Interoperability Patterns in Digital Library Systems Federations

Paolo Manghi, Leonardo Candela, and Pasquale Pagano

Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" – CNR, Pisa - Italy
{paolo.manghi | leonardo.candela | pasquale.pagano}@isti.cnr.it

Abstract. Service providers willing to offer functionality over an aggregation of information objects are becoming the key actors in the Digital Library world. This aggregation is usually performed by collecting and processing objects from a federation of data providers – e.g., institutional repositories, data archives – through well-known standard protocols for data-exchange. Higher-level interoperability problems affect these scenarios and determine the quality of the service offered by and the success of these providers initiatives. Interoperability problems are mainly due to the impedance mismatch occurring between data models of the objects to be exported by data providers and the data model of the service provider and can be measured in terms of "structural heterogeneity", "semantic heterogeneity", and "granularity-of-representation heterogeneity". In this paper, we define a basic architecture through which we describe the interoperability patterns arising when constructing such a federations and present the D-NET Software Toolkit as a general-purpose practical solution to these issues.

1 Introduction

In the last decade, research communities increasingly adopted Digital Library Systems (DLSs) to preserve, disseminate and exchange their research outcome, from articles and books to images, videos and research data. The multidisciplinary character of modern research, combined with the urgency of having immediate access to the latest results, moved research communities towards the realization of service providers aggregating content from federations of DLSs. Service providers, which act here as *data consumers*, offer functionality over an aggregation of information objects [1] obtained by manipulating objects collected from a set of DLSs, which act here as *data providers*. Data providers expose *content resources* while the service provider is willing to consume such resources to accomplish its *tasks*.[1] In order to interoperate, the two interlocutors have to face the following challenges: (*i*) "how to exchange objects", i.e., identifying common data-exchange practices, and (*ii*) "how to harmonize objects", i.e., resolve data impedance mismatch problems arising from distinct data models assumed by the two. Typically, data access across different platforms is overcome by adopting XML as lingua-franca and standard data-exchange protocols, such

[1] By data providers we mean DLSs whose collection of objects are useful to a service provider for accomplishing its tasks. In other words, a DLS cannot be a data provider for a service provider that is not interested in its content resources, i.e., these are not useful for achieving its tasks. In this sense, being or not being interoperable is an exclusive problem of a data provider and a service provider, hence of two interlocutors willing to interact to accomplish a task together.

as OAI-PMH [3] and OAI-ORE [4]. If the adoption of these standards represents an important step towards interoperability, it still leaves open core interoperability challenges. These challenges have mainly to do with impedance mismatch issues that can be classified as:

Data model impedance mismatch – the mismatch between the service provider data model (i.e., the XML schema capturing structure and semantics of the information objects to be generated) and the data providers data model (i.e., the XML schema capturing structure and semantics of the information objects to be collected and elaborated).

Granularity impedance mismatch – the mismatch between XML encodings of information objects at the service provider and data providers sites, which may consider different levels of granularity. For example, one DIDLXML file may represent (i.e., package) a set of the information objects, together with relationships between them, namely a "compound object"; a MARCXML file, instead, typically represents the descriptive metadata of one information object.

When service providers and data providers feature different data models or granularity of representation, specific solutions to achieve interoperability must be devised and implemented.

In this paper we shall describe the interoperability patterns which generally surface when constructing DLS federations featuring data model and granularity impedance mismatch, and present the D-NET Software Toolkit, today used in several real-case scenarios, as a practical general-purpose solution to those. In particular, Section 2 introduces the terminology and defines a basic architecture for DLS federations. Section 3 depicts DLS federation interoperability issues and sketches ideas for solutions. Section 4 presents the D-NET Software Toolkit as an implementation of them. Finally, Section 5 concludes the paper.

