Information Sciences xxx (2009) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Management of evolving semantic grid metadata within a collaborative platform

Michael Hartung[a,*], Frank Loebe[b], Heinrich Herre[c], Erhard Rahm[b]

[a] Interdisciplinary Center for Bioinformatics, University of Leipzig, Härtelstraße 16–18, 04107 Leipzig, Germany
[b] Department of Computer Science, University of Leipzig, P.O. Box 100920, 04009 Leipzig, Germany
[c] Institute of Medical Informatics, Statistics and Epidemiology, University of Leipzig, Härtelstraße 16–18, 04107 Leipzig, Germany

ARTICLE INFO

Article history:
Received 1 December 2008
Received in revised form 24 April 2009
Accepted 1 August 2009
Available online xxxx

Keywords:
Semantic metadata
Grids
Collaborative environments
Schema evolution
Data migration

ABSTRACT

Grid environments, providing distributed infrastructures, computing resources and data storage, usually show a high degree of heterogeneity and change in their metadata. We propose a platform for collaborative management and maintenance of common metadata for grids. As the conceptual foundation of this platform, a meta model is presented which distinguishes structured descriptions and classification structures that both are modifiable. On this basis, the system allows for the creation and editing of grid relevant metadata and provides various search and navigation facilities for grid participants. We applied the platform to the German D-Grid initiative by establishing the D-Grid Ontology (DGO).

1. Introduction

Grid computing offers scientists a distributed infrastructure for collaboration and provides massive amounts of computing, storage, and data resources. Grid initiatives, e.g., the German D-Grid,[1] are highly complex and involve many heterogeneous components. They offer resources of different types (e.g., hardware or software resources) which are distributed among many participating organizations. These include universities, research centers and enterprises, which themselves have affiliated persons or take part in different grid projects representing individual communities such as medicine or physics.

Metadata at varying levels of detail is needed to describe these grid resources as well as the participating organizations, projects, and persons. Frequently, grid metadata is managed independently in each participating project, i.e., each particular project is responsible for its own specific metadata. This may be appropriate for the management of project-specific or domain-specific metadata, for example, biomedical grid projects typically use life science ontologies for data annotation. On the other hand, there are common types of metadata which apply to all grid projects. Information about projects, grid resources and organizations can be managed in an integrated form, and should be accessible online and directly editable for all authorized participating persons and projects. Furthermore, metadata especially about resources should be offered to grid applications and services, e.g., through metadata service interfaces. Providing integrated access to grid metadata permits projects to better exchange information about their ongoing work. For example, grid participants can more easily notice related work in other projects, so that cooperation can be improved and duplicate efforts be reduced.

It is important that a metadata management system offers simple user interfaces for the extension and change of the metadata (usability aspect), since persons of different domains with diverse technical backgrounds (e.g., computer scientists,

* Corresponding author.
E-mail addresses: hartung@izbi.uni-leipzig.de (M. Hartung), loebe@informatik.uni-leipzig.de (F. Loebe), hherre@imise.uni-leipzig.de (H. Herre), rahm@informatik.uni-leipzig.de (E. Rahm).
[1] http://www.d-grid.de.

0020-0255/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2009.08.008

physicians, or librarians) meet in a grid's virtual organization [13]. In addition, providing metadata management over a long-term period requires the ability to handle changes to the data model of the metadata (evolution aspect), since grid applications are highly evolving and hence generate new or changed requirements on their metadata. For instance, if new metadata about special hardware resources or industry organizations should be captured, the current (active) data model needs to be adapted in order to incorporate these new requirements. Furthermore, initial design errors on the data model may lead to ineffectiveness and should therefore be corrected. Hence, a metadata management system requires facilities that support the change of a running data model considering the captured metadata.

We make the following contributions in this paper:

- We propose a simple yet flexible meta model suitable for managing semantic grid metadata. A foundational analysis of existing systems and their usage leads to a novel explicit distinction between content types for structured information and ontological categorization for content classification in the meta model.
- We address the evolution of the data model on the basis of meta model elements. In particular, we discuss relevant evolution operations and their data migration effects.
- We describe a web-based and wiki-like platform using the defined meta model. The platform supports the collaborative creation and editing of grid metadata as well as the evolution of the employed data model. It further addresses usability issues such as powerful search, navigation and visualization capabilities.
- We present an application of our platform, namely the D-Grid Ontology (DGO) system of the German D-Grid initiative available under http://buell.izbi.uni-leipzig.de/dgo. In particular, we outline the current organization of the semantic metadata.

The remainder of the paper is organized as follows: in Section 2 we describe models for the collaborative management of grid metadata, with a focus on the meta model level. Evolution at the data model level is discussed in Section 3. Section 4 presents the data model of DGO, while specific features of the platform concerning usability are illustrated in Section 5. Implementation details are provided in Section 6, related work in Section 7. We conclude with a summary and an outlook on future work.

2. Models of the platform

We build on a three-layered representation for the management of metadata (see Fig. 1) differentiating between the following layers: *meta model*, *data model* and *instance data*. The data model (or schema) is specific to a particular grid or virtual organization, e.g., D-Grid, and prescribes the structure of possible instances and their semantic annotations. The meta model defines the constructs which can be used for defining the data model, in particular for describing the structure of instances (content) and the use of ontologies for semantic annotation of instances. In this section we describe the meta model, whereas Section 4 focuses on the D-Grid Ontology (DGO) with its data model and instances.

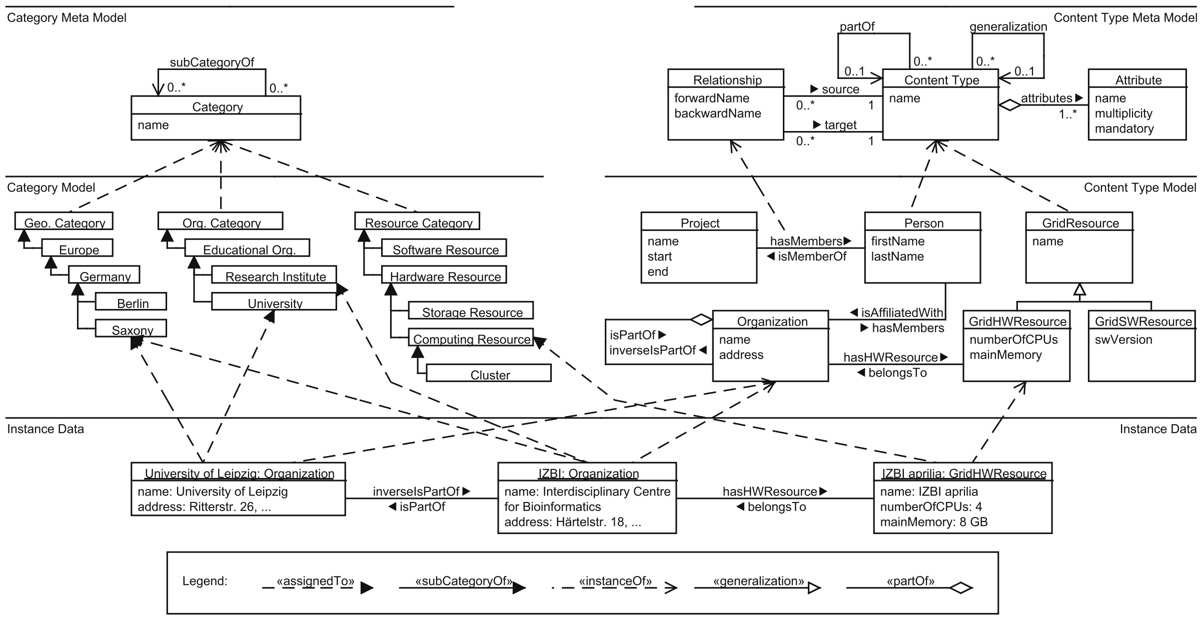


Fig. 1. Three-layered representation of metadata.

