Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2014) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

CRIS 2014

Providing an application-specific interface over a CERIF back-end: challenges and solutions

Dragan Ivanović[a], Nikos Houssos[b]*

[a]*University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia*
[b]*National Documentation Centre, National Hellenic Research Foundation, Athens, Greece*

Abstract

This paper presents a case of presenting information modelled in CERIF through an application-specific programming interface which does not require CERIF expertise by the developers. The CERIF data model is semantically rich and can be used for detailed description of entities of scientific / research activity. A sophisticated application interface is required to exploit the full range of CERIF capabilities. On the other hand, it is obvious that data about scientific-research entities are utilised by users and software developers with various level of familiarity with the CERIF model. Because of the diversity of CRIS systems' users' knowledge of CERIF, in certain cases it is useful to create an additional application interface with elements of some simple model which can be easily understood and used by users with low level or without any knowledge of CERIF. The article presents the design and implementation of a wrapper that enables bidirectional conversion of entered data using a simple model application interface to a CERIF back-end and the use of the wrapper in the ENGAGE project as part of an open infrastructure for Public Sector Information.

Peer-review under responsibility of euroCRIS.

Keywords: CERIF data model; wrapper; JPA; ENGAGE; REST API

1. Introduction

Information about scientific / research activity can be found in various systems such as: research information systems (CRIS), institutional repositories, digital libraries, electronic journals. Interoperability of such systems in

* Corresponding author. Tel.: +0-000-000-0000 ; fax: +0-000-000-0000 .
E-mail address: nhoussos@ekt.gr

Peer-review under responsibility of euroCRIS.

Europe and beyond is an important enabler for better scholarly communication services to scientific communities. During the two last decades, the CERIF model for information systems scientific-research activity has been developed and has evolved into model with rich expressive capabilities which enables semantically accurate, detailed and flexible description of the scientific/research activity domain and facilitates interoperability. Development of CRIS systems based on the CERIF model is rapidly increasing in recent years.
The CERIF model might take considerable effort to learn for newcomers. In some CRIS systems, data about scientific research entities should be entered and utilised by researchers with low level or without any knowledge of CERIF model. CERIF/CRIS systems can also be utilised as a back-end of web applications that provide simplified user interfaces, easily comprehensible by users not familiar with CERIF. Furthermore, and more importantly, CERIF/CRIS systems can provide –via suitable APIs -research information as open data and third parties such as enterprises, public organisations and developer communities, possibly outside the research domain, might find this data suitable for reuse in their applications, systems and end–user services. An application-specific, simplified, CERIF-agnostic API might be more attractive to developers in such a case. This is despite the fact that indeed the use of a standardised, full-fledged programmatic interface over a CERIF database would be ideal in most occasions, since it provides maximum flexibility and reuse potential; such an interface, based on REST technology, is being currently developed within the CRIS Architecture and Development Task Group of euroCRIS. However, in certain situations, the ability to provide a less generic, application-specific and developer-friendly, simpler API, can extend the reach of CERIF-based systems and their information to wider communities and more use cases. This is an approach analogous to what has been identified, mainly in the field of information representation for cultural heritage material, as an important pattern of maintaining a balance between the need for sophisticated data representation standards and simple, application-specific graphics user interfaces (e.g. data entry forms) for end users[7]. Essentially, the approach is to maintain the information in an advanced, expressive back-end model and provide on top of it various customised user interfaces that are intuitive and easy to utilise for users. In an analogous manner, our solution extends this approach from graphical user interfaces to programming interfaces and provides a simple, application-specific API over the generic and flexible CERIF data model. Likewise, a "API pidgin" is proposed in the current article, as a way to create a programming interface that is easy to learn for developers of any data modelling background, while maintaining in the back-end information the needed modelling sophistication and expressiveness, i.e. avoiding the "metadata pidgin" approach[8] that leads to loss of information / semantics.

