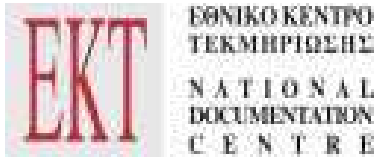
Providing an application-specific interface over a CERIF back-end: challenges and solutions

Dragan Ivanovic[a,c], Nikos Houssos[b,c]

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



12th International CRIS Conference, Rome, 13-15 May 2014

Outline

- Problem definition
- Proposed solution
- Use of the solution in the ENGAGE project
- Summary – Conclusions
- Acknowledgement: Supported by the ENGAGE project (FP7-INFRA, Ref Code: 283700)

CERIF/CRIS APIs

- APIs over CERIF/CRIS databases: a necessity
  - Enable the reuse of data by other applications
- A CERIF REST API currently being developed by euroCRIS (Architecture and Development TG)
- Ideal situation
  - Every CRIS system provides a standard CERIF API
  - Every developer client system utilises the standard CERIF REST API to interact with CRIS

The issue

- In real life diversity is the norm
  - Developers might not be familiar with CERIF
  - Vocabularies / terminology might be different across domains – developers are not familiar with CERIF terminology
  - Developers might wish to quickly reuse CERIF/CRIS data (e.g. in mash-ups) in non-CRIS systems
- A domain-specific API might be useful in certain cases (but ***not*** always / ***not*** even in most cases)

Examples (simplified)

- http://example.org/dataset/12345

  instead of

  http://example.org/cfResProd/?class=Research+data+sets+and+databases&classScheme=Output+Types&id=1234
- http://example.org/software/12345

