The ACGT Generic Multilevel data integration approach

Project Number:	FP6-2005-IST-026996
Deliverable id:	D7.10
Deliverable name:	The ACGT Generic Multilevel data integration approach
Submission Date:	July 2010

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D 7.10
Document name:	The ACGT Genetic Multilevel data integration approach
Document type (PU, INT, RE)	PU
Version:	1.0
Submission date:	07/30/2010
Editor: Organisation: Email:	Alberto Anguita UPM aanguita@infomed.dia.fi.upm.es

Document type PU = public, INT = internal, RE = restricted

ABSTRACT: This deliverable aims at giving a detailed view on the approach adopted in ACGT for the integration of distributed, heterogeneous biomedical data sources. The exposed approach was adopted with the aim of covering the requirements gathered during the project duration. With the developed tools and modules, we provided the ACGT platform with a data integration architecture that has the ability of dynamically adding new sources and capable of performing integrated access to distributed and heterogeneous multi-level data sources. Apart of the approach, detail is given on the end-user oriented tools developed in the semantic mediation layer.

KEYWORD LIST: Clinical trials, Semantic Mediation, Ontologies, Wrapper

MODIFICATION CONTROL			
Version	Date	Status	Editor
0.1	07/20/2010	Draft	Alberto Anguita
1.0	07/30/2010	Final	Alberto Anguita

List of Contributors

- Alberto Anguita, UPM
- Fatima Schera, IBMT-FHG
- Holger Stenzhorn, USAAR
- Martin Dörr
- Mathias Brochhausen USAAR
- Ulf Schwarz USAAR

Contents

1 Executive Summary

This deliverable aims at giving a detailed view on the approach adopted in ACGT for the integration of distributed, heterogeneous biomedical data sources. The exposed approach was adopted with the aim of covering the requirements gathered during the project duration. With the requirements in hand, a complete approach for generic multilevel data integration was designed. This approach includes formats, protocols and algorithms. The main characteristics of the general approach are, namely: i) transparency: the end users will have the impression of being working with a single database covering all integrated data, and the underlying heterogeneities will be hidden, ii) flexibility: with dynamic source integration and global schema updating protocols, the system is flexible enough to adapt to the highly evolving biomedical domain and iii) performance: key design decisions were targeted at providing a high performance platform at the cost of cutting freedom of use—extensive testing proved that the end user experience is not affected by this feature cut. This document gives detail on the adopted approach, as well as on the end-user oriented tools developed in the semantic mediation layer.

2 Introduction

2.1 Purpose and structure of this document

This document describes the multi-level data integration approach implemented in the ACGT Platform. In this section an introduction the issues involved in the integration of heterogeneous biomedical data sources is given. Section 3 gives a global view of the approach adopted for data integration in ACGT. Section 4 describes the mapping format developed for the semantic mediation layer. Section 5 gives details of the query translation and result integration processes. Section 6 describes other tools and modules composing the semantic mediation layer. Section 7 gives a brief overview of the relation of the semantic mediation layer with other components of the ACGT platform. Finally, section 8 provides the conclusions of this deliverable.

2.2 Introduction

The ACGT project was created with the aim of offering a technological platform to support clinicians in the design and conduction of clinical trials on cancer. This platform includes trial design tools and data analysis tools among others. One feature of special relevance is the possibility to access distributed, heterogeneous sources in a homogeneous manner. To do so, the platform incorporates a data integration layer that offers semantic mediation services.

Biomedical data integration

The problem of data integration in the biomedical domain has been in the bioinformaticians agenda for over a decade. It is a problem of great interest, since clinical researchers more and more often employ techniques which require the integrated access to heterogeneous data. The latest approaches in cancer research rely on the integration of clinical and genomic data to find genetic signatures that help identifying the best treatments for each specific patient. Automating this process, however, poses new problems. Different vocabularies, units of measure, data codifications and structures, or even access interfaces and query languages are adopted for each database, even if they contain semantically equivalent data. To deal with all these heterogeneities and offer the end user a homogeneous access point to these data, The ACGT Semantic Mediation layer adopts a series of approaches based on past research on this field.

3 The ACGT approach for multilevel data integration

The design of the ACGT semantic mediation layer began with the gathering of requirements from the end users (in this case, clinicians). The system to develop had to be able to offer integrated access to clinical trial databases as well as legacy databases. Databases to integrate included from clinical sources to genomic data repositories. In addition, good response times were an important factor, as well as standard adoption and ease of use.

