# DERIVING SUB-COURSES FROM CPS PACKAGES BY METADATA TAXONOMY FILTERING

Michael Sonntag, Jörg R. Mühlbacher
Institute for Information Processing and Microprocessor Technology (FIM)
Johannes Kepler University Linz
Altenbergerstr. 69, A-4040 Linz, Austria
{sonntag, muehlbacher}@fim.uni-linz.ac.at

**ABSTRACT**
Creating learning objects (LO) is expensive, so reuse is very important. One possibility is dividing larger packages into small LO, allowing reuse through new assembly. However, this always requires some "glue" between the individual LO to create a new coherent, complete, and larger one. Another possibility is creating a single larger course and deriving various smaller or specific versions from it automatically. These are then complete, or at least much easier to finish, and even matching in look and feel. Although not universally suitable, often for instance a general overview or a detailed sub-topic can be extracted. The only necessity for this is annotation of smaller parts according to an arbitrary taxonomy. According to this taxonomy the LO is then filtered to produce a subset containing only specifically annotated elements. This paper describes a tool for deriving such subsets through converting taxonomy annotations into XML, allowing the use of XPath expressions for filtering.

**KEY WORDS**
Instructional technology, Learning objects, Metadata, Taxonomy filtering

## 1. Introduction

The size of learning objects (LO; [1]) varies significantly. While some describe even single pictures with additional context as LO [2], this approach of making them "tiny" has also been criticized [3]. Contrasting this theoretical discussion, many teachers still create very large LO in practice, typically according to the widely used Content Packaging Specification (CPS, [4]): An existing presentation, covering a two-hour lecture or practice, or even a complete course lasting a full semester, is converted to a single package and often enhanced through additional animations, videos or interactive elements. While such LO are easier to create, they suffer from limited reusability: Only the whole package can be transferred. If smaller elements are needed, they can be extracted, but then only as the "plain" content data, i.e. without their educational context like metadata, references to other elements, etc., in essence creating a new LO from individual resources.

One difficulty of this approach is that any changes to the original (large) LO will not propagate to derived ones. If a course consists of many independent LO, only those changed need to be updated, which might be initiated and performed automatically. It is therefore desirable to link these smaller elements back to their source.

A similar consideration applies to small LO as well, although in a later lifecycle stage. When having been (re-)used to create a compound LO, they might be updated, corrected, enhanced, etc. where currently used. But such changes will not reflect back to the "original" version, perhaps located in some repository [5]. This prevents e.g. corrections to pass to other compound LO including the same, perhaps previously erroneous, small part.

Therefore a method is needed to extract smaller parts from a large LO, breaking it up into its constituting, in the future semi-independent, LO. These smaller parts cannot just be single resources or distinguished according to their type: e.g. a single video files might be a small LO, but an image from a webpage is probably useless without some other page elements, like explanatory text. This is the reverse direction as Wiley [6] describes: LO are not Lego pieces and therefore cannot be combined arbitrarily, but should rather be compared to atoms. Similarly, a compound LO cannot just be split apart according to level of navigation depth, file size/type, length, etc., but must be partitioned according to subject and didactic aspects. Such a method should satisfy the following demands:

- Require explicit semantic annotation: Currently only humans can decide upon an exact content description. To ensure consistent annotation the semantics, i.e. their meaning, must be defined explicitly and clearly.
- Require few/no additional annotation. Metadata is often omitted by authors. Expecting additional one specifically and only for extracting parts is unrealistic.
- Reuse existing metadata standards: The annotation used for selecting parts should ideally be semantically identical to the standard or at least very close. Reinterpretations should be avoided.
- Reuse existing selection standards: Creating a new query or expression language would inhibit adoption by authors. Ideally selection would possess a gentle learning curve, allowing easy start, but being powerful enough for complicated tasks too.

We have integrated a system for selecting subparts of LO modeled according to CPS into a transformer for creating offline views of learning content (see below). This allows creating course representations consisting of a subset of a large LO. Selection is based on the LOM (Learning Object Metadata; [7]) individual elements are annotated with. This allows selection criteria according to a variety of methods, depending only on the information present in the metadata. Examples are subject, keywords, target group, and abstraction level. For actual selection, XPath expressions [8] are used. These however do not operate on the XML metadata representation as contained in the package, but rather a modified one described below, where syntax better matches semantics. The system has been applied in practice to materials for a Java programming course, which is used as an example for expressions.