CRIS systems should be clearly based on the CERIF data model, an established standard able to appropriately represent the research domain. In addition to that, given the diversity of CERIF expertise among the audience of CRIS systems, it can be sometimes important for certain CERIF-based systems to support two kinds of application programming interfaces:

- An interface with elements of CERIF model entities for users knowledgeable with respect to CERIF. A standardised programmatic interface for CERIF systems, based on REST technology, is being currently developed within euroCRIS.
- An interface with elements of some simple model which can be easily understood and used by users without any knowledge of CERIF model concepts. This can be useful in cases, for example, where the information in a CERIF-based system is provided as open data, which can be useful to parties or communities (e.g. open source developers) that are not aware of the research domain and CERIF.

The paper presents an approach for building a CERIF model wrapper which enables conversion of entered data using simple model application interface to CERIF data base and vice versa. The wrapper is being utilised in the ENGAGE FP7 project.

2. Related work

There are a few examples of conversion of some data model entities to the CERIF data model entities. Math-Net International information and communication system data model is one such example (http://www.math-net.de/). Mapping between the Math-Net and CERIF data model is described in CERIF TaskGroup[1]. Common European format for Curricula Vitae (http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2002:079:0066:0072:EN:PDF) describes a person and her or his

personal qualifications mostly as it is defined by the CERIF data model[1]. Conversion of MARC 21 bibliographic format to the *cfResPub* entity of CERIF is described in the paper *CERIF compatible data model based on MARC 21 format*[2]. Furthermore, conversion of metadata about PhD dissertations expressed in Dublin Core and ETD-MS format to the CERIF data model is presented in the paper *A data model of theses and dissertations compatible with CERIF, Dublin Core and ETD-MS*[3].

Also, there are a few generic tools for describing and execution of data conversion between various models. One such a tool is YAT[4]. This tool defines special declarative, rule-based language for describing conversions called YATL. Also, there is an open-source tool called biblio-transformation-engine (https://code.google.com/p/biblio-transformation-engine/). The tool is a generic framework for implementing data transformation workflows for the needs of digital libraries, repositories and archives[5]. It enables the modularisation of transformation code and promotes reuse of individual components of a transformation and separation of concerns for developers. Moreover, it is included in the core distribution of the DSpace platform since version 3.0.

This paper provides a specific conversion tool which can convert some non-standardize data to the CERIF standardized data model.

3. CERIF entities and implementation aspects of a CERIF back-end

The CERIF standard describes a formal data model that enables interoperability between research management systems and this model contains information about people, projects, organisations, publications, patents, events, prizes, equipments[6]. The CERIF entities are divided into the groups. The core group contains *Project*, *Person* and *Organization* entities. Entities *ResultPublication*, *ResultaPatent* and *ResultProduct* belong to the result entities group and these entities contain metadata about scientific research results. There are also CERIF 2nd entities which can be linked with CERIF core and result entities. There is also a semantic layer which enables classification of entities and relations between entities according to some classification scheme. Other entities of the CERIF data model are linked with semantic layer through the *cfClass* entity. For the needs of mappings ENGAGE project data model entities the the *cfClass* entity is used for classification of scientific domains and categories of data sets.

The mostly used data model for data bases is relational data model. CERIF data are usually stored in some relational database management system, such as: MySQL, PostgreeSQL, Microsoft SLQ, etc. On the other hand, the mostly used programming language model is object-oriented. CRIS systems and other systems based on CERIF data model are usually implemented using some object-oriented programming language, such as Java, Python, C#. There is a bias between relation data models and object-oriented programming languages which can be bridged using some ORM tool such as Hibernate, TopLink, etc. Moreover, there is a standardized API for this purpose for the Java applications called JPA (Java persistence API). This API is just a wrapper around some ORM tool which implements this API. This enables writing of Java code which is ORM vendor-agnostic. So, Java code invokes some methods throw the standardized JPA, and real providers of data (some ORM tool) can be changed without impact on the rest of Java code.

