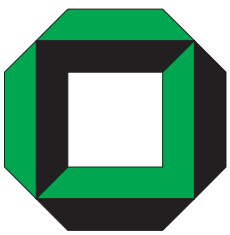
A Strategy for the Modularization of Courseware[1]

Khaldoun Ateyeh Michael Klein Birgitta König-Ries
Jutta Mülle

{ateyeh | kleinm | koenig | muelle}@ipd.uni-karlsruhe.de

December 23, 2003

Technical Report Nr. 2003-3



University of Karlsruhe
Faculty of Informatics
Institute for Program Structures and Data Organization
D-76128 Karlsruhe, Germany

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

Abstract

In order to enable courseware reuse, learning platforms nowadays require the materials to be decomposed into small independent learning units. When trying to fulfill this need, authors face the problem of not knowing how to determine suitable learning objects in their content. What is the appropriate size of one such object? The rather general and abstract definitions for learning objects found in the literature are not very helpful for answering this question. What authors need is an operational definition, which can be directly applied to the learning materials. This paper proposes such a set of formal yet practical definitions by describing learning objects along their contents and resource type and shows how these definitions are used by our platform, SCORE.

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 reusable learning objects. Thus, Authors face the problem of not knowing how to determine suitable reusable learning objects in their content. Most of the time they are confronted with the following questions: 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 or guidance by which authors could determine well-sized reusable learning objects in their material. The definition used in LOM (Learning Objects Metadata) suffers from the same problem:

> *Learning Objects are defined as any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning. [7]*

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. We also show how these definitions are used by our system for courseware reuse (SCORE) to support authors in finding appropriate learning objects and to support instructors in putting courses together.

2 Scenario

Before introducing formal definitions in the next section, we will give in this section an overview of the process of exchanging and reusing courseware. Thereby we will introduce different types of learning units and will show 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 order to make the material collected in the pool (re-)usable, it needs to be appropriately structured and described. In the following we will describe two phases: The first one is the creation and filling of the repository. In this phase, our teachers act as *authors*. In a second phase, the repository is used to develop and teach courses. In this phase, the teachers act as *instructors*. The phases do not occur in sequential order, but can interleave and overlap.

Creating and filling the repository. In this step, the teachers act as authors. They agree on a collection of terms that describe their subject. In our example, some of these terms could be *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*. In our example, knowledge about *functional dependencies* is needed before tackling *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 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*. In our example, there will be a container called *database systems* containing among others another container *relational database systems* which in turn holds materials about the *relational algebra* and so on. In the following, these containers 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. An atom is a piece of learning material that is described by exactly one term of the ontology and one (or in some special cases, see below, several) resource types. Each part of learning material is associated with the one term in the ontology that describes it best. For instance, an introduction to the relational algebra or an exercise requiring the usage of several algebraic operators would be included in the *relational algebra* module, a definition of the join would be part of the *join operator* module. 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. There will be material that is not described by a leaf term, but by an inner node of the graph like an exercise requiring different algebraic operators. In this case, this material is stored in the appropriate higher-level module. Also, it may happen, that materials need to be described by more than one resource type. E.g., an explanation and an example may be so interleaved that it is not possible to divide the material up.

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 ma-

terial 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, 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.

The following sections will deal in more detail with the material presented here: Section 3 contains formal definitions of all concepts used while section 4 introduces our implementation.

3 A Formal Model

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

An overview of the concepts and their relationships is given in Figure 1 and Figure 2. While Figure 1 describes the process of building a repository, Figure 2 describes the process of using it.