The rest of the requirement gathering process and the testing of the platform was done with the help of two partners involved in clinical trials on cancer (the SIOP [SIOP] and the TOP [TOP] trials, related to Nephroblastoma tumor and breast cancer, respectively). Thanks to these, a detailed map of the types of heterogeneities that should be dealt with was drawn (described in deliverable D7.1 "Consolidated requirements on ontological approaches for integration of multi-level biomedical information"), and from it, an architecture of the layer was designed. The main characteristics of this architecture were: i) the division of syntactic heterogeneities from semantic heterogeneities, using a wrapper-based approach to deal with the former, ii) an ontology-based approach for dealing with the semantic heterogeneities, which would act as global schema for a distributed integration approach, and iii) distributed approach for integration, adopting a query translation-approach. The wrapper-based approach for integration is widely used by the scientific community, as it helps separating the integration problem into two smaller ones. The wrappers would hide the technological details of each data source to be integrated, thus solving the syntactic heterogeneities—i.e. access interfaces, query language and database schema types—. This way, every repository to integrate would appear to an upper layer as an RDF-based resource [RDF] with SPARQL [SPARQL] querying capability, and accessible though the OGSADAI [OGSADAI] platform. All these technologies were selected due to being versatile and widespread enough to be considered standards.

The ontology-based approach was motivated by the need of a semantic framework that would provide a common vocabulary covering the domain of the data sources to integrate. To this end, the ACGT Master Ontology (MO), covering the domain of clinical trials on cancer, was developed. The MO contains over a thousand classes and over two hundred relations—although efforts to reduce its size and complexity are currently undergoing—, and its development was primarily guided by the previously mention trials—TOP and SIOP. The main goal was to create a sound and solid semantic framework for the domain of post-genomic clinical trials on cancer. Creators of the MO are currently negotiating its incorporation in the OBO Foundry [OBO]. More information on the MO can be found in the paper published by Brochhausen et al. [Brochhausen et al. 2010].

The distributed approach was adopted due to the highly evolving nature of biomedical databases. By adopting a centralized approach, where the data of the sources is physically transferred to a central repository, we would have come up with a platform not flexible enough to adapt to the emergence of new sources, of the updating of existing ones. In the distributed approach, no data is stored, and the underlying sources are accessed each time a query is received. Queries are automatically translated into sub-queries for the needed sources, and their results are integrated and presented to the end user—or client application—as a unique result. The user still has the impression of being working with a global repository containing all the data, as this process is performed transparently, at the cost of higher computational cost for solving incoming queries. Section 5 describes this process in detail.

Given that the schema of the global database was an ontology—actually, their RDF elements—and that the underlying database wrappers offered RDF-based databases with SPARQL querying capability, the obvious decision for designing the interface of the semantic mediation layer was using SPARQL as query language. This language is a W3C recommendation since 2008, so from the standardization point of view it was the correct decision. This language is however a little too cumbersome and might be too difficult to handle by non-technical users, so a query builder tool was developed to hide its complexity. This will be detailed later in the document.

Several tools, formats and APIs were developed together with the semantic mediation module (SM), composing the final ACGT Semantic Mediation Layer architecture. These include a Query Tool for building SPARQL queries for the SM, a Mapping Tool for creating mappings of legacy databases, a Mapping Format for representing mappings between RDF schemas, and a Mapping API for managing mapping formats. Figure 1 shows the architecture of the semantic mediation layer.