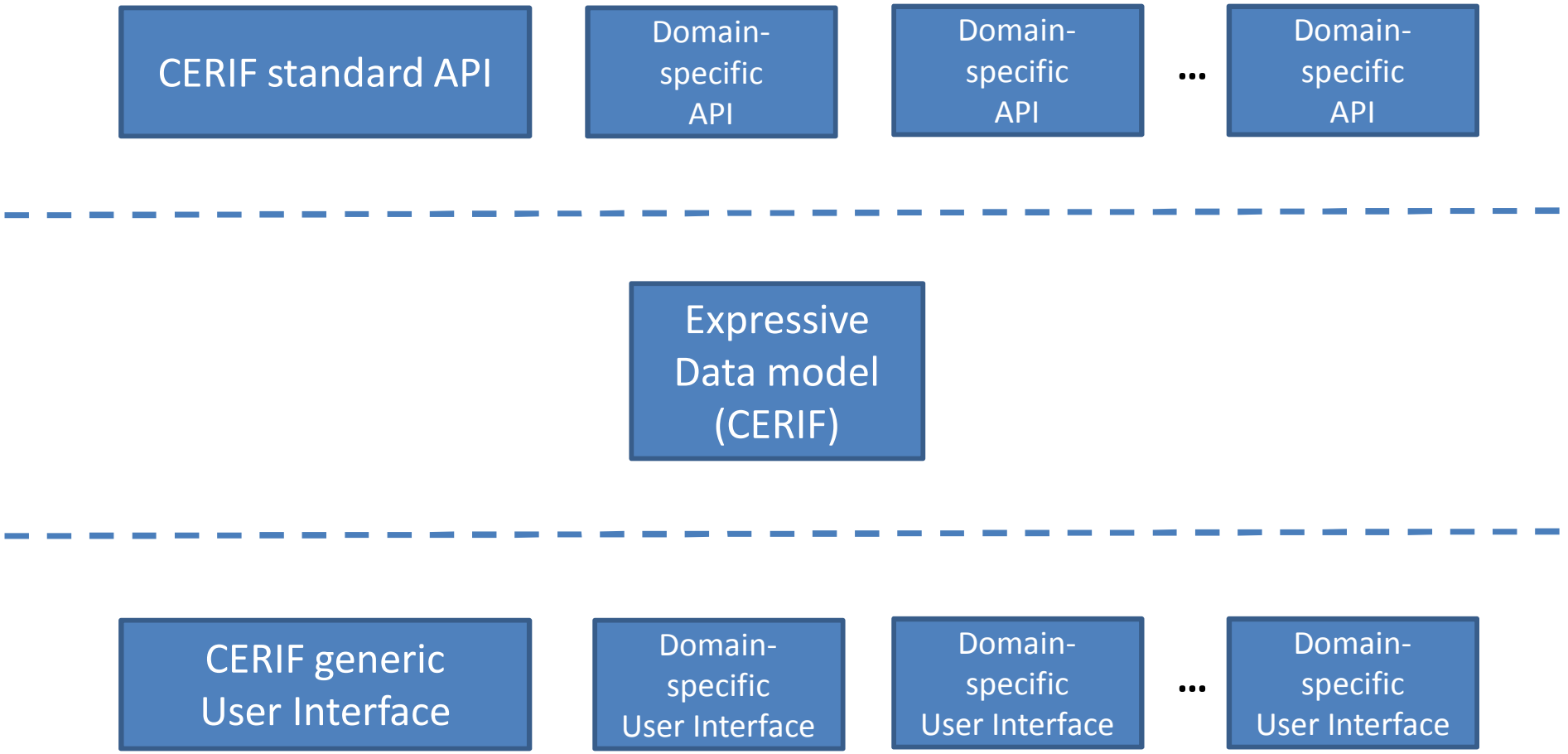
  instead of

  http://example.org/cfResProd/?class=SPSS&classScheme=Software+Package&id=12345

The main idea

- Core of an ***interoperable*** system: information expressed with a sophisticated data model – able to represent data across systems without information loss
- Needs sophisticated, flexible models like CERIF – user interfaces and API
- Sometimes certain types users might not need or wish to become familiar with the core data model
- This might include data curators, administrative personnel developers
- Simple, restricted, easy to learn user interfaces and APIs might be useful ("pidgin" approach) – ***warning***: do ***not*** use "pidgin" as the data model
- Similar approaches have been applied in cultural heritage (CIDOC-CRM) and libraries (MARC) – where domain user interfaces and APIs may be useful besides generic ones

APIs specialisation – analogy with user interfaces



The solution

- Core: a sophisticated data model
- A simpler API (besides the generic CERIF API)
- Easier to grasp and use for developers (API “pidgin”)
- Similar to the approach of creating custom, application-specific forms for data entry over sophisticated data models
- ***Word of warning***: apply with care, might lead to non-interoperable environments
- This is mostly a “necessary evil” solution, but useful in some cases

Representation of Datasets in CERIF

- Dataset-> cfResProd entity with an appropriate classification
  - For example: “Research data sets and databases” using CERIF Semantics classification (classification scheme: Output Types).
  - Application specific vocabulary (e.g. “Dataset” classification)
- Datasets usually consists of one or more files -> cfMedium entity connected with cfResProd

The ENGAGE platform

- International platform for Open Public Sector Information
- Main goal: 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.
- Platform in operation: www.engagedata.eu

