A Practical Strategy for the Modularization of Courseware[*]

Khaldoun Ateyeh Michael Klein Birgitta König-Ries Jutta Mülle
{ateyeh, kleinm, koenig, muelle}@ipd.uni-karlsruhe.de

1 Introduction

In order to enable courseware reuse, it is necessary to decompose learning materials into manageable learning objects. Most scientific projects and organizations that are concerned with courseware reuse and exchange like Ariadne [2], IMS Global Learning Consortium [4], Educause [3], the IEEE Learning Technologies Standards Committee [8] with its Learning Object Metadata (LOM) [7], and the German Competence Center "Learning Lab Lower Saxony" (L3S) [6] recognize the importance of modularization for courseware reuse and exchange, but very few address the problem of how to split (existing) learning materials into learning objects. Authors face the problem of not knowing how to determine suitable reusable learning objects in their content. What is the appropriate size of one such an object? Which characteristics must such an object have in order to be described as reusable? Existing approaches do not offer answers to these questions or at most give a theoretical definition of learning object, which authors cannot concretely apply to their content. A typical example is the often cited definition by WILEY: *A learning object is any digital resource that can be reused to support learning* [13]. It excellently describes what a learning object is, but does not offer any practical instructions by which authors could determine well-sized learning objects in their material. What is missing is an operationalization of the learning object definition offering authors an advice on how to decompose the learning materials into reusable learning objects. A proposal for such a definition is given in this paper. It is based on the idea of describing learning materials along two dimensions: their contents and their resource types. An extended version of this paper is available as technical report [1].

2 Scenario

Before introducing formal definitions in the next section, we will give an overview of the process of exchanging and reusing courseware as well as the different types of learning units, thereby introducing both the different concepts and how they are used.

A group of teachers at neighboring universities, among them Anna and Chuck, has noticed that the topics they teach in their database classes overlap to a certain extent. They decide that it would be more economical to collect their teaching materials into one big pool and to allow all of the teachers to access these materials. The resulting repository should contain materials that exist already, but should also allow for new material to be introduced. In

[*]This work was partially funded by the *ViKar*-Project [12] within the program *Virtual University* by the state of Baden-Württemberg.

order to make the material collected in the pool (re-)usable, it needs to be appropriately structured and described.

Creating and filling the repository. In this step, the teachers act as authors. They agree on a collection of terms that describe their subject, e.g., *database systems, relational algebra, join operator, relational model, normalization, functional dependency*. In the next step, relationships between these terms are established: A term may be more generic than another one, e.g. *relational algebra* is more generic than *join operator*. Certain knowledge, e.g., on *functional dependencies* may be a prerequisite to understand something else, e.g., *normalization*. The result of this step is a *domain specific ontology* of the topic area modeling two types of relationships: The *isSubtopicOf* relation and the *isPrerequisiteFor* relation.

The second task for the authors is to agree on the *resource types* that should be placed in the repository. Resource types describe in what function or for what purpose material can be used. Examples for these types are *introduction, theorem, example, definition*. The result of this step should be a complete list of admissible resource types.

We envision the following logical structure for the repository: For each term in the ontology, there exists a corresponding container in the repository which holds all the materials described by that term. Containers for the upper terms of the ontology, as *relational database systems* also hold ”smaller” containers for the less generic terms, as *relational algebra*, that will be called *modules*. Modules do not only contain other modules, but also pieces of learning material called *atoms* described by exactly one term of the ontology and typically one resource type. Given this logical structure, the next task for the authors is to fill the repository with material. In our example, the authors first have to decompose their existing materials according to the ontology. Consider, e.g., Anna who teaches a class on relational databases. She finds that the most precise description of her entire course is *relational database systems*. Now, she tries to find parts of the course like the chapter described by the term *relational algebra* and an enclosed section on the *join operator*. The latter is a leaf term of the ontology, thus, no more precise term is available. Anna now looks at her material on the join operator and splits it up according to the resource types. This results in learning atoms, e.g. an example for the usage of the join operator, that she can then store in the repository. Material that is not described by a leaf term, but by an inner node of the graph like an exercise requiring different algebraic operators is stored in the appropriate higher-level module.

Instead of relying on existing material, it is also possible to create new material and store it in the repository. Adding material to the repository should not be considered a one time activity, but can and should go on throughout the lifetime of the repository.