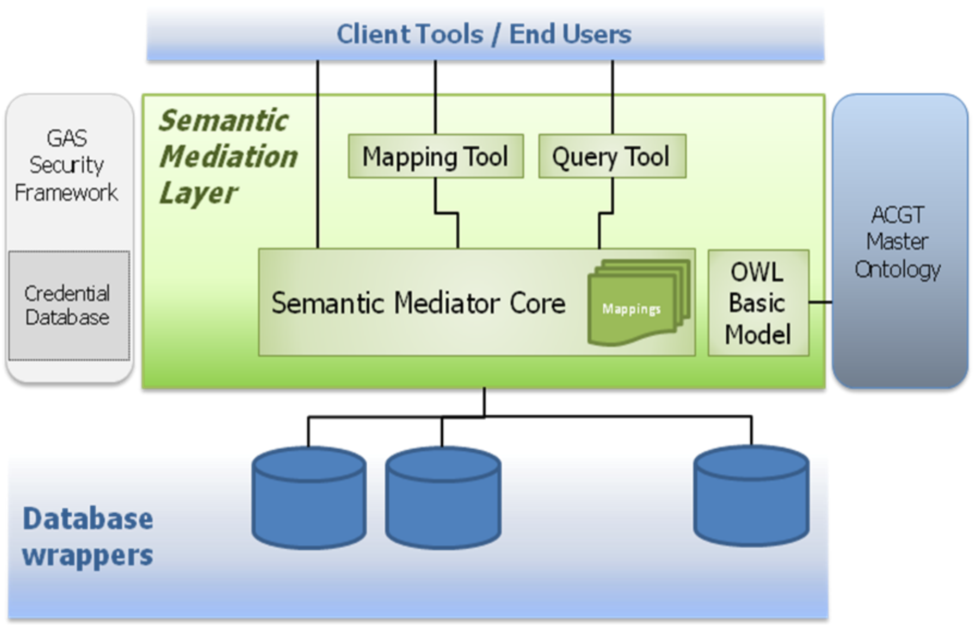


Fig. 1: The semantic mediation layer architecture.

The individual components depicted above are described in detail in the next sections of the document.

4 The Mapping Format

The query translation approach adopted for the semantic mediation process requires the automatic translation of queries in terms of one global RDF schema to sub-queries in terms of a set of RDF schemas of the underlying sources. One precondition is that the global schema covers the union of the domains covered by the subschemas—and possibly more—so elements of the global schema can be directly translated into elements of the subschemas, and vice versa. To annotate how these elements are related, we developed the ACGT Mapping Format. This is an XML format which allows describing pairs of semantically equivalent atomic elements in two RDF schemas. XML files based on this format are the so called mapping files.

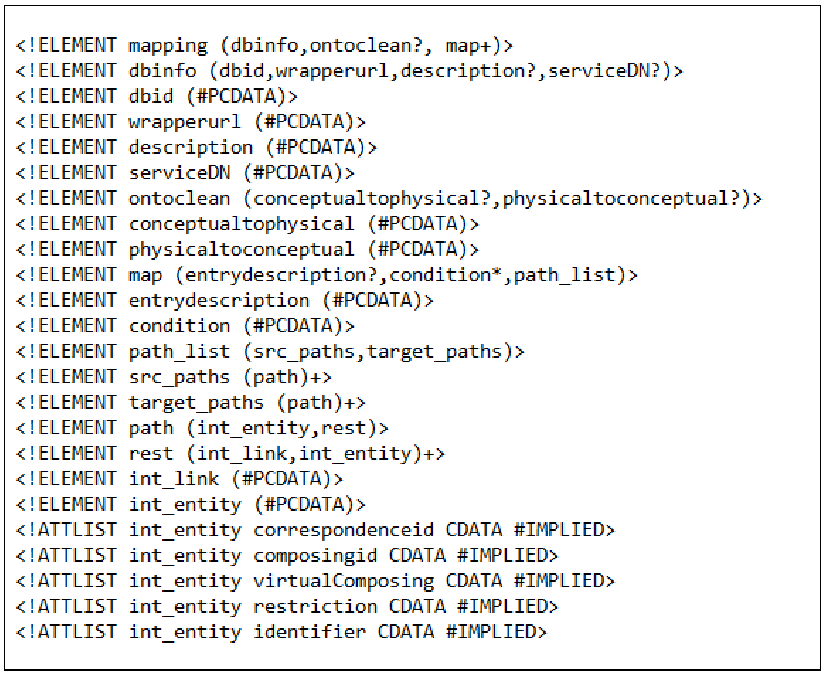
The first question is "what to consider an atomic element in RDF?". In RDF, the elements that have any meaning are the paths, thus an atomic element is the smallest possible path. Anything equal or above that can be used in a mapping, from a simple path with just two classes and a relation linking them, to a full set of paths with bounds between them—as long as they form a connected graph. This is what we can view in the mapping, and the semantic correspondence of two views is an entry. Thus, the mapping of an RDF database with the Master Ontology is a collection of entries, each containing a view on the Master Ontology, and its equivalent view on the database being integrated.

As we said, a view can contain just one path, or several. In case it contains several paths there must be a link between those paths, so the graph that they form is connected. These links are specified as classes in the paths which refer to the same instance—e.g. path 1 is "HumanBeing1 isA Patient" and path 2 is "HumanBeing2 hasA Name", with HumanBeing1 referring to the same instance as HumanBeing2. In the mapping views, this is specified through internal bounds, which allow indicating classes in different paths referring to the same instance.