The meta model consists of two main parts, *content types* and *categories*. Content types define the meta information for instantiable information or content (instance data), called *content items*. Each content item instantiates a particular content type, i.e., that content type determines the structure of the information held by a content item. Categories, on the other hand, are not directly instantiable. Content items are merely assigned to categories. Accordingly, categories serve for semantic annotations of content items and are therefore modifiable with less impact on content items. Given these characterizations of content types and categories on the meta model level we outline their structure and further details in the following sections.

2.1. Content types

A content type has a *name* and a set of *attributes* describing properties for content items. An attribute has a *name*, a data type and a *multiplicity* of one or many. The latter allows for arbitrarily many values of that attribute within a content item. Attributes may also be defined as *mandatory*, i.e., they must be specified during content instantiation (e.g., the first and last name of a person). The attribute's data type restricts the permissible values, e.g., date, URL or string. Furthermore, allowed values can be restricted to an enumeration type to guarantee well-defined terms. We further distinguish between *generic* and *specific* attributes. Generic attributes are predefined and exist for all content types, e.g., the 'ID' attribute for unique object identification. Specific attributes describe application-specific properties of content types.

Content types can be interrelated by binary *relationships* of a specified cardinality. Relationships are managed bidirectionally and thus consist of a forward and backward relationship. Hence content items participating in a relationship are accessible from both directions. For instance, assume that a content type Person has a relationship with a content type Organization. When a content item A of Person 'isAssociatedWith' a content item B of Organization (forward relation), we also maintain that B is connected to A through a 'hasMembers' relationship (backward). In order to keep our model simple and flexible, we currently do not use relationship attributes.

In addition to such application-specific relationships we support two general kinds of relationships with predefined semantics: *generalization* and *partOf*. Firstly, content types can be part of generalization hierarchies supporting inheritance. Hence, derived content types reuse the metadata of their predecessors in the generalization hierarchy and may define additional attributes or relationships. The topmost (root) nodes of the generalization relation are called *base content types*. For instance, a base content type 'GridResource' may inherit its attributes and relationships to more specific content types such as 'GridHardwareResource' or 'GridSoftwareResource'. Secondly, the partOf relationship interrelates content types to construct aggregation hierarchies. For example, we use a recursive partOf relationship between organizations. Such partOf hierarchies are used in our platform to support navigation and to specify the context of content items. For instance, there may be several items called 'Department of Computer Science'. Their meaning only becomes clear by considering their predecessors within the organizational partOf hierarchy, e.g., to differentiate between 'University of Leipzig'/'Department of Computer Science' and 'TU Munich'/'Department of Computer Science'.

2.2. Categories

Categories have a *name* and are hierarchically organized by *subCategoryOf* relationships. These relationships are assumed to form directed acyclic graphs (DAGs) of categories. Moreover the subCategoryOf relationship involves different semantics depending on what categories are interrelated, e.g., in Fig. 1 'Germany' is part of 'Europe' or a 'University' is an 'Educational Organization'. *Roots* are special categories without predecessor for the subCategoryOf relationship and therefore act as entry points of a category structure.

We build on this simple yet flexible category model to broadly support semantic annotations, i.e., the ontological structuring and classification of content items (instance data). Categories can be used to manage content items of different content types *independently* of the content structure. In particular, content items can be categorized into multiple categories (e.g., according to distinct aspects). Notably, the assignment of content items to categories exhibits the character of annotations (see 'assignedTo' associations in Fig. 1). Such associations may be used in many cases, e.g., to instantiate categories or to associate objects to a geographical category. For example, the content item 'University of Leipzig' may be assigned to a 'University' category and a 'Saxony' category.

Categories can be used to improve the navigation within the platform and to support semantic queries. For instance, if someone is interested in all universities participating in a grid, she may navigate through the organization category structure to the university category to see all associated university organizations.

2.3. Distinguishing content types and categories: motivations and effects

Recently, a number of systems of different kinds implicitly embody facilities to handling data in a way related to the above explicit distinction of categories and content types in our meta model. Tagging systems [14] like Del.icio.us[2] or BibSonomy[3] collect keywords/tags (analogous to our categories) from all users assigned to entries typically sharing their

[2] http://delicious.com.
[3] http://www.bibsonomy.org.

appearance in a structured form, e.g., bibliographic entries in the case of BibSonomy. Another related case derives from the application of ontologies in bioinformatics. Ontologies like the Gene Ontology (GO) [2,34] are primarily used for data annotation, e.g., protein data (corresponding to content types) is annotated with GO terms (which is comparable to assigning categories) specifying the protein's function or frequent appearance in particular cell components.

The success of many of these systems supports the view that the suggested distinction between categories and content types is reasonable. In brief, the main idea is to combine benefits from a structured, database-oriented approach with an ontological approach for classification and annotation. Slightly closer inspection reveals the following observations on existing systems, in the terminology of our meta model. Content type models form a specific kind of domain model, typically with a rather small number of content types and their features (i.e., attributes and relations) and very limited use of generalization and inheritance. Due to the attributes and relations, however, content items capture information at a fine-grained level. The number of content items usually vastly exceeds the size of the content type models. Accordingly, the purpose of the category model is to provide means for navigation, filtering, and selection of content items by means of categories in well-known (poly)hierarchical arrangement.[4] In many cases, the assignment of categories to content items captures aspects that are, if at all, only very weakly related to those features of the content items covered in the content type model. For instance, a protein database may primarily describe the structure of proteins, whereas annotations to GO terms relate those proteins to functional aspects or involvement in certain biological processes.

Given these views, a number of potential benefits of the explicit distinction between content types and categories can be identified:

- Focused, quality-oriented information acquisition
  Capturing detailed information via the content model is restricted to that information which is vital for the purposes of the system. The structure as imposed by the features of content types contributes to high quality of the information.
- Exploitation of category hierarchies
  Content items can be classified directly and with minimal effort by users (and therefore browsed and searched semantically), without effects on content type information, yet adding information on content items.
- Mono- vs. polyhierarchical classification
  Generalization between content types is limited to a single supertype, which allows for clear management of the detailed content type model, in particular of feature inheritance. In contrast, the category model may rely on a polyhierarchical structure of categories. This is "more expressive" with respect to classification purposes. It remains manageable because the descriptive features of content types are not entangled with classification.
- Handling changes
  The distinction allows for a decoupled evolution of the content type model and the category model of a system (see the next section).
- Reuse
  Accepted category systems, e.g., established externally on a community basis, allow for their reuse as category model. They provide a common ground for information integration, e.g., for creating mashups of geospatial or temporal data, without an overall integration with the content type model and its interdependences.

As discussed above, the coverage of distinct aspects in the category and content type model is applicable to an increasing number of systems. Extensions to the meta model may consider some integration among the two meta model parts, e.g., in the form of automated category suggestions on the basis of features defined in the content type model.

Since large amounts of content items require long-term maintenance and adaptations at the model level, in the following section we focus on aspects of model evolution, including the effects of the separation of content types and categories in greater technical detail.

3. Evolution of data models

An active data model may require changes due to new or revised requirements on metadata or due to errors in the initial design that must be corrected. Therefore, the data model cannot be tied statically to an implementation of a system based on the overall meta model.