The CERIF JPA library is Java library which enables programmatic access to CERIF databases using the Java Persistence API (https://code.google.com/p/cerif-jpa-persistence/). It is tested with data covering the entire CERIF standard, using both MySQL and PostgreSQL relational database management systems. The library provides easy-to-use functionality for persisting the various CERIF entities (core, result, linked etc.). Furthermore, it has open-architecture and enables easy extending for application-specific needs.

The wrapper proposed into this paper is based on the CERIF JPA library.

4. The wrapper architecture and implementation

The wrapper is implemented using Java platform, Spring framework, open source libraries written in Java including the CERIF JPA library described at the end of the previous section. The wrapper contains of the following

components (Fig. 1):

- Model – This component includes Java beans representing simple model used by simple interface (not CERIF based interface)
- Converters – There are a single converter for each simple model entity. Those classes provide mappings from simple model entities to CERIF JPA entities and vice-verse.
- Repositories – This component includes classes that provide CRUD operation over CERIF database. Those classes extend CERIF JPA library classes by implementing specific finders over CERIF entities needed for improving performance of conversion between simple model entities and CERIF model entities.
- Services – This component includes classes that provide CRUD operations for simple model entities
- REST API – This component includes classes which expose CRUD operations for simple model entities over REST API. Further details and examples on this component are provided in Section 4.1. So, these classes expose methods of Services component classes. REST API enables access from some remote system which can be implemented by any language and programming platform. This means that application interface and the wrapper can execute on different machines and that they can be implemented in different programming languages.

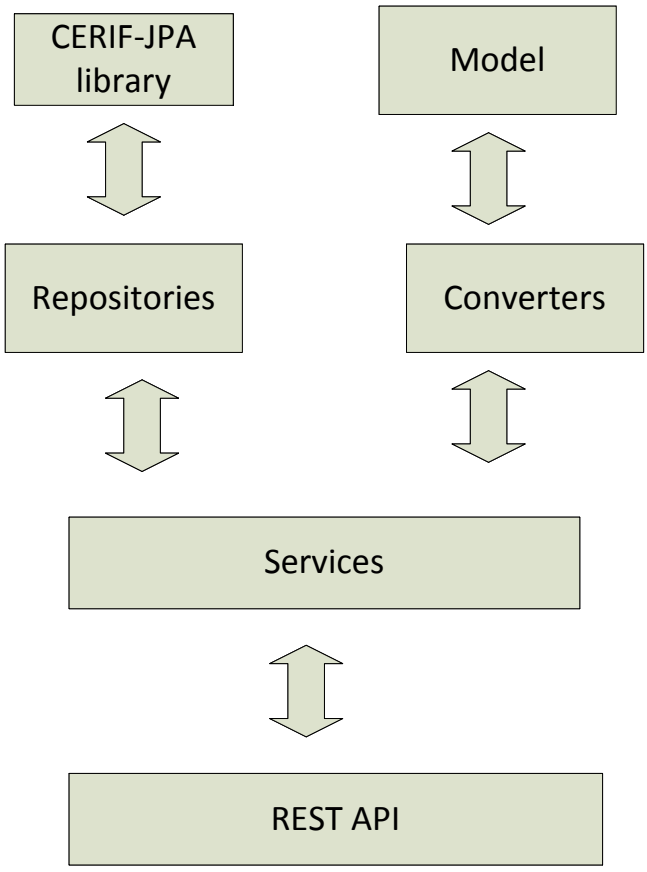

Fig. 1. The wrapper architecture

4.1. The REST interface of the CERIF back-end

The REST interface of the CERIF back-end exposes CRUD operations for simple model entities over REST API. Essentially, the public methods of Services component classes constitute the operations offered by this interface.

The REST API enables access from some remote system which can be implemented by any language and programming platform. This means that application interface and the wrapper can be on different machines or even more application interface can be implemented by different programming language than the wrapper.

