ECP 2007 EDU 417008

ASPECT

Conformance Testing Tools

Version 2

Deliverable number	*D3.2.2*
Dissemination level	*Public*
Delivery date	*June* 2010
Status	*Final*
Author(s)	*Ingo Dahn (KOB), Patricia Heckmann (KOB), Sabine Orth (KOB),Bram Vandeputte (KUL)*
Internal Reviewer(s)	*Jim Ayre (EUN)*

e*Content*plus

This project is funded under the *e*Content*plus* programme[1],
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

[1] OJ L 79, 24.3.2005, p. 1.

Summary

The central objective of conformance testing is facilitating interoperability between various systems. Automated conformance testing is a major tool to catch syntactic errors in content before it gets delivered to the end user. It also has a place during the development of a specification. Examples, which are provided to illustrate specifications, are an important guidance for content providers and system developers and motivate them to take up the specification. Therefore, it is crucial that these examples have been tested for correctness. Conformance tests can be difficult to complete, in particular when specifications contain semantic real-world references that cannot be tested automatically. We need to allow for the fact, therefore, that automated conformance testing cannot always guarantee interoperability. Nevertheless, experience shows that not doing automated conformance tests considerably increases the likelihood of non-interoperability.

Within the ASPECT project, automated conformance testing is provided by Work Package 3 for the following specifications.

- *SCORM 2004*. This uses the official ADL Test Suite in its latest versions of 1.1.1 of October 27th 2009.
- IMS Common Cartridge. This uses the latest official IMS Common Cartridge Test System Version 1.0.7. For use in the ASPECT project this test system has been extended (see Section 2.2.2).
- *IMS LODE* specification with its parts *Registry* and *ILOX* and associated metadata profiles. A number of different test systems have been produced for these profiles in particular to meet requirements from Work Package 2.

In D3.2.1 (Dahn et all., 2009), which was released in April 2009, we provided an introduction to conformance testing and presented the available tools which were most relevant for the ASPECT project. We refer to this Deliverable for a more detailed presentation of the relevant context of the current document.

In the following sections we discuss the development of conformance testing that has occurred in the field of Technology Enhanced Learning since the delivery of D3.2.1. We cover developments done within the ASPECT project as well as developments done elsewhere.

The document is organized according to the specifications against which tests will be made. We begin with the description of recent developments for testing packaged content as specified by the SCORM and Common Cartridge specifications. This is followed by the discussion of recent developments for testing metadata for individual learning objects and for collections of such objects.

The final part is concerned with generic conformance testing issues. This part develops a design for integrating vocabulary testing into a generic conformance testing system. The need for better support for testing usage of external vocabularies was articulated in particular in Work Package 2 of the project. Working with conformance testing in Work Package 5 showed that the correct and complete handling of references is a particular challenge. To facilitate this, a Content Package Referencer has been developed which checks for some popular flaws and corrects them where possible. It is also described in the Section on Generic Conformance Testing.

Index of Contents

Index of Figures

1 Common Cartridge Testing

1.1 *Interoperability*

In February 2010, KUL provided the ASPECT project with the latest version of the Moodle Learning Management System which supports the import of Common Cartridges. This version was used with the sample cartridges that had been produced for the ASPECT Showcase by Klascement and the University of Ljubljana (sees DeliverableD3.5 (Heckman et al. 2009). These cartridges make use of tests and discussion forums as specific features of the Common Cartridge specification.

These resources have now been all successfully imported into the Icodeon Common Cartridge Framework and into Moodle. The Icodeon Framework allowed access to particular resources from within other systems. This was tested with access from a specific web site and from within both Blackboard and Moodle. Access from the ePortfolio system Mahara was not successful because of restrictions to web access imposed by Mahara .

Import of cartridges into an LMS provides teachers with direct access to the cartridge's elements. For example, a teacher may re-combine quizzes from a cartridge with other quizzes to build up her own tests. This access to common cartridge elements was verified with Moodle. Moodlehonours a cartridges access control by refusing import of cartridges with access restricted content. More refined access control, as foreseen by the specification, is not supported.

The current Moodle version is still not correctly handling some issues (for example multiple choice questions with multiple correct answers). Using the test system helped ensure that the problems found were not caused by flaws in the imported cartridges. The issue has been reported to the Moodle Developer Tracker where it is tracked under #MDL-21856.

Another bug discovered in the project, which prevented Moodle from importing a local Common Cartridge if it did not have full Internet access, has been fixed by the Moodle developers.

In February 2010, the session of the IMS Common Cartridge Application Profile Management Group paved the way for the further work on Version 1.1 of the specification. This version aims to support calls to interactive software from within a cartridge using launch sequences as specified in the IMS Basic Tools Interoperability specification. It was decided to remove the requirement to support access control from the conditions to be met for claiming the Common Cartridge Conformance Label.

In April 2010 the ASPECT project provided 15 cartridges from Siveco, the University of Ljubljana, The Open University and ITC Lithuania to the IMS Interoperability Challenge which is to be held in May 2010 in Long Beach, California, USA.

1.2 *Common Cartridge Test System*

1.2.1 Corrected Errors

As reported in D3.2.1, the IMS Common Cartridge Test System is an instance of a generic test system. From this generic test system, specific test systems can be generated for each application profile which has been encoded with the SchemaProf Application Profiling Tool which is also described in D3.2.1.

The Common Cartridge Test System has been used by Work Package 3 as well as by individual content providers to test cartridges for conformance with the specification. Errors found in cartridges concerned mostly missing or misspelled references to files, typos and use of wrong vocabulary terms.

A problem with the IMS Common Cartridge Test System, discovered through its use in the ASPECT project, was the risk that files of users could be overwritten without notice if a non-empty directory was used for collecting the test system reports. As a consequence of observing this problem, the test system has been modified so that now, by default, it creates its own safe reporting directory.

A problem with handling file names with special characters, like '[', in file names in the manifest of a cartridge was discovered by other users and corrected. A similar problem, also discovered in the project, still continues to exist: Resources which reside on the web have to be referenced in cartridges with a web link file in a specific format. The Common Cartridge specification does not specify how special characters, like '&', should be encoded in these web link files. This problem has been reported to the IMS Common Cartridge Application Profile Management Group.

A further problem caused the test system to crash if the zip compression of a cartridge was incorrect or if some XML files in the cartridge where not well-formed. This error is now caught and reported to the user.

It is worth noting that the problems mentioned above have been corrected in the *generic* test system. Thus, by contributing to the discovery and correction of those errors, the ASPECT project has also contributed to the quality of future test systems for other application profiles of other specifications which will be build using this generic system.