While changes primarily affect the data model itself, e.g., a content type or a category, it further may have impacts on other model parts, especially content items. For instance, a modification on a content type predominantly impacts content items that instantiate the concerned content type. Furthermore, others that are related to the modified content type may need an update as well, to ensure model consistency. These impacts should be treated automatically wherever applicable to reduce the manual effort of a user that executes a change on the data model. Hence, automatic migration of a relatively high number of affected content items saves time and reduces human errors compared to manual migration. Furthermore, an information preserving evolution of content items should be provided if possible. Utilizing the hierarchically organized structures of the data model, e.g., the content type generalization, the loss of information on content items resulting from

[4] Note that "category" is used here with a technical bias, rather than its analytical understanding in a (formal) ontological sense. Classification systems or terminological systems [38] are comparable to category models; likewise our subCategoryOf relation is similar to the "broader-than" relation, cf. [25].

some data model changes can be reduced. For instance, the deletion of a content type would typically result in deleting all instantiated content items. However, the migration to the super content type saves the concerned content items of course without specific information provided by the deleted content type.

In the following, we describe what types of change exist on the basis of the meta model. Instance level changes like the creation and modification of content items are assumed for information systems by default. Accordingly, we focus on changes in the data model and outline their semantics. Particularly, we present a *change classification* followed by a detailed description of the basic change operations.

3.1. Change classification

We basically differentiate between three general types of change that are applied to elements at the meta model level in order to support data model changes. The three general types of change are *additions*, *modifications*, and *deletions* of system information. Since the data model level is concerned, changes can be related to the notion of information capacity [17] (intuitively, the "expressiveness") of data models, where they can have different impacts. Additions increase and deletions reduce the information capacity. Only modifications are not restricted to a particular effect in this regard, since they may simply preserve information capacity, but likewise can increase or decrease the latter. Modifications further exhibit a variety of options for refinement, e.g., *renaming*.

The set of change operations that may generally be considered arises from applying the general change types to the meta model elements in Fig. 1, augmented by appropriate refinements of modification operations. Fig. 2 provides an overview of all resulting changes that we consider relevant for most applications of the system. The upper part of the classification is organized according to the model structure components that change operations apply to, whereas the leaves correspond to the basic change operations. The next sections summarize and exemplify these change operations in more detail.

3.2. Changes to content type models

As the taxonomy of changes shows, we distinguish between changes on content types, attributes and relationships in the content type model. Note that these changes are similar to those proposed in related work on object-oriented databases [5] since the content type model possesses object-oriented-like structures and elements, e.g., the generalization hierarchy or relationships. Hence, we can reuse experiences and techniques of schema evolution in object-oriented databases to support

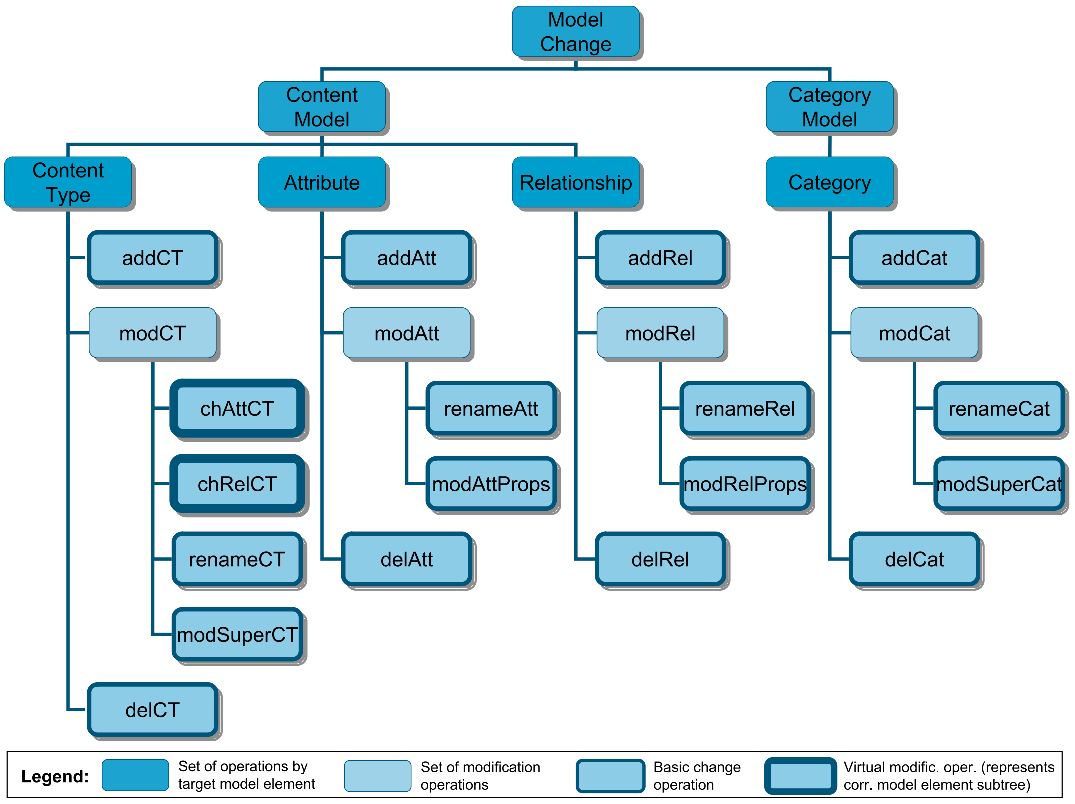


Fig. 2. Classification of selected change operations.

Table 1
Changes on content type model.

Change	Parameters	Description
addCT	*ct_name* *super_ct* *attributes* *relationships*	A new content type with *ct_name* will be created and added to content type model by considering the specified *attributes, relationships* and the optional *super_ct*
renameCT	*ct* *new_name*	The name of *ct* will be changed to *new_name*
modSuperCT	*ct* *super_ct*	The super content type of *ct* will be changed to *super_ct*
delCT	*ct*	Content type *ct* will be removed from the content type model
addAtt	*ct* *att_name* *properties*	A new attribute *att_name* including *properties* will be added to the content type *ct*
renameAtt	*ct* *att_name* *att_new_name*	The attribute *att_name* of *ct* will be changed to *att_new_name*
modAttProps	*ct* *att_name* *properties*	Properties of attribute *att_name* of *ct* will be changed to property settings specified in *properties*
delAtt	*ct* *att_name*	The attribute with name *att_name* will be removed from *ct*
addRel	*ct* *forward_name* *backward_name* *ct_ref*	A new relationship with forward relation *forward_name* and backward relation *backward_name* will be created between content type *ct* and the reference content type *ct_ref*
renameRel	*ct* *rel_name* *rel_new_name*	The relationship *rel_name* of *ct* will be changed to *rel_new_name*
modRelProps	*ct* *rel_name* *properties*	Properties of relationship *rel_name* of *ct* will be changed to property settings specified in *properties*
delRel	*ct* *rel_name*	The relationship *rel_name* will be removed from *ct*

changes in the content type model. Particularly, we describe changes on content types followed by attribute and relationship changes (including partOf) as displayed in Table 1.

Utilizing *addCT*, a new content type will be added to the content type model. The specified attributes and relationships consist of further meta information expressed as properties, e.g., mandatory or multiplicity (for unspecified properties a default property value is used). Note that both *addCT* and *renameCT* may have an impact on other content types due to relationships with the new/renamed content type. *modSuperCT* for changing the super content type and *delCT* for removing an existing content type affect further model parts to an even greater extent. Particularly, affected content items of a deleted content type can be treated variedly depending on the intension of the user. Accordingly, migration to the super content type (if one exists) or immediate deletion are possible. Furthermore, sub content types may be assigned a new super content type or will become base content types. Finally, content types related to the removed content type need an update in their relationships and consequently an adaptation of their content items. For instance, removing 'GridHWResource' from the content type model in Fig. 1 may result in migration of content items to 'GridResource'. An immediate deletion of affected content items is also conceivable, however with the consequence that these content items are no longer accessible within the platform. Furthermore, the content type 'Organization' needs an update to remove its relationship to 'GridHWResource'.

addAtt enables the addition of a new attribute to an existing content type and accordingly to its sub content types. Concerning content items, additions of optional attributes have no immediate impact since values of a newly added attribute are considered as null until they are explicitly specified by a user. Furthermore, the name and the properties of an attribute, e.g., mandatory or multiplicity, can be changed with *renameAtt* and *modAttProps*, respectively. Altered properties are captured in a *properties* set consisting of (property – new value) entries. Properties not included in *properties* remain unchanged. Note that impacts of property modifications depend on the altered property. For instance, an increase of multiplicity from '1' to '3' has no effects on instantiated content items. In contrast a change of the mandatory status from 'no' to 'yes' results in an adaptation (default value setting) of content items who currently have no captured attribute values. No longer used attributes can be removed from a content type by applying *delAtt* which always results in an instance migration in order to exclude the concerned attribute and corresponding values from instantiated content items.

