# AMSL – Managing Electronic Resources for Libraries Based on Semantic Web

Andreas Nareike[1,2], Natanael Arndt[2], Norman Radtke[1,2],
Sebastian Nuck[2], Leander Seige[1] and Thomas Riechert[3]

[1] Leipzig University Library
Beethovenstr. 6, 04107 Leipzig, Germany
{lastname}@ub.uni-leipzig.de

[2] Institute for Applied Informatics (InfAI) e.V.
Hainstraße 11, 04109 Leipzig, Germany
{lastname}@infai.org

[3] Leipzig University of Applied Science (HTWK)
Gustav-Freytag-Str. 42A, 04251 Leipzig, Germany
thomas.riechert@htwk-leipzig.de

**Abstract:** In libraries a change from physical resources to electronic resources, with new licensing models and lending processes, has taken places. The existing managing infrastructure is not yet suitable for the upcoming requirements and does not provide support for flexible and extensible data models for being future-proof. In this paper we present a system that uses the generic RDF resource management system OntoWiki for managing library resources. OntoWiki is extended by components for adapting the generic system to the given domain, e.g. by using data templates. In addition the Linked Data capability of OntoWiki is used and extended to import various metadata to enrich the managing resource. Consequently using Linked Data further enables libraries to build up a Linked Data infrastructure in the library domain.

## 1 Introduction

Electronic resources are gaining more and more importance in modern libraries. While in the past every step in the classical workflow of acquiring, cataloguing and lending was related to a physical item, nowadays libraries have to deal with new categories of media and publication formats, such as e-journals, e-books or databases. Since those resources are not limited to physical items and can be copied without loss of information, new business models have to be developed by publishers. Current examples are pay-per-view, short term loan, big deal and Patron-Driven-Acquisition (PDA) but due to new markets, publication formats and user expectations, new models will keep evolving in the future. Since libraries have to manage these new complex models on their side, we introduce a new system for managing electronic resources in libraries with semantic web technology. This involves

various kinds of information to be stored and interconnected, such as contacts, contracts, packages, agreements and licenses.

In addition to the necessity of managing these resources, our endeavour contains a need for data integration: In the past years, many information sources have evolved on the vendor market which are necessary for the electronic resource management (ERM) or can help to improve it. All this information is available from different sources like Publishers, the German National Library[1], the Journal Database[2], authority files, Integrated Library Systems (ILS), Application Programming Interfaces (API) or DBpedia [KBAL09], while exported in heterogeneous formats e.g. MARC[3], ONIX-PL[4], COUNTER[5], knowledge bases, Microsoft Access databases, RDF and SPARQL.

Following conventional component based software engineering approaches, a system managing this complexity of information resources would need one component for each class of resources. With each new requirement or change in the data structure, the components have to be adjusted or new components have to be developed. In contrast to this conventional component based software engineering approach, we want to shift the complexity from the application layer to the data model. Our method of choice is to use the Resource Description Framework (RDF) [BGM14] and Linked Data [BL09] to specify this data model. Ideally, this means that the software engineering approach can be agnostic regarding the underlying data model. In practice, there will be some additional domain-specific components to further increase the usability in a library context. This design results in an extensible and adaptable software system which is able to manage all information within the life-cycle of an electronic resource.

## 2 Requirements for Electronic Resource Management in Libraries

Our electronic resource management solution for library data can be structured in two main aspects: (1) managing electronic resources and (2) enriching the electronic resources by aggregating and integrating bibliographic and business metadata from different sources. The aspect of managing electronic resources (1) is mainly done by creating, editing, linking and exporting electronic resources and their data, like license terms and contracts or the vendors and their contact data. Since we manage a lot of data, exporting and reporting is also a very important feature as providing a change history is one, too. Lastly, we want to provide the possibility to quickly address any resource in the system without the need to specify search fields, much like one would use a full text search. To achieve aspect (2), we further need to be able to import all that heterogeneous data into our data model to enrich existing electronic resources. External data will often differ in the file format as well as in the chosen model. Due to this, we also need tools to convert and transform data from different sources. Besides these two main requirements there are other non-functional