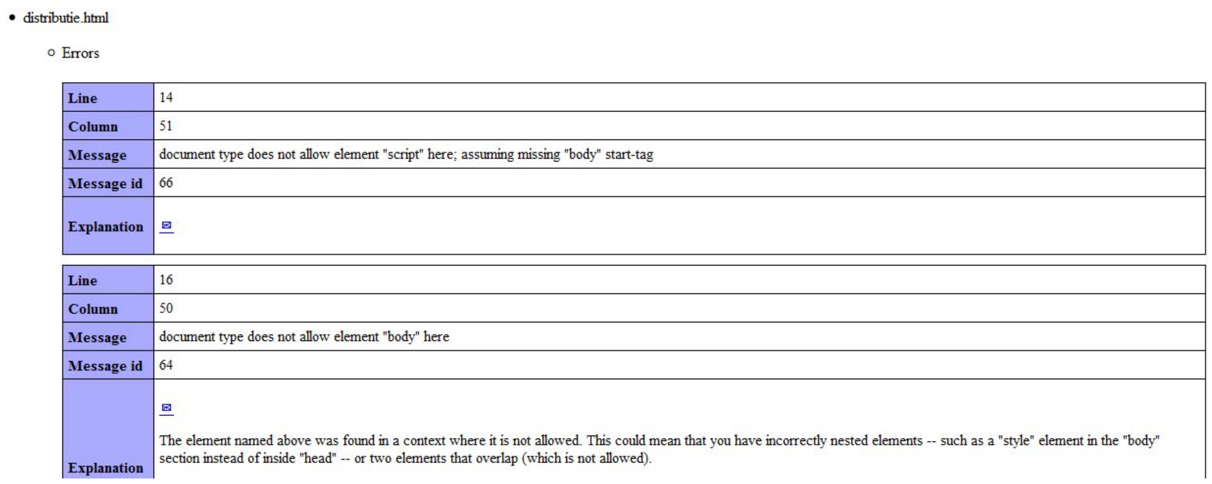
The Common Cartridge Test System performs only tests for issues that have been specified in the Common Cartridge Specification. However, common cartridges may also contain other parts, like html pages which contain the proper content. Validation of these html pages with respect to the html specification is beyond the scope of the standard Common Cartridge Test System.

1.2.2 Adding HTML Testing

Common Cartridges mostly include a number of HTML pages which contain the proper content. The structure of these pages should be governed by the W3C HTML specification (W3C, 1999). Making sure that the HTML pages in Common Cartridges conform to this W3C specification is desirable in order to exclude other sources of non-interoperability beside those considered in the Common Cartridge specification. This has been realized on an experimental level in the reporting period.

This activity also made use of the extensibility of the Generic Test System. In fact it integrated a W3C Mark-up Validation Service (W3C, 2010) into the Common Cartridge Test System. To achieve this, the web service had to be wrapped as a new test system that could be bundled with other test systems. The wrapper has to produce a self-description to be read by the bundled test system and containing its interface description. Moreover, the wrapper needs to convert the error messages produced by the W3C service into a format that can be understood by the generic test system. Finally, style sheets need to be provided which the test system controller can use to integrate the error reports as HTML pages with the error reports of the other test systems and into the summary error reports.

Such a wrapper has been successfully implemented and tested with Common Cartridges from WP5 partners (see screenshot below).



- distributie.html
  - Errors

Line	14
Column	51
Message	document type does not allow element "script" here; assuming missing "body" start-tag
Message id	66
Explanation	✉

Line	16
Column	50
Message	document type does not allow element "body" here
Message id	64
Explanation	✉ The element named above was found in a context where it is not allowed. This could mean that you have incorrectly nested elements -- such as a "style" element in the "body" section instead of inside "head" -- or two elements that overlap (which is not allowed).

Figure 1 An HTML Mark-up Error Report

1.3 A New Online Testing System for Common Cartridges

For repositories and Learning Management systems, it is important to do tests on import capabilities. Towards this end, Thor and Per Anderson at Learning Components, Inc. are working on an online version of a Common Cartridge testing tool which is available for beta testing at http://validator.imsglobal.org/. There are 4 different ways to test a cartridge: URI, direct input, just a manifest, or upload a cartridge. The first three input methods just test the validity of the manifest. It is only when a cartridge is uploaded that the full set of tests is run.

Features of this online tool emphasized by the authors are:

- All schemas used for validation testing are found at permanent URLs on the IMS web site.
  As many current cartridges use older URLs that no longer resolve to a valid location, the test tool provides a warning that this is the case, and then re-maps to the proper URL so that testing may continue.
- This tool uses the ISO standard version of Schematron (released on 2010-01-25) instead of the older Schematron 1.5.
- The tool may be
  - launched from the command line as a stand-alone Java program
  - referenced from within a server-side script such as a JSP page
  - utilized by a SOAP web service.

The SOAP Web Service is located athttp://validator.imsglobal.org/CCValidationService?tester. The WSDL that describes the details of the web service in a way IMS developers can use can be found here:

http://validator.imsglobal.org/CCValidationService?WSDL. The main thing not fully implemented yet is the Schematron rule processing for all of the CC profiles.

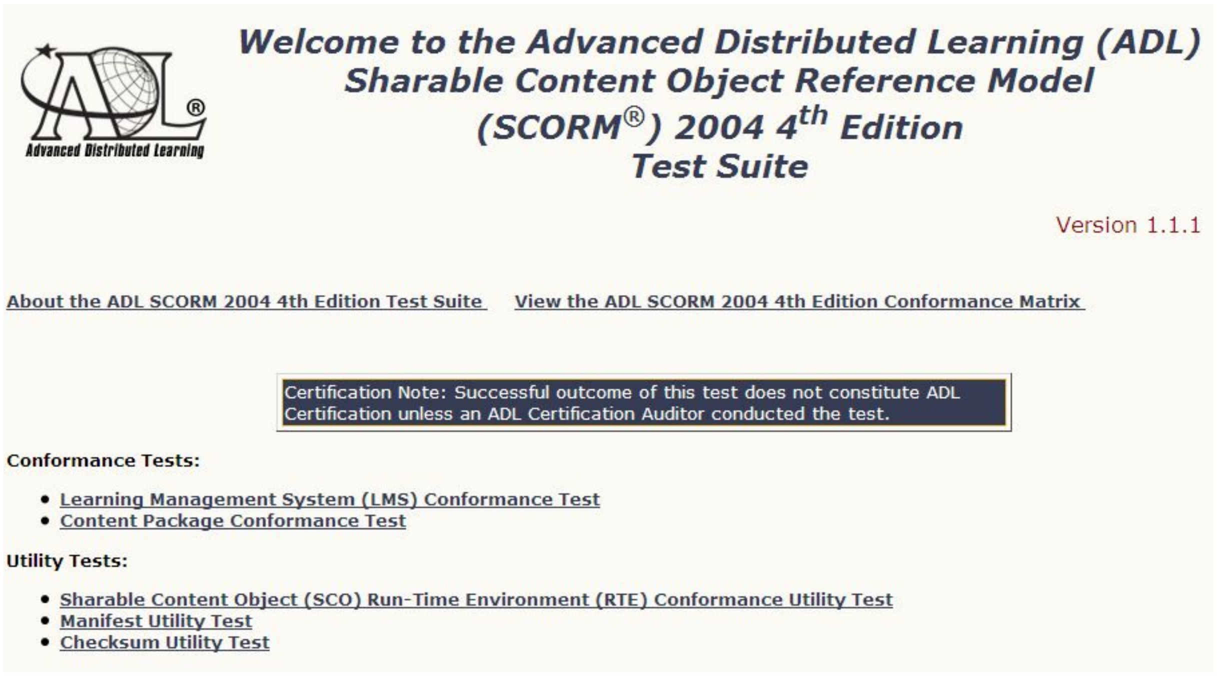
The version of this tool published Mid March has been tested in Work Package 3 and problems have been reported on the Common Cartridge Alliance Forum.

