Automating Metadata Generation:
the Simple Indexing Interface

Kris Cardinaels, Michael Meire, Erik Duval
Dept. Computerwetenschappen, Katholieke Universiteit Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
{kris.cardinaels, michael.meire, erik.duval}@cs.kuleuven.ac.be

ABSTRACT
In this paper, we focus on the development of a framework for automatic metadata generation. The first step towards this framework is the definition of an Application Programmer Interface (API), which we call the Simple Indexing Interface (SII). The second step is the definition of a framework for implementation of the SII. Both steps are presented in some detail in this paper. We also report on empirical evaluation of the metadata that the SII and supporting framework generated in a real-life context.

Categories and Subject Descriptors
H.3.1 [**Information Systems**]: Information Storage and Retrieval – *Content Analysis and Indexing*

General Terms
Design, Algorithms, Experimentation

Keywords
Learning Objects, Metadata Generation

1. INTRODUCTION
One of the main concerns in learning technology research is the problem of acquiring the critical mass to establish real reuse. There are several aspects to the solution of this problem. Many projects focus on the creation of content and how this can be made easier, faster or cheaper [25, 14]. Other projects focus on interoperability aspects [11]. The aspect we focus on in this paper is the creation of metadata. Without appropriate metadata no learning content will be really reusable because it will be difficult or impossible to identify and retrieve it.

Learning object metadata has been researched in several projects. One great achievement was the development of the IEEE LOM standard [17], based on the original ARIADNE pedagogical header definition, and adopted in the widely deployed ADL SCORM reference model [1]. The creation of these metadata, however, currently turns out to be a problem for most systems:

- Most reuse initiatives are still struggling to achieve critical mass,
- Many learning objects only have a very limited set of metadata associated to them [20].

We consider several reasons why users often do not make the learning objects available for reuse or do not create metadata for those objects (see also [12, 14]). Most importantly, the current tools available for metadata creation are not user friendly. Most tools directly relate to some standard and present that standard to the users. The user has to fill in a substantial number of electronic forms. However, the standards were not meant to be visible to end users. A direct representation of these standards on forms makes it very difficult and time consuming to fill out the correct values for the metadata in substantial quantities. The slogan that "electronic forms must die" addresses this specific concern.

A possible solution to this problem is the automatic creation of learning object metadata. In this way, the users do not have to bother with the metadata if they do not want to. This can be compared with search engines on the web that index web pages in the background without any intervention of the creator or the host of the site. In our approach, if the user wants to correct, add or delete metadata, he will still be able to do so, but most users will not need to spend time on it.

In this paper, we introduce a framework to set up an automatic metadata generation system as a web service. This web service generates IEEE Learning Object Metadata although in the future other metadata schemas should be supported as well. Depending on the type of learning object document, the created set can be rather small or more extensive. We at least try to generate a metadata set that contains all the mandatory elements defined in the ARIADNE Application Profile [2].

2. AUTOMATIC METADATA CREATION

2.1 Introduction
In many learning management systems, metadata can be associated with learning objects manually, or they can be generated partially by the system (see Figure 1). It is our opinion that manual creation of metadata might be feasible in small deployments, but that it is not an option for larger deployments where a considerable number of learning objects are to be managed for each user. The system should offer functions comparable to search engines and classifiers for the web (see also [21]). Search engines index web pages automatically without manual intervention of the users or the creators of the pages.

If learning management systems can offer a similar functionality for learning objects, the users will provide much more easily a great number of learning objects and real reuse will become feasible.

Figure 1: A combination of manually and automatically filled in metadata in the Blackboard LMS

2.2 Metadata sources

In [4] and [5], we already introduced different methods for automatic metadata generation. Metadata can be extracted from different sources that are available to the system. We distinguish four main categories of metadata sources:

1. Document *content* analysis: One obvious source for metadata about an object is the object itself. An object-based indexer generates metadata using the object independent from any specific usage. Typical content analyzers are keyword extractors, language analyzers for text documents or pattern recognizers for images.

2. Document *context* analysis: When an object is used in a specific context and data about that context are available, we can rely on the context to obtain information about the object itself. One single learning object typically can be deployed in several contexts which provide us with metadata about it. Section 2.3 provides more details on this kind of metadata source.

3. Document *usage:* Real use of objects can provide us with more flexible and lively metadata than the sometimes more "theoretical" values provided by other metadata sources, or even by human indexers. Systems that track and log the real use of documents by learners are therefore a valuable source. These logs for example store the time spent reading a document or solving exercises. This metadata source category could be considered as a "usage context", and as such as a special case of document context analysis.

4. Composite documents structure: In some cases, learning objects are parts of a whole but stored separately [5]. In such a case, the metadata available for the whole is an interesting source for metadata about a component. Not only is the enclosing object a source, also the sibling components can provide relevant metadata (also look at [28]). For example, one slide in a slide show often gives relevant context about the content of the next slide. This could be considered as a special case of document context, namely "aggregation related context". It also closely relates to the issue of "content packaging".