Using the repository. Teachers, now act as instructors. Assume, Chuck (in the role of *instructor*) has agreed to teach a new course on SQL. Thus, the system will display the contents of the *SQL* module: atoms containing material relevant to SQL as a whole, but also modules dealing with subtopics, e.g. *projection, selection, embedded SQL*. From these, he chooses the ones he wants to use in his course and the system allows him to pick from the contents of this module: suitable examples, definitions and so on. After having this material, Chuck needs to determine the order in which he wants to present it. Again,

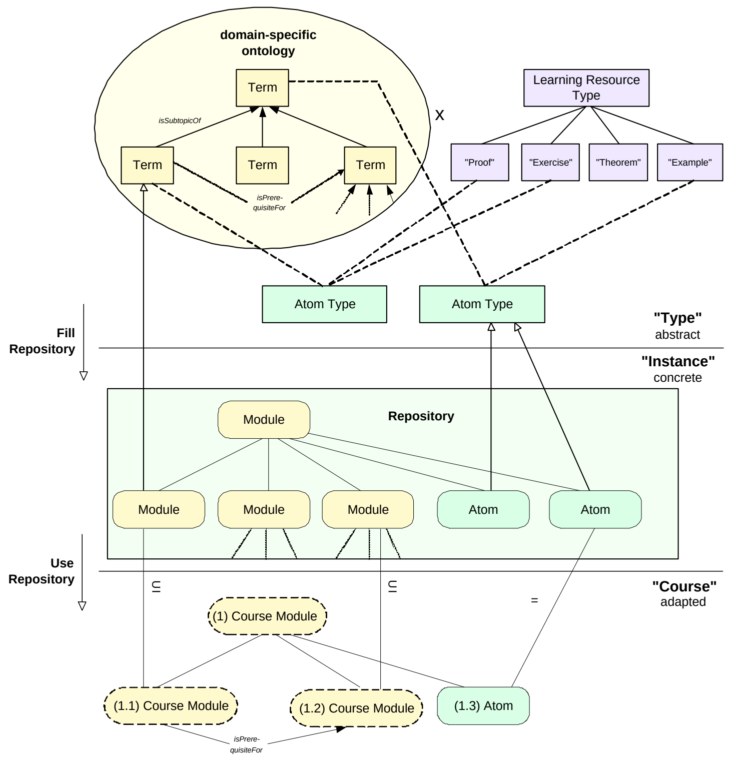

Figure 1: Concepts and relationships of the formal model to organize learning objects

the system helps him by displaying the prerequisites laid down in the ontology. For some subtopics, Chuck may not find appropriate material in the repository. In this case, he will switch back into his role as *author*, develop the appropriate material, and add it to the repository for future use.

3 A Formal Model

This section introduces a formal model defining the basic concepts needed in the process of courseware modularization. Special emphasis lies on how to operationalize the introduced concepts in order to help authors to construct reusable, extensible, and maintainable repositories for their learning objects. An overview of the concepts and their relationships is given in Figure 1.

As described in Section 2, in order to modularize teaching and learning material on a subject, the authors have to agree on a collection of terms that describe the subject and on relationships between these terms [9]:

Definition 3.1 (Domain Specific Ontology) A domain specific ontology *is a directed graph with domain specific terms as nodes. The edges between the terms are typed. There exist two types of edges:*

- isSubtopicOf: *An edge from term t_1 to term t_2 is of type* isSubtopicOf *iff. t_2 deals with a more generic topic than t_1. Edges of this type are unique and mandatory for inner terms, i.e. each term (except for the root of the graph) is subtopic of exactly one other term. Therefore, the terms form a tree when regarding the* isSuptopicOf *edges only.*

- isPrerequisiteFor: *An edge from term t_1 to term t_2 is of type* isPrerequisiteFor, *iff.* t_1 *needs to be understood by a learner before he is able to learn* t_2. *The* isPrequisiteFor *relationship builds an acyclic graph.*

In the following, we will sometimes talk about layers or leaves of the ontology. In these cases, we are always referring to the graph with respect to the *isSubtopicOf* edges.

Learning objects are characterized by their content (described by the terms of the ontology) and by their intended usage. For instance, a certain learning object can be used as introduction, as definition, etc. Therefore, we introduce *learning resource types*:

Definition 3.2 (Learning Resource Type) *A* learning resource type *describes the intended usage of content of learning objects.*

A learning object is classified by one or more learning resource types. This can be "Example", "Definition", "Explanation", "Theorem", "Proof", "Exercise", "Motivation", "Conclusion", "Test/Quiz", "Introduction", "Simulation", or "Scenario" [11][1].

Based on these definitions, we can now introduce the *atom type* as an abstract description of learning objects, more precisely of atoms.