In addition, we need to be able to specify the equivalent of individual classes in the conceptual view—corresponding to the MO—and the physical view—corresponding to the database. This way, if a query requires retrieving the values of "Class_A", we will be able to determine from which classes in the underlying databases we must collect the data to fill the column under "Class_A". This is specified through external bounds.

Finally, the mapping format allows imposing additional restrictions over an entry. These are conditions that must be fulfilled for the entry to be "used" in the translation of a query—or part of a query. These are used in a view when the meaning of that view actually depends on the value of some instance—restrictions are only employed in physical views.

In figure 2 we can see the DTD document describing the valid XML mapping documents.



```
<!ELEMENT mapping (dbinfo,ontoclean?, map+)>
<!ELEMENT dbinfo (dbid,wrapperurl,description?,serviceDN?)>
<!ELEMENT dbid (#PCDATA)>
<!ELEMENT wrapperurl (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT serviceDN (#PCDATA)>
<!ELEMENT ontoclean (conceptualtophysical?,physicaltoconceptual?)>
<!ELEMENT conceptualtophysical (#PCDATA)>
<!ELEMENT physicaltoconceptual (#PCDATA)>
<!ELEMENT map (entrydescription?,condition*,path_list)>
<!ELEMENT entrydescription (#PCDATA)>
<!ELEMENT condition (#PCDATA)>
<!ELEMENT path_list (src_paths,target_paths)>
<!ELEMENT src_paths (path)+>
<!ELEMENT target_paths (path)+>
<!ELEMENT path (int_entity,rest)>
<!ELEMENT rest (int_link,int_entity)+>
<!ELEMENT int_link (#PCDATA)>
<!ELEMENT int_entity (#PCDATA)>
<!ATTLIST int_entity correspondenceid CDATA #IMPLIED>
<!ATTLIST int_entity composingid CDATA #IMPLIED>
<!ATTLIST int_entity virtualComposing CDATA #IMPLIED>
<!ATTLIST int_entity restriction CDATA #IMPLIED>
<!ATTLIST int_entity identifier CDATA #IMPLIED>
```

Fig. 2: DTD schema for mapping documents.

4.1 The Mapping API

XML mapping files can be built manually, although it turns quite complicated if the file must contain many entries. In addition, there are client tools that must build these files automatically. With this in mind, we developed a Java API for generating, editing and exploring mapping files. This API allows the programmatic handling of entries and paths in a mapping, including the creation of new elements and the edition of existing ones. Searching operations are also supported, allowing clients to find specific views, or find the existing views subsuming a given one. One view subsumes another view if the former represents a generalization of the later. This capability is used by in the translation process, since the translation of views must also consider specification/generalization of concepts—i.e. if “A” translates into “B”, a specification of “A” must also be translated into “B”.

5 Query translation and result integration processes

The main component of the ACGT semantic mediation layer is the Semantic Mediator (SM). This module is in charge of receiving user queries, translating them for the necessary underlying databases, and integrating the retrieved results to form a unique result set. The complete process is divided in five different steps, listed below.

1. Query decomposition into query views
2. Identification of involved databases
3. Generation of sub-queries for underlying databases
4. Sub-result gathering from databases
5. Sub-result integration into final result set.

The paragraphs below are devoted to describing each of the previous steps in detail.

5.1 Query decomposition into query views

Given that what the system is capable of only translating views, an incoming query must first be decomposed into its corresponding query views. This poses the first problem: the number of possible view decompositions of a query grows exponentially with the number of elements it contains. For a medium size query, we might easily reach thousands of possible decompositions. If we tried to calculate them all, each query would require too much time to be processed (in the range of hours). We therefore decided to cut the number of calculated decompositions. The paths contained in a view cannot be divided into smaller paths, and views can only be formed with sets of complete paths linked with restrictions of the type "FILTER (?var_1 = ?var_2)". This restriction drastically reduces the space of possible queries. However, the tests performed subsequently with end-users proved that it did not affect their experience with the system. In addition, the query decomposition process complexity scales now linearly with the size of received queries. For large size queries, this process takes no more than a second.

5.2 Identification of involved databases

After we have compiled a list of views contained in the original query, we must compare each of them with the conceptual views contained in all the active mappings. If any comparison reports success, that specific view will have a translation in the given mapping. This comparison must take into account the subsuming condition. If the given view is a specification of a conceptual view in a mapping, it must also be translated.