Therefore in the future we will look at things like SCORM Content Packages and IMS Content Packaging.

The first category is rather straightforward to understand and does not need much further explanation. The third and fourth are not yet further investigated by us. Therefore we will not elaborate on them here. We are however in the process of also dealing with those kind of metadata sources in our framework. The second category, the context analysers, is worth some more explanation.

2.3 Context analysers

Learning objects can be used in several contexts; each context contains metadata that might be usable for the automatic indexers. Some typical context types we can observe are:

- Creator profiles
  In [4], we classified this indexation as profile-based indexation. Every learning object is authored by one or more people. Quite often information about these people is available from different sources. A creator or indexer profile groups this information, so that it can be used when generating metadata for a document of that person.
  Those profiles can both be generated manually and automatically. In the manual case, the user provides some pre-filled templates with information that is likely to be correct for most of the learning objects.
  The profile can also be filled in (semi-)automatically. At the K.U.Leuven, for example, course information for each teacher is available on websites and personnel information is available in the administrative SAP backbone. Course information includes metadata about the audience of the course, the language in which the course is taught, the duration of the sessions, and so on.

- Learning content management systems
  If learning objects are stored together with their metadata, available metadata can be used as a source for newly introduced learning objects. This information is typically used if the new object is related to another object already stored in the system (as a new version of the existing one, for example) [3]. Moreover, similarity searches [4] can be used to search for similar objects in the system, so that their existing metadata can be used to create new metadata.

- Learning management systems
  Learning management systems can provide rich contextual information, like the courses in which the object is used, how many times the document was used or downloaded, etc. As such, it actually does both document context analysis and document usage analysis.

3. DIFFERENT SOURCES – DIFFERENT VALUES

Relying on different sources of metadata augments the process of generating metadata automatically. However, the sources may generate different values for the same metadata element. In this section, we present the options we have to overcome this problem. First we discuss the need for "correct" metadata, without requiring a

formal approved metadata set for each learning object. Then we present four options in solving conflicts between indexers.

3.1 Correctness of metadata

In the first implementation of the ARIADNE Knowledge Pool System, we used to distinguish between validated and unvalidated pedagogical headers or metadata instances. A newly introduced metadata instance for a learning object was given the status 'unvalidated' and only validators could change the status to 'validated'. Validation was a process of checking the metadata values for their correctness. If some of the values in the metadata were incorrect, the validator could change those values or the original indexer had to modify them. Note that the validation process did not focus on the learning object, but rather on the metadata [10].

Quite quickly, however, it became clear that this system did not work as expected. Only validated learning objects could be used in the ARIADNE system, other objects would not be included in the results to queries. So, any user that introduced new material had to wait until validation before it could be deployed in a course. A certain pressure was put on the validators to do their job quickly, but probably also inaccurately.

The same problem arises in other systems, even if those systems spend a lot of effort on the validation process. For example, the Merlot [18] system uses peer review for their contents, but only about ten percent has been reviewed: in the Science and Technology category 465 of 4536 documents had a peer review record associated in the database at the time of writing.

The question we ask ourselves is whether metadata can be incorrect. The difficulty is that we cannot define correctness in terms of right or wrong in case of metadata – or at least not for all metadata elements. Of course some values may be "better" than others, but that does not necessarily imply that the latter values should not be used.

3.2 Conflicts between indexers

The framework we present in the next section uses different classes of *indexers* that can work in isolation from each other. Each indexer generates values for some metadata elements, and as such a subset of a metadata set. These subsets have to be combined into one resulting metadata record for the learning object.

The subsets of metadata that different indexers generate can overlap. In this case, there may arise a conflict between the indexers, that has to be solved. There are several strategies to solve the conflicts; depending on the element, one strategy might work better than another:

1. Include all the generated values in the resulting metadata set,
2. Propose the options to the user and let him/her decide which one to use,
3. Try to find out which indexers are the most likely to be correct and use their value in the result,
4. Apply heuristics to decide on the value.

The first option – including all the values in the resulting set – is the easiest to implement and might be feasible for some metadata elements. For example, a list of concepts could contain all the keywords extracted by several indexers. In some systems, however, the metadata set is strictly defined so we cannot implement this as an overall strategy for all the elements.

The second option could be used in a small system with only a low number of new entries per week or month. In larger systems, however, we would lose all the benefits of automatic indexing as the user has to spend time on controlling all the values and decide which one to use.

In our opinion, the third option is most interesting in many cases. Every generated value will get an associated value which is the degree of certainty of the indexer about that value. We call this value the *confidence value* in our framework. Every indexer determines such a value for the metadata elements it generates. In case of conflict, this strategy will prefer a value with a higher confidence value over one with a lower value.

The fourth option applies in certain cases if heuristics are known about the metadata elements. In that case, the heuristic will provide the solution about the conflict. An example element for which heuristics can apply is the document language. A lot of families of languages exist and in those families the differences between languages might be very small. For example Italian and Catalan are closely related to each other but are different languages; the same applies to Afrikaans and Dutch. If one indexer decides the language is Catalan, the heuristic might say to use Italian. In either case, if the document is used in an Italian or a Catalan environment, the users will understand the contents and thus be able to use the object. Applying Catalan for the document language however could be more precise but the value Italian is not wrong.