3.1 Defining and Filling the Repository

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.*

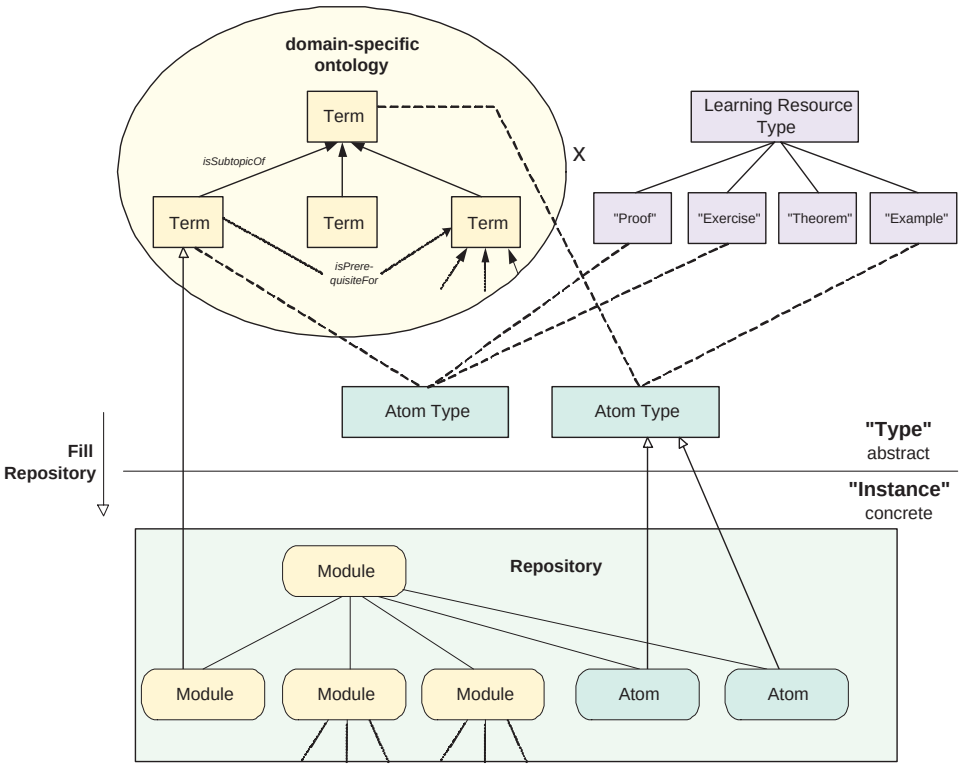


Figure 1: Concepts and relationships of the formal model to organize learning objects

- 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 the concept of *learning resource types*:

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

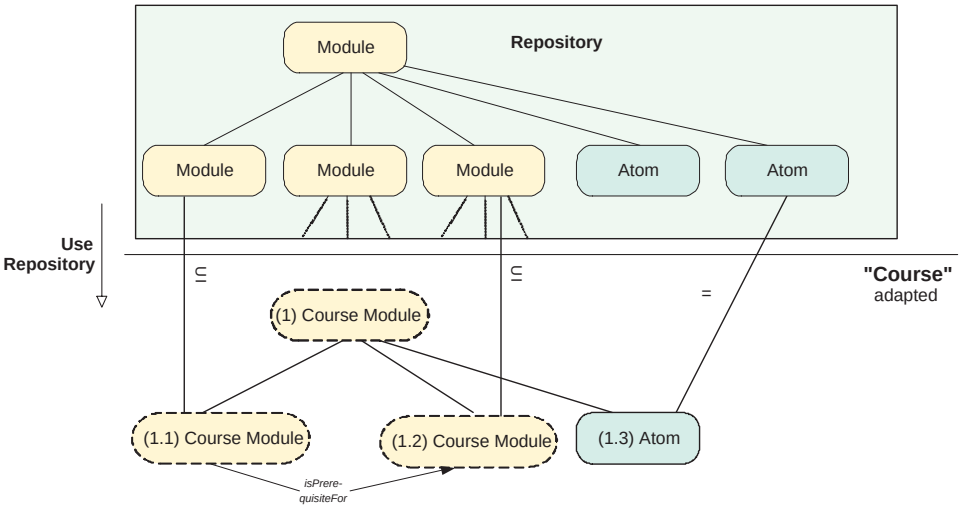


Figure 2: Concepts and relationships of the formal model to organize learning objects

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"}), with the terms *SQL* and *3rd normal form* and the learning resource types "Exercise" and "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 part of Figure 1 have been defined. In the following, we will introduce the concepts of the second part of Figure 1, i.e. concrete instances of learning objects and the resulting repository.

