Tailoring Training Courses Using XML-based Metadata

Silvia Hollfelder, Rodolfo Stecher, Peter Fankhauser
FhG-IPSI, Darmstadt, Germany
hollfeld|stecher|fankhaus@ipsi.fhg.de

Abstract

A flexible course production environment that builds courses from annotated modular learning fragments needs efficient mechanisms for the generation, storage and retrieval of huge metadata collections, since the selection and composition of learning fragments is based on metadata annotations. In this paper, we describe our metadata model, semantic interoperability issues of metadata annotations, and the system architecture for the management of XML-based metadata as well as learning fragments that we use in the Teachware on Demand project.

Keywords: Metadata Management, XML Storage, Semantic Interoperability, Bottom-Up Course Generation, Computer Based Training (CBT)

1. Introduction

The development of electronic course material suitable for different learners takes much effort and incurs high costs. Furthermore, professional trainers have huge expenses to keep course content up to date. This problem occurs especially in areas where knowledge and skills change rapidly, such as in the IT domain. Thus, we need new approaches to support *(semi-)automatic* course generation in order to keep up with current knowledge and perhaps even to adapt materials to individual user needs. In the project *Teachware on Demand*[1], we develop tools and an infrastructure to support the automatic generation of courses, while also aiming to achieve a high degree of reusability of course content. A popular, promising approach is to dynamically compose courses "on-demand". The idea is to segment existing course material (e.g., slides, text books, animations, videos) into so-called learning fragments. Such learning fragments represent typically self-contained units which are appropriately annotated with metadata. Tailored training courses are requested and generated "on-demand" by assembling single fragments, such as in [Ariadne99], [Caumanns00], [Seeberg99], [Vrabic01]], [L300], [Universal], among others. Using this approach that is similar to modularization in software engineering, we intend to achieve a high degree of *reusable content*, and have learning fragments that can be used in new contexts. This reduces costs for the development of training courses. User adaptability is achieved by means of allowing users to specify queries, and dynamically construct courses. As a result, a course may contain fragments from various sources such as text books, instructional movies, or slides from several knowledge providers. Note that there is a tradeoff between granularity (size) of learning fragments, reusability, and annotation effort. Having small units of learning fragments increases the annotation effort, but implies better reusability since small fragments can be composed more flexible

1.1 Annotation of Learning Fragments

The starting point in the course authoring process are the learning fragments. In our approach, a learning fragment is a self-contained, modular piece of course material. These fragments are annotated according to our metadata schema that provides efficient mechanisms to retrieve fragments with respect to the specific needs of a course [Caumanns01]. Our schema [Caumanns01] is a project-specific extension of the standardized LOM (Learning Object Metadata) [LOM00] schema developed by IEEE, describing the annotation of learning fragments. For our schema, we extended existing LOM-categories (e.g., presentation aspects) and specified additional categories (e.g., IT-specific, Index) to enable an efficient pre-selection of learning fragments. For example, the new category "IT-specific" describes attributes that are needed for training IT-professionals, such as the programming language used for the description of a concept (e.g., language C++ for sorting algorithm). Extensions for other training domains are possible.
To compose courses from a set of learning fragments, an appropriate modeling of conceptual dependencies between fragments is needed. For example, fragment A introduces the concept (topic) "quick sort", and fragment B displays the drawbacks of concept "quick sort". Thus, fragment B can not be understood without having knowledge of fragment A. In our approach, we do not annotate such conceptual fragment-to-fragment relations

[1] Teachware on Demand is funded by the German Ministry of Research and Education (bmb+f).