Once again, had we not reduced the count of generated query views, this process would require too much computational power. With the imposed restriction, this step is usually performed again in around one second.

5.3 Generation of sub-queries for underlying databases

With the list of query views containing a translation in the available mappings, we are ready to generate the sub-queries for the underlying databases. The physical views from the mappings will form the paths of the sub-queries, and the external bounds together with the variables in the SELECT clause of the original query will provide the necessary information to complete the sub-queries. In addition, we will have to retrieve from the data sources the fields required to solve any cross-constraint contained in the original query (a constraint involving elements which must be retrieved from different sources). Given that these type of restrictions cannot be solved at the local sources, we must retrieve the extra data necessary to solve them at an upper level.

5.4 Sub-result gathering from databases

With the sub-queries already generated, we must send them to the corresponding data sources. In our case, every data source has a corresponding database wrapper (and therefore a common invocation format). Database wrappers are deployed as OGSADAI resources, so each invocation requires the construction of a perform document. In addition, the database wrapper can be configured with a credential system for ensuring secure data access. In this case, the perform document will have to be further configured.

After all database wrappers have been invoked (this task is performed in parallel, to reduce response times), their results are retrieved and parsed. The failed accessions are considered as empty results. The results are obtained in SPARQL Result Format so they must be processed before handling the data contained in them. These data are temporarily stored into an RDF-based auxiliary database. This RDF-based auxiliary database is populated with the results retrieved from the local sources. The purpose is to obtain the result to the original query from it. The data stored in it is subsequently retrieved with an auxiliary query—basically, the original query without the constraints that were solved at the local databases—, enabling the use of cross constraints.

First, the RDF Schema for this database is created. For each local source queried, a class representing one table of results is created—we will call this table class. Then, one class is created for each variable in the results of a local query—we will call this variable class. These classes are provided with a datatype property to store their actual results. The class representing the table of results is linked to the classes representing its variable by means of object properties. Figure 3 depicts the relation between one table of results from a local source and the RDF Schema of the auxiliary database.

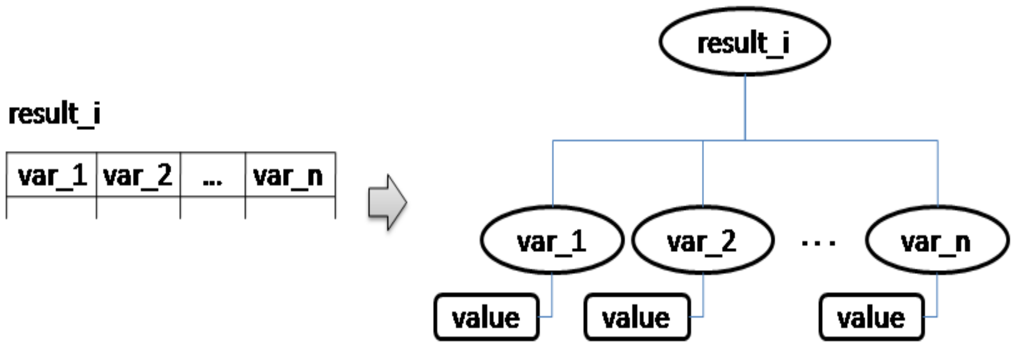


Fig 3: Generation of the RDF-based auxiliary database containing the sub-results.

When the RDF Schema is complete, the database is populated with the actual results. The procedure is as follows: for each row of results in a table, one instance of its table class and one instance of each of its variable classes are defined. The values in the row are fed into the datatype properties of the corresponding variable classes. This way, values belonging to the same row are related with each other.

This procedure is repeated for all results obtained from local sources. The generated auxiliary query is able to retrieve and correctly integrate the partial results, producing a single result which corresponds to the original query.

5.5 Sub-result integration into final result set

The final step involves generating the result to the original query. This is accomplished by performing the auxiliary query against the temporary repository. This generates an SPARQL Result document which contains the answer to the query posed by the user to our system. This result can be returned in different possible formats, including a CSV file. If desired, the user can specify in the original invocation that the result must be uploaded to the ACGT DMS for subsequent processing. Our system will take care of placing it in the specified location and returning the data needed to locate it in the DMS.

Figure 4 depicts the complete process of query translation and result integration.