A new bidirectional relationship between two content types is created with *addRel*. Due to the bidirectionality of relationships an addition has effects on both ends, i.e., both involved content types, sub content types and consequently all instantiated content items need to be altered to include the new relationship. Renaming (*renameRel*) and property modification (*modRelProps*) of relationships are similar to those of attributes. Finally, deleting an existing relationship (*delRel*) always impacts other model parts. For instance, the deletion of the relationship belongsTo/hasHWResource can be specified as *delRel*(GridHWResource, belongsTo) or *delRel*(Organization, hasHWResource). The bidirectionality thus leads to changes of both content types. Hence, content items of 'GridHWResource' and 'Organization' need to be migrated in order to remove relationship entries for belongsTo and hasHWResource, respectively.

3.3. Changes to category models

Changes in the category model concern categories and their position in the hierarchy. We identified four basic change operations summarized in Table 2. *addCat* with parameters name of category and optional super categories is applied to create and add a new category to the category model. As an example, a new category 'StorageResource' with super category 'Resource' is created by *addCat*(StorageResource, {Resource}). With *modSuperCat* the structure (hierarchy) of categories can be altered. For instance, a super category change from 'Resource' to 'HardwareResource' for 'StorageResource' (*modSuperCat*(StorageResource, {HardwareResource})) describes a move within the category hierarchy. Note that such a change has no impact on associated content items since they are merely assigned to categories. In contrast, the renaming of a category (*renameCat*), e.g., 'StorageResource' is renamed to 'DataResource' (*renameCat*(StorageResource, DataResource)), has effects on associated content items. Particularly, assignments of content items need to be migrated to the renamed category otherwise the assignments will become obsolete.

The deletion of an outdated category is supported by *delCat*, e.g., the deletion of 'DataResource' with *delCat*(DataResource). This change operation has two consequences that need to be considered. On the one hand, sub categories of the removed category need to be migrated. For instance, the sub categories of 'DataResource' will be given new super categories caused by the deletion of their current super category. Thus, these categories will become sub categories of 'HardwareResource' since this is the super category of 'DataResource'. If no super category exists, these categories will become root categories with no super category. On the other hand, assignments to the removed category need to be adapted. Basically, assignments of content items can be migrated to the super category, e.g., 'HardwareResource' for 'DataResource'. In the absence of a super category the concerning assignments will be lost.

3.4. Comparing content type and category evolution

The previous sections have described data model change operations and their possible impacts on other model parts for the content type and the category model. This section compares evolution aspects of both models with each other. Particularly, we outline specifics of evolution in each model and describe benefits of managing changes (evolution) in a decoupled way.

A content type model usually exhibits a relatively flat structure in terms of the content type generalization hierarchy. By contrast, content type attributes and relationships support detailed representation of structured information, i.e., they describe content types in more detail and act as a schema for content items. The development and refinement of a content type model is normally performed in the following way. Designers typically start with a small number of general content types and refine them over time, e.g., by adding extra attributes or more specific sub content types. The focus of evolution is therefore rather on attribute and relationship changes than on complex modifications of the generalization hierarchy (e.g., content type restructuring). Changes on content types primarily have impacts on content items due to the instantiation between content items and content types. These impacts mainly result in adaptation of content items to include (exclude) attributes or relationships of content types. However, content type changes mostly do not influence the category model and category assignments of content items (an exception form content item deletions caused by a base content type deletion).

Category models are usually complex, i.e., they are often possibly deep and cross-linked hierarchical structures. The focus of changes is more on the enhancement of those hierarchies (e.g., by adding or moving categories) for categorization than on the detailed description of a category itself. Contrary to content types, content items are at most affected indirectly when category changes occur. Particularly, only the assignments of content items to categories need to be adapted since a content item does not instantiate a category directly, it is 'only' annotated with a number of categories. This fact allows for more complex changes in the category model (e.g., moves or restructuring of categories) resulting in no, or only a few, simple

Table 2
Changes on category model.

Change	Parameters	Description
addCat	*cat_name* *super_cat*	A new category with *cat_name* and super categories specified in *super_cat* will be created and added to the category model
renameCat	*cat* *new_cat_name*	The category *cat* will be renamed to *new_cat_name*
modSuperCat	*cat* *super_cat*	The super categories of *cat* will be replaced by the entries in *super_cat*
delCat	*cat*	The category *cat* will be removed from the category model

changes in the instance data (adaptation of category assignments for content items). Analogous to changes in the content type model, category model changes do not influence the other model part. Altogether, this is a clear improvement over evolution approaches that, e.g., operate on complex graph structures where even more and complicated side effects of changes need to be considered.

Comparing evolution and its impacts for the two decoupled models a potential use case of the system arises. Ontology engineers may provide the current version of an ontology under development in terms of the *content type model*, whereas users may categorize content items according to their intuitions by means of the *category model*. The combined information may then be used for evaluating hypothetical extensions to the content type model (i.e., the ontology under development).

4. Sample application – the D-Grid Ontology

D-Grid started in 2005 as a Germany-wide grid initiative. Its aim is to provide a common grid infrastructure for e-Science projects in Germany and to prove the viability and advantages of grid usage in different scientific domains. D-Grid entails many community projects, e.g., for medical and physics applications, and a common integration project (DGI).

Currently, metadata about D-Grid and its structures is highly heterogeneous and distributed across many websites and project-specific repositories, e.g., information about projects, persons, or available hardware and software resources. Furthermore, there are almost no relations or explicit semantic links between these independently maintained information objects. The goal of our metadata platform is to integrate and semantically categorize this heterogeneous information in a common system and to offer it to all D-Grid participants, applications and interested users. New participants in D-Grid can thus quickly inform themselves about ongoing work in D-Grid projects and the organizations and persons involved. Furthermore, resource providers, i.e., institutes providing hardware or software to the grid, can specify parameters about their resources which may be useful for scheduling and distribution of grid applications. Our platform semantically categorizes its content within a so-called D-Grid Ontology (DGO). It simplifies the manual creation and maintenance of metadata using a collaborative, wiki-like platform. Through the use of the meta model including content types and ontological annotations high data consistency and quality is pursued.

On the basis of our meta model described in Section 2, we use four basic grid content types in the DGO model, namely *Person*, *Project*, *Organization* and *GridResource* (see content type model in Fig. 1). As an example, the content type Person uses attributes such as first name, last name, email or phone number for the registration of personal information. Furthermore, relationships to content items of other content types show a person's semantic neighborhood, e.g., the projects a person is working in ('isMemberOf') or the organization to which a person is affiliated ('isAffiliatedWith'). Furthermore, DGO exploits recursive partOf relationships for projects and organizations. In particular, 'D-Grid' is the topmost project of DGO and contains a number of sub projects such as 'MediGRID', 'HEP-Grid' or the 'Integration Project (DGI)', which themselves include further sub projects. Furthermore, DGO uses several category hierarchies for ontological classification of content items (see category model in Fig. 1). Every content item of DGO is assigned to a minimum of one category. For instance, a community project such as 'MediGRID' is assigned to the category 'Community Project' (in terms of project type) and 'D-Grid I' (funding aspect) since it was funded as one of the starting projects of the D-Grid initiative.