The remainder of the paper is structured as follows: After a discussion of taxonomy filtering the sample implementation is described. Then follows an example how elements might be annotated and what sub-LO can be derived from it. Section five consists of an evaluation, describing advantages and limitations. The paper ends with conclusions on the usefulness of the method described.

## 2. Taxonomy filtering

While it would be possible to let users select sub-LO by XPath expressions directly on a metadata representation, this results in a complicated and error-prone solution. Additionally, XML is not the only representation for metadata: RDF (Resource Description Framework, [9]) is available too, which is syntactically different and would require completely new selection expressions. Theoretically seen this is also problematic: selecting semantic parts according to their syntactic representation instead of their content mixes layers. E.g. all resources belonging to a single subordinate LO need not be represented in metadata as children of a single parent item, and with no other intermingled content. Only when it is secured that syntax exactly mirrors semantics, both in a specific LO for a certain task as well as conceptually for all instances regarding a software tool, this could be acceptable.

Another option would be foregoing metadata information and selecting parts directly from the navigation specification in a CPS package. However, this is also problematical. While separating parts into chapters can be useful (each chapter as a sub-LO), the depth of a section typically has no semantics. Creating a "general overview" is therefore not possible through e.g. selecting all parts up to a depth of three navigation links: In brief chapters this might be detailed content, in longer ones still an overview. This would also cause problems with special sections like appendices. These occur “normally” within the navigation, but are annotated differently with metadata.

Filtering could be based on arbitrary base information, but as identified above, already existing and well-defined metadata should be used. If possible, it should also retain its original semantics to avoid introducing errors on repurposing. Because of its prevalence and practical importance, LOM in the IMS variant [7] was selected as the base. It is almost identical, and being actively aligned with, the IEEE version [10] and part of the widely used SCORM specification [11]. From the information contained the “classification” element was selected, which consists of a description, some keywords, purpose, and a taxon path (Figure 1, top). A taxon path states all terms from the top of the taxonomy to the most specialized applicable term. Only purpose and taxon path are of interest here; description/keywords are intended for display to humans and not for automatic processing (and omitted in the picture).

The purpose states which characteristic of the resource is described by this classification entry. While it can be set arbitrarily, a vocabulary is defined which may be useful here, although typically used to describe larger LO as a whole. E.g., “discipline” is usually stated solely for the LO, but not for sub-elements. Still, when present and sufficiently detailed, it can be used for subdividing LO.

The taxon path consists of a single source referencing the definition of the taxonomy and a taxon list. These are described through an id, the “official” value from the taxonomy, and an "entry", a human-readable description of the referenced taxonomy element. The entry can be in any language, but the id should be unique for a specific taxonomy. The syntax of this information in XML representation poses an implementation problem: While IMS defines the taxon path as taxons containing taxons (nesting), IEEE (and probably the next IMS version too) specific these as taxons after taxons (flat list). This example also confirms the problems of working on the syntactic instead of the semantic layer as discussed above.

As the query language XPath [8] was selected, because it is easy to understand and with its "axes" concept matches taxonomies. While other approaches exist (see [12] for a Datalog example), these are typically difficult to understand for novice users. Contrastingly in XPath expressions are hierarchically arranged terms with a specific relation between them, which is easy to understand. The relation is typically specialization; every sub-term “is-a” parent term, resembling a taxonomy. Additionally, taxonomies belong to a certain specification or standard. In XML terms this could be called a namespace. A further advantage of XPath is, that although getting started is easy, it allows very complex expressions as well when necessary.

Because XPath works on a syntactic level but matching should take place on the semantic one a transformation is required. Here a small variation is made in favour of handling: While conceptually the important matching element is the taxonomy id, it would be hard for humans to create a selection based on them. Therefore, the taxon value is used as the primary data element instead. Matching in expressions is still possible on the id, too.

Unlike [13], the result is still XML and not RDF to better support queries formulated in XPath. Both structure and query are more natural and probably easier to understand for users than RDF and query languages based on specific types of logic.

The transformation works as follows:

1. An empty XML document, and for each classification a new root element, is created. So any number of classifications is available simultaneously. These elements are named after the purpose of the classifications, with the source added as an attribute.
2. The first taxon value is added as child of the purpose element, with its id as attribute. The namespace for it is created from the source of the taxon path. This allows ignoring it for more simple selection and when only a single type of classification is present.
3. For each further taxon a new sub element is nested within the previous (similar to the LOM XML binding) taxon representation. The nesting ensures, that XPath expressions can be written according to the child axis, regardless how the taxons are described in the source; see the changing standards above.