[1] Deutsche Nationalbibliothek (DNB): `http://portal.dnb.de`
[2] Zeitschriftendatenbank (ZDB): `http://www.zeitschriftendatenbank.de`
[3] MARC standards, The Library of Congress: `http://www.loc.gov/marc/`
[4] ONIX for Publications Licenses (ONIX-PL): `http://www.editeur.org/21/ONIX-PL/`
[5] National Information Standards Organization (NISO): `http://www.niso.org/schemas/sushi/`

requirements our solution should fulfil. The data model should be adaptable to offer the possibility to react to changes in the vendor market. And of course other libraries should be able to use the ERM or participate collaboratively in an existing instance, so they can interchange metadata and build up a Linked Open Data (LOD) infrastructure within the library domain.

## 3 State of the Art

Existing solutions which are currently used in libraries and are relevant for our endeavour can be distinguished in systems for managing print resources, electronic resources or more generic any kind of resources and tools for aggregation and integration of relevant data sources.

Traditionally, libraries have been primarily concerned with managing print resources. When it comes to electronic resources like e-books or e-journals, the situation is significantly changed. Not only are there different licensing models with accompanying complex access models, the contract terms and subscriptions terms often change with every new business year. Although Integrated Library Systems for managing print resources (e.g. LIBERO[6] used at the Leipzig University Library) are already available and in use in libraries, they are not prepared to be used for the management of electronic resources.

Recently *Next Generation Library Systems* (or *Unified Resource Management Systems*) which constitute a new approach to Electronic Resource Management have emerged.While those management systems imply a modern multitier architecture, there are also commercially driven cloud-based systems that come as Software-as-a-Service. Due to the commercial nature, these systems are not transparent and it is difficult to evaluate how well standards are implemented and which requirements have been incorporated [JAC+04, JAA+12]. Also these products come with their own commercial knowledge bases, which is an important part of the licensing model which makes it difficult to integrate external knowledge bases. Furthermore, these products have in common that the underlying data models are not sufficiently generic (as required in section 2) and can't be extended, as the structure needs to be scaled to fit new structural informational requirements.

A more general approach for managing electronic library resources is to use generic resource or document management systems e.g. specially adapted Wikis. OntoWiki [HDMA09, FMT+14] is a semantic Wiki system for managing resources of the Web of Data using RDF and OWL. OntoWiki is highly extensible and serves as an application development framework for knowledge intensive applications [FMT+14]. Besides the core functionality of organising RDF resources in classes, creating, querying and editing the RDF resources, it provides an application programming interface for third party extensions. The Erfurt-Framework is the abstraction layer, which is used to provide access to the underlying triple store for the high-level application. With the Linked Data Server and Linked Data Wrapper components OntoWiki can be used for publishing and consuming Data on the Linked Data Cloud resp. Web of Data.

[6] LIBERO – Library Management System: `http://www.libero.com.au/`

## 4 Using OntoWiki and RDF for Electronic Resource Management

One approach of building a highly agile system for managing electronic resources while being flexible to react to changing requirements to the data model is to implement a service oriented architecture (SOA) with an ecosystem of components which can be combined and extended as needed. In contrast to setups of SOAs in producing industries, the library electronic resource management doesn't consist of highly complex business processes with many combined tasks. Instead, we have a low complexity in the workflow management (e.g. simple processes like recording a new license) but a rather high complexity in the data model of interconnected resources. Implementing the system by developing individual customised components for each single step in the workflow is very expensive since each of them has to be adjusted, once new requirements arise for the data model. Our approach is to use a generic software framework for working on an highly expressive data model expressed in RDF and OWL. As a consequence, new requirements only require modification in the data model while the generic software can be used unchanged.