The current version of DGO (as of November 2008) categorizes and interrelates about 40 projects, 150 organizations, 300 persons, and 75 grid resources. There are about 950 bidirectional relationships between content items.

5. Usability features

In the following, we describe some of the features of our platform to illustrate its usability. In particular, we firstly illustrate how semantic metadata is displayed within the platform. Furthermore, we present navigation and search capabilities as well as options for creation, classification and editing of content. For hands-on experience the interested reader may directly use the system (after registration) under http://buell.izbi.uni-leipzig.de/dgo.

5.1. Content visualization

Each content item is shown on its own article page, providing information about its name, basic attributes, relationships, category classifications, explanations (free text), images and versioning. Relationships to other content items are presented as hyperlinks. Specific tabs allow for the direct change of content pages, in particular editing, renaming or category assignment.

Our platform exploits Web 2.0 techniques, such as maps and navigable trees, to display semantic metadata in different forms. In particular, we use Google Maps[5] to geographically locate content items such as organizations or D-Grid hardware resources on a map. This feature may be helpful for users to find regionally close D-Grid participants to promote a possible cooperation. For instance, the sample map in Fig. 3 (left) includes all organizations currently participating in D-Grid. When selecting a location, e.g., Leipzig, all organizations in this place participating in D-Grid are listed and may be further explored. For each location on the map, we use the partOf structure among content items to aggregate all corresponding items for display.

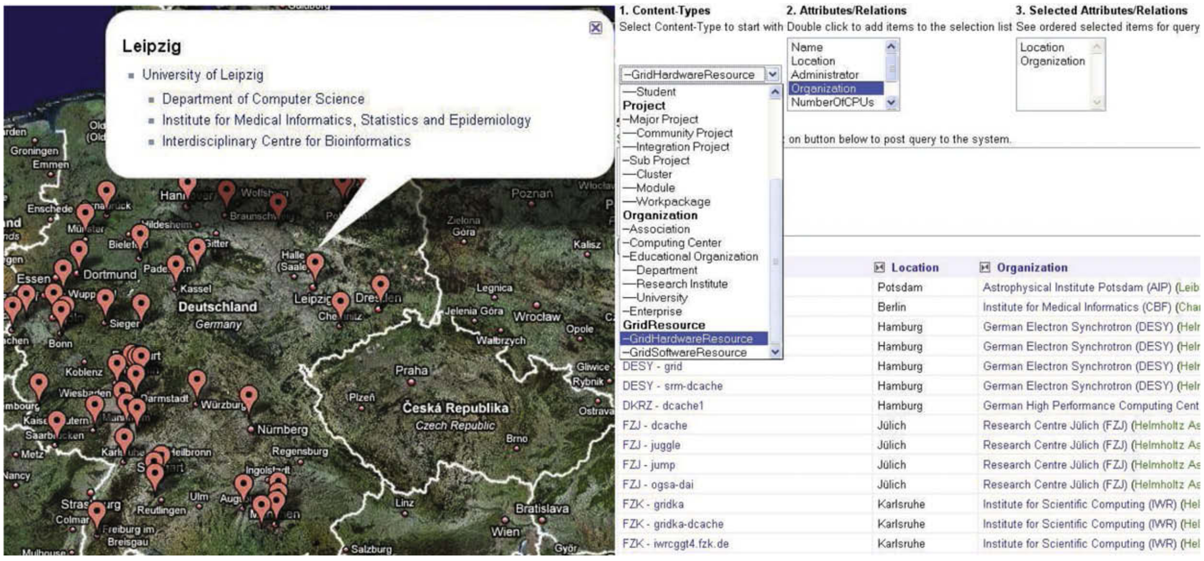
[5] http://maps.google.com.

Furthermore, we employ the partOf relationships to generate trees representing hierarchical structures such as organization or project structures.

5.2. Search and navigation facilities

The platform provides different search and navigation facilities. A simple text search supports keyword-based search over all attributes of content items. Furthermore, semantic query capabilities on content types and categories are provided. In particular, a query generator (Fig. 3 right) for interactive specification of semantic queries is available so that users need not learn a complex query syntax. The results are presented in tables which can be interactively sorted on different attributes or relationships, e.g., person name or the affiliated organization.

The platform also provides extensive navigation capabilities for content retrieval with the help of a category browser (Fig. 4 left). It dynamically generates a navigation tree representing categories and content items in an integrated form, by associating content items to their most specific categories. For instance, with a few clicks a user can navigate from the top category ‘Person’ to ‘Researcher’ or ‘Professor’ to see all associated content items. Every node of the tree links to the corresponding category page or content item article.



Leipzig
- University of Leipzig
  - Department of Computer Science
  - Institute for Medical Informatics, Statistics and Epidemiology
  - Interdisciplinary Centre for Bioinformatics

1. Content-Types
Select Content-Type to start with
−GridHardwareResource
—Student
Project
−Major Project
—Community Project
—Integration Project
−Sub Project
—Cluster
—Module
—Workpackage
Organization
−Association
−Computing Center
−Educational Organization
—Department
—Research Institute
—University
−Enterprise
GridResource
−GridHardwareResource
−GridSoftwareResource

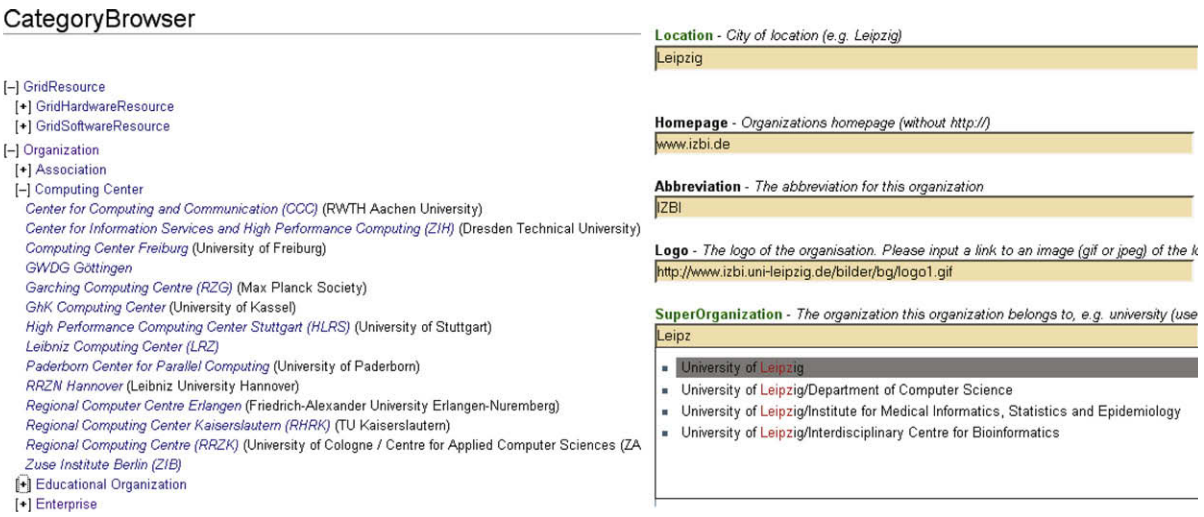
2. Attributes/Relations
Double click to add items to the selection list
Name
Location
Administrator
Organization
NumberOfCPUs

3. Selected Attributes/Relations
See ordered selected items for query
Location
Organization

on button below to post query to the system.

