RDFBuilder: A tool to automatically build RDF-based interfaces for MAGE-OM microarray data sources

Alberto Anguita, Luis Martin, Miguel Garcia-Remesal, Victor Maojo

This paper presents RDFBuilder, a tool that enables RDF-based access to MAGE-ML-compliant microarray databases. We have developed a system that automatically transforms the MAGE-OM model and microarray data stored in the ArrayExpress database into RDF format. Additionally, the system automatically enables a SPARQL endpoint. This allows users to execute SPARQL queries for retrieving microarray data, either from specific experiments or from more than one experiment at a time. Our system optimizes response times by caching and reusing information from previous queries. In this paper, we describe our methods for achieving this transformation. We show that our approach is complemen-tary to other existing initiatives, such as Bio2RDF, for accessing and retrieving data from the ArrayExpress database.

1. Introduction

The unprecedented growth of biomedical data sources in the last decade has changed the way biologists do their job. Whereas ten years ago they gathered and analyzed data from one or few sources – private databases were frequently managed by one institution alone – nowadays, collaborative efforts usually imply working simultaneously with several heterogeneous databases to extract new and valuable information. Different areas within biomedicine have already benefited from this distributed approach. Many current efforts focus on integrating microarray data with other types of data – for instance, clinical data – to find new genetic signatures for improved diagnosis and better treatment selection. Zirn et al. describe this process with Wilms tumor patients, obtaining genetic biomarkers that help to determine the best treatment for each patient [1]. However, these efforts are often hampered by the complexities of working with several data sources in different formats, or with different interfaces. Thus, data standardization is a major issue.

To provide unified and homogeneous access to the vast amounts of available biological data, bioinformaticians have to deal with both semantic and syntactic heterogeneities across different sources [2]. Over the last few years, semantic homogenization efforts have been commonly addressed by applying semantic web technologies [3–5]. A significant example is the development of ontologies that act as unified vocabularies for the biomedical domain [6,7]. To name a few examples, the Foundational Model of Anatomy (FMA) contains a symbolic representation of the phenotypic structure of the human body [8], the Gene Ontology (GO) offers a representation of gene and gene product attributes across species [9], and the ACGT Master Ontology represents the domain of cancer research [10]. All these ontologies are currently part of the OBO Foundry [11], a consortium created with the goal of establishing a set of principles and good practices for ontology development. In the case of syntactic heterogeneities – i.e.,

differences in database models, query languages, and interface types – the efforts focus on developing standard models and languages for describing, storing and accessing data. RDF [12] has been accepted as a *de facto* data model for this purpose in the biomedical domain, being SPARQL the most adopted associated query language. In this context, numerous approaches for "RDFizing" existing biomedical data and offering a common method for accessing the vast amounts of available data have been proposed. Examples of these approaches are YeastHub [13], an RDF structure and a web-based application for integrating both RDF datasets and data in tabular format into a centralized RDF repository, or Bio2RDF [14], a project that aims to unify and convert several existing public biomedical databases into a common RDF format.

Besides all the existing efforts to standardize data in the biomedical domain, there also exist more specific efforts targeting concrete areas and topics in the life sciences. For instance, many standards have been created to describe data in particular areas [15]. There are standards such as CellML [16], for describing cellular and subcellular processes, BioPAX [17], a data format for biological pathway data, or SBML [18], a data format for representing models of biochemical reaction networks. For microarray experiments, this domain-specific standardization effort has been primarily led by the FGED society [19]. This organization supports the development of languages, ontologies and standards for this domain – i.e., MIAME, MAGE-OM, MAGE-ML and MGED Ontology. One of these standards, the MAGE-ML language, provides a uniform manner of expressing genomic data in XML format. This language has been adopted by the most important microarray experiment public repositories, namely, ArrayExpress [20], GEO [21] and CIBEX [22]. By using it, all these databases datasets can be retrieved as XML documents in MAGE-ML form.

Developing domain-specific standards in biomedicine helps to solve problems of data interoperability and data exchange arising in biomedicine. On the other hand, developing too many standards may take us back to the starting point of having to deal with incompatible formats [23,24]. MAGE-ML is an example of this situation. This language provides a formal and solid base for ensuring interoperability among microarray-related repositories. On the other hand, data integration systems capable of accessing RDF based data sources cannot handle these data. In this regard, we propose a method for automatically providing RDF-based access to MAGE-ML data sources. The presented method is implemented by means of a programmatic interface and a web-based software system that provide RDF-based access to any database compliant with the MAGE-ML format. These software components are easily adaptable to any FGED-compliant database, apart from ArrayExpress, and can be configured in a matter of minutes. We also describe a real-case scenario with the ArrayExpress database.