[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".

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"}). Another example is a Postscript file with an exercise about the 3rd normal form, that may also be used as example, so it is of type (*3rd normal form*, {"Exercise", "Example"}).

As described in Section 2, the stepwise decomposition of teaching material results in the creation of atoms. Starting with the material for an entire course, in a first step, an author will divide it into ever smaller parts described by exactly one term of the ontology. For as long as it seems practical, he then chooses parts of parts that are described by a more specific term. In a second step, the author will divide each part along the learning resource types into atoms. The general goal one should observe is to assign each resulting part the most specific term applicable and the smallest possible set of learning resource types. Atoms that belong 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 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).

3.2 Using the Repository

So far, we analyzed the process of defining and filling a reusable courseware repository and we defined the concepts needed to allow conducting them effectively. In this section we will define another concept needed in the process of using the repository to build context specific courses. For every term

in the ontology there exist one module in the repository. The module contains all types of material from all participants that can be associated with the correspondent term in the ontology. By all means, it is not conceivable that these kind of modules can be directly re(used). Considering our scenario from section 2 we have shown that in order to be able to (re)use the learning modules in the repository, instructors have to be able to adapt them to their own needs such as learning form, target group and pedagogical strategy. This reflection leads to the concept of the Course Module.

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.*

Thus, a course module covers a specific view of an author on a learning module according to its use in a specific context. Atoms, modules and course modules are stored in a repository:

Definition 3.7 (Repository) *A* repository *is a collection of modules and atoms.*

Each learning object in a repository belongs to one or more subjects and for each exists a domain specific ontology on which the author group or the community has agreed. Moreover, each term of the ontology there exists a related module in the repository. A repository may also contain modules related to different domain specific ontologies.

3.3 Extending the Repository

To keep our scenario simple, we have assumed that a course module always corresponds to one specific term in a common and fixed ontology. However, in practice authors or instructors when build new courses, they often need to add new terms. The reason why these terms are not found in the ontology is: These terms are context specific and when defining a common domain specific ontology, generic terms are normally chosen so that the ontology can serve a wide variety of interests and situations. Or, these terms represent new development in the domain area and they are not yet integrated in the ontology. One way to deal with this situation is to allow course specific views on the ontology. This means, for every course there exist a course view on the ontology. A course view can consist of a mixture of ontology

terms and course specific terms. Modules cover course specific terms will be treated as course modules. Views provide an effective instrument to deal with the dynamic aspects of the domain specific ontology such as extensibility, adaptability and flexibility. It also allows to define different levels of access right to the learning materials. This way, authors and instructors can choose whether they want to share their specific modules with others or not.

4 Metadata

In the SCORE of courseware reuse metadata are used to describe the different aspects of a learning object such as technical, pedagogical and legal aspects. This will enhance the access and reuse of courseware by allowing effective and intelligent search mechanisms. In this section we will briefly describe the SCORE metadata standard. SCORE metadata standard is based on the LOM metadata standard. It defines a subset of the LOM Standard and it extends it on SCORE specific elements.

When defining the SCORE metadata standard we paid attention to keep the metadata set as small as possible. We believe that this will motivate authors and content provider to mark their content with metadata.

The LOM standard is divided into 9 sections. SCORE uses the sections 1,2,4,7,8 and 9. In the following we will describe each of the section used in SCORE. For a complete description of the meta data see [11].

1. general (section 1): this section contains general characteristics of the learning object such as the title, the language and a general description of the learning object. The information in this section is used to provide a simple search functionality in the SCORE system.

2. life cycle (section 2): this section is very important for the management of the learning object especially in cooperative development environment such SCORE. It contains information about the life cycle of the learning object such as version status and the persons contribute to the learning object.

3. technical (section 4): this section capture all technical requirements and characteristics of the learning object such as format and size. This kind of information is very important in the process of choosing learning objects that fit technically together and that suitable for the technical environment of the user

4. educational (section 5): this section describes a suitable educational context where the learning object can effectively be used. Examples for educational characteristics are target group and interactivity type.