2 SCORM Testing

The SCORM 2004 4th Edition Test Suite Version 1.1.1 contains the compliance testing software, procedures and supporting documents for organizations to test their LMSs, SCOs and SCORM Content Packages. This version supersedes all previous versions.

After installation the user will find the Test Suite in the folder "ADL". **It is important to notice, that the Test Suite is released and tested only on Microsoft Windows Vista SP1 and XP SP3 on the Intel Pentium Platform using Microsoft Internet Explorer 7 and Internet Explorer 8.**Other browsers like Mozilla Firefox donot work, as tests done within the project have shown. On the other hand, these tests showed also that the ADL SCORM Test Suite works well under Windows 7 with the aforementioned versions of Internet Explorer.

Starting the Test Suite opens the browser with two compliance test and three utility tests.


Welcome to the Advanced Distributed Learning (ADL) Sharable Content Object Reference Model (SCORM®) 2004 4th Edition Test Suite

Version 1.1.1

About the ADL SCORM 2004 4th Edition Test Suite   View the ADL SCORM 2004 4th Edition Conformance Matrix

Certification Note: Successful outcome of this test does not constitute ADL Certification unless an ADL Certification Auditor conducted the test.

Conformance Tests:

- Learning Management System (LMS) Conformance Test
- Content Package Conformance Test

Utility Tests:

- Sharable Content Object (SCO) Run-Time Environment (RTE) Conformance Utility Test
- Manifest Utility Test
- Checksum Utility Test

Figure 2 ADL SCORM Test Suite

2.1 Conformance Tests:

2.1.1 Learning Management System (LMS) Compliance Test

This test verifies the compliance of an LMS to the Run Time Environment specification described in the SCORM 2004 4th Edition.

Revision: The instructions for LMS Test Content Packages CO-06, CO-09, and CO-13b were updated to display the correct precondition rule of ,*"If obj1 activity progress known and not completed, then skip"*.

Use: There are two options for a test: Choosing "New LMS Conformance Test" will initiate the first step of the LMS Conformance Testing procedure. Choosing "Load Saved LMS Conformance Test" opens a list with saved tests. The test starts at the last step it completed before.

2.2 Content Package Conformance Test

The purpose of the test is to verify compliance of SCORM 2004 4th Edition Content Packages. The content packages can be packed as zip-files or not packed with an imsmanifest.xml file located at the root of the package.

Revision: The Content Package Conformance Test and Manifest Utility Test were updated to enforce that parents of a given element are correct in both name and XML namespace.

Use: First step is to give a name for the package to identify it. The type - Content Package (PIF) or non content Package (non PIF) - must be selected. Step three is to choose the type of the SCORM Application Profile. Now the file can be uploaded. A click at the button "Begin Test" starts the test.

2.3 Utility Tests

2.3.1 Sharable Content Object (SCO) Run-Time Environment (RTE) Compliance Test This test verifies compliance of individual Sharable Content Objects to the Run-time Environment specification.

Use: A step-by-step instruction helps to go through the test. Each test of a SCO may or may not exercise all possible paths through the SCO. Printing the log file is at any time possible.

2.3.2 Manifest Utility Test

The purpose of this test is to verify compliance of SCORM 2004 4th Edition IMS Manifest only. The test subjects includes packages in the form of a zip-file with the imsmanifest.xml located at the root of the package.

Use: In four steps it is possible to test the content aggregation manifest or resource manifest. Just choose the options and upload the file in the last step.

2.3.3 Checksum Utility Test

This new test verifies that test logs were produced by the expected content. A successful comparison indicates that the summary log was created as the result of a successful Content Package Conformance Test run with the given Content Package.

Use: Upload the selected file and check against the Checksum Value or the (uploaded) Log File.

The ADL Test Suite is a comprehensive tool for experts/developers to test SCORM packages. Developers mostly use it in combination with RELOAD Editor. The Test Suite can be downloaded from http://www.adlnet.gov/Technologies/scorm/SCORMSDocuments/2004%204th%20Edition/Test%20Suite.aspx

3 LODE Testing

3.1 LODE Specification Released a IMS Internal Draft

The new IMS Specification on “Learning Object Discovery and Exchange” (LODE) was released on March 2nd, 2010 in draft form to IMS Contributing Members and to members of the IMS Developers Network. The objective of this specification is to facilitate the discovery and retrieval of learning content stored in repositories. The LODE specification consists of two parts:

- The Registry part with the XML schema imsloreg_v1p0.xsd for specifying the description of registries and the ways in which they can be accessed. The specification defines the following root data elements
  - *metadataCollection*
  - *contentCollection*
  - *target*
  - *protocol*
  - *accessPolicy.*
- The ILOX part with the XML schema with the XML schema imsloilox_v1p0.xsd for specifying the description, at different levels of detail, of the learning objects to be exchanged. ILOX documents can have one of the root elements
  - *work*
  - *expression*
  - *manifestation*
  - *item.*

These two core specifications are augmented with more specifications for detailed descriptions of the communication protocols to be used, for example OAI-PMH or SQI. Application profiles that come with the LODE specification add vocabularies to be used for values of particular data elements.

3.2 LODE Challenges for Testing

A special feature of the LODE Registry specification is the requirement that some fields specify schemas that are to be used at other points. For example, the specification of a contentCollection name and location of a protocol XML schema while a target element (i.e. the description of an end point of a service) must contain an element as defined by this schema in order to be used with the registry described. This requirement concerns usage of data, not the structure of data. This can be tested only if both the registry description and the target description are known.