The implementation of AMSL[7] based on the OntoWiki Application Framework [HDMA09, FMT+14] with Erfurt Framework to gain access to our knowledge base and data model.[8] With its generic edit, query and visualisation components the data is accessible for collaboratively working on one instance and multiple instances via Linked Data. The edit and visualisation functions are adapted for domain experts by using a newly introduced flexible way of expressing data templates in RDF. External data is imported into the knowledge base using the generic Linked Data components and specialised adaptors. For faster access to the complex knowledge base, an adaptor Elasticsearch index is maintained.

### 4.1 The Data Model

A great amount of our data is imported from external knowledge bases. This is mostly bibliographic data about titles and publishers, scientific fields of research, etc. There is usually little or no choice regarding the vocabularies or the data model and we import this data 'as is'. To conveniently use this data, we will create some alignments for these vocabularies.

For business data about license types and conditions, information about contact persons and information about the budget, we have to create resources by our own. Our main focus is on using common vocabularies and vocabularies that have been developed by the library community.

For bibliographic data we use ontologies of the Dublin Core Metadata Initiative[9] and the Bibliographic Ontology Specification[10] (BIBO). For concepts we could not find in existing

[7] AMSL project: `http://amsl.technology/`

[8] AMSL project code repository: `https://github.com/AKSW/OntoWiki/tree/deployment/erm-hd`

[9] DCMI Metadata Terms: `http://purl.org/dc/terms/` and `http://purl.org/dc/elements/1.1/`

[10] Bibliographic Ontology Specification: `http://bibliontology.com/`

vocabularies mentioned before, we developed the vocabulary BIBRM[11] that is aligned to the ideas of the Electronic Resource Management Initiative (ERMI) [JAC+04]. For business data, we use the Academic Institution Internal Structure Ontology[12] (AIISO) and the Friend of a Friend vocabulary[13] (FOAF) to manage organisations, contacts and academic institutions.

### 4.2 Knowledge Base Concept for Collaborative Work in Library Consortia

OntoWiki is able to manage a large number of Knowledge Bases stored in the triple store. In addition, it provides user management that grants read and/or write access for each knowledge base. To meet the requirements for data aggregation and data enrichment described in section 2, we developed a knowledge base management concept.

We differentiate our knowledge bases into three kinds according to their role. Knowledge bases belonging to category (A) provide additional information for ERM and might contain complete dumps. Category (A) also includes knowledge bases providing data about faculties, organisations and contacts which was created mostly by hand. Knowledge bases of type (B) can be considered as stores for auxiliary data. They provide triples linking one resource with another, e.g. an ISSN with a ZDB resource. A knowledge base of this type could be obtained by exporting and transforming data from an Integrated Library System. Auxiliary knowledge bases are usually hidden even for administrators and are not edited within OntoWiki.

The types (A) and (B) both have in common that they will help to connect a high number of resources to their metadata counterparts right from the start, so resolving ISSNs and fetching resources from the ZDB linked data endpoint is only necessary if something is missing.

The final type (C) of knowledge bases are the ones used as working model and contain data that is mostly relevant to only one library. All the business data about contracts, licenses, prices etc. will be stored and edited there. To achieve multitenancy capability for the whole system there is one knowledge base of type (C) for each tenant (project partner).

To use the triples of knowledge bases belonging to (A) and (B) in the working models of (C), OntoWiki supports the OWL property `owl:imports`. As a result, triples of one knowledge base will be accessible in another knowledge base, as if they were imported. This enables the users to use the linking triples and access the metadata of e-journals with a few clicks. The advantage of using `owl:imports` is that the triples of a knowledge base can be used in many different knowledge bases without actually duplicating triples. Furthermore, having a separate knowledge base facilitates updating or even exchanging the knowledge base.

Data stored in (A) and (B) is shared knowledge of all libraries, therefore user privileges of OntoWiki will be set to allow read and write access for every contributor for these

[11] Vocabulary for Library ERM: `http://vocab.ub.uni-leipzig.de/bibrm/`
[12] Academic Institution Internal Structure Ontology: `http://vocab.org/aiiso/schema`
[13] FOAF Vocabulary Specification 0.99: `http://xmlns.com/foaf/0.1/`

knowledge bases. Since business data is more sensitive, the models of type (C) will only be read- and writable by users of that institution.