directly, since this leads to high annotation efforts for a huge amount of fragments, but employ a loose coupling mechanism. Therefore, we introduce so-called concept taxonomies. A concept taxonomy, that is modeled using the BaseWeP approach [Billig00], describes the structure of concepts (hierarchy or network). The category "Index" of our schema extensions specifies conceptual relations between fragments and concepts (topics) to be taught. We distinguish between pre- and postknowledge of concepts referring to a fragment. The pre-knowledge (pre-requisites) concepts specify which concepts have to be known for the understanding of a fragment, and the post-knowledge relations denote gained knowledge by studying a fragment. Note that we can use multiple concept taxonomies. The vocabulary to be used for the conceptual annotations is specified by coupling a fragment to a concept taxonomy by means of an entry in category "Index".
The metadata annotations are encoded in XML [XML99]. This enables efficient access to any attributes of semi-structured metadata by means of XQL-queries. Further, by using XML we are able to handle many optional attributes in our schema. XML also provides mechanisms for data exchange and data integration, thus other metadata schemas can be re-used for our purpose.
As XML-based schema, we used the LOM DTD developed in the Multibook project [Seeberg99] as basis and extended it by our project specific annotations. Hereby we accomplish interoperability with "pure" LOM-annotated fragments. As a consequence, we can use any LOM-annotated learning material from various knowledge sources, which is not possible by using proprietary vocabularies such as the Learning Material Markup Language (LMML) [Süß00].
XML-encoded metadata are stored in the so-called Repository. Since each learning fragment is annotated separately, we get around 80 metadata entries (around 60 entries in LOM and around 20 entries due to our extensions) for one fragment. This results in huge metadata collections. For example, the fragmentation of a text book with 300 pages may lead to around 1000 fragments or more, assuming that we segment each paragraph of that book. In our storage system, a metadata instance (including index) needs around 13 KB storage capacity. We will run our first tests on course generation with around 5000 fragments, but plan a many times larger quantity of fragments for a good training system. Thus, we need efficient ways to store and query metadata. Learning fragments (e.g., text paragraphs) can be encoded in XML and are stored in the Repository. Non XML-encoded fragments (e.g., media data) are stored in external fragment storages.

1.2 Course Compilation and Presentation

Based on the available metadata annotations, on-line training courses are generated semi-automatically by selecting appropriate course fragments and by structuring them into a training course, which is a composition of fragments. The selection and composition is based on a query that specifies concepts to be taught and restrictions (e.g., author, source, date of authoring). For these specified concepts, any *pre-knowledge* concepts are retrieved and appropriate fragments are selected. This means that the *course structure* is build *dynamically* on the corresponding pre-knowledge conditions of fragments to be selected [Caumanns00]. We call this *bottom-up* approach, that stands in contrast to *top-down* course composition, where the course structure (e.g., a table of content) is static and given a priori. Further, didactic, content-specific and company-specific needs can be considered for the selection of fragments. The structuring and selection processes are strongly correlated. In case that no fragments are found that map exactly to the restrictions (e.g., the date of authoring is older than specified in the course request), the algorithm selects fragments that do not map optimally to the specification, but also fit to the course. This proceeding avoids the ad hoc generation of missing fragments for a requested course. As a result, complete training courses can be generated. A course trainer also has the option to refine an automatically composed course.
The complete training course is then presented to the end user who has access to a web portal or to a company's knowledge base. In the following, we describe our system architecture for the metadata management.

2. Metadata Management

2.1 Teachware on Demand System Architecture

The Teachware on Demand architecture consists of Repositories, a Mediator, Concept Storage and Fragment Storage Systems, and Clients (see Figure 1).
The huge amount of metadata we have to deal with indicates the need for efficient storage and query mechanisms. This is achieved using a commercial XML storage technology as part of the Repository, namely P-DOM [PDOM00], that we extended to our application specific needs. The advantages of using P-DOM are the good performance, the ease of extensibility of the metadata schemas, and the possibility of co-existence of old and new schemas (DTDs) in our repositories. P-DOM is a persistent DOM (Document Object Model) [DOM00]. DOM is a platform- and language-neutral interface for XML. It provides a standard set of objects for representing XML documents, a standard model of how these objects can be combined, and a standard interface for accessing and manipulating them. Many of the existing implementations of DOM interface are designed to

work in main memory only. P-DOM is a lightweight persistency solution for the DOM implemented in Java. It employs a self-describing, compact file format to handle XML-data, makes file access fully transparent in order to avoid impedance mismatch (the object oriented nature of DOM and the navigational methods it provides are difficult to map to a data only API of a database or file system). A caching object manager synchronizes persistent images of objects with their counterparts in main memory, which are controlled by the Java garbage collection [Huck99]. P-DOM provides rather low level methods for storing and accessing objects. Thus we have created high level, comfortable functions to handle them.
We built on top of P-DOM some generic P-DOM interaction functions (such as executing a query without the burden implied by low level access) and some specific function for our needs (such as searching the metadata matching specific concepts, taxonomies and restrictions). To store a metadata instance, it is parsed into a DOM object, an identifier is generated, and it is stored in P-DOM. When there is a request (query), the repository retrieves metadata matching the specified concepts and restrictions and sends the results back to the Mediator.