A feature of the LODE ILOX specification, which requires special consideration for testing, is that some sub-elements are mandatory only if the element is not used as a root element or dependent on some values in the instance document. Also, allowed values at some places may depend on values found at particular other places in instance documents. Such requirements cannot be expressed in XML schemas and require therefore special efforts for testing. ILOX requires that Name is from an unspecified vocabulary containing at least a certain set of values. As the possible values are unspecified, this cannot be tested.

3.3 ASPECT Contribution to LODE Testing

The ASPECT project has been actively involved, through EUN, in the development of the LODE specification. The specification has been used in the project and the project has provided profiles and test systems.

3.3.1 Standalone Test Systems

In D3.2.1 we reported on the delivery of conformance testing systems for the Learning Resource Exchange Metadata Application Profile (LRE-MD) Version 4.0 (Beta 2). LRE-MD is a domain profile of the ILOX part of the LODE specification. It consists of two application profiles.

1. A profile of the loose version of the Learning Objects Metadata Specification IEEE LOM (IEEE, 2002) and
2. A profile of an early version of the IMS Learning Objects Discovery and Exchange Specification (IMS LODE) ILOX Specification 1.0 Draft 2.0 (IMS, 2010)

Since then.this specification has been augmented by a more restricted “core” version of the profile for which a test system has also been produced.

Following the release of the internal draft of the LODE specification, two further test systems have been provided: One for the Registry part and one for ILOX part. Both are currently beta tested. These will be further refined as the vocabularies, which are used in the recommended profiles, become available. Currently they support a profile using the OAI-PMH and the SQI protocol. Usage experience in Work Package 2 shall inform the further development of the profile to be tested. The test systems shall be adapted accordingly by Work Package 3 to support the testing requirements that emerge during the course of this work.

3.3.1.1 Usage

As for the Common Cartridge Test System, these test systems (that build on the generic test system) are installed on Windows (32 and 64 Bit), Mac (64 Bit only) and Linux (32 and 64 Bit) operating systems by simply unpacking the zip files provided, presuming that a current Java Runtime Environment V1.6 or later is available. They are capable of handling several test items in one run. Unless there are restrictions due to a specific configuration of the local internet access, users simply select the items under test and click “Validate”. Error reports are complete with line numbers of error locations. They are displayed in the local web browser after a click on the respective button.

Providing a test system for the third existing “recommended” version has been deferred until it will be precisely defined in the profile documentation what conformance for the recommended version means (it could mean that <u>only</u> recommended fields are used or that <u>all</u> recommended fields are used).

3.3.1.2 License Information

While the system has been configured within the ASPECT Project, it is licensed fromIMS, thatholds the rights in the Generic Test System, for all kinds of usage including commercial usage. The source code of the system is not open. The underlying profiles have been submitted to the IMS Application Profile Repository.

3.3.1.3 Limitations

The tool can be called only through its GUI. It does not perform vCard conformance tests for embedded personal data. Specific vocabularies are encoded into the schemas. The latter shall be augmented by the possibility to embed complex profiles in VDEX format using the design described in the last Section.

3.3.2 Online Metadata Validation Service

The ASPECT Online Metadata Validation Service can test a variety of metadata formats. We discuss it here in connection with the LODE specification since this is among the most recent applications of the service within the project.

3.3.2.1 Objectives and Use Cases

When metadata are harvested, a quick but secure check of syntactic metadata quality is needed with respect to an Application Profile. This cannot be provided with manual testing of metadata quality as potentially a need for the validation of thousands of instances is expected. Metadata providers should be provided with an error report to help them to improve their metadata instances.

A lot of the metadata being exposed is mapped from another scheme, as a result of which the provider needs to map the metadata into the scheme that was agreed upon. As a lot of errors can occur during this mapping, validating a single mapped instance at this stage could prove very valuable in speeding up the mapping process.

Another use case occurs at the time when metadata are collected. The harvesting process is the first step where a metadata repository gets hold of the metadata. Adding a validation step at this point has the advantage of only consuming metadata that is parseable and that conform to the Application Profile.

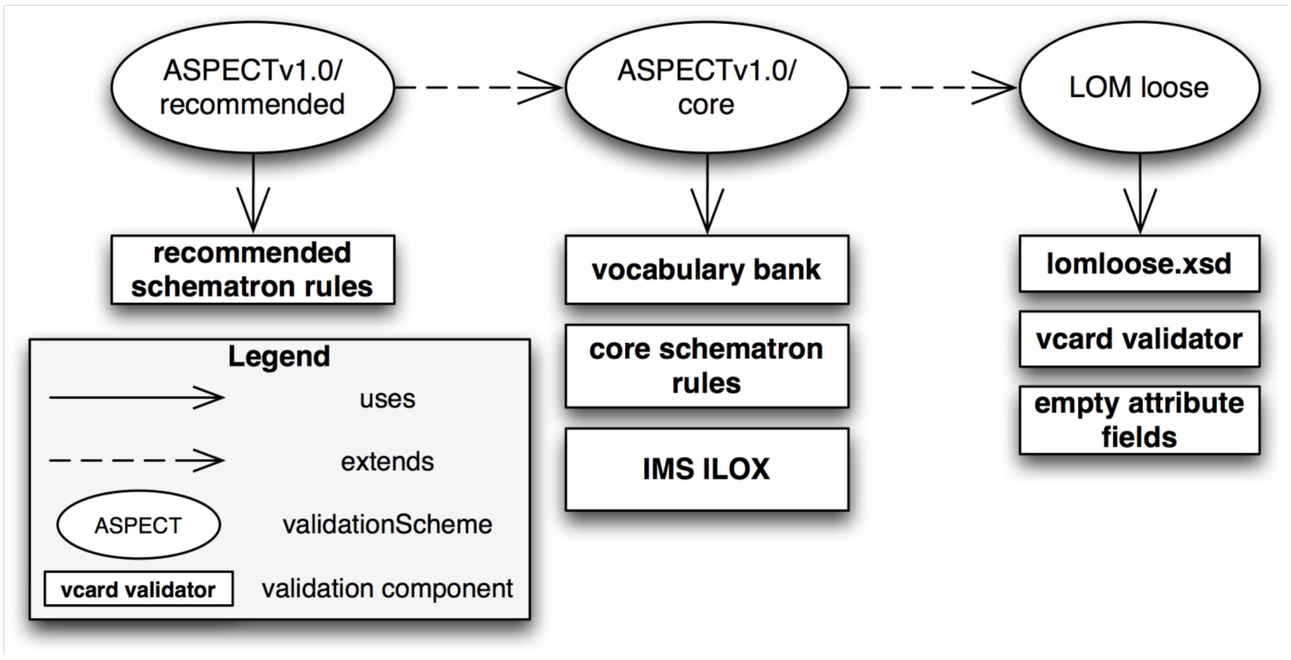
At some point the collected metadata can be enriched with extra metadata, and so validation of the new metadata can also be useful at this stage.