2. Computational methods and theory

The goal of our system is to provide SPARQL-based access to the ArrayExpress public database. To do this, we have developed a software module that transforms ArrayExpress experiments data into a single equivalent RDF document. This is accomplished by dynamically exploring the MAGE-ML documents describing each experiment and translating their content into an RDF-based format. The resulting document is an RDF representation of the MAGE-OM object model (http://www.ebi.ac.uk/microarray-as/aer/magepages/mage_all.html). The generated RDF classes and properties effectively resemble this specification. This RDF is dynamically generated from the data structure contained in MAGE-ML documents, so the transformation algorithm incorporates no previous information about the MAGE-OM model. This is accomplished by following a set of transformation rules, described below:

- MAGE-OM classes are translated into RDF classes. For example, the *ArrayGroup* class contained in the *Array* package produces an equivalent RDF class.
- Properties relating MAGE-OM classes are translated into RDF object properties. For example, the *bioAssayFactorValues* property linking *BioAssay* and *FactorValue* in Mage-OM produces an equivalent RDF object property named http://aewrapper#bioAssayFactorValues.
- Attributes in MAGE-OM are translated into RDF datatype properties. For example, the *arrayIdentifier* attribute from the *Array* class in MAGE-OM produces an equivalent RDF datatype property named http://aewrapper#arrayIdentifier_string.

The element names are preserved whenever possible. Thus, the *ExperimentDesign* class in Mage-OM produces an RDF class with an equivalent name. MAGE-OM package names are added as prefixes to their respective class names, and the http://aewrapper# namespace has been selected for the generated models. Thus, the above MAGE-OM class produces a class in the RDF model named http://aewrapper#Experiment.ExperimentDesign.

The described element generation rules are not enough to effectively construct the equivalent RDF model. The MAGE-OM specification contains some incompatibilities with RDF due to the possibility of name-sharing attributes to coexist – e.g. *length* as an integer attribute in *BioSequence*, or as a float attribute in *ArrayGroup*. In order to adhere to the RDF specification, a disambiguation process is carried out. Generated datatype properties include an additional suffix corresponding to the actual domain datatype. Therefore, *value* (the string attribute in the class *NodeValue*) is translated into the RDF datatype property named http://aewrapper#value_string.

3. System description

3.1. *System overview*

RDFBuilder is designed to provide access to the ArrayExpress microarray database through an RDF-based interface using the SPARQL query language. To this end, the system automatically translates the MAGE-ML schema of the required experiment or set of experiments to an equivalent RDF model – MAGE-ML entities are translated into RDF according to the method described in the previous section. The resulting RDF model is queried with the SPARQL query issued by the user, and

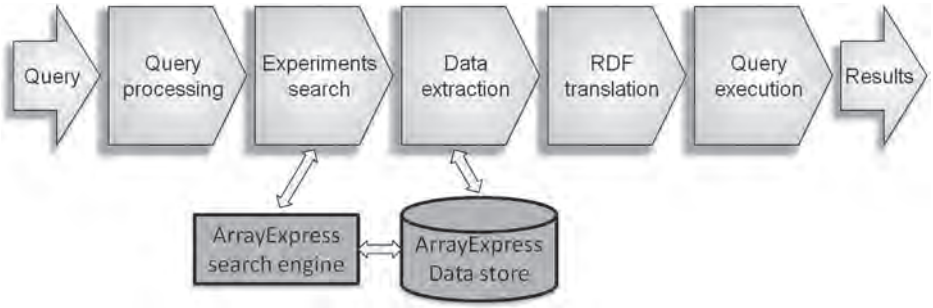

Fig. 1 – Description of the query and data translation process carried out by RDFBuilder, including communication with ArrayExpress.

results are returned in SPARQL result format. The generated RDF instances are, in addition, stored in a cache repository, allowing the speed-up of subsequent data requests. The system can be accessed in two different ways: (i) a programmatic interface encapsulated in a JAR file and (ii) a web interface accessible using a regular internet browser.

Together with the SPARQL query, the user must provide a valid ArrayExpress experiment identifier or a set of keywords to determine which experiments are to be accessed. When keywords are provided, RDFBuilder employs the ArrayExpress native functionality to extract a related list of experiments. The system automatically downloads the non-cached experiments and produces an equivalent RDF model from each of them. To do this, RDFBuilder employs the MAGE Software Toolkit that offers Java based access to MAGE-ML documents. Through this library, the tree of elements that compose the MAGE-ML document is extracted using simple Java methods. However, instead of creating a detailed mapping of the available methods into a predefined set of RDF classes, we adopt a technique that allows a more generic analysis of the document. Using the reflection programming technique [28], RDFBuilder dynamically resolves object types and invokes the available methods in order to retrieve the microarray data. The methods signatures themselves provide the name of the MAGE-ML entities being retrieved. This way, no precompiled set of MAGE-ML elements to retrieve is required, and evolution of the software upon updates in the MAGE-ML standard is completely automatic.