5. relation (section 7): this section covers the structural relations of the learning object to other learning objects. To define multiple relationships, there may be multiple instances of this category. If there is more than one target learning object, then each target shall have a new relationship instance. Examples for structural relationships are: is bart of, is based on and requires. This information is very valuable in a (semi)automatic generation process of course modules

6. annotation (section 8): this section is used to store user remarks on the learning object. Within SCORE, these remarks can be added only form authors and instructors to share their assessments of learning objects, suggestions for use, etc.

7. classification (section 9): this section captures the relationship of a learning object to the domain specific ontology.

5 Implementation

In this section we will briefly describe the SCORE [10] platform "System for Courseware Reuse". SCORE implements the modularization concept described in Section 3 and supports the scenario for courseware production and (re)use described in Section 2. Therefore, the SCORE platform provides tools to support the processes of defining, filling and (re)using a shared courseware repository. Figure 3 gives an overview of the system architecture of SCORE, Figure 4 shows a screen shot of the system. As you can see, SCORE is a complete learning platform with components for authoring, learning, and teaching. SCORE system architecture is a typical three tier architecture. Layer 1 is the data management layer. It manages learning object repositories as well as all SCORE users. It also manages learning and or teaching process information such as information produced in the learning process and the different communication and cooperation facilities such as user annotations, chats, forums and instructor virtual meetings rooms. In order to allow a transparent and database technology independent access to the data, score offers an EJB interface to the database. Layer 2 provides common services that can be shared from the different user environments. Finally, Layer 3 offers user specific environments. Altogether score distinguishes between three main roles: student, instructor and author and thus

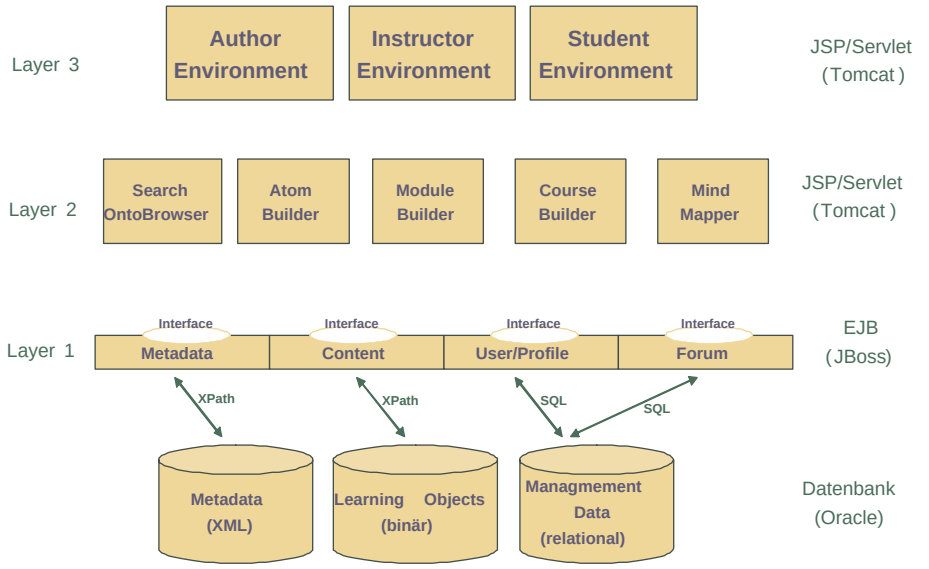


Figure 3: Layered system architecture of our learning platform SCORE.

provides three different environments for student, author and instructor. In the following, we will discuss some components of SCORE that are relevant for conducting of the tasks described in this paper from an author's and instructor's perspective.

Defining a shared repository: As mentioned in Section 2, in a first step, cooperating authors agree on a collection of terms that describe their subject or domain. In a second step, they set relationships between these terms and build a domain ontology. The SCORE OntoBrowser tool supports the authors by conducting these steps. OntoBrowser provides a graph drawing tool helping authors to easily define their ontology. OntoBrowser will then generate an RDF based ontology file. The second classification schema is the resource type classification schema. This sort of classification is part of the SCORE metadata standard[2]. The SCORE metadata standard [11] is a subset of LOM guaranteeing the possibility of courseware exchange with other learning systems.