3.3.2.2 Functions Realized

The validation service consists of independent reusable components and supports extra plug-ins and validation of vocabularies. The Figure 3 below illustrates the hierarchical configuration of several validation schemes, where these components are used. Recently the metadata

validation service has been augmented by a possibility to test OAI-PMH targets.

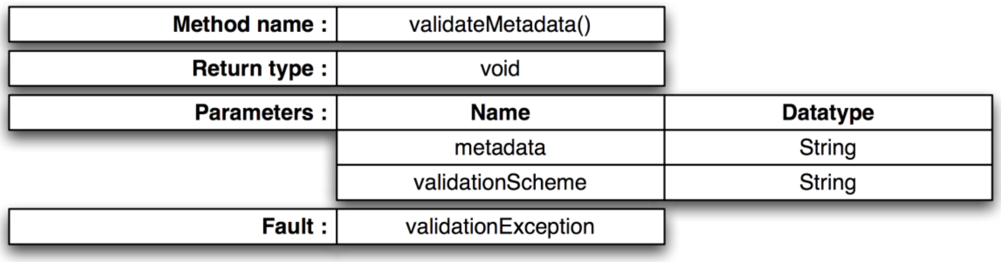
The implementation has proved to be powerful and scalable. A 3-level error reporting has been implemented, supporting both a very detailed report on every error, as well as a summary of all the errors present in a metadata collection.



3.3.2.3 Usage

The validation service can be used online without the need of any installation whatsoever. It is sufficient to select the Validation Scheme to use, copy paste the metadata to be validated and click validate at http://ariadne.cs.kuleuven.be/validationService .

The service is also embedded in the ARIADNE Harvester, and can be used in a very similar way at http://ariadne.cs.kuleuven.be/lomi/index.php?title=Harvesting_Metadata . For online tools and widgets, a REST based web service is available for easy integration. A standalone Java library is also available.



Method name :	validateMetadata()	
Return type :	void	
Parameters :	**Name**	**Datatype**
	metadata	String
	validationScheme	String
Fault :	validationException	

Figure 4 Aspect Online Metadata Validation service

3.3.2.4 License Information

This software is available under a LGPL license. The validation service has been used in several EC-funded e-learning projects, such as MELT, MACE, ICOPER, STELLAR and

ASPECT. In addition, it is currently used in the Share.TEC system being developed at the Open University of the Netherlands.

3.3.2.5 Limitations and Known Issues

At the moment the used technologies only support validation of XML metadata, although, in theory, extra technologies and components can be plugged in to support other formats. It is not possible to validate content or the structure of resources. There is no stand alone application available yet and, finally,the error feedback mechanism could be improved.

4 Generic Testing Issues

Deliverable D3.2.1 described how test systems for various application profiles, including Common Cartridge and the LRE Metadata Specification Application Profile, could be generated from a Generic Conformance Test System. These concepts are also explained in a recent publication (Dahn, Zimmermann 2010). This generic test system supports validation of XML files with respect to arbitrary XML Schemas and validation with respect to Schematron rules. Moreover, it can test the correctness of internal references and perform specific tests for simple types that cannot be expressed by the languages of XML Schema or Schematron.

4.1 Adding Vocabulary Testing – the Concept

4.1.1 What is Vocabulary Testing?

Vocabularies are lists of terms that can be used to describe particular properties. Often vocabularies contain additional information, for example descriptions of the meaning of the respective term, relations between different terms or translations of terms into multiple languages. Terms in vocabularies are referenced by preferred entries. These preferred entries can be anything from abstract identifiers to terms in a preferred language.

In many cases different domains can use the same specification but require the use of different domain specific terms in their respective controlled vocabularies. The LRE Service Centre provides users with access to the LRE Vocabulary Bank for Education at [http://aspect.vocman.com/vbe/home] collecting vocabularies from many sources.

Vocabulary testing has to provide two kinds of tests:

1. A test that the items under test use only specified vocabularies
2. A test that entries used in test data do in fact exist in the specified vocabularies.

The usage of vocabularies can be encoded in specifications and application profiles in two different ways. They can be *encoded by value*, by allowing only specific values for particular XML tags or attributes, or they can be *encoded by reference* to a separate vocabulary document. The second method makes it possible to use vocabularies in standard formats, like IMS VDEX, while the second requires pre-processing vocabularies to enable the integrationof the allowed entries and vocabulary terms into other documents like XML schemas. Usage of vocabularies encoded by value can already be tested by the generic test system. An ongoing objective of ASPECT Work Package 3 is also to provide support for vocabulary encoding by reference. The design that has been developed for this objective is described below.

4.1.2 The IMS VDEX Specification and its Usage in the ASPECT Project

The most common case of specifyingvocabulary usage is the requirement that, at a certain point in a document, a term from a particular vocabulary must be used. However there are more complex situations to be mastered like the following taken from the LRE Metadata Application Profile. Consider the following example (EX) of a fragment of a document to be tested.

```
<manifestation>
<name>
<vocabularyID>LRE.manifestationNameValues</vocabularyID>
<value>package in</value>
</name>
<parameter>
<vocabularyID>LRE.packageInValues</vocabularyID>
<value>scorm_v2004</value>
</parameter>
```

The LRE Metadata Application Profile specifies that the *value* (in this case *scorm_v2004*) must be from a specific vocabulary (in this case identified by *LRE.packageInValues*) **if the *value* child of the preceding *name* tag holds the text *packagein*.**

To test whether this requirement is satisfied, it is not enough to know that *parameter* refers to a vocabulary and where this vocabulary can be found. It must be determined beforehand whether the test is applicable at all and this requires looking to a different place in the document – the *../name/value* text content! We shall come back to this example of a conditional vocabulary test later on.

There are even more vague specifications around, for example “The parameter element can refer to any vocabulary in a particular vocabulary bank”. This is problematic, as testing for conformance with this requirement can lead to different results in cases where the content of the vocabulary bank changes. We assume that conformance test results need to be reproducible in order to be of value. This requirement is suggested by the approach that the semantics of a profile is given by the set of its conformant document, as described in the IMS Application Profiling Guidelines (IMS, 2009). Making the semantics of an application profile time dependent is likely to introduce an additional theoretical complexity, beyond what is needed to support current practice.

This does not exclude changes in a vocabulary bank. It only means that the profile used for testing must also change if the vocabulary bank changes and that these changes must be reflected in the test system for the profile. Then test reports need to specify precisely which version of the vocabulary bank and which version of the profile have been used for testing. In this case, by taking a snapshot of the vocabulary bank at a certain point in time, it is possible to download all relevant vocabularies for use with a generic vocabulary test system.

Perhaps, in the future, application profiles will emerge that allow only use of “some” vocabularies from “some” vocabulary banks. However,currently profiles such as this donot exist; there is also not enough experience – and not enough practical relevance – to implement vocabulary testing in this way.