### 4.3 Templates for System Librarians

Initially, we were looking for a way to formally document in which way we combine different vocabularies to describe our data. Vocabularies are often not very precise in their constraints with the intention to be as widely usable as possible. The Dublin Core property `dc:creator` does not specify the kind of the creator (author, composer, painter, etc.) and also does not link the creator to other vocabularies (e.g. FOAF). To build a consistent database, it is not feasible to choose a semantic for each new date, i.e. to decide and write down for each new creator, that it also is a person as defined by the FOAF vocabulary. Especially for people with little or no experience in RDF, it is also easier to just fill out a form without having to worry too much about the vocabularies. On the most non-technical oriented level, we wanted to provide such forms that can be used on a daily basis. However, it is worth noting that forms, by 'shaping' the data, actually implement the underlying logic of an application or the application profile. On a more general level, when the application logic is sufficiently clear and machine-readable, forms are just one expression of this logic. An application profile should also

- filter imported data and possibly even apply transformation rules
- express the used ontology, without having to look at the data itself
- define/build an abstract API or SPARQL queries [TWSWG13] to access the stored data in a specific way

As we stated above, a requirement for an ERM system is the ability to quickly adapt to changes in the general framework without having to restructure the whole data store. Consequently, we store the template definitions in the triple store which is editable by the stakeholders. Our core idea is to define templates which bind a class and list properties that an instance of this class should use. The terms 'class' and 'property' here refer to the RDF/OWL constructs e.g. `rdfs:Class`, `owl:Class`, `rdf:Property`, `owl:DatatypeProperty` or `owl:ObjectProperty`. More specifically, listing 1 shows an example template.

```
@prefix bibrm: <http://vocab.ub.uni-leipzig.de/bibrm/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix bibo: <http://purl.org/ontology/bibo/> .
@prefix rdavocab: <http://rdvocab.info/> .

<http://vocab.ub.uni-leipzig/bibrm/Template/Periodicals> a bibrm:Template ;
    bibrm:bindsClass bibo:Periodical ;
    bibrm:providesProperty dc:title, bibo:issn, rdavocab:placeOfPublication .
```

Listing 1: An example template for a periodical

This template states how the class `bibo:Periodical` is used in the context of the application. Note that `bibo:Periodial` is a class of an external namespace, so we should not modify this class resource directly. However, a subclass of `bibo:Periodical` could be defined by using `rdfs:subClassOf`. This is a somewhat common approach, but requires additional reasoning when it comes to interlinking on the Web of Data. Occasionally, the use of subclasses might even be seen as ontology hijacking.

```
<http://ld.zdb-services.de/resource/2206346-8> a bibo:Periodical ;
    dc:title "Nature physics" ;
    bibo:issn "1745-2481" ;
    rdavocab:placeOfPublication "Basingstoke" .
```

Listing 2: A resource derived from the template in listing 1

The simple resource that is modelled after this template is shown in listing 2. It's left open, if the properties listed in the template are optional or mandatory, but we also provide properties that have a stricter semantic. Right now we use templates to describe our resources which can be seen as a formalised cookbook.

Another core usage are forms that simplify the creation of new resources. These forms are directly derived from a template. For instance, when creating a new resource of type `bibo:Periodical`, the template in listing 1 would generate a form that has input fields for `dc:title`, `bibo:issn` and `rdavocab:placeOfPublication`.

Where available, a suitable widget will be used. For instance, in the case of the publication place, an URI is often preferred to a simple string, e.g. instead of 'Heidelberg', `http://dbpedia.org/resource/Heidelberg` is more usable. Here we can provide a widget that uses auto-completion based on labels for URIs, so typing 'Heidelberg' will produce the a suitable resource. By using `rdfs:range`, we can further specify where OntoWiki will look for resources for auto-completion.

When importing external resource data, there might be more triples that are not really needed or wanted. For example, when resolving ZDB resource in listing 2, we will actually get more triples. By evaluating our template for `bibo:Periodical`, we can suppress unwanted triples when displaying a resource.