A simple example is shown in Figure 1. The first part is the original XML representation according to LOM, the second the generated structure for use through XPath.

**Original (LOM):**

```
<classification>
  <purpose>
    <source>LOMv1.0</source><value>discipline</value>
  </purpose>
  <taxonPath>
    <taxon>
      <id>9.3</id>
      <entry><string language = "en">Information
                                    Science</string></entry>
    </taxon>
    <taxon>
      <id>9.3.1</id>
      <entry><string>Information Processing</string></entry>
    </taxon>
    <source>
      <string>http://www.example.com/science</string>
    </source>
  </taxonPath>
</classification>
```

**Converted for filtering:**

```
<discipline source="LOMv1.0">
  <InformationScience id="9.3"
            xmlns ="http://www.example.com/ science">
    <InformationProcessing id="9.3.1"/>
  </InformationScience>
</discipline>
```

**Figure 1: Transformation example**

## 3. Implementation

The system has been integrated into existing Java software to transform CPS packages into an offline version. It has recently been added into the CPS editor jCAPT (see [14] for more information and download), where it, in addition to conversion, serves for previewing packages. Within these programs it has to filter the individual elements according to visibility anyway, which is a standard feature of CPS. So integrating a further filtering step is straightforward. Although not supported by this software, it would be simple to not convert the result but export it to a (new) CPS package of only matching items.

The first step in filtering is converting the source metadata into the new form as described before. While this is possible programmatically, a more flexible approach was chosen: an XSLT stylesheet transforms a specific metadata record into the new form. This stylesheet is applied to each metadata record within the items separately. The resulting XML document is kept in memory only and then used as input for the filtering stage. Employing stylesheets brings the additional advantage that source changes, both in structure and content, can be easily integrated: The various version of IMS and IEEE LOM are already marked with different namespaces. These can be integrated into stylesheets and easily extended for new variations, obviating program changes.

As "entry" values of taxons are arbitrary strings, a translation to valid XML element names is necessary. Because there most Unicode characters are allowed, this is not difficult. E.g. spaces are left out (Figure 1: "Information Processing" → "InformationProcessing"). Similar methods are used to convert the taxon path source into an URI.

The filtering stage consists of applying the expression entered by the user to the resulting document and converting the return value to a Boolean. Only if "true" the item is shown in the output, otherwise it is suppressed. In the software this means that also related resources, i.e. the learning content, are not packed into the offline version.

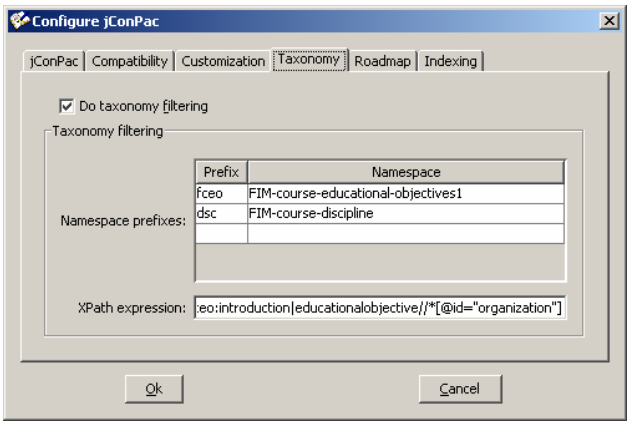


**Figure 2: UI for configuring taxonomy filtering**

What has been an advantage above, the different namespaces, is a slight difficulty in the user interface (UI). As the taxon path source is converted into a namespace, the XPath expression must necessarily contain this namespace, typically as a prefix for abbreviated writing. Therefore the UI consists not only of the expression to use for filtering. Additionally a box for entering the namespaces with their prefixes is present, which are used in the expression (see Figure 2). These prefixes may differ from those in the metadata, as XPath expression matching uses the full namespaces for comparison; prefixes are used only for easier construction and better readability. However, most expressions will only require a single namespace. Exceptions are when several different classifications, not only several values from a single one, are combined for selection.

## 4. Example

The system has been tested through an introductory course on Java programming. The manifest and the learning materials contain not only the lecture part, but also the accompanied practice, i.e. exercises assigned to students (see Figure 4 for the basic structure; some chapters omitted for clarity and size). However, these are actually two separate courses, although tightly integrated (lecture and practice). Therefore material must be provided separately too: students may also (re-)take only one. The annotation combined with taxonomy filtering allows creating both practice and lecture material. Some elements are common, e.g. introductory notes on course organization, while others are specific, like required software and its installation (practice) and final exam information (lecture).