4.1.3 The Usage of a Vocabulary Test System in Generic Conformance Testing

The Generic Test System (GTS) is designed to be extensible by new tests. The development designed here makes use of this feature. On start-up, the Launcher component of the GTS analyzes the properties of the test systems it has at its disposal and the initial tests to be made. Next, it analyzes the items under test (IUT). The GTS then tries to execute for each IUT the tests specified by the profile. It does this by calling specialized test systems, like the Vocabulary Tester (VT), providing them with the information required to do the tests. As part of a GTS, Test Systems must be designed to be re-usable in the context of an unforeseen variety of application profiles.

In fact, the VT assumes that the GTS, or a Test System which has previously determined the need for vocabulary testing, has prepared the data needed. For testing usage of VDEX vocabularies, the VT needs to know

- what is the source (i.e., what is the vocabulary to be tested) and
- what are the value entries to be tested
- where can the respective vocabularies be found.

Beside these, the VT needs to be called with some more technical information – for example where it has to write its logging data to.

Note that one IUT may use many different profiles in a domain profile and each of these profiles may use many vocabularies. The VT needs to know which profile a specific test refers to. Moreover, the VT needs enough information to determine the **location of the vocabulary usage** in the IUT in order to create meaningful error reports which point the user to the location of errors. We realize that software development is required for two tools:

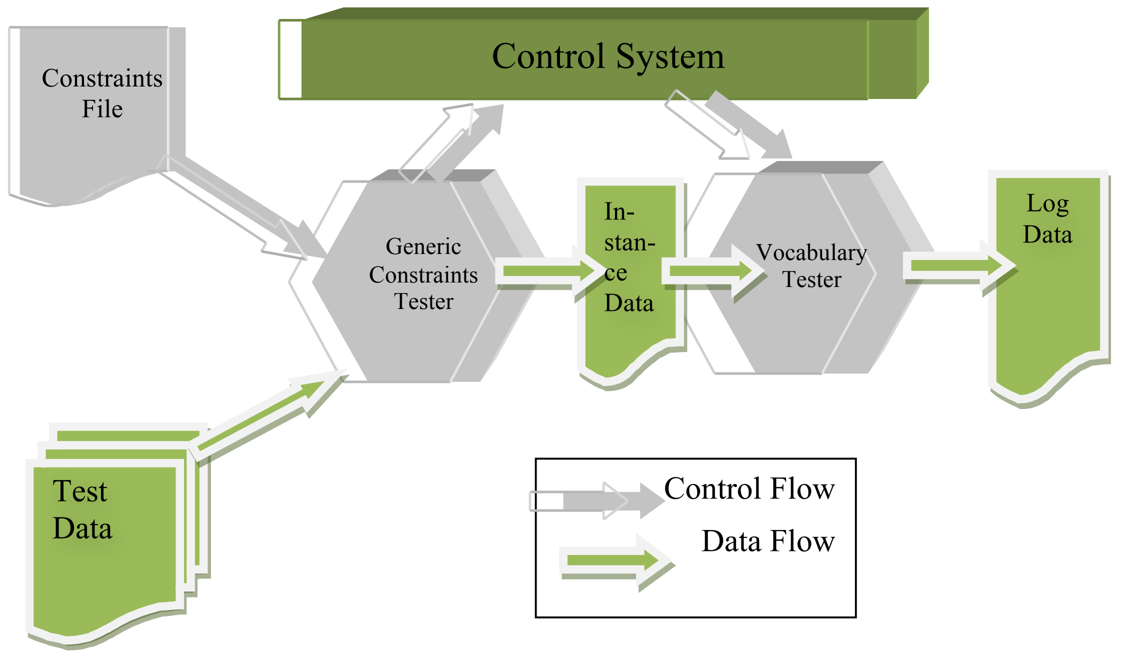
1. The GTS needs to be augmented to prepare the relevant data for the VT
2. The VT needs to be developed from scratch.

This will enable vocabulary testing for all future kinds of appropriately encoded application profiles. We note that, for an easy capturing of specifications for using vocabularies, the SchemaProf tool also needs to be augmented to save vocabulary requirements in the format needed by the GTS. This latter part, which is not required to build a working vocabulary tester, shall be touched upon in the Section “Integrating Vocabulary Testing into Application Profiling”.

From an abstract point of view, vocabulary testing is a special case of testing so-called additional constraints (see IMS Application Profiling Guidelines at (IMS, 2009)). Additional constraints need to be used when the specific test to be made can be only determined at run time. These constraints specify, for a location in a test document, which tests are to be called depending on a number of parameters. The values of these parameters are either given in advance by the profile or specification (so-called static parameters) or the profile specifies in the constraint XPath expressions (so-called dynamic parameters) which are evaluated at test time to retrieve the respective parameter values which are needed to determine which tests are actually needed.

File testing is another type of additional constraints. File constraints specify that locations in a test document contain references to files which must have specific properties (for example to be valid with respect to another application profile). File constraints are already supported by the Generic Test System.

We designed a revision of the GTS architecture which splits constraints testing into a Generic Constraints Test System (GCTS) which extracts data, like file names or vocabulary identifiers, from test files and specific constraint test system, like file testers or vocabulary testers, doing the actual tests on the provided parameters. The Constraints File contained in the application profile, informs the Generic Constraints Tester where it can find the relevant data in the Test document. The following figure shows this flow of data and control.



In the case of vocabulary testing, the

- Generic Constraints Tester shall determine, based on the profile constraints file, the places in the test document where vocabulary testing is needed and it shall extract for these places the line number, the name of the vocabulary and the value of the entry to be tested.
- It shall write an Instance Data File containing these data.
- Then it informs the GTS Control System that the Instance Data File should be passed to a specific vocabulary tester.
- The Control system calls the specific Vocabulary Constraints Tester through the interface described in the next Section.
- This specific vocabulary tester extracts the data it can use from the Instance Data File, performs the tests. The Instance Data File is in a generic format which is specified at http://iwm.uni-koblenz.de/xsd/inst_addc_v1p0.xsd. Therefore, the Vocabulary Tester

can be designed in a generic way, independent of any particular test data format or application profile.

- Then the vocabulary tester writes a log file which is converted later on by the GTS into an HTML error report page.

4.1.4 Specifying a Vocabulary Tester Interface

As a first step, the interface between the GTS and the VT has been specified as follows. The interface is divided into two parts – command line arguments and an XML file containing the data from the IUT prepared for testing.

The VT shall be called by the GTS from a command line using the following parameters.

