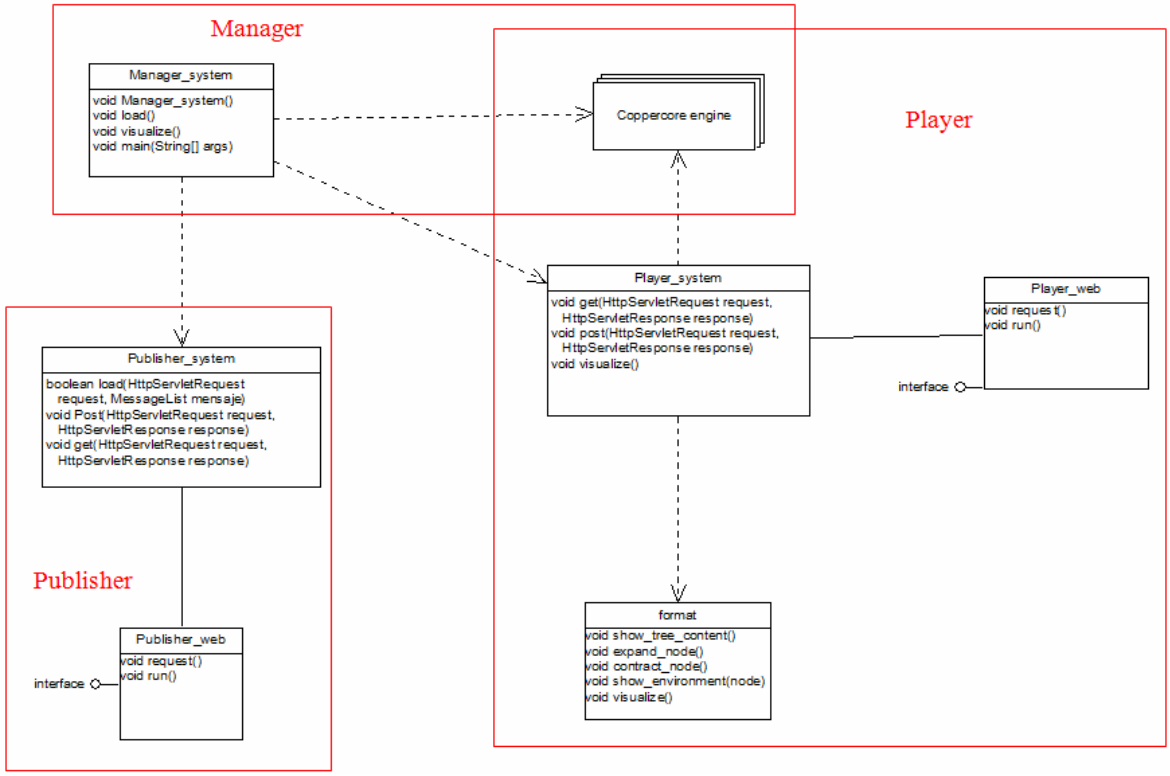

Figure 2. Class architecture of the software developed

The administration system consists of an application with a graphical interface that offers to the user these functions:

1) Load a UoL in the run engine, calling the publication application.
2) Create, or select an execution previously created, that will be supported by the run engine for a certain UoL.
3) Create, or select a user created previously, who will be the user that will operate with the execution.
4) Assign one of the roles defined by the UoL to the assigned user.

All these operations are implemented with events that are transformed into calls to the interface of Coppercore, using the following classes of this engine:

- *org.coppercore.common.Message*
- *org.coppercore.common.MessageList*
- *org.coppercore.delegate.LDCourseManagerDelegate*
- *org.coppercore.delegate.LDEngineDelegate*
- *org.coppercore.dto.RunDto*
- *org.coppercore.dto.UolDto*
- *org.coppercore.exceptions.CopperCoreException*
- *org.coppercore.exceptions.ConfigurationException*
- *org.coppercore.interfaces.LDCourseManagerHome*

Finally, the application for visualization is activated by the Administration System, once selected the UoL, the execution and the user (with a selected role). This application consists of a Web interface that receives the parameters of user and execution. The content of the interface is created from requests to the class *Player_web*. This class reads the content tree included in the UoL, and returns XML code that is transformed by means of extensible stylesheet XSL (W3C, 2006), so that it can be visualized by a Web browser. The set integrated by XSL files for format conversion, Javascript files to control how to expand or to contract the content tree or to show the context of a certain node of the tree, and the style sheets CSS, is represented as the *Format* class in figure 2.