2 Digital Library System Federations

Figure 1 shows the basic architecture of a DLS federation. Data providers manage a collection of information objects that match a given data model. In particular, information objects may be publications, audio and video material, compound objects (i.e., sets of interlinked information objects), or "surrogates" of all these, namely metadata descriptions of information objects. In general, we can assume that data providers handle a "graph" of information objects, whose structural and semantic complexity depends on the data model to which it conforms. Typically, data providers expose a "view" of their objects collection, by identifying the subset of objects to be exported and, for those, the structural aspects to be revealed to the world.

Similarly, the service provider operates a collection of information objects matching a local data model. Such a collection is obtained by bulk-fetching, manipulating and then aggregating information objects exported by the individual data providers.[2]

As highlighted in Figure 1, in the Digital Library world the basic interoperability issues occurring between a service provider and data provider to exchange information objects are typically overcome by adopting XML as lingua-franca in combination with standard data-exchange protocols (e.g., OAI-PMH). XML files have a labelled tree structure and can

[2] Other federative approaches are possible, for example adopting distributed search as interaction mechanisms, but these are out of the scope of this paper.

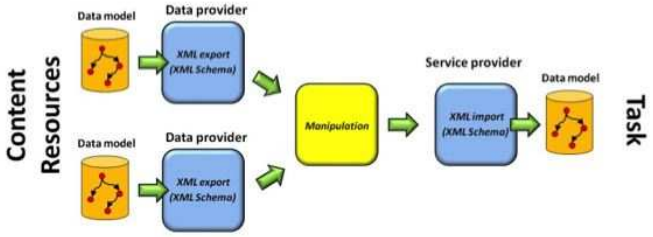

Fig. 1. DLS federation: basic architecture

therefore represent any kind of information objects; in principle XML files can also contain the payload of another file (e.g., a PDF) instead of a reference to a payload.

On the data provider side, information objects are associated with an XML schema that captures the essence of their data model and special "exporting components" are developed to generate and return the relative XML representations in response to service provider's requests. On the service provider side, a similar but inverted situation occurs. Information objects have a correspondent XML schema and a special "importing component" is constructed, capable of converting an XML file onto an information object instance.[3] Note that having the same data model does not mean to have the same XML schema representation.

3 Interoperability Patterns

Observing Figure 1, it is clear that if the XML schema and the intended semantics of the element values of all data providers and the service provider match, no interoperability issue occurs. The service provider can smoothly collect from data providers information object XML representations and directly aggregate them locally into one collection. When this happens, it is because common agreements have been established or an interoperability solution has been already realized. When this is not the case, data and service providers cannot interoperate due to *data model* or *granularity* impedance mismatches and solutions must be devised.

3.1 Data Model Impedance Mismatch

Data model mismatches occur at the level of the data provider and the service provider data models when the relative XML schema views have paths of XML elements (paths in the following) and/or the relative value domains (leaves in the following) that do not exactly match. In this case interoperability issues arise because, due to either structural or semantic heterogeneity, the service provider cannot directly aggregate the information object XML representations of the data providers. Typically, depending on the kind of mismatch, the

[3] Note that, in some cases, data provider or service provider implementations may manage their information objects directly as XML files, onto native XML databases or full-text indices.

solution consists of software components capable of overcoming such differences by applying appropriate XML file transformations (see Figure 2). Implicitly, such transformations convert information objects from one data model onto another.

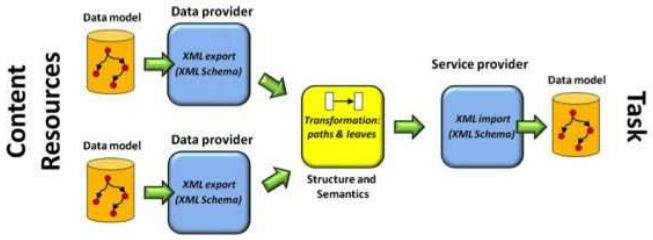

Fig. 2. Interoperability issue: data model mismatch

In particular, we can identify two kinds of mismatch, strictly related with each other:

Structural heterogeneity occurs when the XML schema of data provider and service provider are not equal, i.e., when their paths do not match. Typical examples are: the service provider paths are a subset of the data provider paths, service provider has paths that are different but correspond to data providers paths (i.e., using different element names or hierarchies to describe the same concept), service provider has paths that do not have a data provider path correspondent (i.e., service provider data model is richer than data provider's one).