Short name	Description	Example
dbd	Document Base Directory: Directory where the data for the current test are held	"c:\\...\<tempDirfor TestFile>/"
idf	Instance data file: Absolute path to an XML file containing IUT specific data	“c:\\..\instanceConstraints.xml”
lf	Log File: Absolute path to the log file to be written	“c:\\..\logfile.xml”
ll	Log level	WARN
pid	Profile Identifier: Name of the profile to be used for testing	“Vocabulary Test Profile”
tf	Test File: Name of the test file	“c:\\...\ilox_scorm.xml”

This design assumes that the location of the vocabularies used for testing can be determined from the name of the profile. In the section “Keeping Vocabularies in Test Systems” below we design a way to achieve this.

Data specific for a particular test are given in a the Instance Data File, say instanceConstraints.xml, specified by the idf command line parameter. This file contains descriptions of vocabulary constraints in the following format.

```
        <ic:vocabularyConstraint constrID="VDEX Constraint">
                <ic:context><!--The root of the part of the document which holds the
parameters -->
                        <ic:line>139</ic:line>
                        <ic:tag>parameter</ic:tag>
                </ic:context>
                <ic:vocabularySourcetype ="vdex-flatTokenTerms"><!--The element holding
the name of the vocabulary -->
```

```
            <ic:content>
                <ic:line>140</ic:line>
                <ic:tag>vocabularyID</ic:tag>
                <ic:content>LRE.packageInValues</ic:content>
            </ic:content>
        </ic:vocabularySource>
        <ic:vocabularyElement><!--The element holding the vocabulary entry-->
            <ic:content>
                <ic:line>141</ic:line>
                <ic:tag>value</ic:tag>
                <ic:content>Not scorm_v2004</ic:content>
            </ic:content>
        </ic:vocabularyElement>
    </ic:vocabularyConstraint>
```

As output of a vocabulary test the VT has to deliver an XML file containing its error reports. The format of this XML file is not predefined. Instead, the VT has to provide to the GTS two style sheets that transform this XML file into an html file that can be integrated into the global error report from the GTS to the user and to another xml file in a specified format listing only the number of errors, warnings and fatal errors. The latter file is used by the GTS to summarize the reports of all test systems for a specific IUT.

4.1.5 Preparing the GTS for Vocabulary Testing

Each machine-readable SchemaProf application profile contains a file describing so-called additional constraints. Such constraints are specified requirements for the structure of files in the IUT which cannot be encoded in XML schemas or in Schematron rules. Requirements to use specific vocabularies are of this type.

For vocabulary constraints, the application profile contains an additional constraints'file which keeps the profile-dependent data to be used by the GTS to call the VT.

The requirement from the LODE profile governing the case in (EX) can be described in the additional constraints' profile

```
<vocabularyConstraint cnd="cnd0" constrID="VDEX Constraint">
<constraintLocation location="//lors:parameter"/>
<vocabularySource type="vdex-flatTokenTerms">
<constraintLocation location="./lors:vocabularyID"/>
</vocabularySource>
<vocabularyElement>
```

```
<constraintLocation location="./lors:value"/>
</vocabularyElement>
</vocabularyConstraint>
```

Conditions are defined in the main profile file and can be referenced by the additional constraints file. For example cnd0 would contain the XPath "../lors:name/lors:value/text()='package in'".

This file from the profile is read by the General Constraints Test System (GCTS). This analyzes the IUT and writes the instance specific constraints' file mentioned above which contains the concrete values from the IUT.

Note that the GCTS does not have to know anything about the type of constraint tests it is working for – all it does is to:

- Extract from the IUT the concrete data for testing.
- Replace the constraintLocation tags, containing XPath expressions, with instanceLocation tags containing the respective content, line numbers and tag names.

Therefore, the same GCTS can also be used for other types of additional constraints, like file constraints, constraining specific properties of files like their size or mime type.

4.1.6 Alternative Concepts

The design described in the previous Section has been drafted to satisfy the following requirements:

1. Do not make any assumption about the profile which requires the vocabulary test.
2. Keep the development of the VT separate from the development of the GTS, in particular it is required that all issues specific for vocabulary testing are handled in the VT only.
3. Avoid additional assumptions about the behavior of other test systems or of the GTS.

Meeting these requirements allows one to extend or replace the VT at a later stage as necessary. Moreover, the integration of the VT in the designed way shall provide experience that can be useful for future integration of other test systems as the calls to the VT do not rely on particular features of vocabulary testing.

Nevertheless, alternative possibilities have been explored before fixing the design described. These alternatives are

1. Encoding the vocabulary constraint in a profile by "abusing" a Schematron rule to issue a call to a VT. The GTS uses this mechanism for triggering tests that specific values are of a specific type. It turned out to be difficult to prepare the data required by the VT using the restricted possibilities of the Schematron rule language. Moreover, this approach would later require more complex modifications of the SchemaProf application profiling tool which already writes information about vocabulary constraints into the additional constraints' file in the profile.

2. The specification of the root element, which keeps source and value together, is given in a separate profile. This requires the development of a specific profile for each set of vocabularies which may be used at a particular location in IUTs. This would require overwriting the specification of the root element given in the base specification which is more than just a profiling operation, since it changes the structure of the existing specification. It would require an ad hoc solution for each vocabulary/base specification. These "profiles" would not be acceptable for the IMS Application Profile Registry.
3. For each vocabulary permitted by the intended application profile, a simple type is defined which consists of an enumeration of all entries of the respective vocabulary. Another type keeps all permitted vocabulary identifiers. A conditional modification in the profile states that, for each permitted vocabulary identifier, the entries in the corresponding value locations in the IUT must be enumerated in the simple type describing this vocabulary. This alternative has the advantage that no special vocabulary test system is needed and that,*in case of fixed vocabularies,*conformance can be tested by any XML validator. It has the disadvantage of requiring complex changes in the test system whenever a vocabulary is changed while, in the solution described in the previous section, it is sufficient to replace the changed vocabulary in the test system.

4.1.7 Keeping Vocabularies in Application Profiles

In order to test whether an entry is from a vocabulary identified by a vocabulary identifier, a test system must be able to locate the respective vocabulary. Unfortunately, there is no standard for locating vocabularies similar to the XML schemaLocation attribute for locating XML schemas. Instead, we propose to keep all vocabularies permitted for a profile in a special directory in the profile named "Vocabularies".