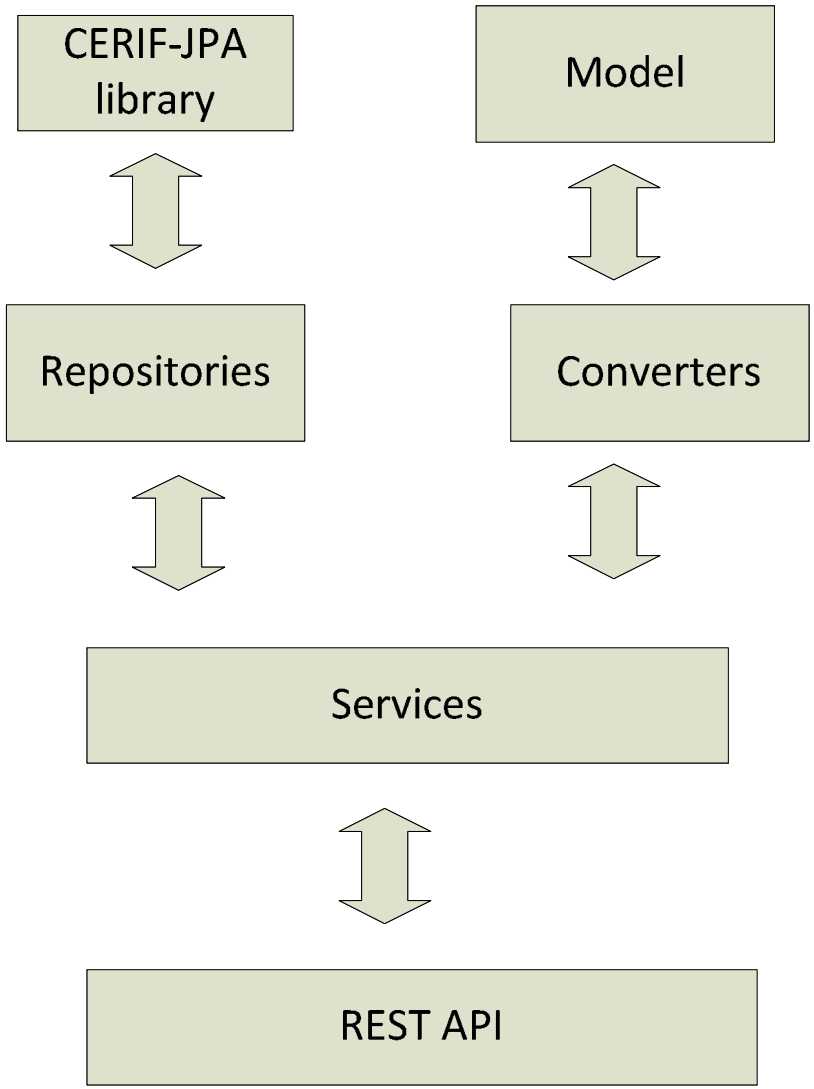
Use of the solution in ENGAGE

- CERIF is applied as the sophisticated data model in the platform back-end
- A wrapper has been developed to expose information and functionality (CRUD) according to a simpler API
- Usage of the simpler API by certain other components of the ENGAGE architecture
- Utilised also for the ENGAGE API to third parties

Implementation aspects

- A wrapper developed using open source technologies
  - Java / Spring framework
  - Java Persistence Architecture / Hibernate
  - CERIF JPA https://code.google.com/p/cerif-jpa-persistence
- RDBMS-agnostic solution - tested successfully with MySQL and PostgreSQL (PostgreSQL in the production system)
- Modularisation of code in different building blocks (e.g. data model, conversion, interface with DB)

System internal architecture



REST API

- REST chosed due to simplicity, ubiquity, familiarity of developers
- Level-2 REST API
  - According to Richardson et al. classification, 2007)
- Data exchanged in JSON
- Preserves certain CERIF features (e.g. multi-linguality)

Main conversion patterns

- 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.

  Example: Dataset instead of cfResultProduct.

- Separate API class for a CERIF classification scheme.

  Example: SoftwarePackageUsed, AnalysisUnit in the ENGAGE API.

Conclusions - Summary

- CERIF is a primary vehicle for representing research information.
- Due to its expressiveness, CERIF can generate many other formats/models.
- APIs are critical for interoperability and reuse of information in CRIS systems – ideally a standard CERIF API should be used.
- There are cases that a simpler, constrained API might be useful to facilitate and increase the utilisation of CRIS data. Such an implementation is fully feasible, as demonstrated in the ENGAGE project.
- Future work: automatic generation of converters based on templates (patterns) and XML configuration

Thank you!

- Questions / comments?