[2]When defining the SCORE metadata standard we paid attention to keep the metadata set as small as possible. We believe that this will motivate authors and content provider to mark the content with metadata

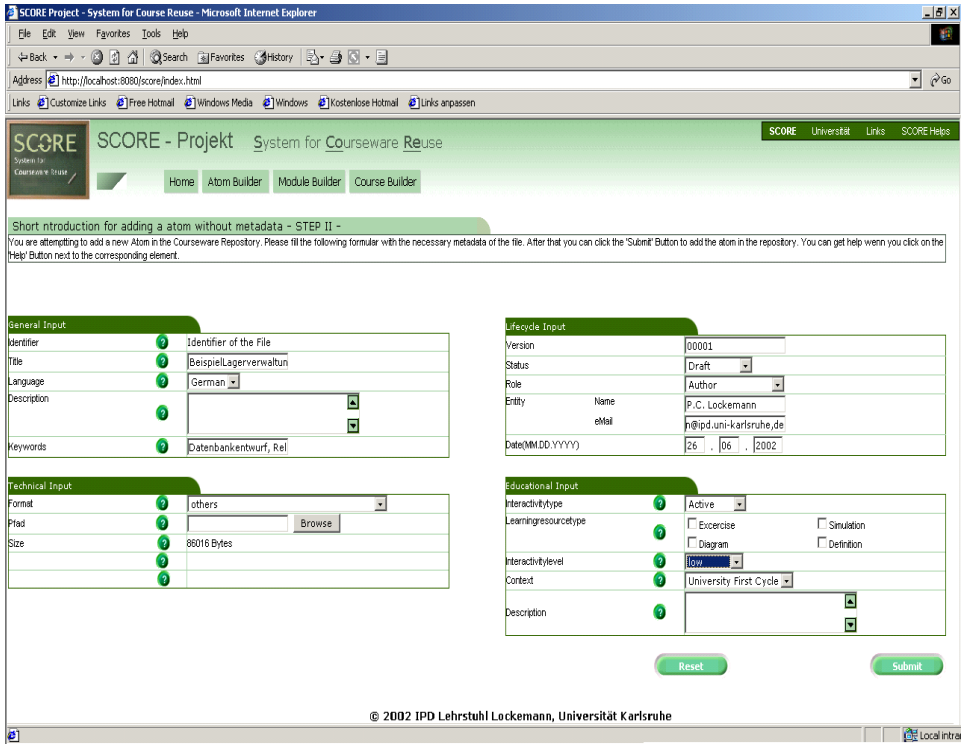


Figure 4: Screenshot of SCORE's AtomBuilder.

Filling the repository: To fill the repository with content, the SCORE AtomBuilder tool is used. The AtomBuilder allows authors to mark their learning atoms with SCORE metadata, match them to the right term and to insert them in the SCORE repository. The process of marking learning atoms is very fatiguing and time consuming, therefore the AtomBuilder has been designed to minimize the amount of metadata to be filled in manually by the author. Instead, the AtomBuilder fills in most of the metadata about the learning atom such as technical and author information automatically.

Using the repository: SCORE provides different tools that facilitate the reusability of the courseware stored in the SCORE repository. To explain how SCORE supports instructors to combine their courses out of existing learning modules let us go back to our scenario in Section 2 where Chuck decided to build a course on SQL. To achieve this he could have used the SCORE OntoBrowser to navigate through the ontology in order to get a picture of the terms and their relationship as well as the materials existing for each term. After that, he might have used the CourseBuilder to specify

his course by selecting the most appropriate terms and if needed adding new ones. With the help of the ModuleBuilder, he could process the modules, that is select, add, and arrange the learning atoms and sub modules to meet his pedagogical strategy and target group.

6 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. Furthermore, we have briefly described how the shown concepts are supported form our learning platform SCORE. An example on how this approach can be applied in the praxis is described in [5]. In the meanwhile we are implementing the SCORE platform.

Bibliography

[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.