After translating each experiment data into its corresponding RDF model, a full merge is performed in order to produce a single RDF model from all experiments. Since all experiments stick to the same schema, this procedure is straightforward. This RDF represents the data of interest for the user. The final step consists on launching the original query against this model. The obtained result is the product returned to the user.

RDFBuilder is composed of a set of libraries developed in the Java programming language and a web interface enabling direct access by end users. The web service tool provided as the interface for direct query access was developed in AJAX, according to a strong model-view-controller design pattern [25]. The business logic is provided by a Java API containing the implementation of the algorithms to map MAGE-ML and translate SPARQL queries to the native microarray database language. RDFBuilder is also able to automatically parse the results that are retrieved by the underlying microarray database, returning a higher semantic level SPARQL result set. Fig. 1 illustrates the whole process.

3.2. *Accessing RDFBuilder*

The API is designed to resolve queries in SPARQL. The schema for generating these queries was dynamically created from the MAGE-OM model, as described previously, and can be downloaded from http://www.bioinformatics.org/RDFBuilder/wiki/. As shown in this RDF schema, the prefix for all classes and relations is http://aewrapper. Thus, http://aewrapper:Experiment.Experiment should be specified to refer to the *Experiment* class found in the package of the same name in the MAGE-OM model.

Users can perform two different types of queries with RDFBuilder, as described below.

1. *Experiment-specific queries*. These include a SPARQL query together with the identifier of an existing ArrayExpress experiment (e.g. "E-GEOD-21068"). These queries are aimed at retrieving data solely from the specified experiment. The provided SPARQL must conform to the RDF schema described previously.
2. *Keyword-based queries*. Opposed to the experiment-specific queries, keyword-based queries are targeted at a group of experiments related with a set of keywords. They resort to an ArrayExpress feature which permits to infer a set of microarray experiments from a list of keywords. These queries are therefore composed of a SPARQL query and the list of keywords (e.g. "organism,' "ratus,' "mammary,' and glands"). Results from the related experiments are automatically integrated and presented to the user as a single SPARQL result set.

The SPARQL query in both types of queries is provided to the API as a String. RDFBuilder performs the corresponding parsing process, assisted by the Jena library [26].

The described functionality can be accessed through the *es.upm.gib.aewrapper.queryprocessing.QueryProcessor* class of the API. Its methods allow performing the two types of queries and always return a path to the generated results file. The sample code in Fig. 2 shows how to launch a user query using our programmatic interface.

```
import es.upm.gib.aewrapper.queryprocessing.QueryProcessor;
.
.
.

// The SPARQL query to be submitted
String query =
  "PREFIX base: <http://aewrapper#>\n " +
  "SELECT ?id ?name ?desc_text\n" +
  "WHERE {\n" +
  "  ?exp base:identifier_string ?id .\n" +
  "  ?exp base:name_string ?name .\n" +
  "  ?exp a base:Experiment.Experiment .\n" +
  "  ?exp base:descriptions ?desc .\n" +
  "  ?desc base:text_string ?desc_text .\n" +
  "}";

// the query submission code
// (this query is submitted to a single experiment)
String resultPath =
  QueryProcessor.processQuery(query, "E-TABM-199");

// the path to the results file is printed on console
System.out.println("Path to results " + resultPath);
```

Fig. 2 – Example of generic Java code to use the API to formulate a simple query.

The JavaDoc documentation attached to the API details how to correctly invoke RDFBuilder and the causes for possible exceptions – i.e., error parsing the query, unable to access the specified experiment, etc. The API needs a directory in the hard drive to be configured as experiment cache. In this directory, the API will place the downloaded experiments from ArrayExpress, as well as the generated result files. For further detail, please refer to the README.TXT file packed with the library.

4. Examples

This research was initially supported by the European Commission-funded ACGT (Advancing Clinico-Genomic Trials) Integrated Project [27]. The main goal in ACGT was to develop a platform to support clinical trials involving the integration of phenotypic and genomic data. For that purpose, the ACGT platform incorporated a data integration layer for homogenizing heterogeneous databases. This data integration layer adopted the ACGT Master Ontology (MO) as a global schema for representing the underlying data. The core component of this layer was the Semantic Mediator, a software module capable of dynamically translating queries in terms of the MO into heterogeneous database queries, and then integrating the generated results. The final ACGT infrastructure was proven to be viable across a variety of real world scenarios, all of which involved integrating microarray data to some extent. Testing focuses on how our tool can improve configuration, execution and performance issues in these scenarios.