	Location	Organization
	Potsdam	Astrophysical Institute Potsdam (AIP) (Leib
	Berlin	Institute for Medical Informatics (CBF) (Char
	Hamburg	German Electron Synchrotron (DESY) (Helr
	Hamburg	German Electron Synchrotron (DESY) (Helr
DESY - grid	Hamburg	German Electron Synchrotron (DESY) (Helr
DESY - srm-dcache	Hamburg	German Electron Synchrotron (DESY) (Helr
DKRZ - dcache1	Hamburg	German High Performance Computing Cent
FZJ - dcache	Jülich	Research Centre Jülich (FZJ) (Helmholtz As
FZJ - juggle	Jülich	Research Centre Jülich (FZJ) (Helmholtz As
FZJ - jump	Jülich	Research Centre Jülich (FZJ) (Helmholtz As
FZJ - ogsa-dai	Jülich	Research Centre Jülich (FZJ) (Helmholtz As
FZK - gridka	Karlsruhe	Institute for Scientific Computing (IWR) (Hel
FZK - gridka-dcache	Karlsruhe	Institute for Scientific Computing (IWR) (Hel
FZK - iwrcggt4.fzk.de	Karlsruhe	Institute for Scientific Computing (IWR) (Hel

Fig. 3. Organizations of D-Grid on a map and query generator.



CategoryBrowser

[–] GridResource
  [+] GridHardwareResource
  [+] GridSoftwareResource
[–] Organization
  [+] Association
  [–] Computing Center
    Center for Computing and Communication (CCC) (RWTH Aachen University)
    Center for Information Services and High Performance Computing (ZIH) (Dresden Technical University)
    Computing Center Freiburg (University of Freiburg)
    GWDG Göttingen
    Garching Computing Centre (RZG) (Max Planck Society)
    GhK Computing Center (University of Kassel)
    High Performance Computing Center Stuttgart (HLRS) (University of Stuttgart)
    Leibniz Computing Center (LRZ)
    Paderborn Center for Parallel Computing (University of Paderborn)
    RRZN Hannover (Leibniz University Hannover)
    Regional Computer Centre Erlangen (Friedrich-Alexander University Erlangen-Nuremberg)
    Regional Computing Center Kaiserslautern (RHRK) (TU Kaiserslautern)
    Regional Computing Centre (RRZK) (University of Cologne / Centre for Applied Computer Sciences (ZA
    Zuse Institute Berlin (ZIB)
  [+] Educational Organization
  [+] Enterprise

Location - *City of location (e.g. Leipzig)*
Leipzig

Homepage - *Organizations homepage (without http://)*
www.izbi.de

Abbreviation - *The abbreviation for this organization*
IZBI

Logo - *The logo of the organisation. Please input a link to an image (gif or jpeg) of the l*
http://www.izbi.uni-leipzig.de/bilder/bg/logo1.gif

SuperOrganization - *The organization this organization belongs to, e.g. university (use*
Leipz
- University of Leipzig
- University of Leipzig/Department of Computer Science
- University of Leipzig/Institute for Medical Informatics, Statistics and Epidemiology
- University of Leipzig/Interdisciplinary Centre for Bioinformatics

Fig. 4. Category browser and editing of content.

5.3. Creation and editing of content

For every content type the system provides an interactive input form to create and change content items including their assignments to categories (Fig. 4 right). These forms are dynamically created from the current meta information (attributes, relationships) of a content type. They consist of different kinds of form fields, in particular mandatory, autocomplete-aware, single-/multivalued and category assignment fields and free text boxes. The different kinds of fields are marked with different background colors and labels to improve user interaction and the input dialog.

As soon as a user clicks on an autocomplete field or types some letters into it, typed value suggestions are offered for selection, which simplifies input and reduces the number of duplicate entries. For example, an input field capturing a relationship to the content type 'Organization' (e.g., a person's affiliation) suggests organization items matching the input. Furthermore, if an attribute is restricted to a controlled vocabulary (i.e., an enumeration datatype), we suggest values matching current entries of such a vocabulary. In multivalued fields a common separator divides multiple entries for a single attribute or relationship. Category assignment fields employ autocompletion to simplify categorization and to guarantee correct category assignments.

6. Implementation

The presented platform builds upon a widely used semantic wiki implementation, the Semantic Media Wiki (SMW) [21]. SMW, in turn, extends the MediaWiki[6] implementation, which is also used by Wikipedia. MediaWiki provides a powerful infrastructure for collaborative management of text-based articles. It is also aware of categories and sub categories, but links between articles in MediaWiki are untyped (have no semantics) and search capabilities are limited to simple text searches. SMW introduces semantic properties for wiki articles and thus supports a semantic annotation and enhanced querying of wiki contents.

We extended MediaWiki and SMW in several directions. Firstly, we introduce content types to capture semantic metadata in the form of structured content. Particularly, the template feature of MediaWiki is utilized to describe content types. Hence, for each defined content type of the content model a corresponding template exists. For instance, the content type Person is described by a Person template including specifications such as the email attribute or relationships to other content types (e.g., the affiliation relationship). Furthermore, templates are used for the visualization of content items within the platform as well as the automatic generation of graphical edit forms based on content type descriptions.

We introduced bidirectional relationships (on the basis of SMW semantic properties) between content types to automatically maintain referential integrity and to provide better navigation capabilities. The implementation of this feature requires checks during the edit of a content item to ensure the integrity. Particularly, if new or changed relationship values are inserted the referenced content items need to be updated as well. For instance, if Person A leaves Project P, i.e., the project entry is removed from A, the implementation ensures that A is also removed from the project member list of P (backward relationship). Furthermore, if a relationship references an unknown content item, we automatically generate this based on knowledge from the bidirectional relationship and corresponding values.

We support graphical UIs for content creation and change, e.g., autocompletion to avoid duplicates. As mentioned before the UIs are automatically generated based on content type descriptions captured in templates. Moreover, we utilize Web 2.0 techniques for novel visualization and interaction options, e.g., dynamic generation of maps for content items and interactive specification of semantic queries. In order to generate these maps, we utilize location attributes of a content type as well as partOf relationships between content items. Currently, the location attributes represent the city, e.g., of an organization. The geographical coordinates (latitude/longitude) of a city needed for the map visualization is obtained from a publicly available web service.[7] Another implemented feature is the display and usage of contexts. This feature utilizes partOf relationship entries to compute the context of content items. For instance, one can differentiate between a 'Department of Computer Science' located at the 'University of Leipzig' or the 'TU Munich'. Hence, confusions are reduced, e.g., during the creation of content or in query results.

An evolution environment supports changes on a running data model, i.e., content types and categories can be altered to incorporate new or changed requirements. The implementation is based on the concept of data model evolution discussed in Section 3. Particularly, users can use the operations presented in Fig. 2 to modify the content type model as well as the category model. Possible adaptations on content items are performed automatically, i.e., the system checks what content items are affected by a particular data model change and migrates them to be consistent with the modified data model. For instance, the deletion of a content type attribute adapts affected content items by removing the attribute and corresponding values.

7. Related work

7.1. Semantic wikis

Our approach builds upon established wiki technology [22] and its combination with semantic technology, cf. [29,36]. The initially visible distinction between semantic wikis originating from 'classical' wikis, e.g., the Semantic MediaWiki (SMW)

[6] http://www.mediawiki.org.
[7] http://www.geonames.org.

[21], and editors for knowledge bases or ontologies with wiki-like, collaborative features, e.g., IkeWiki [31] or OntoWiki [3], is currently diminishing [6]. Classical ontology editors such as Protégé[8] focus on the initial design of ontologies at the schema level. They support many advanced ontology language features (e.g., logical class definitions such as unions, intersections and complements) making their user interfaces rather complex. By contrast, our platform aims at the web-based collection and maintenance of content items and their categorization.