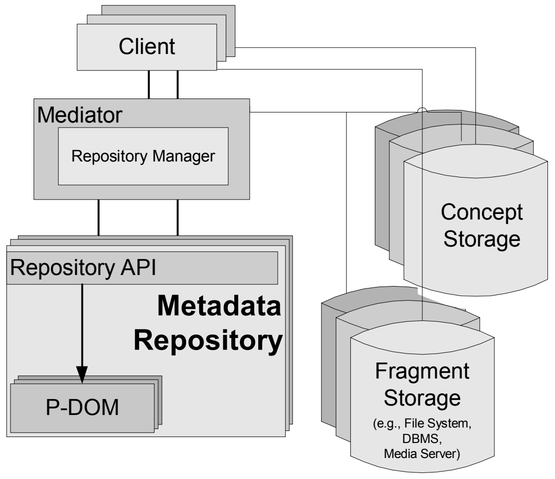

Figure 1: Teachware on Demand System Architecture

The Teachware on Demand Repository consists of P-DOM and a repository interface (Repository API). This API provides services for the management (access, update, insert and delete) of P-DOM objects and XML-documents to the clients and to the Mediator. Each Repository uses a simple version manager service to guarantee data consistency for editing operations.
To enable the coupling with existing, proprietary learning systems, learning fragments (i.e., textual documents, slides, videos) are stored in the Teachware on Demand architecture externally on so-called Fragment Storages (e.g., File System, Media Server, DBMS). The administration of learning fragments is also left to these storage components. As exception, XML-encoded learning fragments can be stored in the Repository.
A Concept Storage contains concept taxonomies referenced in the metadata. These taxonomies are also stored externally to enable a flexible binding of any existing taxonomy (e.g., ACM Computing Classification System, Dutch Basic Classification) for the metadata annotation.
A so-called Mediator provides transparent access for all client requests to all distributed Repositories. A client sends a course request to the Mediator without having knowledge of all available Repositories. Then the Mediator evaluates all Repositories for the requested metadata and sends the results back to the client. Thus, it is responsible to keep track of all available Repositories by means of a Repository Manager that enables to add new ones, to drop existing ones, and to choose a Repository for the storage of a new file on a insert request (if not specified by the client). It uses a parsing service to translate the restrictions sent by the Client into a XQL query. A course request is executed over all available Repositories and the result is returned as a set of metadata to the Client. The Mediator provides also methods to access external Fragment Storage components. This loose coupling implies a consistency problem because the fragment storage is out of the systems control. Thus, a mechanism to keep metadata consistent is needed.
The communication between the Repository, the Mediator and the Client is based on the Simple Object Access Protocol (SOAP) [SOAP]. SOAP is used because it is a lightweight, broadly accepted, XML-based access protocol for interchanging structured and typed information in a decentralized and distributed environment that can be implemented directly over HTTP. The metadata to be sent to the Repository are organized in an IMS package [IMS-CP00].

2.2 Semantic Interoperability

By using XML, only syntactic interoperability is accomplished. Within the project, semantic interoperability needs to be achieved for the following reasons.
There exist a huge set of online learning material sources (e.g., HTML-pages), that are already annotated with various metadata vocabularies, such as Dublin Core (DC) [DC99]. Standardized descriptions of multimedia content in MPEG-7 will probably be available in the near future. As we need a unique data model for the course compilation, these metadata descriptions needs to be transformed to our project-specific scheme. Hereby, we have to solve interoperability for various vocabularies (e.g., Dublin Core to LOM [LOM99]), and have to handle missing metadata entries.
The course compilation algorithm works basically on specified pre-knowledge conditions of fragments. As fragments are annotated independently, cycles in pre-knowledge conditions can occur during the sequencing process of fragments to a course. Another problem is that annotated concepts can refer to various detail levels of a concept hierarchy and thus the concepts are more specific or general (e.g., “sorting algorithm” versus “quick sort”). But for course compilation, several fragments that refer to the same concept have to be identified. Further, interoperability between several concept taxonomies is needed for a flexible usage of all available annotated fragments.
To achieve semantic interoperability, we are currently working on several import mechanisms with careful attention on SCORM (Sharable Content Object Reference Model) [SCO00], a specification for exchange of learning content. In the near future, we will investigate in RDF-based [RDF01] modelling and exchange of concept hierarchies by means of OIL (Ontology Interchange Language).