For this experiment we chose a scenario concerning the integration of relational and microarray databases for the International Society of Pediatric Oncology's SIOP 2001/GPOH trial. There are several reasons for this choice. First of all, the goal is to reproduce a documented research scenario, described in Zirn et al. [1], where tumor samples hybridized in two-color microarrays are analyzed together with relational data from the hospital database (containing phenotypic patient information). Secondly, the changes in the scenario component architecture clearly illustrate how RDFBuilder contributes to improve the efficiency of the scenario in several ways. Although converting microarray data to an RDF format was not one of the initial ACGT goals, we found that doing so would facilitate the whole data integration process considerably.

Fig. 3 shows a comparison of the same scenario architecture. The architecture on the left is the component architecture used in the original SIOP 2001/GPOH scenario. Relational and microarray data clearly follow a different retrieval path until they reach the analysis process. In addition, microarray data have a final annotation process to match the MO terminology – which acts here as a semantic framework for homogenization. Although the process lifts part of the workload of the original scenario as described in Zirn et al.'s. paper, full heterogeneous data integration is only achieved with the inclusion of RDFBuilder. The architecture on the right side makes use of our tool to generate an RDF-compliant wrapper in the data integration layer retrieving microarray data in a way that can be directly processed by the Semantic Mediator. Clearly, several processes and components are no longer needed when using RDFBuilder, further automating data processing, and achieving better results in the end. After the mediation process, all the data are completely homogenized by means of the MO. This way it can be reused in other type of processes or scenarios, as well as integrated with other data types curated by the Semantic Mediator. Furthermore, the complete process performs better after several uses because RDFBuilder is capable of caching data for further use in future retrievals.

The task that takes longest to complete is the actual downloading of experiment data. However, the implementation of a cache of accessed experiments helps to avoid this bottleneck. The result is a speed increase in the processing of queries with overlapping experiments (cached data validity is checked every time it is accessed). When a user launches the same or a related query to one already run, the system is able to automatically identify the similarities and retrieve results from the cache where previous data are stored. This approach avoids retrieving the same experiment from the ArrayExpress database twice. The API autonomously detects when updated experiment data are available. This prevents user queries from being answered with outdated data. Our experiments show how this helps to reduce the response time by approximately 80% (for an average query involving 9 microarray experiments).

5. Hardware and software specifications

The system has been completely developed in Java, so the hardware required is any that supports and offers a Java virtual machine. Other requirements:

- Operating system(s): platform independent.
- Programming language: Java (1.5 or higher).