Another difference to related systems concerns our meta model. The meta models of many semantic wikis are based on Semantic Web standards, most often RDF [23] and OWL [24] (e.g., WikSAR [4], SweetWiki [6], IkeWiki and OntoWiki). In these systems the ontology categories describe the structure of the content items. In contrast, our meta model explicitly supports both a database-oriented and an ontological part so that the content structure is independent of the ontological content categorization using multiple category hierarchies. These aspects result in a clearly structured system configuration and facilitate a straightforward access and maintenance of grid metadata. While the different parts of our meta model could be expressed by standard ontology languages such as OWL, we do not need the full complexity of OWL and wanted to conceptually differentiate between a database-oriented and an ontological part. The built-in support for these two parts is not found in other wiki implementations. An interesting extension of our platform is import/export functionality for standard languages. For instance, initial category as well as content models can be designed with other tools (e.g., Protégé) and could be imported into the platform. Similarly, a currently developed data model can be exported, e.g., for usage in other applications.

Another feature of our platform is the bidirectionality of the relationships. This can be considered as a simple form of reasoning which still allows for efficient system behavior (cf. also the extension of RDFS termed RDFS-Plus in [1, chapter 7]). Many semantic wikis avoid the use of Semantic Web reasoning for efficiency reasons (cf. [20], [6, p. 87]; exceptions are, e.g., IkeWiki and BOWiki [16]), but offer limited forms, e.g., type inferences. IkeWiki is comparable to our platform in this respect, since users can follow relations in both directions (without offering forward and backward readings). SMW does not support inverse relations itself, but there are some workarounds to produce corresponding behavior.[9]

Finally, we are not aware of semantic wikis with a special semantics for the *partOf* relationship. In our case, this relation is primarily used for sophisticated labeling of content items. Thus it can partially avoid the use of – in many cases manually maintained – context modifiers, e.g., the component in parentheses of ‘Dept. of Computer Science (University of Leipzig)’.

7.2. Schema evolution

Evolution of data models (schemas) was primarily studied in the area of databases and closely related disciplines. The online bibliography on schema evolution presented in [28] and available online[10] categorizes relevant literature along the various research fields and models. Particularly, schema evolution is investigated for relational database systems, object-oriented systems, XML and ontologies. A survey on schema evolution in database systems is given in [30]. Effects of relational schema changes are discussed in [10,17]. In particular, Hull introduced the concept of information capacity [17] to measure effects of changes on instance data. We reuse the definition of information capacity in our system to categorize possible evolution operations for content types and categories regarding their effects on instance data (content items) as described in Section 3.1.

Schema evolution for object-oriented databases was investigated in [5,35], in particular they defined possible changes and proposed techniques for their realization. Evolution in OO databases is similar to the evolution of our content types since we utilize a content type model with inheritable content types which consist of attributes and relationships for structural information representation. The evolution of XML was studied from two directions. While a first scenario dealt with the adaptation of XML schemas (e.g., DTD) if instance documents evolve [19], another scenario investigated evolution on the schema level [8,15,33]. Particularly, change operations for schema elements were proposed and techniques for migrating instance documents were evaluated. Note that the second scenario is more similar to our evolution approach since we study evolution on the data model level (content types and categories) and migrate instances in case of inconsistency. In other words, we do not investigate data model adaptation if instance data (content items) evolve.

Recently a lot of work addressed the evolution and versioning of ontologies (see [11] for a survey) especially in the Semantic Web and for specific ontology representations such as OWL [24] or Frame Logic [18]. Noy and Klein [26] defined change operations to describe the evolution between ontology versions. A formalization of the ontology evolution process was described in [32]. The authors proposed a 6-phase evolution process which can handle critical changes on ontologies unambiguously. The evolution of ontologies is similar to the evolution of our category model. However, our work differs from previous work on ontology evolution since we investigate evolution of both categories (ontological part) and content types (structural part) independently. This distinction allows for a lightweight and decoupled evolution of our models compared to a uniform approach operating on, e.g., a complex graph structure, where even more and complicated side effects of a change need to be considered.

[8] http://protege.stanford.edu.
[9] http://semantic-mediawiki.org/wiki/Help:Inferencing.
[10] http://se-pubs.dbs.uni-leipzig.de/.

7.3. Grid ontologies

The development and application of grid ontologies was already studied in the area of the Semantic Grid [9]. The S-OGSA architecture [7] of the OntoGrid project is a reference architecture for the Semantic Grid extending OGSA [12] to support explicit handling of semantics. Knowledge represented in ontologies (e.g., a VO ontology to describe virtual organizations) can be accessed by ontology services provided within the grid.

Other grid ontologies partly focussing on specific grid sub domains were developed in [27,37,39]. Particularly, the OWL-based Core Grid Ontology proposed in [39] differentiates between core classes such as GridUser, GridMiddleware, GridApplication or GridResource to represent knowledge about grid infrastructures. Furthermore, the authors of [27] investigated a knowledge architecture including reference ontologies for grid computing. The approach provides the foundation for semantic description of grid resources/services to enable interoperability in a grid. The myGrid ontology presented in [37] describes the bioinformatics research domain by a service and a domain ontology to support service discovery. While the domain ontology acts as a vocabulary for describing core bioinformatics data types, the service ontology describes physical and operational features of available services. In contrast to our work these ontologies do not differentiate between a data-oriented and an ontological part. Our D-Grid ontology aims at uniformly and semantically describing the D-Grid initiative, especially persons, projects and organizations involved in D-Grid. Furthermore, none of the approaches proposed an interactive platform to maintain and manage semantic grid metadata collaboratively and online.

8. Summary and future work

We presented a meta model and a platform for the collaborative management of semantic metadata in grids. The platform provides grid participants of large-scale grid initiatives such as D-Grid with a collaborative and web-based way of creating, editing and using grid metadata, e.g., on grid resources, projects, and participating organizations and persons. Furthermore, change management for a running data model including the automatic migration of affected instance data is provided. Particularly, the independence of the content type and the category model allows for a lightweight and simplified evolution of both in comparison to evolution on highly complex and cross-linked data models. We applied the platform within the German D-Grid initiative in order to build a semantic metadata repository for D-Grid and to improve the collaboration between participating projects. The platform is currently running under http://buell.izbi.uni-leipzig.de/dgo and is actively used by D-Grid members.

We see several opportunities for future work. First, the platform can be extended based on new requirements from the D-Grid communities. Second, the utilization of multiple category models for different virtual organizations of a grid can be investigated. So content items may be categorized along individual category models that represent the perception of a specific domain (e.g., physics or medicine). Another point for future work is the automatic categorization of content items. Particularly, attribute values of content items or relationship entries may contain information that should be used for categorization suggestions in order to reduce the manual effort for users. For instance, if a content item is located in Leipzig, a possible category assignment suggestion would be Saxony. Furthermore, if a super organization is assigned to a particular geographic category all its sub organizations will likely possess the same annotation as well.

Acknowledgements

This work is supported by the German Federal Ministry of Education and Science research project MediGRID – Module Ontology Tools (01AK803E) as part of the D-Grid initiative. We are grateful to the anonymous reviewers for their suggestions that helped to improve the paper.