4. AUTOMATIC INDEXING FRAMEWORK

The overall structure of our framework is depicted in Figure 2. For now, the idea is that learning object metadata can be derived from two different types of sources, which represent category 1 and 2 in section 2.2.[1] The first source is the learning object itself; the second is the context in which the learning object is used. Metadata derived from the object itself is obtained by content analysis, such as keyword extraction, language classification and so on. The contexts typically are learning management systems in which the learning objects are deployed. A learning object context provides us with extra information about the learning object that we can use to define the metadata.

Following this idea, the framework consists of two major groups of classes that generate the metadata, namely Object-based indexers and Context-based indexers. The object-based indexers generate metadata based on the learning object itself, isolated from any other learning object or learning management system. The second class of indexers uses a context to generate metadata. By working this way, the framework is easily extensible for new learning object types and new contexts. To be complete, the framework also has some Extractors that for example extract the text and properties from a PowerPoint-file, and a MetadataMerger that can combine the results of the different indexers into one set of metadata.

---

[1] As noted above, in the future we certainly will also look at options 3 and 4.
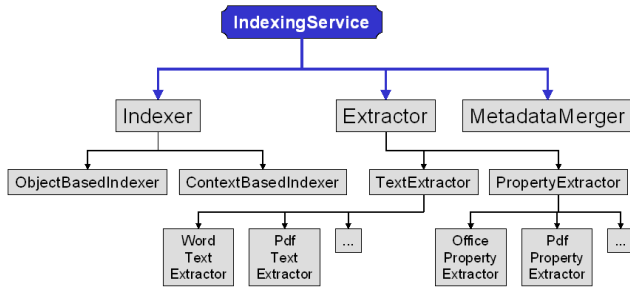
Figure 2: Overall Structure of the Automatic Indexing Framework

Figure 3 and Figure 4 explain the hierarchies of the indexers more in detail. Of course, these hierarchies are extensible with other indexers.

4.1 ContextBasedIndexer

We already explained the use of contexts for learning object indexing. There is, however, one subclass that needs more explanation: FilesystemContextBasedIndexer. A learning object as a file is always stored in some context of the operating system it belongs to. This class represents that file system and contains metadata that the file system stores about the object. Depending on the file system, the metadata attributes vary.

We also implemented some specific LMS contexts. Currently we have classes that generate metadata for a Blackboard document, or an OpenCourseWare object. Basically, these classes mine the consistent context that courses in both environments display. For example, Blackboard maintains information about the user logged in (a reasonable candidate author of learning objects newly introduced), about the domain that the course covers (a reasonable candidate for the domain of the learning objects that the course includes), etc. Similarly, the OCW web site is quite consistent in how it makes that kind of information available to end users. Our indexer for OCW basically mines this consistent structure for relevant metadata about the learning objects referenced in the course web site.
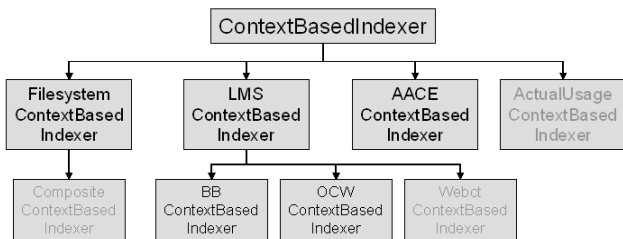

Figure 3: The Class Hierarchy of ContextBasedIndexer

4.2 ObjectBasedIndexer

These indexers often work together with the Extractor classes, to generate metadata that is derivable from the learning object contents. Several specific learning object formats can be implemented as subclasses of MimeBasedIndexer. These classes are able to deal with a particular type of files like pdf documents.

Furthermore, there is a language indexer, that can determine the language of a piece of text.

The ArchiveIndexer class is used to handle bundled learning objects, such as different web pages with links between them or pictures included.

In the future, this category of indexers can be extended by using for example Artificial Intelligence techniques. There exist libraries like iVia [27] that allow for things like keyword extraction, automatic document summarization, etc. We should also look at the domain of information retrieval to see what exists already in that domain.