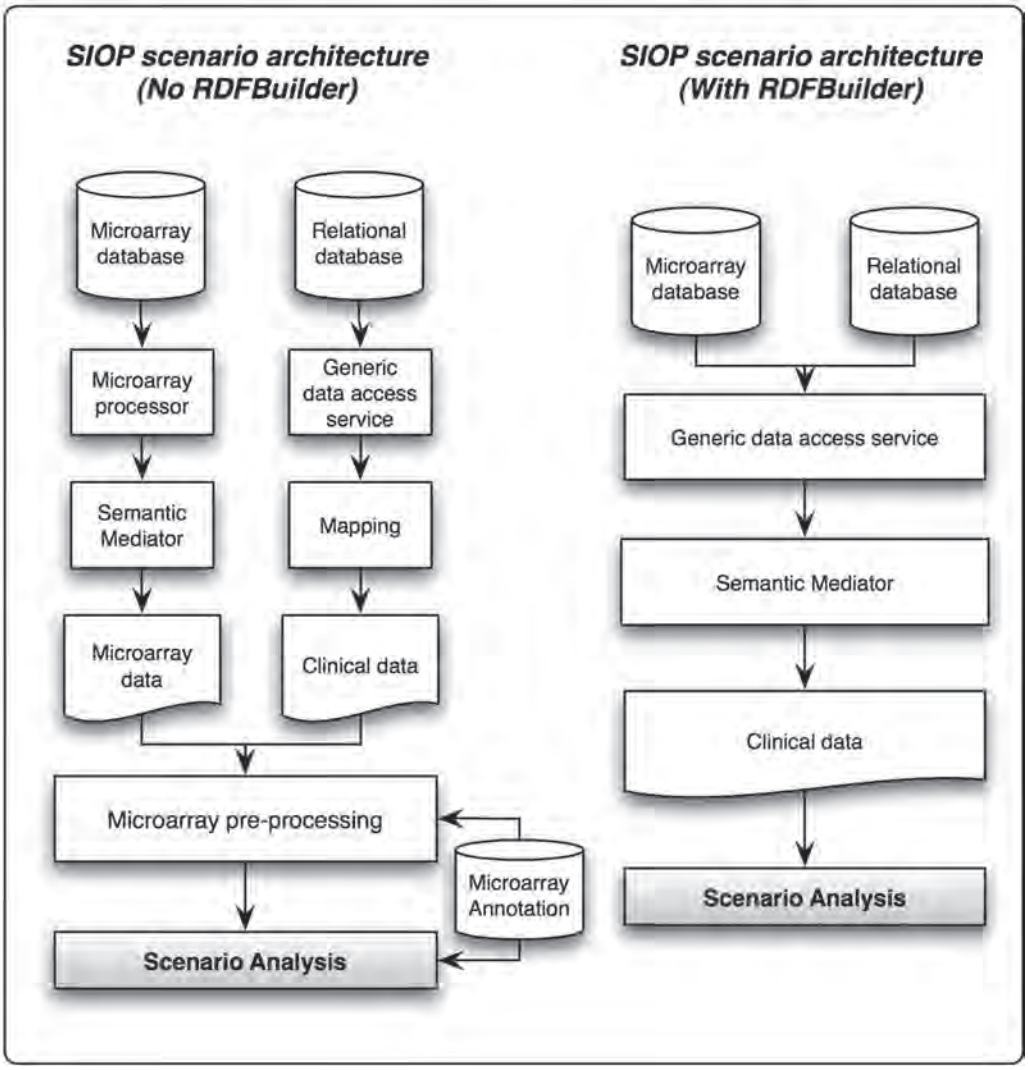

Fig. 3 – A comparison of possible SIOP scenario architectures.

- License of use: LGPL.
- Any restrictions of use by non-academics: None.

6. Availability

The software can be freely downloaded as a Java library. Detailed instructions of use are included, as well as complete JavaDocs. The project homepage is located at http://www.bioinformatics.org/RDFBuilder/wiki/.

The software can be tested online from any browser at http://servet.dia.fi.upm.es:8080/ArrayExpressWrapperWeb/.

7. Discussion

In this paper we have presented RDFBuilder, a tool to automatically translate MAGE-ML compliant microarray databases into RDF. This system acts as a wrapper, enabling the execution of SPARQL queries to retrieve ArrayExpress data. The algorithm used to build and populate the RDF representation of ArrayExpress works in a dynamic, adaptive manner, and can be easily extended to work with any kind of XML schema. Such a dynamic approach might be helpful, given the huge cost of building wrappers able to maintain the expressivity level of the wrapper database.

RDFBuilder offers an algorithm for transparently translating the data contained in the ArrayExpress database into RDF. The focus when designing the tool was to maintain the expressive power provided by MAGE-ML, the native language employed in ArrayExpress database storage. In addition, RDFBuilder supports the retrieval of data from several experiments in an integrated manner, without the need to specify the actual experiments.

A related initiative aimed at providing RDF-based access to biological databases is Bio2RDF [14]. The latter focuses on translating public biological databases in some specific formats (namely, XML documents, flat text files, relational databases and HTML documents) into RDF. With respect to public databases such as ArrayExpress, Bio2RDF retrieves summarized information provided in HTML documents of each instance stored in the database. While Bio2RDF is

designed to cover a large number of data repositories, it hardly digs in the details of each of them. Conversely, our tool is specifically designed to support this type of documents by providing an *ad hoc* RDF schema that covers all the complexity of this language, and thus enables users to issue complex queries to the ArrayExpress microarray repository. While Bio2RDF focuses on supporting a large set of different databases, RDFBuilder concentrates on maintaining the expressivity of the original query systems. In addition, our tool is capable of displaying query results of a set of experiments in an integrated manner. This means that end users are relieved from having to manually merge different results. Our approach can be reused by any database adopting this data model (e.g. Gene Expression Omnibus [21] or the Stanford Microarray Database [29]). In addition, by providing results in SPARQL-Results format and by offering a SPARQL query endpoint, resulting data sets can be integrated with any kind of RDF-compliant data. This includes relational databases whose conversion to an RDF schema is straightforward.

There exists a project at Google Code targeted at the translation of FGED standards to RDF, called magetab2rdf1. This project seems to be focused, from its main Web-based reference, on the translation of MAGE-TAB documents – a tabular representation resulting of the simplification of the MAGE-ML standard – into RDF using existing vocabularies such as the PROV Ontology (PROV-O), the Ontology for Biomedical Investigations (OBI), the Open Biomedical Ontology (OBO), the Experimental Factor Ontology (EFO) and the MGED Ontology.

Conversely, our tool does not resort to any external vocabularies or ontologies to translate the actual data. Instead, our tool aims at translating the full MAGE-ML specification – i.e., the database schema rather than the actual data – into RDF, and for this purpose a new RDF model is obtained as a literal reflect of the MAGE-OM model. Thus, it can be used to translate MAGE-ML compliant databases into RDF. This task, to our knowledge, cannot be accomplished by neither magetab2rdf nor other existing tools. In addition, the method and software presented in this paper offer various functionalities, not yet provided by other readily available tools at the time of writing. First, our solution enables users to directly access ArrayExpress data through SPARQL queries, dynamically retrieving the required databases and automatically translating them into SPARQL-Results format. Access to the ArrayExpress database is therefore completely transparent to users – unlike the magetab2rdf, that accepts a user-provided MAGE-TAB document and translates it into RDF – providing them with a virtual RDF-based view of the ArrayExpress data that can be directly queried using our programmatic interface. Conversely, the magetab2rdf tool generates a plain RDF file that needs to be queried using external tools and/or interfaces. Second, our software includes an innovative feature for automatically integrating data from multiple ArrayExpress experiments in a dynamic, transparent fashion. Since the RDF models generated by our tool share a common schema, they can be automatically integrated without any user intervention. Therefore, users can simultaneously query several experiments, or even specify a set of terms that may be borrowed from different ontologies including GO, OBO, OBI, etc. After the query is processed by our system, the software will automatically present the user the results as if they were retrieved from one single, integrated, database.