For the player creation, the compiled version of the Coppercore engine has been used (available in *http://coppercore.sourceforge.net/downloads.shtml*). This software contains all that is required to execute the

run-time engine, and provides the interface that services the created player. For that, packages' series must be created to replace Java packages from the original Coppercore application; these packages are (figure 3):

- *publisher.war*: This package contains the publication system, to allow the loading by units of learning in the Coppercore engine.
- *CopperCore-client.jar* y *CopperCore-common.jar*: These two packages form the management system, and they manage the unit of learning, executions, users and roles that the Coppercore engine will handle.
- *webplayer.war*: This package contains the visualization application that allows to show the content of the UoL, according to the execution, the user and the selected role.

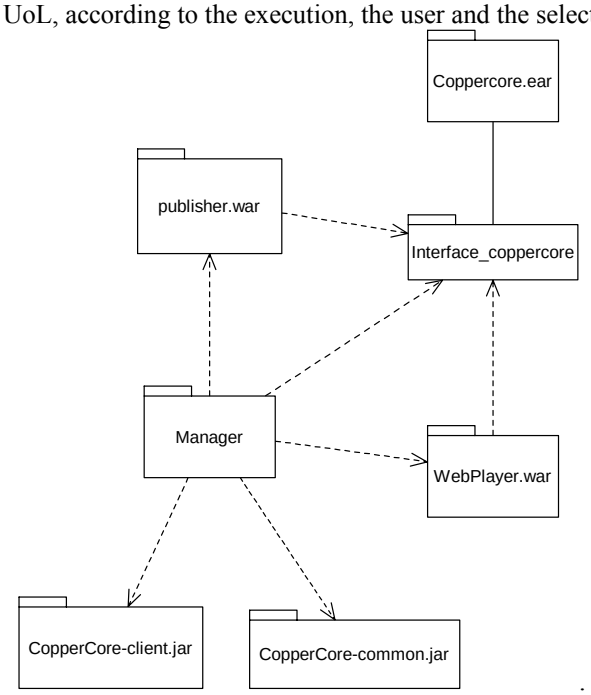

Figure 3. Java packages generated

3. IMS-LD PLAYER OPERATION

In order to be able to execute the Coppercore engine the Java 2 Standard Edition SDK and a Web Java application server, such as JBoss, must be installed. Coppercore engine can be started executing a command *coppercore.bat* within the folder in which the developed application is stored. Next, to start the application, is necessary to execute a batch file named *ldplayer.bat* to configure file system properties.

Once started, the management system is executed (figure 4). First, the Unit of Learning that we want to visualize must be selected, so we press the *Load* button. This button launches the publication system, which opens in a new web browser window (Internet Explorer in figure 5). In that window, the *Browse* button can be pressed in order to select from the file browser the zip package that contains the UoL that is going to be loaded. Once selected, by clicking the *Load* button it begins the UoL loading process in the environment. Once finalized, a summary report will show us the load process result (Figure 6); if it appears some message that begins with the string "(ERROR)", it shows that any failures exist in the UoL that was being loaded. In any another case the load process was correct and we may proceed to close by pressing the *CLOSE* button from the report window.