The size of these classes may vary. At the moment, most classes can generate values for about 5 metadata elements. Depending on the complexity to generate these values, the classes contain only a few lines of code to several tens of lines.
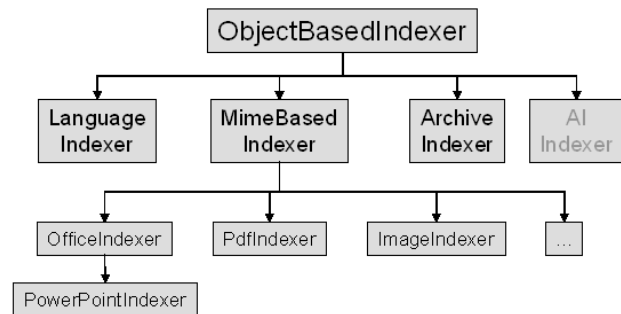

Figure 4: The Class Hierarchy of ObjectBasedIndexer

4.3 Implementing Specific Indexers

Developers wanting to implement their own indexer classes extend the above base classes. Indexers that handle documents or objects of a specific type, e.g. Microsoft PowerPoint files, should extend the ObjectBasedIndexer class. If a specific learning object context like a specific learning management system, should be handled, the ContextBasedIndexer is the appropriate super class. This is what we, for example, did for the Blackboard LMS case, as explained in section **Error! Reference source not found.**. We now briefly explain the interfaces that must be implemented to create these specific indexers.

The overall interface Indexer is only defined to have a general type for the different indexer classes. There are no methods defined in this interface. It is extended by the ObjectBasedIndexer and ContextBasedIndexer.

The method in the ObjectBasedIndexer interface that has to be implemented is defined as follows:

```
public void addMetadata(
      DataHandler lo,
      String fileHandle,
      AriadneMetadataWithConfidenceValue
      metadata);
```

In our framework implementation, this method is called by the indexing service which accepts a learning object and calls the appropriate indexers to create metadata. The first argument in the method is a reference to the learning object itself. The second argument is the file name for the learning object, which is included

as an auxiliary argument to make the implementation easier. The last parameter is the metadata object to which the new metadata will be added.

The class ContextBasedIndexer defines a similar method to create metadata and another method to retrieve a reference to the learning object within the context:

```
public void addMetadata(
        AriadneMetadataWithConfidenceValue
        metadata);

public DataHandler getDataHandler();
```

4.4 Using the classes

We now briefly describe the use of the above presented classes to generate metadata for a learning object in some context(s). If someone wants to obtain metadata, some distinct steps have to be followed:

1. The user has to identify what are the object and the contexts within which the object resides. To simplify this, we introduce a new class, that identifies the learning object, or the context that a learning object resides in, called a *MetadatasourceId*. This top-level class is an abstract class which must be sub-classed by specific classes for each context in which learning objects can be identified. Examples of MetadatasourceId classes are FileSystemMetadatasourceId, BBMetadatasourceId and OCWMetadatasourceId. The last one for example identifies the "OpenCourseWare" context of an OCW document. Concretely, this identifier is nothing more than the URL location of the OCW document, as from that URL we can fully identify the OCW document. The FileSystem context is the one we use to identify the learning object in the context of the OS file system, and so in no specific context of a LMS.

2. The identifying objects we just made, are then fed to the system, which uses the identifiers to create the correct Indexer instances for the metadata generation. The decision on which indexers are applicable for the document is made based on the file type (for example MS PowerPoint) or defined by the context that is provided.

3. For each Indexer instance associated with the learning object, the system sends a request to create metadata for that object.

4. As described in section 3 we need conflict resolving between different metadata instances. For now, we only implement strategy 3, which comes down to working with degrees of certainty for generated metadata values, and choosing the one(s) with the best confidence value. This is implemented by the MetadataMerger and AriadneMetadataWithConfidenceValue classes. The last one represents the metadata instance, with associated confidence values for each metadata element. When adding an element to the instance, the confidence values are checked. Only if the confidence value for the new element is higher than the current one, the new one replaces the old one. Otherwise, the old value is preserved. The MetadataMerger class can merge to existing metadata instances into one, according to the same strategy. In next versions of the framework, we want to make it more flexible, allowing other merging strategies as well.

5. AN AUTOMATIC INDEXING SERVICE

We implemented the above framework as a web service. We briefly explain the methods of this service which we call the *Simple Indexing Interface*: in essence, this is an application programmer's interface to implement the services. This interface is being developed as a part of our research on the development of a global framework for learning object web services. The first initiative in this context has led to the development of the Simple Query Interface standard [24], a definition of web services that enable querying Learning Object Repositories in a standardized way. Our specification for the indexing interface closely relates to SQI and uses the same design principles.

The different methods that should be implemented are given in Table 1. A typical course of action is illustrated in Figure 5.

Session handling methods
startSession
endSession
Session Configuration
setMetadataLanguage
setConflictHandlingMethod
setMetadataFormat
getSupportedConflictHandlingMethods
getSupportedLanguages
getSupportedMetadataFormats
Learning Object Indexing
getMetadata
getMetadataXML

Table 1: Methods of the Simple Indexing Interface

6. THE BLACKBOARD CASE STUDY

6.1 Binding to the Blackboard LMS

To make a proof of concept of the argument that learning management systems are interesting sources of metadata, we designed and implemented a metadata generator for the Blackboard Learning Management System.[2]

We already had some experience with the coupling of Blackboard and Ariadne, as we made a Blackboard Building Block for the Ariadne Knowledge Pool System [26]. This building block makes it possible for users of the Blackboard system to attach documents to a course, as Ariadne documents. This means that the document is not inserted in the Blackboard specific database, but as a learning object in the Ariadne Learning Object Repository. This approach has the advantage that the document is not enclosed within a specific content repository. Instead, it resides in a knowledge pool system that can be accessed from other environments as well.