Our tool can be a valuable resource for biomedical researchers to seamlessly integrate a set of target experiments stored in ArrayExpress and automatically create a single, SPARQL query-able, RDF document integrating the data from the selected experiments. We believe that this can save a significant amount of time to researchers who, even using other existing tools such as magetab2rdf, would otherwise have to manually gather and integrate the individual experiments, which would have to be queried using external tools and/or interfaces. RDFBuilder is currently being used by biomedical researchers of the European Commission-funded p-medicine project [30] – in which the authors currently participate together with other 18 European and Japanese research institutions, universities and companies – to gather, integrate and query MAGE-ML-based microarray experiments. Thus, it can support the creation of custom clinical trials and novel drugs targeted at specific patients based on their clinicogenomic profile. In addition, RDFBuilder can be used both by end users through the provided web interface or by other tools and frameworks, such as Bio2RDF itself. Our tool is not a substitute for this or other existing approaches for biomedical database integration. Instead, it is a complementary approach that enables end users and other existing tools to seamlessly access MAGE-ML in an RDF-compliant form, thus paving the path toward the deployment of biological integration solutions.

RDFBuilder improves query performance for regular microarray database users using of a cache memory that stores partial schemas and results. The system is able to identify when a query or a part of a query is being reused, and retrieve the results much faster than querying ArrayExpress directly.

By using Java, we assure system portability to any platform. The software is licensed under GNU GPL, so its code is available for non-profit exploitation. All the third party libraries that have been used in the system are freeware. If a researcher needs to use the system as a "black box," the included programmatic interface provides easy means for communication. The Java JAR library implementing RDFBuilder is available for free downloading (see Section 6 for detailed information).

Heterogeneity, quantity and disparity of all available biomedical data sources are an obstacle to progress in this field. By enabling RDF-based access to the most widespread format for managing microarray data, RDFBuilder becomes part of the bioinformatics community's global effort to homogenize heterogeneous biomedical data sources. RDF has come to be the *de facto* standard for representing disparate data in biomedicine. With this software, we hope to further support biomedical research's need for distributed data access.

8. Conclusions

Modern research in bioinformatics and medicine involves accessing and combining huge amounts of data from different heterogeneous sources and locations. In a generic scenario, these data sets need to be queried, managed, curated and, in many cases, manually integrated, bridging the heterogeneities

arising out of non standardization caused by tremendous growth that has taken place in the last decade. In this regard, the FGED consortium for microarray databases is working on defining new standards. However, standardization is only the first step in the process of giving the research community reasonable access to a significant number of data sources. Tools are required to facilitate the translation of queries, schemas and data.

With MAGE-OM as the most widespread standard for representing microarray data, there is an increasing need to integrate this standard with other RDF-based biomedical databases. In this paper, we have presented a software tool for automatically wrapping MAGE-OM-compliant data into RDF and providing SPARQL querying capability. Our solution, RDFBuilder, does not require the adoption of any custom ontology or RDF schema, since the generated RDF documents directly reflect the MAGE-OM standard. RDFBuilder was also designed to autonomously adapt to modifications in the MAGE-ML standard without code reimplementation. In addition, the API implements a cache system, which ensures access to updated data while reducing access times for frequently retrieved datasets. Finally, a Web-based interface has also been developed and made publicly available for tool testing, including some example queries.

RDFBuilder has features that facilitate interoperability with earlier approaches implemented to transform biomedical data into RDF format, such as Bio2RDF. Our tool is targeted at bioinformaticians who develop data integration applications. In this context, RDFBuilder solves the syntactic heterogeneities by offering RDF-based access to the ArrayExpress databases. Our approach provides both data translation services – by translating MAGE-ML documents into RDF – and a dynamic SPARQL query endpoint, making it suitable for a wide range of data integration approaches. We firmly believe that RDFBuilder can significantly improve the process of integrating microarray with other biological data.

This work was funded by the European Commission, through the ACGT integrated project (FP6-2005-IST-026996), p-medicine (FP7-ICT-2009-270089) and INTEGRATE (FP7-ICT-2009-270253). The authors would also like to thank Martín Esteban for writing some parts of the code, and to Esther Peinado for developing the Web interface and providing useful suggestions.