Semantic heterogeneity occurs at the level of leaves, under two circumstances:

1. service provider and data provider have corresponding leaves in the relative XML schemas (i.e., the schema are equal or are not equal but a one-to-one mapping between their paths can be identified), but do not share the same formats (e.g., date/time formats, person names) and vocabularies;
2. the service provider has leaves in the XML schema that do not find a direct correspondent in the data provider XML schema; such leaves must be derived by elaborating (i.e., computing over) leaves of the data providers XML files.

Interoperability solutions to data model mismatches consist in the realization of *transformation components*, capable of converting XML files conformant to data providers schema onto XML files conforming to the service provider schema. The logic of the component maps paths in the original schema onto paths of the target schema. In principle, source and target paths may have nothing in common, either the element names or hierarchical structure of the elements. Similarly, the values of the leaves of the output XML files may completely differ from the ones in the input XML files, in domain, format and meaning.

Depending on the application scenario the implementation of transformation components may largely differ. We can identify the following cases, together with possible categories of solutions, where, in some cases, the cardinality of data providers in the federation may impact on cost and sustainability.

All data providers have the same XML schema The transformation component should generate XML files conforming to the XML schema of the service provider from the data provider XML files, whose paths are the same. To this aim all leaves of service provider XML files are generated by processing the leaves of the incoming XML files through transformation functions (F). The complexity of the F can be arbitrary: "feature extraction" functions taking a URL, downloading the file (e.g., HTML, PDF, JPG) and returning content extracted from it; "conversion" functions applying a translation from one vocabulary term to another vocabulary term; "transcoding" functions transforming a leaf from one representation format to another (e.g., date format conversions); "regular expression" functions generating one leaf from a set of leaves (e.g., generating a person name leaf by concatenating name and surname originally kept in two distinct leaves). Since only one source XML schema is involved, the component can be developed around the only one mapping necessary to identify which input leaves must be used to generate an output leaf through a given F.

Data providers have different XML schemas The transformation component should generate XML files conforming to the XML schema of the service provider assuming the incoming XML files have different paths, depending on the data provider's XML schema. In principle, the solution could be that of realizing one component as the one described for the previous scenario for each set of data providers sharing the same schema. However, if an arbitrary number of data providers is expected, possibly showing different structure and semantics and therefore requiring different transformation functions, this "from-scratch" approach is not generally sustainable. One solution is that of providing general-purpose tools, capable of managing a set of "mappings" (e.g., Repox[4], D-NET [5]). These consist in named lists (*input paths*, F, *output path*), where the output path (which may be a leaf) is obtained by applying F to the input paths. Mappings can be saved, modified or removed, and be reused while collecting XML files from data providers sharing the same structure and semantics. Similarly, the component should allow for the addition of new F to match unexpected requirements in the future.

3.2 Granularity Impedance Mismatch

In designing an interoperability solution between a data provider and a service provider, the "granularity" of the objects exported by a data provider may not coincide with that intended by the service provider. For example, data providers may export XML files that represent "compound objects", which are rooted sub-graphs of the local object collection. The service provider might be interested in the compound object as a whole, thus adopting the same granularity, or only in some of the objects that are part of it, thus adopting a finer granularity. Hence, in addition to the data model mismatch, a granularity impedance mismatch may arise.