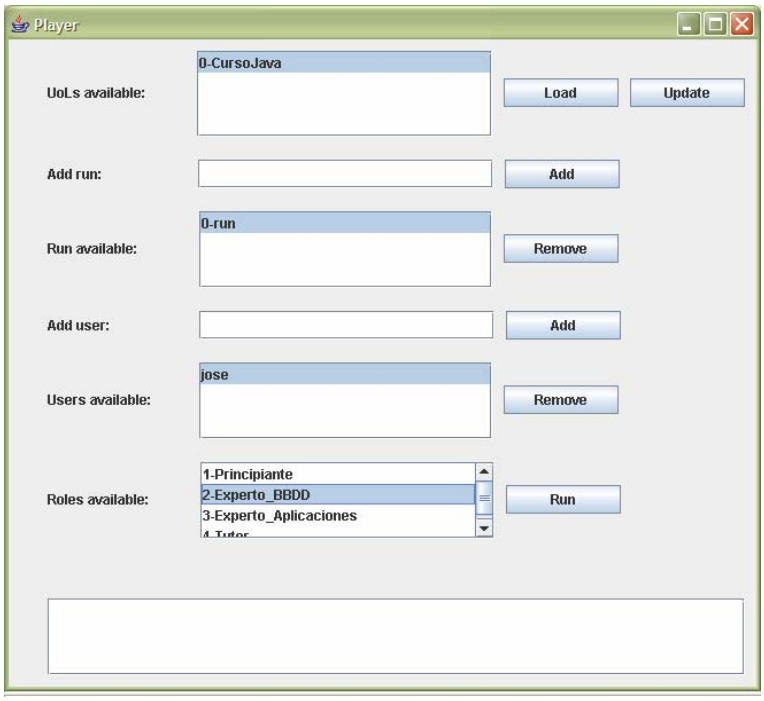

Figure 4. User interface of the Management System

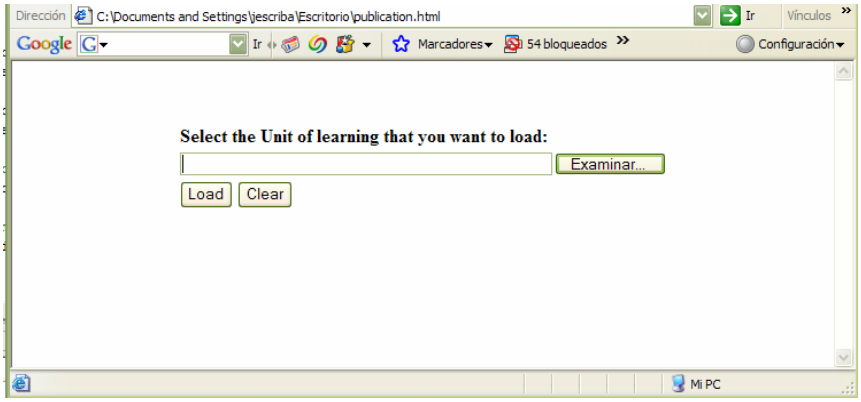

Figure 5. User interface of the Publication System

Next, so that they appear in the management user interface the data about the UoL that has been loaded, is due to press on the button *Update* and select UoL that is desired to execute of the list of units of learning available. The following step would be to give a name to the execution that is going to create; for it a new execution can be created, writing a name in the field *Add run*, and pressing the button *Add*, or one of the list *Run available* can be selected. Then, a user for the execution is due to create; this one can be created writing its name in the field *Add user* and pressing the button *Add*, or one in the list *Users available* can be selected. Finally, the role of the user must be selected from the list *Roles available*; this information is provided by the UoL, so it is not necessary to create no new role. Once selected the Uol, the execution, the user and his role, it has to press on the button *Run* so that the visualization window is opened (figure 7).