[1] B. Zirn, O. Hartmann, B. Samans, M. Krause, S. Wittmann, F. Mertens, N. Graf, M. Eilers, M. Gessler, Expression profiling of Wilms tumors reveals new candidate genes for different clinical parameters, International Journal of Cancer 118 (8) (2006) 1954–1962.

[2] L. Martin, A. Anguita, V. Maojo, J. Crespo, Integration of omics data for cancer research, in: W.C.S. Cho (Ed.), An Omics Perspective on Cancer Research, Springer, Dordrecht, Netherlands, 2010, pp. 249–266.

[3] B.M. Good, M.D. Wilkinson, The life sciences semantic web is full of creeps!, Briefings in Bioinformatics 7 (3) (2006) 275–286.

[4] J. Sagotsky, L. Zhang, Z. Wang, S. Martin, T. Deisboeck, Life Sciences and the web: a new era for collaboration, Molecular Systems Biology 4 (2008).

[5] V. Maojo, M. Garcia-Remesal, H. Billhardt, R. Alonso-Calvo, D. Perez-Rey, F. Martin-Sanchez, Designing new methodologies for integrating biomedical information in clinical trials, Methods of Information in Medicine 45 (2) (2006) 180–185.

[6] D. Perez-Rey, V. Maojo, M. Garcia-Remesal, R. Alonso-Calvo, H. Billhardt, F. Martin-Sanchez, A. Sousa, ONTOFUSION: ontology-based integration of genomic and clinical databases, Computers in Biology and Medicine 36 (7) (2006) 712–730.

[7] R. Alonso-Calvo, V. Maojo, H. Billhardt, F. Martin-Sanchez, M. Garcia-Remesal, D. Perez-Rey, An agent- and ontology-based system for integrating public gene, protein, and disease databases, Journal of Biomedical Informatics 40 (2007) 17–29.

[8] C. Rosse, J.L.V. Mejino, A reference ontology for biomedical informatics: the Foundational Model of Anatomy, Journal of Biomedical Informatics 36 (6) (2003) 478–500.

[9] G.O. Consortium, The Gene Ontology (GO) project in 2006, Nucleic Acids Research 34 (Suppl. 1) (2006) D322–D326.

[10] M. Brochhausen, A.D. Spear, C. Cocos, G. Weiler, L. Martin, A. Anguita, H. Stenzhorn, E. Daskalaki, F. Schera, U. Schwarz, S. Sfakianakis, S. Kiefer, M. Doerr, N. Graf, M. Tsiknakis, The ACGT Master Ontology and its applications – towards an ontology-driven cancer research and management system, Journal of Biomedical Informatics 44 (2011) 8–25.

[11] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. Goldberg, K. Eilbeck, A. Ireland, C. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S. Sansone, R. Scheuermann, N. Shah, P. Whetzel, S. Lewis, The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, Nature Biotechnology 25 (11) (2007) 1251–1255.

[12] L. Martin, A. Anguita, A. Jimenez, J. Crespo, Enabling cross constraint satisfaction in RDF-based heterogeneous database integration, in: 20th IEEE International Conference on Tools with Artificial Intelligence, 2008, pp. 341–348.

[13] K. Cheung, K. Yip, A. Smith, R. Deknikker, A. Masiar, M. Gerstein, YeastHub: a semantic web use case for integrating data in the life sciences domain, Bioinformatics 21 (Suppl. 1) (2005).

[14] F. Belleau, M. Nolin, N. Tourigny, P. Rigault, J. Morissette, Bio2RDF: towards a mashup to build bioinformatics knowledge systems, Journal of Biomedical Informatics 41 (5) (2008) 706–716.

[15] L. Stromback, D. Hall, P. Lambrix, A review of standards for data exchange within systems biology, Proteomics 7 (6) (2007) 857–867.

[16] C. Lloyd, M. Halstead, P. Nielsen, CellML: its future, present and past, Progress in Biophysics and Molecular Biology 85 (2–3) (2004) 433–450.

[17] E. Demir, M. Cary, S. Paley, K. Fukuda, C. Lemer, I. Vastrik, G. Wu, P. D'Eustachio, C. Schaefer, J. Luciano, F. Schacherer, I. Martinez-Flores, Z. Hu, V. Jimenez-Jacinto, G. Joshi-Tope, K. Kandasamy, A. Lopez-Fuentes, H. Mi, E. Pichler, I. Rodchenkov, A. Splendiani, S. Tkachev, J. Zucker, G. Gopinath, H. Rajasimha, R. Ramakrishnan, I. Shah, M. Syed, N. Anwar, O. Babur, M. Blinov, E. Brauner, D. Corwin, S. Donaldson, F. Gibbons, R. Goldberg, P. Hornbeck, A. Luna, P. Murray-Rust, E. Neumann, O. Reubenacker, M. Samwald, M. van Iersel, S. Wimalaratne, K. Allen, B. Braun, M. Whirl-Carrillo, K. Cheung, K. Dahlquist, A. Finney, M. Gillespie, E. Glass, L. Gong, R. Haw, M. Honig, O. Hubaut, D. Kane, S. Krupa, M. Kutmon, J. Leonard, D. Marks, D. Merberg, V. Petri, A. Pico, D. Ravenscroft, L. Ren, N. Shah, M. Sunshine,