### 4.4 Periodical Import Process of License Data

Importing lists of subscribed journals is one of the main methods to solve requirement aspect (2) described in section 2. While the vendors will send their contract details and subscription lists in several formats and data structures to their library partners, we had to figure out which data is the most important for our data model. In dialogue with the librarians of Leipzig University Library, we learned that the import data is composed of both the electronic and the print version of International Standard Serial Number (ISSN), the title information and (optionally) the price of the e-journals. Librarians manually create CSV files which are structured, using the four columns, as mentioned above. This file than can be imported into OntoWiki which will create RDF resources accordingly. On the

import form, the user selects if the journals are part of a license contract or license package and writes a comment describing the contract/package. Each line of the import file will be accepted if there is at least one ISSN given or otherwise be dropped and not further processed. A line of the CSV file results in a new contract item resource (URI generation with hash of timestamp and the first found ISSN) and a link from the contract/package resource to that newly created resource. In listing 3 you can see an example for one CSV line and the resulting triples in RDF in listing 4 after the import process.

```
1234-5678;2345-6789;"An example";990,90
```

Listing 3: Example for one line of the CSV file

```
@prefix bibrm: <http://vocab.ub.uni-leipzig.de/bibrm/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.com/> .

ex:contract1 a bibrm:LicenseContract ;
    bibrm:hasItem ex:1 .

ex:1 a bibrm:LicenseContract ;
    dc:title "An example" ;
    bibrm:PISSN <urn:ISSN:1234-5678> ;
    bibrm:EISSN <urn:ISSN:2345-6789> ;
    bibrm:price "990,90" .
```

Listing 4: Resources created when importing the CSV from listing 3

### 4.5 Resolving ISSN as Identifier for Resources

For electronic journals, the ISSN is the main identifier for collecting and aggregating metadata. The ZDB provides a REST-API that will return RDF result sets for a given ISSN. In AMSL we developed an ISSN resolver that fetches the result set for an ISSN, extracts the important triples that link to the resources provided by the ZDB Linked Open Data service and creates an RDF (cf. listing 5) which is then imported into OntoWiki using the provided Linked Data Wrapper component. In our case, every resource with a URI starting with `urn:ISSN` followed by a valid ISSN will be resolved by the ISSN resolver to import the links to the ZDB resources into our data model. Linking these two kinds of resources helps to meet requirement aspect (2) (cf. section 2).

```
@prefix umbel: <http://umbel.org/umbel#> .

<urn:ISSN:1234-5678> umbel:isLike
      <http://ld.zdb-services.de/data/2754934-3> ,
      <http://ld.zdb-services.de/data/2662590-8> ,
      <http://ld.zdb-services.de/data/2089379-6> .
```

Listing 5: Example of linking a ISSN to resources of ZDB

### 4.6 Recording Changesets for each Resource

Every transaction done by OntoWiki is monitored by the built-in versioning mechanism. Hence, every addition and removal of statements is stored in a separate database. For reasons of comprehensibility, additions and removals relating to the same resource are consolidated into one single change statement event. As a result, every resource builds up a change log which provides an overview about the changes related to the resource. Since these changes are chronologically ordered, the built-in rollback feature allows to restore a resource to an earlier state. In addition to simple information about changes regarding a resource, the versioning also provides aggregated information which can be used to e.g. query recently updated resources by user or a list of users that edited one specific resource.

The versioning information is stored within a separate SQL table in the underlying store and thus is not accessible via SPARQL. The underlying data structure is related to the changeset vocabulary[14], as it already stores all of the required information which can be used to export the data as a knowledge base using the changeset vocabulary.

### 4.7 Using Search Engines for Faster Access to Resources

The full-text search extension replaces the SPARQL-based search function with a scalable, near real-time search powered by Elasticsearch[15]. To achieve this, the full-text search extension makes use of a class based index structure. Hence, all triples of one class are put into one index. The search is triggered via an autocomplete function that suggests results as the user types a search term into an permanently accessible search bar. The matched results are then displayed directly as result of the search. If no results have been found, the user will be redirected to a more detailed search result page which simultaneously triggers a fuzzy search. This makes the search function more robust against typing errors and provides the possibility of restricting the result set to the previously defined classes.