3. Conclusion and Future Work

In this paper, we have presented a flexible course generation environment to take advantage of re-contextualization and re-utilization of learning materials. Courses are dynamically generated on-demand from fragments’ metadata entries stored in the Repositories. A first Repository prototype is available. It enables client applications to store and query XML-based metadata instances via the SOAP protocol. A further investigation is the (semi)automatic generation of metadata. Our project partners are currently evaluating the usability of our bottom-up based approach.

References

[Ariadne00] Alliance of Remote Instructional Authoring and Distribution Networks for Europe (Ariadne). http://ariadne.unil.ch/

[Billig00] Billig, A., Sandkuhl, K. and Wendt, A.*: XML-Based Content Management in Web-Portals: The BaseWeP Approach to Evolution*. In Proc. Internet and Multimedia Systems and Applications, Las Vegas, Nevada. November 2000.

[Caumanns00] Caumanns, J.: *Bottom-Up Generation of Hypermedia Documents*. Multimedia Tools and Applications. Nr. 2/3, Vol. 12, November 2000, pages 109-128.

[Caumanns01] Caumanns, J., Hollfelder, S. (in German): *Web-basierte Repositories zur Speicherung, Verwaltung und Wiederverwendung multimedialer Lernfragmente*. In Proc. of Information Research & Content Management, DGI Online Tagung, Eds. Ralph Schmidt, Frankfurt a.M., May 2001, pages 130-140.

[DOM00] Wood, L., et al: *Document Object Model (DOM) Level 1 Specification (Second Edition),* Version 1.0. W3C Working Draft, September, 2000. http://www.w3c.org/TR/2000/WD-DOM-Level-1-20000929/

[DC99] *Dublin Core Metadata Element Set*, Version 1.1. http://purl.org/DC/documents/rec-dces-19990702.htm

[Huck99] Huck G., Macherius, I., Fankhauser, P.: *PDOM: Lightweight Persistency Support for the Document Object Model,* OOPSLA Workshop Java and Databases, Persistency Options, Denver, 1999.

[IMS-CP00/1] IMS Global Learning Consortium, Inc.: *IMS Content Packaging Information Model*, Version 1.0. Juni 2000. http://www.imsproject.org/content/packaging/cpinfo10.html

[L300] L3 Project Page. http://www.l-3.de/en/index.html

[LOM00] IEEE Learning Technology Standardization Committee*: Draft Standard for Learning Object Metadata*, Version 4.0. http://ltsc.ieee.org/doc/wg12/LOM_WD4.htm

[LOM99]	IEEE Learning Technology Standards Committee (LTSC), *Draft Standard* for *Learning Object Metadata*, Draft Document, Version 3.6, http://ltsc.ieee.org/doc/wg12/LOM3.6.html. Section 8, Complete Dublin Core Mapping
[OIL00]	*Ontology Interchange Language*. http://www.ontoknowledge.org/oil/
[PDOM00]	GMD-IPSI XQL Engine. Homepage: http://xml.darmstadt.gmd.de/xql/index.html
[RDF01]	*Resource Description Framework*. http://www.w3.org/RDF/
[Seeberg99]	Seeberg, C., Steinacker, A., Reichenberger, K., Fischer, S., Steinmetz, R.: *Individual Tables of Contents in Web-based Learning Systems*. In Proc. 10th ACM Conference on Hypertext and Hypermedia. Darmstadt, February 1999.
[SCO00]	SCORM Sharable Content Object Reference Model. Version 1.0, http://www.adlnet.org/Scorm/docs/readme.htm
[SOAP]	*Simple Object Access Protocol*, SOAP. http://www.w3.org/TR/SOAP/
[Süß00]	*Learning Material Markup Language 1.1*. http://daisy.fmi.uni-passau.de/projects/PaKMaS/LM2L/
[Universal]	*Universal Exchange for Pan-European Higher Education*. Project home page: http://www.ist-universal.org/
[Vrabic01]	Vrabic, G, Simon, B.: *Learning Resource Catalogue Design of the UNIVERSAL Brokerage Platform*, In Proc. of ED-MEDIA, June 2001.
[XML99]	Bray, T. et al*: Extensible Markup Language (XML) 1.0* (Second Edition), W3C Recommendation, October 2000. http://www.w3c.org/TR/2000/REC-xml-20001006