Each vocabulary should be given its own directory in the Vocabularies directory, holding all files which define the vocabulary. In addition, the Vocabularies' directory must contain a vocabulary description file named vocabularies.xml. This file should look like the following fragment.

```
<vocabularies>
        <vocabulary type =" vdex/flatTokenTerms">
                <vocabName>
                        <langstring language="en">IMS GLC LODE Manifestation Name
Vocabulary</langstring>
                </vocabName>
                <vocabIdentifier>http://www.imsglobal.org/lode/lriv1p0/manifestationnamevo
                cabularyv1p0</vocabIdentifier>
                <vocabLocation>vocab1/manifestationnamevocabularyv1p0.xml<vocabLocati
                on>
                <vocabSource>URL of the vocabulary</vocabSource>
        <vocabRetrieved>Date and Time of retrieval from URL</vocabRetrieved>
    </vocabulary>
…
</vocabularies>
```

To fill this structure, a vocabulary management tool is needed which:

- gets a vocabulary from a local or remote file store. In a first approach, to be refined later if needed, it may be assumed that each vocabulary is given as a single xml file. Also, in a first approach, this may be confined to retrieving a single vocabulary while, later on, it may be extended to retrieve from a location all vocabularies or all vocabularies with a restricted set of vocabIdentifiers.
- extracts the required data from the vocabulary
- requests from the user additional information where needed
- creates or updates the vocabularies.xml file
- lists all included vocabularies with their
    - vocabIdentifier
    - first vocabName
    - vocabLocation
- allows deleting or updating a vocabulary

Such a tool has been developed within Work Package 3.

4.1.8 The Test Algorithm of the Vocabulary Tester

A re-usable vocabulary tester must be designed independent of the use of vocabularies in a particular specification. That means it must be given its argument in a form which is independent of the context where it is called. The VT interface described above meets this requirement.

As explained in 5.1.4.we assume that a vocabulary tester gets its parameters as command line arguments and out of an instance constraints file. The instance constraints file contains the data that is necessary for the actual test.

Called with the data as described in the interface above, the Vocabulary Tester proceeds as follows:

- Create a data structure out of the XML filescaching the vocabularies,as defined in the profile actually used for testing, with their identifier, vocabulary name and terms.
- Create a data structure out of the instance constraints file containing the file information with the source and value pairs of the test file.
- For each value contained in the instance constraints file:
    - Check if the source corresponding to the value exists as a vocabulary identifier in the vocabulary data structure.
        - If it does: check if the value exists as a term in the vocabulary element of the vocabulary data structure.
    - In case, the vocabulary identifier is unknown, log an error message.

    - In case, the vocabulary identifier is known, but the term is unknown, log an error message.
    - In case, both the vocabulary identifier and the term identifier are known, log a success message.
- Write the XML error log to be transformed to an html page and to a summary by the GTS.
- Exit and signal completion of work.

Note the flexibility of this algorithm. The look-and-feel of the error reports can be easily changed by changing the style sheet used to present the error report. Also,performance can be increased by caching the valid entries belonging to each vocabulary name in each profile.

4.1.9 Thoughts on Integrating Vocabulary Testing into Application Profiling

Implementing the vocabulary testing design described in this document makes it possibleto easily create test systems for application profiles using VDEX vocabularies. Nevertheless, it can be expected that other types of vocabularies can be handled in a similar way by just replacing the VT without any further change in the GTS.

For the user, developing an application profile with vocabularies, the main hurdle is the need to manually edit the additional constraints' file of the application profile. Instead, it would be better to specify vocabulary usage directly in the SchemaProf tool used for application profiling. In fact, SchemaProf already supports capturing vocabulary root, vocabulary name, vocabulary type and vocabulary entry, either as static values or by specifying elements or attributes where these values can be found at test time. It already writes these data to the application profile.

However, they need to be written yet in the correct format into the additional constraints' file. The main difficulty one needs to overcome in order to achieve this is translating the information from SchemaProf, using XPath expressions for XML schemata into XPath expressions in the instance documents which are, of course, not available when the application profile is captured.

A further need is to augment SchemaProf with an easy to use way to manage the vocabularies used in a profile. This could be possibly achieved by integrating the vocabulary management tool described above.

4.2 Preparing for Testing – The ASPECT Referencer

4.2.1 Objectives and Use Cases

Using the Microsoft/Icodeon converter from SCORM to Common Cartridge is the most convenient way to easily produce Common Cartridges. Nevertheless, experience with this tool in Work Package 5 revealed some problematic situations –not so much arising from the tool itself, as from the context in which it is used.

Tracing back reports, that cartridges converted with the tool did not play properly, showed that one possible reason were missing or superfluous references in cartridges. In fact, the IMS Content Packaging Specification requests that each file in a content package must be referenced in the file imsmanifest.xml and that each referenced file must exist. SCORM Packages as well as Common Cartridges are IMS content packages. If the references in a

SCORM package are not according to the specification, the cartridge obtained by conversion from such a SCORM package will have incorrect references as well, leading to interoperability problems. The Common Cartridge test system could detect such referencing problems. The ADL SCORM Test Suite issues a warning, but not an error, in case of such problems which may not be taken as serious as it is for further processing the common cartridges generated from such SCORM packages. Only testing the converted cartridges or using them in a cartridge player revealed the problem.

In order to avoid this situation, a Content Package Referencer was developed which avoids this situation and corrects it automatically as much as possible.

4.3 Outlook

We have described advances in conformance testing that have emerged since the delivery of Version 1 of this document. Many of these advances were developed in the ASPECT project. These included further refinement of Common Cartridge testing and the provision of first test systems for the new IMS LODE specification. Over the coming months, these new conformance testing possibilities need to be used in practice with data from the ASPECT content providers to verify their correctness and usefulness. It can be expected that this will lead to corrections in the test systems as well as in the respective application profiles. Through the participation of EUN in the LODE specification group, ASPECT LODE testers can support directly the process of further development of this specification.

Implementation work shall concentrate on supporting the use of vocabularies in VDEX format. By integrating this support into the Generic test System it will be made available for a broad range of future specifications and application profiles.

5 References

Dahn, Ingo et all. 2009.*ASPECT Project D3.2.1 Conformance Testing Tools, Version 1.* 2009.

Dahn, Ingo and Zimmermann, Sascha. 2010.*Application Profiles and Tailor-Made Conformance Test Systems.* Submitted to Int. Journ. IT Standards and Standardization (JITSR), 2010.

Heckmann, Patricia et all. 2009.*ASPECT Project D3.5 Best Practice Report for Content Use V2*. 2009

IEEE. 2002. IEEE 1484.12.1-2002, Draft Standard for Learning Object Metadata, Piscataway 2002, retrieved from http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf, 2002

IMS. 2009. Application Profiling Guidelines. *IMS Global Learning Consortium.* [Online] May 20, 2009. [Cited: July 16, 2009.] http://www.imsglobal.org/ap/.

IMS. 2010. *IMS GLC learning Object Discovery and Exchange,* CM/DN Draft 2.0. IMS Internal Communication

W3C. 1999. HTML 4.0.1 Specification[Online] 1999. [Cited: April 16, 2010.] http://www.w3.org/TR/REC-html40/

W3C. 2010. W3C Markup Validation Service. [Online] 2010. [Cited: April 16, 2010.] http://validator.w3.org.