## 5 Conclusions and Prospect

Within this paper we presented the use case electronic resource management in libraries and how it can be handled by using RDF resources and managing them with the generic knowledge engineering system OntoWiki. The OntoWiki application framework is open source. For the AMSL project, OntoWiki was extended by components which make it easier to adapt the complete system to the library domain e.g. by using the presented data model (section 4.1) in combination with data templates as presented in section 4.3. Due to the presented knowledge base concept in section 4.2, the complete system is multitenancy capable and can be used collaboratively amongst a library consortium. Using RDF

[14] Changeset Vocabulary: `http://purl.org/vocab/changeset/schema#`
[15] Elasticsearch: `http://www.elasticsearch.org/`

and Linked Data for the resource management and metadata management in libraries can further lead to a Linked Open Data infrastructure for library metadata exchange.

## 6 Acknowledgments

We want to thank our colleagues Lydia Unterdörfel, Carsten Krahl and Björn Muschal from Leipzig University Library, Jens Mittelbach from Saxon State and University Library Dresden (SLUB) and our fellows from Agile Knowledge Engineering and Semantic Web (AKSW) research group for their help. This work was supported by the European Union and Free State of Saxony by a grant from the European Regional Development Fund (ERDF) for the project number 100151134 (SAB index).

## References

[BGM14] Dan Brickley, R.V. Guha, and Brian McBride. RDF Schema 1.1. W3C Recommendation, W3C, February 2014. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[BL09] Tim Berners-Lee. Linked Data. Design issues, W3C, June 2009. http://www.w3.org/DesignIssues/LinkedData.html.

[FMT+14] Philipp Frischmuth, Michael Martin, Sebastian Tramp, Thomas Riechert, and Sören Auer. OntoWiki - An Authoring, Publication and Visualization Interface for the Data Web. *Semantic Web Journal*, 2014. to appear.

[HDMA09] Norman Heino, Sebastian Dietzold, Michael Martin, and Sören Auer. Developing Semantic Web Applications with the OntoWiki Framework. In Tassilo Pellegrini, Sören Auer, Klaus Tochtermann, and Sebastian Schaffert, editors, *Networked Knowledge - Networked Media*, volume 221 of *Studies in Computational Intelligence*, pages 61–77. Springer, Berlin / Heidelberg, 2009.

[JAA+12] Tim Jewell, Jeff Aipperspach, Ivy Anderson, Deberah England, Rafal Kasprowski, Bob McQuillan, Tim McGeary, and Angela Riggio. Making Good on the Promise of ERM: A Standards and Best Practices Discussion Paper. Technical report, NISO ERM Data Standards and Best Practives Review Steering Committee, One North Charles Street, Suite 1905, Baltimore, MD 21201, January 2012. http://www.niso.org/apps/group_public/document.php?document_id=7946&wg_abbrev=ermreview.

[JAC+04] Timothy D. Jewell, Ivy Anderson, Adam Chandler, Sharon E. Farb, Kimberly Parker, Angela Riggio, and Nathan D. M. Robertson. Electronic Resource Management – Report of the DLF ERM Initiative. Technical report, Digital Library Federation, Washington, D.C., 2004. http://old.diglib.org/pubs/dlf102/.

[KBAL09] Georgi Kobilarov, Christian Bizer, Sören Auer, and Jens Lehmann. DBpedia - A Linked Data Hub and Data Source for Web Applications and Enterprises. In *Proceedings of Developers Track of 18th International World Wide Web Conference (WWW 2009), April 20th-24th, Madrid, Spain*, April 2009.

[TWSWG13] The-W3C-SPARQL-Working-Group. SPARQL 1.1 Overview. W3C Recommendation, W3C, March 2013. http://www.w3.org/TR/sparql11-overview/.