The following scenarios typically occur (see Figure 3):

(1:N) each XML file of the data provider translates onto more XML files of the service provider, e.g., un-packaging of compound object XML representations. The solution

[4] Technical University of Lisbon, Instituto Superior Técnico Repox - A Metadata Space Manager, `http://repox.ist.utl.pt`

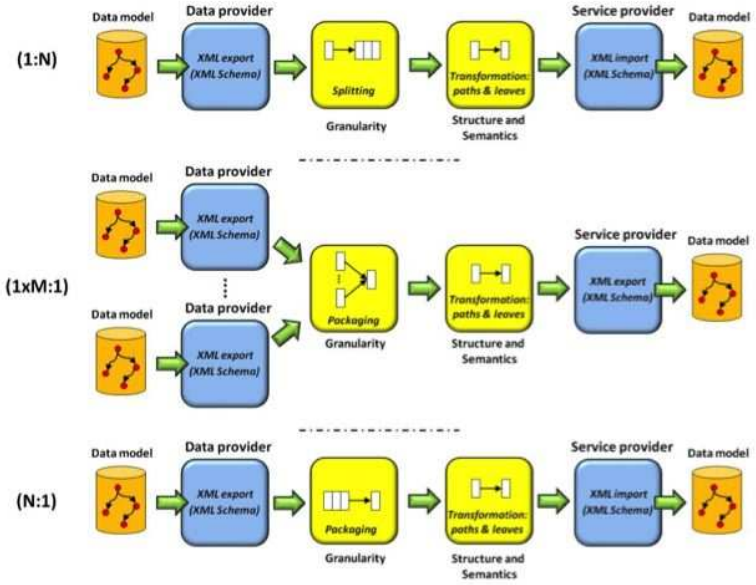

Fig. 3. Interoperability issue: granularity mismatch

requires the realization of a *splitting component*, capable of obtaining a list of XML files from each XML file exported by the data provider. The operation may occur before or after structural and semantic interoperability issues have been tackled.

(N:1) more XML files from one data provider correspond to one XML file of the service provider. The solution requires the realization of a *packaging component* capable of identifying the set of XML files exported by the data provider which have to be combined onto one XML file of the service provider. The logic of such a combination may vary across different application domains, but is often based on shared identifiers and/or external references to those.

(1xM:1) the combination of one XML file for each data providers in the federation corresponds to one XML file of the service provider. The solution is similar to the case (N:1), where the *packaging component* has to be able of managing the XML files exported by a set of data providers and identify those that have to be combined onto one XML file of the service provider.

3.3 Architecture of interoperability solutions

The reader may have noticed that no assumption has been made on which entity is in charge of the realization of the components. In fact, given an interoperability scenario like the one we described, the transformation, splitting and packaging components may be fully or partly realized by the data providers or by the service providers, depending on the level of engagement and agreements established by the federation.

"Bottom-up" federations Some federations are realized by organizations who have control over the set of participating data providers. All interoperability issues are to be solved at the data providers sites and the service provider is a simple object collector and aggregator. Although quite rare in practice, due to the difficulties of autonomous and independent organizations of respecting external guidelines, this is the case for example for DAREnet-NARCIS[5], the service provider of the research and academic institutions of the Netherlands. The relative institutional repositories agreed on exporting their bibliographic metadata records according to Dublin Core XML format and to a precise semantics of the elements (e.g., given vocabularies and formats for dates and creators).

"Open" federations Some federations are attractive to data providers, which are available to adhere to given "data model" specifications in order to join the aggregation. However, to not discourage participation, service providers define "data provider guidelines" that are often limited to the adoption of simple XML schemas and to light-weight best practices on usage of leaves. Therefore, in most of the cases, the realization of leaf transformation components is left to the service provider (e.g., the DRIVER repository infrastructure)[6].

"Community-oriented" federations A community of data providers handling the same typology of content, but in different ways, finds an agreement on a common data model and together invests on the realization of a service provider capable of enabling a federation by solving all interoperability issues that may arise (e.g., the European Film Gateway project[7]). In such scenarios, packaging, if needed, typically occurs at the service provider side, while part of the transformation may also occur at the data provider side, before XML export takes place. If this is not the case, data providers are directly involved in the definition of the paths and leaves transformation specification, while the service provider limits the intervention to the relative implementation.

"Top-down" federations Federations may be the result of the interest of a service provider to offer functionality over data providers whose content is openly reachable according to some declared XML schema (e.g., OAIster-OCLC project[8], BASE search engine[9]). In such cases, it is the service providers that has to deal with interoperability issues.