---

[2] This was an obvious choice for us, because we use Blackboard at K.U.Leuven University. Moreover, the BlackBoard Building Block mechanism allowed us to develop an interface between the ARIADNE repository and the BlackBoard LMS.
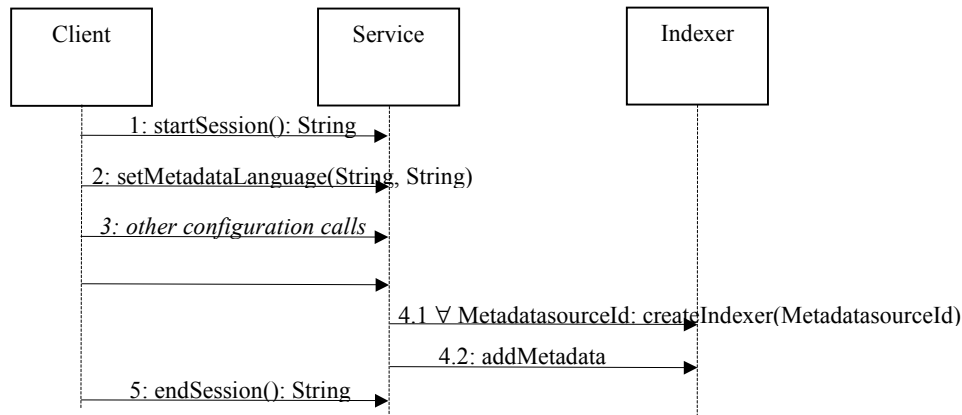
Figure 5: Interacting with an Automatic Indexing Service

It is necessary to ensure that the learning object is inserted in Ariadne in a reliable way, by ensuring that the generated metadata is of good quality. To have some kind of reassurance for this, we implemented a test system for the documents that are already present in the blackboard system. For all these documents, we wanted to automatically generate metadata instances of good quality.

6.2 Blackboard Learning Object Context

In our indexing framework we inserted a component ToledoBBContextBasedIndexer[3] as a subclass of BBContextBasedIndexer, which in turn is a subclass of the more general LMSContextBasedIndexer. The ToledoBBContextBasedIndexer will generate all possible metadata that can be derived from the Blackboard LMS context of a learning object. To retrieve this information we could make use of:

- The file system Blackboard uses internally. For example: all documents of course "XYZ" reside in a directory "XYZ"

- The database used internally by Blackboard to manage all the data.

- The Blackboard API. Blackboard offers a Java API on top of the database and file system. This API allows for example to retrieve all announcements, all staff information, all course documents…

As a first step, we looked at all 18 mandatory fields in the Ariadne LOM application profile. We investigated how many of these elements could be determined from the blackboard context. We found out that 16 out of 18 fields can be established by using only the ToledoBBContextBasedIndexer, without using any other part of our indexing framework. However, it is not because they *can* be determined by the ToledoBBContextBasedIndexer that it should be done there. For example, the file size *can* be determined in this context; but the question is whether it *should* be done there.

Referring to section 3.2 about the conflicts between automatic indexers, we can take 2 approaches:

1. We only generate the metadata that is characteristic to the Blackboard context. So we will not put the file size in the

metadata instance that is returned by it. This would provide us 9 metadata fields including the title and the fields about the classification.

2. We generate everything we can. It is the task of the object that combines the different metadata instances to decide on what is the most appropriate value.

When we apply our complete indexing framework on a Blackboard course document, we can generate metadata for the larger part of the mandatory fields, namely for **17 out of 18 fields**.

The remaining issues for the generated metadata are:

o The pedagogical duration is difficult to determine. A possible heuristic is to consider it as a function of the number of pages. For example, a MS PowerPoint document containing 30 slides can be considered to have a duration of 60 minutes. This is however not a robust approach. One person can take 2 hours for 30 slides while another person can do it in 20 minutes. To solve this issue, we can derive it from a context like the electronic learning environment. If that environment supports the notion of a "lesson", we can deduce the pedagogical duration from the context. For instance, if a lesson takes 2 hours and contains 2 documents, we could take a pedagogical duration of 1 hour for each of them. Note that we can also decide not to include the pedagogical duration. This implies an update to the Ariadne LOM application profile, which currently makes pedagogical duration mandatory. The original idea behind this was that data about the pedagogical duration is required in order to be able to automate the authoring of courses. For instance, suppose an instructor combines several learning objects when preparing a lesson of 2 hours. Then it would make sense to put an implicit constraint that the sum of the pedagogical durations of all documents for that lesson should not exceed the lesson duration of e.g. 2 hours[4]. Although this idea is indeed interesting, we think that at this moment the support by tools, to take advantage of this metadata item, is not present enough. So at this moment, it would not be too much of a problem if we would omit it.

---

[3] Toledo is in fact the name of the Blackboard configuration that is used at K.U.Leuven university.