The choice of a REST interface over SOAP was made mainly due to the following reasons:

- Ubiquity and popularity among developers. REST is a very common standard and is widely used in developer communities.
- Simplicity. REST is very easy to use, at least for the basic CRUD operations, using standard web technologies, in fact the HTTP protocol and its operations. Thus, REST is well supported by simple and robust tools in practically every language used for web development.
- The required interface mostly needed to cover CRUD operations over specific entities, no complex, application-specific business logic. Therefore, a REST API was inherently suitable for this purpose.

5. Case study: Using of the wrapper in the ENGAGE project

The ENGAGE project is funded under the European Commission FP7 Programme. The main goal of ENGAGE project is the deployment and use of an advanced service infrastructure, incorporating distributed and diverse public sector information resources as well as data curation, semantic annotation and visualisation tools, capable of supporting scientific collaboration and governance - related research from multi - disciplinary scientific communities, while also empowering the deployment of open governmental data towards citizens (http://www.engage-project.eu/wp/?page_id=166). Thus, ENGAGE is a door that leads researchers to the world of Open Government Data. By using the ENGAGE platform (http://www.engagedata.eu/), researchers and citizens will be able to submit, acquire, search and visualize diverse, distributed and derived Public sector datasets from all the countries of the European Union.

5.1. The component Model

The ENGAGE platform API uses a simple model, using a few entity names (Dataset, Uploader, Resource, Country), which are immediately accessible to developers. Entities Uploader and Country return data about person who uploaded the dataset and the country responsible for creation of dataset, respectively. The Resource entity holds data about a file which represents a dataset. One dataset can be represented using a few files, thus the entity Dataset is connected with the list of resources.

5.2. The component Converters

The *Converters* component contains classes which implement mappings between the model shown in Fig. 2 and CERIF data model. The Dataset entity is converted to the cfResProd CERIF entity and entities linked with this entity, the Resource entity is converted to the cfMedium CERIF entity. We used some patterns for conversions in the course of implementing the ENGAGE CERIF wrapper. This can be a list of potential mappings that can be reused to systematically construct wrapped APIs over CERIF. A few examples of patterns, as identified in the course of developing the ENGAGE CERIF wrapper, are described below:

- Class name change. Example: ResultProduct instead of cfResProd, Project instead of cfProj, OrganisationUnit instead of cfOrgUnit.
- Separate API class for a CERIF entity classification value. For example “Dataset” instead of cfResultProduct.
- Separate API class for a CERIF classification scheme. For example SoftwarePackageUsed, AnalysisUnit in the ENGAGE API.

The part of those mappings for entity Dataset is shown in Table 1. All mappings for the ENGAGE platform model to the CERIF data model can be obtained by contacting the authors of this paper via email.

Table 1. Mappings

ENGAGE	CERIF
state	ResultProduct_Class
title	ResultProductName
description	ResultProductDescription
author	Person_ResultProduct
license	ResultPublication_ResultProduct
categories	ResultProduct_Class
country	ResultProduct_Country
resources	ResultProduct_Medium

5.3. The component Repositories

We created the following classes in the repository package for extending CERIF JPA library classes by implementing specific finders over CERIF entities needed for improving performance of conversion between simple ENGAGE model entities and CERIF model entities:

- ResultProductCustomRepository
- ResultPublicationCustomRepository
- PersonCustomRepository
- MediumCustomRepository
- ClassCustomRepository

For example, ResultProductCustomRepository implements the finder findByClassId(Long classId) which returns a list of ResultProduct which are classified using provided classId. It can be used for retrieving a list of Dataset which are in certain state, firstly we retrieve a list of ResultProduct classified using certain classId which represents some Dataset state, then this list can be converted to the list of Dataset entities using component *Converters*.