4 D-NET Software Toolkit

The D-NET Software Toolkit was designed and developed within the DRIVER and DRIVER-II EC projects and is being used and extended in EFG and OpenAIRE projects. Its software

[5] DAREnet: Digital Academic Repositories, `http://www.narcis.nl`
[6] The DRIVER Infrastructure, `http://search.driver.research-infrastructures.eu`
[7] The European Film Gateway project, `http://www.eureopanfilmgateway.eu`
[8] OAIster-OCLC project, `http://www.oclc.org/oaister/`
[9] BASE: Bielefeld Academic Search Engine, `http://www.base-search.net`

is open source, developed in Java and available for download[10]. D-NET implements a Service Oriented Architecture (SOA), based on the Web Service framework, capable of operating run-time environments where multiple organizations can share data sources and services to collaboratively construct DLS federations [5]. To this aim, D-NET provides several services with which developers can "assemble" applications for the collection and manipulation of information objects from a pool of data sources. Specifically, a D-NET infrastructure is made of two main logical layers: the system core, called enabling layer, whose function is to support the operation of the application layer, which consists of the services forming the applications.

4.1 Enabling Services

The enabling services support a framework where developers can combine services to build a running application. The ResultSet Enabling Service, for example, can create, delete, operate a set of ResultSets, i.e., "containers" for transferring lists of objects between a "provider" service and a "consumer" service. Technically, a ResultSet is an ordered list of files identified by an End Point Reference (EPR), which can be accessed by a consumer through paging mechanisms, while being fed by a provider, in order to reduce response delays and limit the objects to be transferred. D-NET services are designed to be organized into workflows by relying on ResultSet EPRS, i.e., they are conceived to accept as input parameters and return as results of their invocations ResultSet EPRs. Other enabling services offer functionalities for service orchestration and autonomicity, safe communication, and discovery [2].

4.2 Application Services

D-NET services involved in the construction of Digital Library System federations are *mediation services*, i.e., information object collection components, and *manipulation services*, i.e., splitting, packaging and transforming components. We shall see that D-NET also offers service kits for the realization of advanced service providers, capable of managing information objects of arbitrary data models. All D-NET services return and accept objects encoded as XML files through a ResultSet.

Mediation services D-NET offers *Object Collection Services* for accessing data providers exposing content through OAI-PMH interfaces (e.g., repositories, archives) or through remote file systems with folders containing XML files. Moreover, special *Data Provider Management Services* allow data provider administrators to "register" their DLSs to the federation, and D-NET administrators to automatically fetch objects from the registered DLSs through *Object Collection Services.*

Manipulation services Services in the manipulation area are capable of handling objects to transform them from one XML schema to another. Bulk transfer of XML files from a service to another is performed by sending a ResultSet EPR to the service, which will then actively pull the XML files from it for local usage. Manipulation services implement splitting, packaging and transformation components and include:

[10] D-NET Project, `www.d-net.research-infrastructures.eu`

MDStore Service An MDStore Service manages a set of *MDStore units*, each capable of storing XML files of a given XML schema. Consumers can create and delete units, and add, remove, update, fetch, get statistics on XML files from-to a given unit.

Feature Extractor Service A Feature Extractor Service generates a ResultSet of XML files from a ResultSet of input XML files by applying a given extraction algorithm to given leaves of the input file. Examples are: extracting the histograms of image payload objects; extracting full-text of PDF payload objects; generating payload objects from payload objects (DOC to PDF). Algorithms can be plugged-in as special software modules, whose invocation becomes available to consumers.