[4] Of course this is not always the case, e.g. if the documents include some study at home. But the example is just illustrative.

o   The authors: When a user is logged in and uploads a document, it is known who does this. So during the document upload, we have the information about the author[5]. However, in our case, we were processing the already existing documents at a later moment. And unfortunately, nowhere in the Blackboard system is maintained who has uploaded the document.
We have solved this issue by taking the instructors of the course as authors of the document. For courses with only 1 instructor, this is probably correct in the majority of the cases. Even when a teaching assistant uploads a file, it is maybe not a problem to take the instructor as author.

o   The accuracy for fields concerning top-level classification (ScienceType, MainDiscipline and SubDiscipline) depends on the way course numbers or identifiers are chosen. For Toledo the classification is determined pretty well in a considerable part of documents. For example, in Toledo all courses in the field of Philosophy start with a "W", so for all documents, used in that course, we can take "Human and Social Sciences" as ScienceType, "Human Sciences" as MainDiscipline and "Philosophy" as SubDiscipline. However, this highly depends on the particular Blackboard configuration. A possible extension and solution for this is to take into account the instructor information: for example instructors who work at the department of Computer Science produce documents about computer science.

o   The lowest level classification element (MainConcept) more or less represents keywords for the document. Here we have several options among which the label of the file within Blackboard or the directory name in which the file resides. E.g. for the course "Multimedia", one of the topics is "XML". The course documents about that topic could be structured in folders like "Multimedia → XML → Introduction". Then it would be a reasonable approach to take "Multimedia", "XML", and "Introduction" as main concepts, because they are all relevant keywords. The assumption then is that instructors make folders with relevant names. A possible extension in the future would be to use other techniques like AI approaches for automatic keyword extraction. [27]

o   To determine the language of documents, we use a java library. This seems to do a good job if there is enough text to process. So, to determine the documentLanguage field, it seems to be a viable solution. For now, we are also using it to determine the language of e.g. the MainConcept. The only problem is that this consists of only one or a few words, which makes the job of determining the language a lot harder. Because in the particular case study of Toledo, we know that the courses are in English or Dutch, we have limited the possible languages to "en" and "nl". But even then, it is often not correct.

6.3   Evaluating the results

6.3.1   What and how:

To evaluate the framework for Toledo Blackboard documents, we took some of the documents of a Toledo course and compared manually generated metadata with automatically generated metadata

for those documents. For the comparison to be valid, we only considered manually generated metadata that already existed. That way, we tried to avoid biases, like a metadata creator paying more attention to his task because he knows that his metadata will be evaluated.

To get such comparable data, we took all documents of a Toledo course about programming (in Dutch: "Beginselen van Programmeren"), and then searched our Ariadne KPS for documents with the same name. This gave us several results, because some of the documents were inserted both in Ariadne and in Toledo. For these documents we could retrieve the (manually created) metadata from the Ariadne KPS. We then used our AMG framework, which gave us automatically generated metadata for it.

To compare the generated metadata we are using the tool XMLUnit[6]. It allows us to compare two pieces of XML in an automatic way, taking into account the specific properties of XML. The result for 1 particular document is summarized into a more human-readable table (Table 2).

Metadata field	Value, automatically generated by AMG framework	Value, manually generated with the Ariadne-SILO tool [7]
Manual and automatic value differ, but this difference is normal, understandable and/or not meaningful or not important		
documentType	expositive	Expositive
packageSize	83.9	84
publicationDate	02-02-2004	28/10/2003
creationDate	29-10-2004	28/10/2003
operating_system_type	Multi-OS	MS-Windows
accessRight	private	Restricted
author/postalCode	B-3001	3001
author/affiliation	K.U.Leuven	KULeuven
author/city	Heverlee	Leuven
author/tel	+32 16 327538	/
author/department	Afdeling Informatica	Computerwetenschappen
author/email	Henk.Olivie@cs. kuleuven.ac.be	henk.olivie@cs. kuleuven.ac.be
phdAnnotations/@type	multiValued	string
Manual and automatic generated values differ… to be investigated whether it matters for the "quality"		
mainDiscipline	Civil Engineering / Architecture	Informatics / Information Processing
documentLanguage	nl	en
documentFormat	Narrative Text	Slides
title	Werken met Java	Praktisch werken met Java
title/@lang	nl	en

Table 2: Comparing manually and automatically generated metadata for a document in a Toledo Blackboard course

---

[5] Assuming the document is uploaded by the author. This is not always the case.

[6] XMLUnit, http://xmlunit.sourceforge.net

[7] Actually the SILO tool also does some things automatically, like: when you are logged in with a certain user profile, it can capture things like the first en last name, which are used as name for e.g. the author.