Additionally two more "extracts" are used. Firstly, a subset is created according to the topic, i.e. selecting a single or several chapters. These could be used for special courses, as repository for refreshing, or for obtaining smaller parts, e.g. to transfer them to mobile devices with limited resources. The second subset serves as an example for a cross-section: all introductory parts are combined. While probably not very useful for teaching, it could serve as a kind of extended table of contents, showing prospective students all the topics contained in an overview.

For classifying the various parts two simple custom taxonomies were created. The first describes the topic of a section. This is a classical subject hierarchy starting with "Java" at the top and then covering sub-parts like "arrays", "strings", etc. The second taxonomy is used to describe which part of a course some material belongs to and what role it fulfills there. This hierarchy starts with "lecture" or "practice" (other possibilities are irrelevant for this specific course) and then continues with "slides", "examples", "tasks", etc. An example of a classification taken from the course is shown in Figure 3.

```
<metadata xmlns="http://ltsc.ieee.org/xsd/LOMv1p0">
<lom><classification>
    <purpose>
        <source>LOMv1.0</source><value>discipline</value>
    </purpose>
    <taxonPath>
        <source>
            <string language="en">FIM-course-discipline</string>
        </source>
        <taxon>
            <id>Java</id>
            <entry>
                <string language="en">Progr. Lang. Java</string>
            </entry>
        </taxon>
        <taxon>
            <id>Strings</id>
            <entry>
                <string language="en">Strings</string>
            </entry>
        </taxon>
    </taxonPath>
</classification></lom>
</metadata>
```

**Figure 3: Taxonomy annotation example**

### 4.1. Lecture/Practice extraction

For deriving the two separate materials the following expressions based on the educational objective are used:

- Lecture: educationalobjective/fceo:lecture
- Practice: educationalobjective/fceo:practice

For both, the namespace prefix "fceo" is mapped to "FIM-course-educational-objectives". As shown in Figure 5 and Figure 6, only presentation and examples, respectively tasks and their solutions are present. Not shown in this example is, when a chapter is only present in one part of the course. For instance, a section on software engineering would perhaps be part of the lecture only. It would be omitted completely in the practice, including the chapter heading, etc. One specialty of this approach is, that, unlike in the CPS specification, visibility is inherited: if a chapter is invisible, all sub-parts are omitted too.

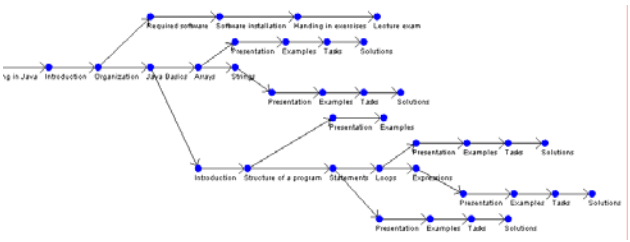


**Figure 4: Complete course**

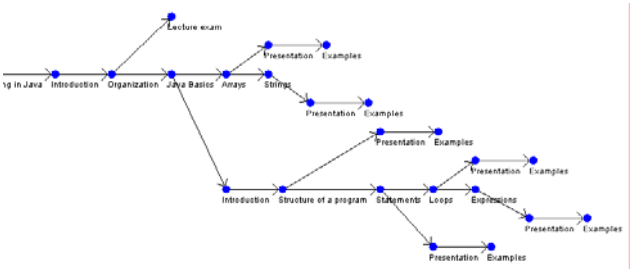


**Figure 5: Lecture subset**

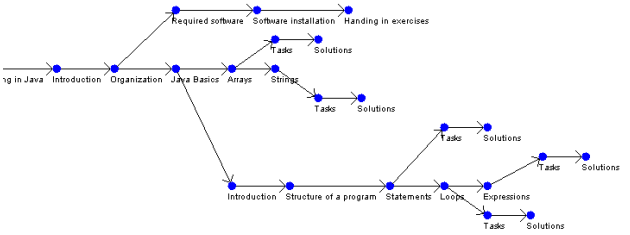


**Figure 6: Practice subset**

Therefore parent elements must be annotated according to all classifications that some child elements might possess. Otherwise, these would not be reached during the inspection to be checked against the expression, and therefore also never shown in the result. This was a design decisions to keep the system simple and intuitive, as local experience showed that teachers are easily confused by the non-inheritance of visibility as specified by the CPS specification: visible child elements are shown in place of their invisible parents.