Definition 3.3 (Atom Type) *An* atom type *is a 2-tuple of exactly one term of the domain specific ontology and a set of learning resource types.*

Examples of atom types are (*SQL*, {"Exercise"}), (*3rd normal form*, {"Exercise", "Example"}). The latter atom type may, e.g., be used for learning objects that contain material about the *3rd normal form* and may be used as example and as exercise. Atom types may belong to arbitrary terms on any level of the domain specific ontology.

Until now, the concepts of the upper third of Figure 1 have been defined. In the following, we will introduce the concepts of the second part of the figure, i.e. concrete instances of learning objects and the resulting repository. Let us first introduce the atom, the smallest unit of learning objects that we consider:

Definition 3.4 (Atom) *An* atom *is an instance of an atom type.*

An example is a set of powerpoint slides containing examples of SQL queries, therefore resulting in the type (*SQL*, {"Example"}).

As described in Section 2, the stepwise decomposition of teaching material results in the creation of atoms. Atoms belonging to the same term in the domain specific ontology belong to the same learning object, which we call module. To each term in the ontology there exists exactly one corresponding module, which contains all learning objects related to that term.

Definition 3.5 (Module) *A* module *is a collection of materials belonging to exactly one term in the domain specific ontology. It comprises all learning objects which are related to that term, i.e. atoms and other (sub-)modules.*

Remember that there exists a module for each term in the ontology. This implies that authors do not need to actively define modules and do not need to sort material into modules. The modules are implicitly defined during the creation of the ontology. By instantiating

[1]In the metadata standard LOM [7] the term "learning resource type" is used slightly differently: LOM mixes the content-based types introduced in our definition, with technical types like "figure", "graph", or "table".

an atom from a certain atom type it automatically becomes part of the appropriate module (and implicitly also of the modules this module is part of through the *isSubtopicOf* relationship).

Up to now, we have examined how to decompose material into smaller units, i.e. atoms and modules, and thus, how to fill the repository. The following definition describes how material can be reassembled to form a new course (denoted as *course modules*), and thus how to use the repository:

Definition 3.6 (Course Module) *A* course module *is a special module type which evolves from a module by selecting (sub-)modules and atoms from it according to the specific course context, and bringing them into a total order which is compatible with the partial order of the isPrerequisiteFor relation.*

To keep our scenario simple, we have assumed that a course module always corresponds to one specific term in a common and fixed ontology. In practice, there exist several cases where the adaptability and extensibility of the domain specific ontology is inevitable. We deal with this by allowing the community to jointly change the ontology as well as allowing single authors to set up private ontology views.

4 Conclusions and Future Work

In this paper, we have presented an approach that practically guides authors through the difficult process of dividing existing learning material into reusable learning modules. The approach is based on formal definitions for learning atoms and modules, which on the one hand distinguish between abstract types and concrete instances, and on the other hand differentiate between domain specific terms and domain independent learning resource types. An example on how this approach can be applied in the praxis is described in [5]. In the meanwhile we are implementing a system called SCORE "System for Courseware Reuse" [10] that makes use of this strategy for the modularization of courseware.

References

[1] Ateyeh, K.; Klein, M.; König-Ries, B.; Mülle, J.: A Strategy for the Modularization of Courseware. Techn. Report No. 2003-3, Dept. of Informatics, Universität Karlsruhe, 2003.

[2] Alliance of Remote Instructional Authoring and Distributing Network for Europe (ARIADNE). http://ariadne.unil.ch

[3] Educause. http://www.educause.edu

[4] IMS Global Learning Consortium. http://www.imsproject.org

[5] Klein, M.; Ateyeh, K., König-Ries, B.; Mülle, J.: Creating, Filling, and Using a Repository of Reusable Learning Objects for Database Courses. BTW-Workshop Datenbanken und E-Learning, Leipzig, Februar 2003.

[6] Learning Lab Lower Saxony (L3S). http://www.learninglab.de

[7] LOM working draft v5. http://ltsc.ieee.org/doc/wg12/LOM_WD5.doc

[8] IEEE Learning Technology Standards Committee (LTSC). http://grouper.ieee.org/groups/ltsc/

[9] What is an Ontology? http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[10] SCORE project. http://www.ipd.uka.de/SCORE/en/index.html

[11] Metadata catalog of the SCORE project. http://www.ipd.uka.de/SCORE/xsd/score_v1.xsd

[12] Virtueller Hochschulverbund Karlsruhe (ViKar). http://www.vikar.de

[13] Wiley, D.A.: Learning objects need instructional design theory. In A. Rossett (Ed.) *The 2001/2002 ASTD Distance Learning Yearbook*. McGraw-Hill, New York. 2002.