6.3.2   Results

We split up the table into two parts. The first part contains the metadata fields for which the difference is understandable or not important. For example "expositive" and "Expositive" were regarded as being different by the tool, but the only difference is the capitalization. Another difference is the creation date and publication date. However this also is normal because of our way of working, i.e.: the same document was at some moment inserted in the Ariadne KPS (therefore publication and creation date 28 October 2003) and at some time in Toledo (so publication date on 2 February 2004). The metadata creation date for the Toledo version was the time we used our framework for the document. Regarding the operating_system_type, the framework sets Multi-OS as a default. In the future we will change this to be more flexible and to allow that certain types are not Multi-OS. However, in this particular case, this value is even more correct than the manual one, because a pdf is not Windows-specific. Then, concerning the data about the author, the automatic values are also more correct, as the framework contacts the K.U.Leuven LDAP-server to retrieve the information, whereas the manual data is of course manual, and liable to errors.

The second part of the table contains more pertinent differences. The most important one is maybe the mainDiscipline. As we mentioned in section 6.2, the automatic value is not always as accurate as we would want. Concerning the document language, the chosen course is a bit particular as it is a course about programming. As a consequence it consists of pieces of explanation (in Dutch) and pieces of Java code. Therefore both are in a way correct, although you can argue that the explanation is most important to determine the language, and therefore the automatic value is more correct. But we must admit that this is certainly not always the case. For some of the documents of the same course, the automatic value was "fy" and thus less accurate than the manual one. The title also differs, but in this case not in a very meaningful way. Moreover, the language of the title-field differs, and in this particular case, the automatic value is the correct one. The documentFormat is a tricky issue. The document is a pdf, but actually it is the conversion of a PowerPoint to PDF.

6.3.3   Analysis/conclusions

Sometimes the manual value is better, sometimes the automatic value is better. But from this first, of course limited evaluation exercise, our framework seems to do pretty well, and it can certainly compete with the manually generated values. With of course the big difference that our framework does not require manual input, which ensures the scalability!

7. THE OPENCOURSEWARE CASE STUDY

Another case study is a framework component for MIT OCW [31] courses, which is a free and open educational resource for faculty, students, and self-learners around the world. In this particular case, everything that could be taken as input for our framework is online, in particular as a course homepage or a course document. For example, the source code of such a homepage contains visible information and invisible, meta-information. Fortunately for our framework, the information online is kept in a consistent way, so that we could develop a OCWContextBasedIndexer that analyzes the online pages and derives metadata from them. And of course, because that OCWContextBasedIndexer is plugged in the rest of the

framework, we could benefit from things like the LanguageIndexer to determine the language of documents!

So, this case study illustrates very well the extensibility of the framework, namely: the more the framework gets extended, the more also the indexing of the OCW courses will get better. For example, once we have extended our WordIndexer with methods to get the Microsoft Word specific metadata, or once we have added extra methods like for keyword extraction, this will also be of benefit to the OCW course indexing.

8. CONCLUSIONS AND FURTHER WORK

In this paper, we presented research about automatic metadata generation for learning objects. We mentioned several issues that have to be tackled and presented solutions for all of them. In general, it is our opinion that metadata can be generated automatically without a great loss of accuracy and with a lot of benefits for the users – both content creators and content users. We also implemented and tested a framework.

We are currently developing extensions for the framework to handle automatic metadata generation for specific cases. Comparable to the Toledo and OCW implementation we described above, we are now looking at, for example, proceedings of the AACE conferences from which we can use the papers published in the proceedings as learning objects. From the proceedings we can extract interesting information about those papers and generate metadata for them. Another idea is to investigate citation sites like citeseer or DBLP[8] from which a lot of information about research papers can be found.

As a second extension we are doing research on combing the automatic indexing service with the options described in [5] about composite learning objects. Composite learning objects contain a lot of information about their components and components provide information about their siblings. This is closely related to the research going on about content packaging and ontologies in learning objects [28].

Because we do not want to be limited to LOM, we will also look at ways to generate metadata for other metadata schemas like Dublin Core.

All of these efforts will be made available on the accompanying web site we made [30]. There you will find documentation, news about our work, and links to, for example, the SourceForge location of our project.

9. REFERENCES

[1] Advanced Distributed Learning, available online http://www.adlnet.org.

[2] ARIADNE, available online http://www.ariadne-eu.org.

[3] B. Doan, W. Kekhia and Y. Bourda, *A semi-automatic tool for the indexation of learning objects*, in Proceedings of ED-MEDIA World Conference on Educational Multimedia, Hypermedia and Telecommunications, pages 190-191, 2002.

[4] K. Cardinaels, E. Duval and H. Olivié, *Issues in Automatic Learning Object Indexation*, in Proceedings of ED-MEDIA

---

[8] Digital Bibliography & Library Project, available at: http://dblp.uni-trier.de/.

World Conference on Educational Multimedia, Hypermedia and Telecommunications, pages 239-240, 2002.

[5] K. Cardinaels and E. Duval, *Composite learning objects: exposing the components*, in Proceedings of the 3rd Annual ARIADNE Conference, pages 7, 2003.

[6] K. Cardinaels, M. Meire and E. Duval, *Simple Indexing Interface, draft specification*, version 0.2, Octobre 2004.