### 4.2. Selecting the string chapter

Selecting a specific content chapter is simple. For example, selecting only the part on strings would use the expression "discipline//dsc:Strings" with the namespace "dsc" mapped to "FIM-course-discipline" (see Figure 7 for the resulting structure).

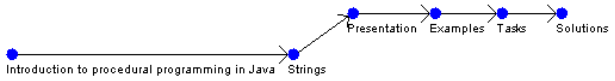


**Figure 7: String chapter**

Noteworthy within the expression are the double slashes, resulting in all elements containing somewhere the annotation "Strings" to be included. Although not present in the example, this would include not only "Java/Strings", but also "API/Strings" or "HelperClasses/Strings", reducing the expression complexity and allowing easy extension.

### 4.3. Introductory version

A more complex expression is needed for generating an introductory course, i.e. including only the general sections but leaving out examples, tasks, etc. For this "educationalobjective/fceo:lecture/fceo:introduction | educationalobjective//*[@id="organization"]" is used (see Figure 8). The prefix "fceo" is mapped identical as above.

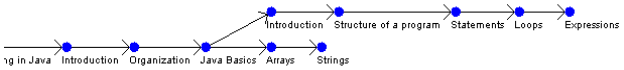


**Figure 8: Introductory version**

This expression consists of a union of the introduction section from the lecture and all organizational parts. For the latter a different approach than before was taken. Instead of using the (language-dependent) entry here the id is used exemplarily. Wherever in the hierarchy the id is "organization", this element is included in the output.

## 5. Evaluation

The approach of creating an intermediate representation for classification metadata results in several advantages:

- Regardless of the metadata format (version/specification) the selection is always identical. This is possible because the differences are only of syntactical nature and the same content can be represented in all of them. Expressions are therefore portable between versions but also between learning materials.
- A clear and simple syntax according to a public standard draws upon existing expertise by users. The learning curve starts slowly, so getting started is simple. Simultaneously complicated selections (see example on introductory version) are possible as well.
- Separation from the syntax allows also different source representations, e.g. derived from basic metadata in RDF. Simultaneously the (previously purely technical) format is changed to a more semantically-oriented representation: sub-classes are described by consecutive steps in the expression and the area they belong to is converted to an easily understandable namespace.

However, some difficulties exist as well:

- As the main variant is based on the language-dependent description, translated or misspelled content might lead to erroneous results. Unfortunately, this is rather difficult to prevent. While using the ID is always possible (see above) this would need extensive UI support for constructing the expression. Such support is only possible when the taxonomy is known exactly and completely. As the taxonomy itself is not included in the package, an improved user interface would only be possible if the taxonomy were integrated into the software or retrievable through a standard method and in a standardized format.
- Conversion to element names and namespaces is required. While omitting spaces is quite simple to recognize and remember for users, other changes which might be necessary could complicate the expression. As users must re-do these in their mind to create correct and matching expressions, this may become a liability for complex naming schemes. However, especially the taxonomy values can be expected to be "ordinary" words for the creator of learning material and not consist of special characters (signs). A different approach could be modeling the taxon path source not as a namespace but rather as an attribute. In this case no transformation would be required. However, then the syntax would not be as similar to the semantics as possible and additionally collisions might occur through the same term being used in two separate classifications, which is avoided here. Moreover, expressions would become more complex.

The method has been applied to a course on Java programming. Experiences with this system have been positive, as only a single source material package had to be created and kept updated. An additional benefit is the encouragement of adding metadata. While simple and straightforward here, this metadata is available for later reuse and for discovery in repositories as well. As not only the course as a whole is described but its sub-elements as well, reuse of parts is made easier.

## 6. Conclusions

While the approach is already useful and has proven its suitability, further enhancements are possible. As identified previously, integrating at least some classifications into software would help both authors during annotation [15] as well as users in constructing selection expressions. Adding these taxonomies should not be based on direct hard-coding but rather in a standard exchange format. This would allow institutions to add their custom taxonomies in addition to publicly available ones. A specification that could be used is VDEX [16], which supports both vocabularies and taxonomies.

Another aspect for further improvement is annotation itself. While technically simple to do, its amount could perhaps be reduced. As previously identified, all elements within a navigation path must be marked accordingly for the "lowest" item to show up after filtering. While this possesses the advantage of always working correctly, even when extracting parts as new LO, it requires extensive annotation. One possible approach to reduce the amount of annotations is their inheritance.