5.4. The component Services

The component *Services* contains classes which implement basic CRUD operation over the ENGAGE platform model (Fig. 2). There is a separate class for each entity shown in Fig. 2. For instance, the class DatasetServices implements the following methods:

```
public Dataset findById(Integer id);
public List<Dataset> findByLanguage(String languageCode);
public Dataset findByResource(Integer resourceId);
public List<Dataset> findByUploader(Integer uploaderId);
public List<Dataset> findAll();
public List<Dataset> findAll(Integer offset, Integer limit);
public void save(Dataset dataset) throws PersistenceException;
public void delete(Integer id) throws DeleteException;
```

5.5. The component REST API

This component contains classes which export the *Services* component classes' methods over REST API. It is implemented using Spring MVC framework annotations. A short indicative fragment of the REST API for the Dataset entity is provided in the Table 2:

Table 2. REST API

Java API	REST API
public Dataset findById(Integer id);	GET /datasets/*{id}*
public List<Dataset> findByLanguage(String languageCode);	GET /datasets/languageCode/{code}
public Dataset findByResource(Integer resourceId);	GET /datasets/resourceId/{id}
public List<Dataset> findByUploader(Integer uploaderId);	GET /datasets/uploaderId/{id}
public List<Dataset> findAll();	GET /datasets/
public void save(Dataset dataset) throws PersistenceException;	POST /datasets/ (create) PUT /datasets/{id} (update)
public void delete(Integer id) throws DeleteException;	DELETE /datasets/*{id}*

6. Conclusion

It is obvious that the CERIF data model is semantically rich and that can be used for detailed description of entities of scientific-research activity. Also, it is obvious that data about scientific-research entities are entered by users with diversity of CERIF model knowledge. Because of diversity of CRIS systems' users' knowledge of the CERIF model, CRIS systems should have CERIF based data model and two kind of application interface:

- Interface with elements of CERIF model entities for users with high level of knowledge of CERIF model concepts.
- Interface with elements of some simple model which can be easily understood and used by users with low level without any knowledge of CERIF model concepts.

A CERIF model wrapper is presented in this paper. This wrapper enables conversion of entered data using simple model application interface to CERIF data base and vice versa. Also, this paper presents usage of this wrapper in the ENGAGE project. The future work on development of this wrapper should enable definition of conversions between simple model and CERIF model through XML files without implementation of classes for these conversions. So, customization of this wrapper for different usage should be as simple as possible. In this way, the presented wrapper could be used by many systems.

Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Infrastructures Programme (FP7 INFRASTRUCTURES/2007-2013) in the frame of the ENGAGE project (grant agreement number: 283700).

References

1. Jeffery, K., Lopatenko, A. and Asserson, A. (2002), "Comparative Study of Metadata for Scientific Information: The place of CERIF in CRISs and Scientific Repositories", *Proceedings of the 6th International Conference on Current Research Information Systems, University of Kassel, August 29 - 31, 2002*, pp. 77-86.
2. Ivanović, D., Surla, D. and Konjović, Z. (2011) 'CERIF compatible data model based on MARC 21 format', *The Electronic Library*, vol. 29, no. 1, pp. 52-70, DOI: 10.1108/02640471111111433.
3. Ivanović, L., Ivanović, D. and Surla, D. (2012) 'A data model of theses and dissertations compatible with CERIF, Dublin Core and EDT-MS', *Online Information Review*, vol. 36, no. 4
4. Cluet, S., Delobel, C., Siméon, J., & Smaga, K. (1998), "Your mediators need data conversion!". *ACM SIGMOD Record*, Vol. 27, No. 2, pp. 177-188.
5. Stamatis, K., Konstantinou, N., Manta, A., Paschou, C., & Houssos, N. (2012). Biblio-transformation-engine: An open source framework and use cases in the digital libraries domain. In *7th International Conference on Open Repositories, Edinburgh*.
6. Asserson, A., Jeffery, K. and Lopatenko, A. (2002), "CERIF: Past, Present and Future: An Overview", *Proceedings of the 6th International Conference on Current Research Information Systems, University of Kassel, August 29 - 31, 2002*, pp. 33-40.
7. Doerr, M. (2003). The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI magazine* 2003;**24**(3), 75.
8. Baker, T. A Grammar of Dublin Core. *D-Lib Magazine* 2000;**6**(10): 3.