[7] W. B. Cavnar, and J. M. Trenkle, *N-Gram-Based Text Categorization*, Proceedings of Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, UNLV Publications/Reprographics, pp. 161-175, 11-13 April 1994.

[8] Dublin Core Metadata Element Set v1.1. Available at: http://dublincore.org/documents/1999/07/02/dces/

[9] CanCore Application Profile. Available at: http://www.cancore.ca

[10] E. Duval, E. Forte, K. Cardinaels, B. Verhoeven, R. Van Durm, K. Hendrikx, M. Wentland-Forte, N. Ebel, M. Macowicz, K. Warkentyne, and F. Haenni, *The ARIADNE Knowledge Pool System*, Communications of the ACM 44 (5), pp. 73-78, May, 2001.

[11] E. Duval, *Learning technology standardization: making sense of it all*, International Journal on Computer Science and Information Systems, 1(1), pages 33-43, 2004.

[12] E. Duval, and W. Hodgins, *A LOM Research Agenda*, WWW2003 Conference, 20-24 May 2003, Budapest, Hungary.

[13] E. Duval and W. Hodgins, *Making Metadata go away - Hiding everything but the benefits*, presented at DC2004 conference, Shanghai, China, 2004.

[14] J. Greenberg, A. Crystal, W. D. Robertson and E. Leadem, *Iterative Design of Metadata Creation Tools for Resource Authors*, in Sutton, S. Greenberg, J., and Tennis, J. (Eds.), 2003 Dublin Core Conference: Supporting Communities of Discourse and Practice – Metadata Research and Applications. DC-2003: Proceedings of the International DCMI Conference and Workshop. September 28 - October 2, 2003, Seattle, Washington. Syracuse, NY: Information Institute of Syracuse, pp. 49-58.

[15] J. Greenberg, *Metadata Extraction and Harvesting: A Comparison of Two Automatic Metadata Generation Applications*, Journal of Internet Cataloging, 6(4): 59-82, final draft available from http://www.ils.unc.edu/mrc/mgr_index.htm.

[16] R. Heery and M. Patel, *Application profiles: mixing and matching metadata schemas*, Ariadne, issue 25, 2000. Available at: http://www.ariadne.ac.uk/issue25/app-profiles/intro.html

[17] IEEE, *Standard for learning object metadata*, available on http://ltsc.ieee.org/wg12.

[18] Merlot Peer Review, http://www.merlot.org/home/PeerReview.po

[19] J. Najjar, E. Duval, S. Ternier, and F. Neven, *Towards interoperable learning object repositories: the Ariadne experience*, Proceedings of the IADIS International Conference WWW/Internet 2003 (Isaias, P. and Karmakar, N., eds.), vol 1, pp. 219-226, 2003,

[20] J. Najjar, S. Ternier, and E. Duval, *User behavior in learning object repositories: an empirical analysis*, Proceedings of the ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications (Cantoni, L. and McLoughlin, C., eds.), pp. 4373-4379, 2004.

[21] D. Rehak, Good & plenty, googlezon, your grandmother and nike: Challenges for ubiquitous learning & learning technology, in PGL Workshop on E-Learning Objects and Systems, Orlando, Florida, June 3-4, 2004.

[22] D. Rehak and R. Mason, *Keeping the Learning in Learning Objects*. Available at: http://www.lsal.cmu.edu/lsal/expertise/papers/

[23] SingCore Application profile. Available at: http://www.ecc.org.sg/eLearn/MetaData/SingCORE/index.jsp

[24] B. Simon, D. Massart and E. Duval (eds.), *Simple Query Interface Specification*, CEN/ISSS Workshop on Learning Technologies, 2004.

[25] K. Verbert and E. Duval, *Towards a global architecture for learning objects: a comparative analysis of learning object content models*, in Proceedings of the ED-MEDIA 2004 World Conference on Educational Multimedia, Hypermedia and Telecommunications, pages 202-208, AACE, 2004.

[26] P. Vandepitte, L. Van Rentergem, E. Duval, S. Ternier, en F. Neven, *Bridging an LCMS and an LMS: a Blackboard building block for the Ariadne knowledge pool system*, Proceedings of ED-MEDIA 2003 World Conference on Educational Multimedia, Hypermedia, and Telecommunications (Lassner, D. and McNaught, C., eds.), pp. 423-424, 2003.

[27] iVia Open Source Virtual Library System: http://infomine.ucr.edu/iVia/

[28] ALOCoM ontology and component architecture, K. Verbert. Information available at http://ariadne.cs.kuleuven.ac.be/alocom/publications.php

[29] M. Hatala and S. Forth, *System for Computer-aided Metadata Creation*, Proceedings of 12th International Conference of The World Wide Web Consortium (WWW2003), Budapest, May 20-24, 2003. Available at: http://www.sfu.ca/~mhatala/pubs/hatala-forth-www2003.pdf

[30] Homepage of our work on automatic metadata generation: http://ariadne.cs.kuleuven.ac.be/amg/

[31] MIT OpenCourseWare, http://ocw.mit.edu/index.html