Classification inheritance is already present in a certain sense as items not annotated at all are implicitly defined as similar to the annotation of their nearest ancestor. If this ancestor is visible, all sub-elements without annotation are visible as well. Interesting here is however the reverse direction: If an element is marked for instance as "A/B/C", what can be said about its unannotated ancestors, so that these perhaps need not be annotated? All ancestors are probably at least of type "A/B", otherwise the specific classification would not be contained at that place. It cannot however be determined whether there are elements of type "A" only: all up to and including the root might be "A/B" too in a specialized course.

But a significant problem occurs here: If the current item is not part of a homogenous course but rather an independent LO integrated at this specific position, this approach might err. The reason is that this method implicitly assumes that the first occurrence of a sub-classification is annotated. However, especially in the case described, this need not be so. Elements might be present without metadata and only in the "middle" suddenly, e.g. through including external resources, the annotation appears. Moreover, nothing can be derived about siblings. They might be of the same classification (metadata omitted), of a parent one (general overview on the detailed information in the current item), of a sub-classification (additional details), or an unrelated one (separate chapter).

Therefore currently manual annotation is the only reliable possibility and should be supported by tools to ease the workload of authors and encourage them to add this information. Some help could be given through the guidelines outlined briefly, but a completely automatic system seems not feasible in this way. Additionally, as the IMS/IEEE LOM specifications provide no way to specify that some metadata was derived automatically as compared to manual annotation, parts extracted as new LO would then contain this information as misleadingly "authoritative" from the content author.

Automatically deriving subsets of courses through their classification provides benefits based on existing metadata. Through this additional use case for metadata, authors might be encouraged to add metadata to the LO created by them, which is currently still a problem. Through enabling the creation of a single master course with various automatically derived sub-courses, problems through missing updates or duplicated materials can be avoided, reducing maintenance efforts.

## References


[1] A. Ip, I. Morrison, M. Currie, What is a learning object, technically? *WebNet2001 Conference* http://users.tpg.com.au/adslfrcf/lo/learningObject(WebNet 2001).pdf

[2] M. Sosteric, S. Hesemeier. When is a Learning Object not an Object: A first step towards a theory of learning objects. *International Review of Research in Open and Distance Learning*, 10/2002 http://www.irrodl.org/content/v3.2/soc-hes.html

[3] P. R. Polsani, Use and Abuse of Reusable Learning Objects, *Journal of Digital Information*, *3*(4), 2003 http://jodi.ecs.soton.ac.uk/Articles/v03/i04/Polsani/

[4] IMS Global Learning Consortium: Content Packaging Specification. http://www.imsglobal.org/content/packaging

[5] Z. Shen, Y. Shi, G. Xu, A Learning Resource Metadata Management System Based on LOM Specification. *The 1st International Conference on Web-based Learning (ICWL2002)*. http://media.cs.tsinghua.edu.cn/~pervasive/paper/200209-1.pdf

[6] D. A. Wiley, Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In: A. D. Wiley (Ed.). *The Instructional Use of Learning Objects*, Bloomington, IN: Agency for Instructional Technology, 2002, 3–23 http://www.reusability.org/read/chapters/wiley.doc

[7] IMS Global Learning Consortium: Learning Resource Meta-data specification. http://www.imsglobal.org/metadata

[8] W3C: XML Path Language (XPath). http://www.w3.org/TR/xpath

[9] W3C: Resource Description Framework (RDF) http://www.w3.org/RDF/

[10] IEEE Standard for Learning Object Metadata 1484.12.1-2002

[11] Advances Distributed Learning Initiative: Sharable Content Object Reference Model (SCORM) http://www.adlnet.gov/scorm/

[12] J. Brase, W. Nejdl. Ontologies and Metadata for eLearning. In: S. Staab, R. Studer (Eds.) *Handbook on Ontologies*. Springer 2004, 555-574. http://www.kbs.uni-hannover.de/Arbeiten/Publikationen/2003/Ontologies_for_elearning.pdf

[13] C. Qu, W. Nejdl. Integrating XQuery-enabled SCORM XML Metadata Repositories into an RDF-based E-Learning P2P Network. *Educational Technology & Society*, 7 (2) (2004), 51-60

[14] jCAPT – Java Content Assembling and Packaging Tools. http://jcapt.fim.uni-linz.ac.at/

[15] A. Brasher, P. McAndrew. Metadata vocabularies for describing learning objects: implementation and exploitation issues. *Learning Technology Newsletter* 5 (1), 24-27 http://lttf.ieee.org/learn_tech/issues/january2003/#9

[16] IMS Global Learning Consortium: Vocabulary Definition Exchange. http://www.imsglobal.org/vdex/