References

[1] D. Allemang, J. Hendler, Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL, Morgan Kaufman Publishers, Burlington, Massachusetts, 2008.
[2] M. Ashburner et al, Gene ontology: tool for the unification of biology, Nature Genetics 25 (1) (2000) 25–29.
[3] S. Auer, S. Dietzold, T. Riechert, Ontowiki – a tool for social, semantic collaboration, in: I.F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, L. Aroyo (Eds.), The Semantic Web – ISWC 2006: Proceedings of the Fifth International Semantic Web Conference, Athens, Georgia, USA, November 5–9, Springer, Berlin, 2006, pp. 736–749.
[4] D. Aumüller, S. Auer, Towards a semantic wiki experience – desktop integration and interactivity in WikSAR, in: S. Decker, J. Park, D. Quan, L. Sauermann (Eds.), Proceedings of the ISWC 2005 Workshop on The Semantic Desktop: Next Generation Information Management and Collaboration Infrastructure, Galway, Ireland, November 6, CEUR-WS.org, Aachen, Germany, 2005, pp. 212–217.
[5] J. Banerjee, W. Kim, H.-J. Kim, H.F. Korth, Semantics and implementation of schema evolution in object-oriented databases, in: U. Dayal, I.L. Traiger (Eds.), SIGMOD '87: Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data, ACM, New York, 1987, pp. 311–322.
[6] M. Buffa, F.L. Gandon, G. Ereteo, P. Sander, C. Faron, SweetWiki: a semantic wiki, Journal of Web Semantics 6 (1) (2008) 84–97.
[7] O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, C. Goble, An overview of S-OGSA: a reference semantic grid architecture, Journal of Web Semantics 4 (2) (2006) 102–115.
[8] T. Dalamagas, A. Meliou, T. Sellis, Modeling and manipulating the structure of hierarchical schemas for the web, Information Sciences 178 (4) (2008) 985–1010.
[9] D. De Roure, N. Jennings, N. Shadbolt, The semantic grid: past, present, and future, Proceedings of the IEEE 93 (3) (2005) 669–681.
[10] C. Ewald, M. Orlowska, Characterization of the effects of schema change, Information Sciences 94 (1–4) (1996) 23–39.

[11] G. Flouris, D. Manakanatas, H. Kondylakis, D. Plexousakis, G. Antoniou, Ontology change: classification and survey, Knowledge Engineering Review 23 (2) (2008) 117–152.

[12] I. Foster, C. Kesselman, J. Nick, S. Tuecke, Grid services for distributed system integration, Computer 35 (6) (2002) 37–46.

[13] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, International Journal of High Performance Computing Applications 15 (3) (2001) 200–222.

[14] S.A. Golder, B.A. Huberman, Usage patterns of collaborative tagging systems, Journal of Information Science 32 (2) (2006) 198–208.

[15] G. Guerrini, M. Mesiti, D. Rossi, Impact of XML schema evolution on valid documents, in: A. Bonifati, D. Lee (Eds.), WIDM '05: Proceedings of the Seventh Annual ACM International Workshop on Web Information and Data Management, ACM, 2005, pp. 39–44.

[16] R. Hoehndorf, J. Bacher, M. Backhaus, S.E.J. Gregorio, F. Loebe, K. Prüfer, A. Uciteli, J. Visagie, H. Herre, J. Kelso, BOWiki: an ontology-based wiki for annotation of data and integration of knowledge in biology, BMC Bioinformatics 10 (Suppl. 5) (2009) S5.

[17] R. Hull, Relative information capacity of simple relational database schemata, SIAM Journal on Computing 15 (3) (1986) 856–886.

[18] M. Kifer, G. Lausen, J. Wu, Logic foundation of object-oriented and frame-based languages, Journal of the ACM 42 (4) (1995) 741–843.

[19] M. Klettke, H. Meyer, B. Hansel, Evolution: the other side of the XML update coin, in: ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops, IEEE Computer Society, 2005, p. 1279.

[20] M. Krötzsch, S. Schaffert, D. Vrandečić, Reasoning in semantic wikis, in: G. Antoniou, U. Aßmann, C. Baroglio, S. Decker, N. Henze, P.-L. Patranjan, R. Tolksdorf (Eds.), Reasoning Web: Tutorial Lectures of the Third International Summer School 2007, Dresden, Germany, September 3–7, Springer, Berlin, 2007, pp. 310–329.

[21] M. Krötzsch, D. Vrandečić, M. Völkel, H. Haller, R. Studer, Semantic wikipedia, Journal of Web Semantics 5 (4) (2007) 251–261.

[22] B. Leuf, W. Cunningham, The Wiki Way: Collaboration and Sharing on the Internet, Addison-Wesley Professional, Reading, Massachusetts, 2001.

[23] F. Manola, E. Miller, RDF Primer, W3C Recommendation, World Wide Web Consortium (W3C), Cambridge, Massachusetts, 2004.

[24] D.L. McGuinness, F. van Harmelen, OWL Web Ontology Language Overview, W3C Recommendation, World Wide Web Consortium (W3C), Cambridge, Massachusetts, 2004.

[25] A. Miles, S. Bechhofer, SKOS Simple Knowledge Organization System Reference, W3C Working Draft, World Wide Web Consortium (W3C), Cambridge, Massachusetts, 2008.

[26] N.F. Noy, M. Klein, Ontology evolution: not the same as schema evolution, Knowledge Information Systems 6 (4) (2004) 428–440.

[27] M. Parkin, S. van den Burghe, O. Corcho, D. Snelling, J. Brooke, The knowledge of the grid: a grid ontology, in: M. Bubak, M. Turala, K. Wiatr (Eds.), Proceedings of the Sixth Cracow Grid Workshop, Cracow, Poland, ACC Cyfronet AGH, 2006, pp. 111–118.

[28] E. Rahm, P.A. Bernstein, An online bibliography on schema evolution, ACM SIGMOD Record 35 (4) (2006) 30–31.

[29] D. Riehle, J. Noble (Eds.), Proceedings of the 2006 International Symposium on Wikis, ACM, New York, 2006.

[30] J.F. Roddick, Schema evolution in database systems: an annotated bibliography, ACM SIGMOD Record 21 (4) (1992) 35–40.

[31] S. Schaffert, IkeWiki: a semantic wiki for collaborative knowledge management, in: Proceedings of the 15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE 2006, Manchester, UK, June 26–28, IEEE Computer Society, Los Alamitos, California, 2006, pp. 388–396.

[32] L. Stojanovic, A. Maedche, B. Motik, N. Stojanovic, User-driven ontology evolution management, in: A. Gómez-Pérez, V.R. Benjamins (Eds.), EKAW '02: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, Ontologies and the Semantic Web, Springer, London, UK, 2002, pp. 285–300.

[33] H. Su, D. Kramer, L. Chen, K. Claypool, E.A. Rundensteiner, XEM: managing the evolution of XML documents, in: K. Aberer, L. Liu (Eds.), Proceedings of the Eleventh International Workshop on Research Issues in Data Engineering, RIDE 2001, IEEE Computer Society, 2001, pp. 103–110.

[34] The Gene Ontology Consortium, The Gene Ontology project in 2008, Nucleic Acids Research 36 (2008) D440–D444.

[35] M. Tresch, M.H. Scholl, Schema transformation processors for federated objectbases, in: S.C. Moon, H. Ikeda (Eds.), Proceedings of the Third International Symposium on Database Systems for Advanced Applications, DASFAA 1993, Daejon, Korea, World Scientific Press, 1993, pp. 37–46.

[36] M. Völkel, S. Schaffert (Eds.), SemWiki2006 – From Wiki to Semantics: Proceedings of the First Workshop on Semantic Wikis, Budva, Montenegro, June 12, CEUR Workshop Proceedings, vol. 206, CEUR-WS.org, Aachen, Germany, 2006.

[37] K. Wolstencroft, P. Alper, D. Hull, C. Wroe, P. Lord, R. Stevens, C. Goble, The myGrid ontology: bioinformatics service discovery, International Journal of Bioinformatics Research and Applications 3 (3) (2007) 303–325.

[38] S.E. Wright, G. Budin (Eds.), Handbook of Terminology Management, John Benjamins, Amsterdam, 1997.

[39] W. Xing, M. Dikaiakos, R. Sakellariou, A core grid ontology for the semantic grid, in: Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006), IEEE Computer Society, 2006, pp. 178–184.