Project: SCORM Compliant Content Packaging for Wiki-based Content Development

Student: Michael Aram Mentor: Gustaf Neumann

Abstract

In this paper we present an early prototype of a wiki-based SCORM authoring platform, which was developed in the context of the *Google Summer of Code 2009*. We based our implementation upon OpenACS' existing wiki application XoWiki and tried to realize the concepts of SCORM (e.g. organizations) using existing implementations (e.g. the `categories` package).

1 Introduction

SCORM The Sharable Content Object Reference Model (SCORM) [2] is a well established standard in the e-learning world. As a reference model it acts as an additional layer above other established e-learning standards, coming from a variety of organizations (IMS, ADL, ...). It focuses on the interoperability of learning management systems. To achieve that, it standardizes the way in which learning material should be packaged, so it can easily be transported (content aggregation model). In short, these *content packages* are folders that contain learning resources which can be displayed by a web browser and a file that contains meta information about these resources.

Moreover, SCORM specifies how the communication between a content package and a learning management system should take place (run time environment). Finally, current versions of SCORM provide the means for defining rules whether or not and in which order subparts of the content package can be accessed by the learner (sequencing). SCORM's content aggregation model relies on, and extends, the IMS Content Packaging specification [3].

There are a number of open source editors for SCORM content available, like e.g. the RELOAD Editor[1] or the eLearning XHTML editor (eXe)[2]. However, many of these editors are designed to be standalone client applications and support the traditional idea of one person authoring learning material for an audience.

From the perspective of the vendor of a learning management system, implementing support to handle SCORM packages seems to be a sensible investment, as it opens the system for externally developed learning content.

.LRN and OpenACS .LRN is a powerful open source learning management system based on the Open Architecture Community System (OpenACS), both completely written in Tcl. The xo*-familiy of packages (`xotcl-core`, `xowiki` and `xowf`) established itself as a flexible, generic, object oriented toolkit, enabling rapid development of arbitrary applications for both OpenACS and .LRN. XoWiki [4] is one of the most flexible wiki-frameworks, supporting advanced concepts like for example structured wiki features, multiple access policies, flexible application integration and workflows. In the e-learning context, XoWiki can be used as a single tool

[1] http://www.reload.ac.uk/editor.html
[2] http://exelearning.org

for content-development for (adaptive) content presention, blog style content distribution and assessement. Actually, it is used in huge .LRN installations like the Learn@WU platform of the Vienna University of Economics and Business[3].

2 Implementation Concepts

XoWiki as a wiki application implicitly supports collaborative authoring and easy-to-use wiki-style content creation. However, to turn XoWiki into a SCORM player and authoring environment, it obviously had to be extended. The main fields we had to consider were

- **importing** existing content packages,
- **playing** the content package (i.e. providing a run time environment),
- **authoring**, i.e. changing the contents of the package and its metadata, and
- **exporting** the content as distributable content package.

A SCORM content package may contain a hierarchy of folders. These folders are "ordinary" file-system folders and may be used by the content developer to organize the contents of the package. For example, he might want to put all graphics into an `images` folder. These folders have no didactic meaning and are *not* directly presented to the learner. However, to be able to import a content package as-it-is into an XoWiki instance, we have to represent this hierarchic structure.

Moreover, to play a valid SCORM package, a special JavaScript object has to be provided, the so-called "API Adapter". It is used to exchange information with the learning management system, e.g. the students' name.

In our approach, each SCORM package is represented as a single XoWiki instance. Thus, authoring the actual contents of a SCORM package, i.e. the contained HTML pages and files, basically means editing a wiki instance.

[3]https://learn.wu.ac.at

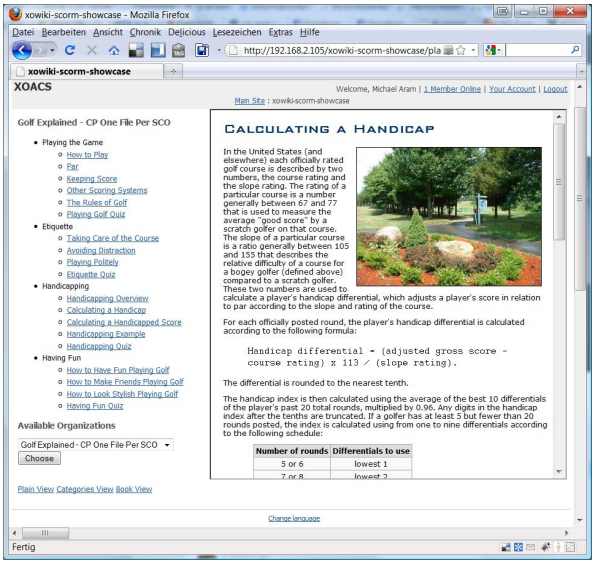

Figure 1: XoWiki-SCORM Player

Figure 1 shows a screenshot of an example SCORM content package[4] played by an `xowiki-scorm` instance.

2.1 Folders

The main challenge regarding the import of content packages into a wiki instance was the lack of a "folder" implementation for XoWiki. So far, each XoWiki instance used to have a single content repository folder associated. This means that all content inside an XoWiki instance was stored inside this single folder in a flat structure.