Transformator Service A Transformator Service is capable of transforming XML files of one schema into XML files of one output schema. The logic of the transformation, called mapping, is expressed in terms of a rule language offering rules for: (*i*) path removal and addition, leaf concatenation and switch, (*ii*) regular expressions over leaves, and (*iii*) invocation of an algorithm through a Feature Extractor Service on a leaf. User interfaces support administrators at defining, updating and testing a set of mappings, which they can save, refine and reuse in the future. The idea is to offer tools for systematizing the process of definition of XML mappings, traditionally consisting in programming XSLT scripts or other sophisticated and ad-hoc code. The tools capture a very common scenario, in which paths to the leaves of the input and output XML schemas can be mapped onto a flat list of labels ("flat record"). The rules are then applied to input label values in order to obtain output label values and automatically generate the corresponding XML files. For all other cases, administrators can opt for a "traditional" solution by uploading mappings defined as arbitrary complex XSLT transformations or adding new algorithms in the Feature Extractor Service, capable of processing the whole XML file as desired.

Object Packaging Service An Object Packaging Service can perform both splitting and packaging operations over XML files provided through ResultSets. Splitting is performed by applying a given xpath query to each of the XML files in an input ResultSet; the XML pieces returned by each query are returned as individual XML files through a ResultSet. Packaging is performed by accepting a set of Result Sets and generating one valid XML file that includes several XML files, one for each of the input ResultSets. The operation accesses the ResultSets incrementally, hence it is important to properly order the ResultSets to obtain the expected XML packages.

"Service provider" services D-NET offers a multitude of data management services, used by developers to build service providers capable of managing objects of various data models in the most appropriate way. These include: Store Services (for payloads of any MIME type), Index Services (for full-text indexing and faceted browsing over XML files of any schemas), Compound Object Services (for the creation of typed-graphs of metadata, payload and relationship objects), Database Services (for managing objects as relational records), OAI-PMH Publisher Services (for exporting XML files through the known protocol), Validation Services (for checking the "quality" of a ResultSet of XML files according to a set of validation rules), Search Services (for querying and browsing XML files into Index Services according to SRW/CQL interfaces).

5 Conclusions

Since 2006, the D-NET software toolkit was constantly refined and extended to accommodate the interoperability issues arising in the construction of DLS infrastructures across different application domains. In this paper we identified the interoperability patterns recurring in these applicative scenarios and schematized the reasoning behind the realization of possible solutions. Finally we presented the D-NET services that were developed to offer general-purpose tools capable of addressing and tackling the requirements of the identified patterns.

Acknowledgments Research partially supported by the INFRA-2007-1.2.1 Research Infrastructures Program of the European Commission as part of the DRIVER-II project (Grant Agreement no. 212147) and by the ICT-2007.4.3 Information and Communication Technologies Program as part of the DL.org project (Grant Agreement no. 231551).

References

1. L. Candela, D. Castelli, N. Ferro, Y. Ioannidis, G. Koutrika, C. Meghini, P. Pagano, S. Ross, D. Soergel, M. Agosti, M. Dobreva, V. Katifori, and H. Schuldt. *The DELOS Digital Library Reference Model - Foundations for Digital Libraries*. DELOS: a Network of Excellence on Digital Libraries, February 2008. ISSN 1818-8044 ISBN 2-912335-37-X.
2. L. Candela, D. Castelli, P. Manghi, and P. Pagano. Enabling Services in Knowledge Infrastructures: The DRIVER Experience. In F. E. M. Agosti and C. Thanos, editors, *Post-proceedings of the Third Italian Research Conference on Digital Library Systems (IRCDL 2007)*, pages 71–77. DELOS: a Network of Excellence on Digital Libraries, 2007.
3. C. Lagoze and H. Van de Sompel. The OAI Protocol for Metadata Harvesting. `http://www.openarchives.org/OAI/openarchivesprotocol.html`.
4. C. Lagoze, H. Van de Sompel, P. Johnston, M. Nelson, R. Sanderson, and S. Warner. Open Archives Initiative Object Reuse and Exchange (OAI-ORE) User Guide - Primer. Technical report, Open Archives Initiative, 2008.
5. P. Manghi, M. Mikulicic, L. Candela, D. Castelli, and P. Pagano. Realizing and Maintaining Aggregative Digital Library Systems: D-NET Software Toolkit and OAIster System. *D-Lib Magazine*, 16(3/4), March/April 2010.
6. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.
7. G. Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer*, 25(3):38–49, 1992.