R. Tang, R. Whaley, S. Letovksy, K. Buetow, A. Rzhetsky, V. Schachter, B. Sobral, U. Dogrusoz, S. McWeeney, M. Aladjem, E. Birney, J. Collado-Vides, S. Goto, M. Hucka, N. Le Novere, N. Maltsev, A. Pandey, P. Thomas, E. Wingender, P. Karp, C. Sander, G. Bader, The BioPAX community standard for pathway data sharing, Nature Biotechnology 28 (9) (2010) 935–942.

[18] M. Hucka, A. Finney, H.M. Sauro, H. Bolouri, J.C. Doyle, H. Kitano, et al., The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models, Bioinformatics 19 (4) (2003) 524–531.

[19] P.L. Whetzel, H. Parkinson, H.C. Causton, L. Fan, J. Fostel, G. Fragoso, L. Game, M. Heiskanen, N. Morrison, P. Rocca-Serra, S. Sansone, C. Taylor, J. White, C.J. Stoeckert, The MGED Ontology: a resource for semantics-based description of microarray experiments, Bioinformatics 22 (7) (2006) 866–873.

[20] H. Parkinson, U. Sarkans, M. Shojatalab, N. Abeygunawardena, S. Contrino, R. Coulson, A. Farne, G. Garcia Lara, E. Holloway, M. Kapushesky, P. Lilja, G. Mukherjee, A. Oezcimen, T. Rayner, P. Rocca-Serra, A. Sharma, S. Sansone, A. Brazma, ArrayExpress – a public repository for microarray gene expression data at the EBI, Nucleic Acids Research 33 (Suppl. 1) (2005) D553–D555.

[21] T. Barrett, D. Troup, S. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I. Kim, A. Soboleva, M. Tomashevsky, R. Edgar, NCBI GEO: mining tens of millions of expression profiles database and tools update, Nucleic Acids Research 35 (2007), Database issue, D760–D765.

[22] K. Ikeo, J. Ishi-I, T. Tamura, T. Gojobori, Y. Tateno, CIBEX: center for information biology gene expression database, Comptes Rendus Biologies 326 (10–11) (2003) 1079–1082.

[23] J. Quackenbush, Standardizing the standards, Molecular Systems Biology 2 (2006) 0010.

[24] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N.P. Chue-Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. Paton, D. Pearson, T. Sugden, P. Watson, M. Westhead, The design and implementation of Grid database services in OGSA-DAI, Concurrency and Computation: Practice and Experience 17 (2–4) (2005) 357–376.

[25] E. Curry, P. Grace, Flexible self-management using the model-view-controller pattern, IEEE Software 25 (2008) 84–90.

[26] B. McBride, Jena: a semantic Web toolkit, IEEE Internet Computing 6 (2002) 55–59.

[27] L. Martin, A. Anguita, N. Graf, M. Tsiknakis, M. Brochhausen, S. Ruping, A. Bucur, S. Sfakianakis, T. Sengstag, F. Buffa, H. Stenzhorn, ACGT: advancing Clinico-genomic trials on cancer – four years of experience, Studies in Health Technology and Informatics 169 (2011) 734–738.

[28] B.C. Smith, Procedural reflection in programming languages, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, 1982.

[29] G. Sherlock, T. Hernandez-Boussard, A. Kasarskis, G. Binkley, J.C. Matese, S.S. Dwight, M. Kaloper, S. Weng, H. Jin, C.A. Ball, M.B. Eisen, P.T. Spellman, P.O. Brown, D. Botstein, J.M. Cherry, The Stanford microarray database, Nucleic Acids Research 29 (2001) 152–155.

[30] N. Graf, A. Anguita, A. Bucur, D. Burke, B. Claerhout, P. Coveney, A. d'Onofrio, N. Forgo, C. Hahn, J. Hintz, B. Jefferys, S. Kiefer, K. Marias, G. McVie, C. Ohmann, A. Persidis, J. Pukacki, S. Rossi, S. Ruping, U. Schwarz, G. Stamatakos, M. Stanulla, H. Stenzhorn, Y. Tanaka, M. Taylor, M. Tsiknakis, P-medicine: a solution for translational research? Paediatric Blood & Cancer 59 (6) (2012) 1101.