We decided to implement these folders using the OpenACS content repository folders. As a consequence, our prototype allows for each XoWiki instance a hierarchy of content-repository folders, instead of only a single root folder.

2.2 Organizations

The IMS CP specification, on which the SCORM content aggregation model is based, allows the structuring of content in so called organizations, which are represented as XML in

[4]One of the "Golf Examples" provided by Rustici Software (http://www.scorm.com/scorm-explained/technical-scorm/golf-examples/).

the manifest file. In contrast to the folders introduced above, these organizations are finally presented to the learner to navigate through the content of the SCORM package. A content package may only have one folder structure, but an arbitrary amount of organizations.

There are already two established methods for organizing the pages inside an XoWiki instance, i.e. categories and page order.

One can use the `categories` OpenACS package to create category trees and relate them to the XoWiki instance. Each wiki page can be mapped to an arbitrary amount of categories. In SCORM terminology, a wiki page acts as a *resource* and the categories which it is mapped to are *items*.

The second approach is based on XoWiki's "book mode" and uses "page order values", which may be assigned to each XoWiki page. This approach is the basis for exporting existing "XoWiki Books" as SCORM content packages, but it limited to only one organization per package.

When importing a content package, our prototype maps existing organizations to both category trees and (the default organization) as page order values.

3 Summary & Outlook

Among the outcomes of this project are the OpenACS packages `ims-cp` and `scorm`, which are prototypical implementations of the respective standards, and the `xowiki-scorm` package, a sub-package of `xowiki` that is based upon them.

Our prototype is a proof of concept and is still subject to changes and development. At the time of writing, the `ims-cp` package implements the most important concepts defined by the IMS CP 1.1.4 specification, i.e. classes to generate and manipulate *manifests* (including *organizations*, *items*, *files*, and *resources*) and render them into an XML representation. Concepts that were not immediately needed for a prototypical implementation (most notably metadata, sub-manifests and external resources) have still to be dealt with, though.

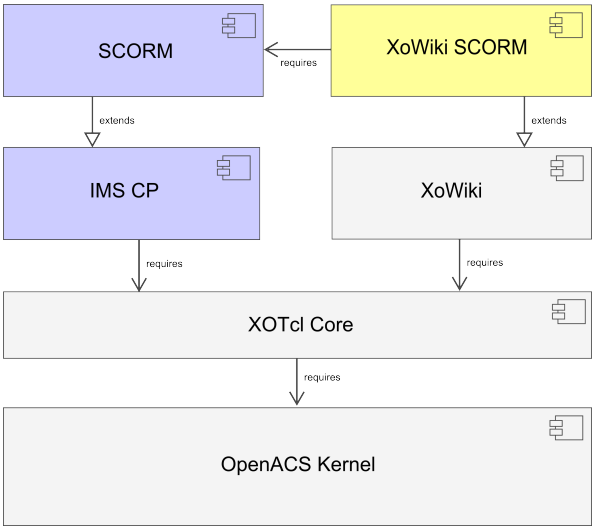

Figure 2: XoWiki-SCORM Architecture Overview

The `scorm` package focuses on version 1.2 of the SCORM. In the current state, it can be seen as a lightweight extension of the `ims-cp` package that contains static resources (e.g. XML Schema files, which are required by all valid SCORM packages) and includes them in the manifest. In addition to that, it provides the client-side JavaScript-based API Adapter together with the server-side SCORM API. However, the server-side API currently only satisfies the most basic needs of content packages, e.g. answering their calls to essential SCORM API functions like `LMSInitialize()` or `LMSFinish()`. The persistence and correct handling of session data generated by a student during a learning session still has to be implemented.

The `xowiki-scorm` package provides the classes for importing the resources of a SCORM content package into an XoWiki instance. The mapping of SCORM organizations to data structures used by XoWiki (i.e. category trees and page order values) is also implemented in this package. Additionally, it contains template files that allow the playing of a SCORM content package (as shown in Figure 1) by providing the client-side API and a dedicated frame, as specified by the SCORM

standard. Finally, an export mechanism has been integrated that dumps the contents of an XoWiki instance into a SCORM compliant content package and considers the associated category trees and page order values when generating the manifest.

In the upcoming months, we will focus on implementing the missing features mentioned above. Development progress will be published on the project's homepage [1].

References

[1] XoWiki-SCORM. URL `http://wiki.tcl.tk/23190`. Homepage of the GSoC 2009 Project: SCORM Compliant Content Packaging for Wiki-based Content Development.

[2] Philip Dodds (ADL) et al. SCORM 1.2 Documentation Suite. Technical report, Advanced Distributed Learning Initiative, 2001. URL `http://www.adlnet.gov`.

[3] Colin Smythe (IMS) and Alex Jackl (IMS). IMS Content Packaging Specification v1.1.4. Technical report, IMS Global Learning Consortium, 2004. URL `http://www.imsglobal.org/content/packaging/`.

[4] Gustaf Neumann. XoWiki Documentation. URL `http://media.wu-wien.ac.at/download/xowiki-doc/`.