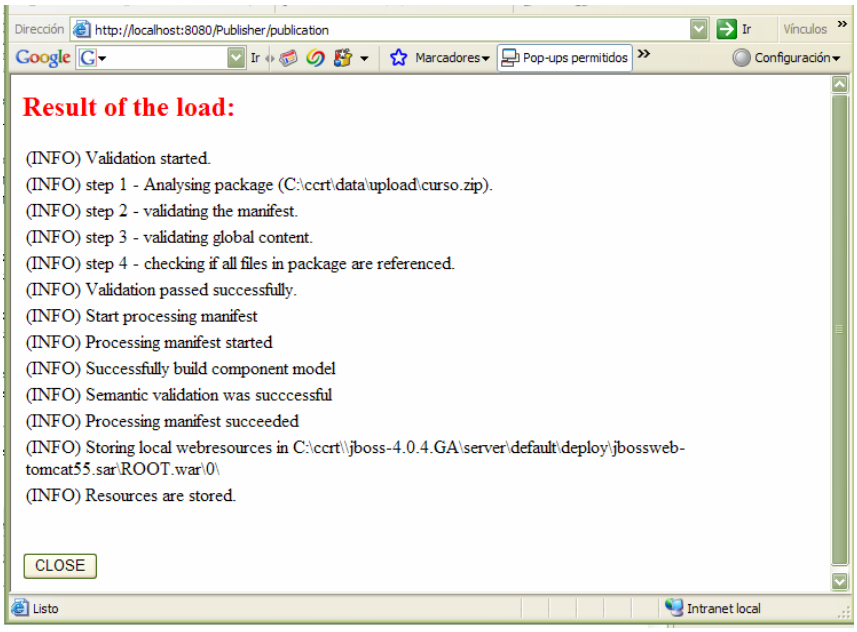

Figure 6. Summary report about the the load of a UoL

In the visualization window (Web browser) then appears the name of the selected UoL, followed by the root of the execution tree. By clicking on the icon with symbol "+", it is possible to unfold the content of the course contained in the UoL (figure 7 shows a course example about "Java language"). After unfolding the content, if it is selected any of the objects, a description of its content appears. In addition, if the object contains an associate environment this will appear in the lower left corner with the word *Environment*.

The execution environment can also be unfolded, since in a UoL, within the environment of each object, it is stored course content that can be visualized depending on the profile of the user. By clicking on any content element, corresponding course materials will be loaded in the left window (figure 7).

4. CONCLUSIONS

The Coppercore run engine provides a complete interface for the creation of players that allows to explore all the characteristics of specification IMS Learning Design in the three levels (A, B and C). The player developed allows to give a vision of the possibilities of the Coppercore engine for level A, supporting the most elementary contents, but it can be extended in order to allow to create more complex environments, like complete forums, in which the users can interact among them and with the tutor to solve their doubts on the education process. And even monitoring how it is developed the process of learning by the different users to obtain statistics of processes and possible improvements of the learning. The authors' intention is to extend the player functionality with characteristics of IMS Learning Design levels B and C, and to integrate it in the EDVI e-learning platform (Barchino et al., 2005) that was previously developed in the University of Alcala. EDVI supports well known specifications in this context: SCORM (ADL, 2006) or IMS QTI (IMS, 2006).

Figure 7. Visualization system playing the learning design of a "Java Language" course

ACKNOWLEDGEMENT

This research is co-funded by: (1) the Spanish Ministry of Industry, Tourism and Commerce PROFIT program (grants FIT-350200-2007-6 and FIT-350101-2007-9) and Plan Avanza program (grant PAV-070000-2007-103), (2) the Spanish Ministry of Education and Science PROFIT program (grant CIT-410000-2007-5) and (3) Castilla-La Mancha autonomous community under the educational innovation cooperation program (grant EM2007-004). Authors also want to acknowledge support from the TIFyC research group.

REFERENCES

ADL, 2006. *Sharable Content Object Reference Model (SCORM)*. ADL Advanced Distributed Learning. http://www.adlnet.gov/scorm/

Barchino, R., Otón, S. and Gutiérrez, J.M., 2005. An Example of Learning Management System. *Proceedings of IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005).*

Hermans, H., Manderveld, J. and Vogten, H., 2003. *Educational Modelling Language*. Open University of the Netherlands. http://hdl.handle.net/1820/77/

IMS, 2003a. *IMS Learning Design Specification*. IMS Global Learning Consortium. http://www.imsglobal.org/learningdesign/

IMS, 2003b. *IMS Simple Sequencing Specification*. IMS Global Learning Consortium. http://www.imsglobal.org/simplesequencing/

IMS, 2004. *IMS Content Packaging Specification*. IMS Global Learning Consortium. http://www.imsglobal.org/content/packaging/

IMS, 2006. *IMS Question & Test Interoperability Specification*. IMS Global Learning Consortium. http://www.imsglobal.org/content/question/

Koper, E.J.R., 2001. *Modelling units of study from a pedagogical perspective: the pedagogical metamodel behind EML*. Open University of the Netherlands. http://eml.ou.nl

W3C, 2006. *Extensible Stylesheet Language (XSL)*. World Wide Web Consortium. http://www.w3.org/TR/xsl/.