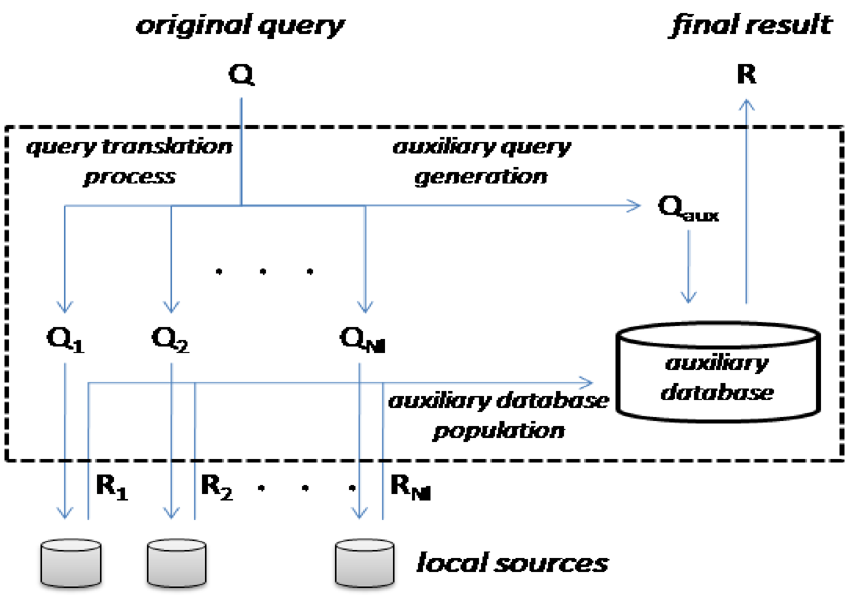


Fig 4: The complete process performed by the Semantic Mediatior to answer an integrated query.

Additional information on this process can be found in the paper published by [Martín et al. 2008].

6 The semantic mediation layer tools & services

A series of tools and services have been developed together with the Semantic Mediator to provide the required features in the semantic mediation layer. These included a web-based tool for creating SPARQL queries, another web-based tool for building mappings with legacy databases and an API for editing and exploring OWL [OWL] documents.

6.1 ACGT Query Tool

Accessing data using the SM requires creating queries in SPARQL. This query language has a moderate to high complexity and is often out of reach for researchers lacking the appropriate technical background. For this reason, we developed a web-based graphical tool that allows creating SPARQL queries just by selecting and clicking elements on the screen. Available elements that can be included in a query are shown to the user with natural language descriptions, so any newcomer can easily construct a query for the data she/he wishes to retrieve. Nevertheless, advanced users can edit the SPARQL query by hand if they wish.

6.2 ACGT Mapping Tool

There are two types of databases that are integrated through the semantic mediation layer. First there are clinical trials designed in the ACGT platform itself. The mappings for these are automatically built by the trial design tool, so there is no need for the end user to design the mapping document as well. Second, there are legacy databases which are already designed outside of the platform. These need specific mapping files developed by the end user. To avoid the complexity of dealing with cumbersome and long XML documents, we developed a web-based graphical tool for creating and submitting mapping files for such data sources. The tool allows the collaborative work by an expert in the domain of the database and an expert in the ACGT MO, so the database manager obtains feedback on which are the elements semantically equivalent to the fields of his database. The tool is accessed through a graphical interface that allows selecting the elements of each database schema that must be mapped.

6.3 OwlBasicModel API

In order to facilitate the handling of the Master Ontology in the Semantic Mediator, we developed a Java API for creating and exploring OWL documents. The focus in this API was only including some of the most common features of the OWL-DL language for ontologies, emphasizing in speed. It also included a feature for automatically generating an RDF document from an OWL document, dropping the elements which were unnecessary for the SM. This feature was employed to generate an RDF version of the Master Ontology, which would act as the global schema for the semantic mediation layer.

7 The semantic mediator layer in the ACGT platform

The semantic mediation layer is just part of a larger technological platform aimed at managing clinical trials on cancer. There are tools for designing new trials and managing their data, and tools focused on performing advanced data analysis. Some of these tools collaborate with the semantic mediation layer components.

The ObTiMA system is used by clinical trial chairmen to design and conduct clinical trials on cancer inside the ACGT platform. This system stores all the CRF data of the clinical trials designed with it, so it implements the database wrapper technology accessed by the Semantic Mediator to integrate data. This allows performing the semantic integration of the clinical trials stored in ACGT. Further details on this topic can be found in the deliverable "D7.9: Formal procedures and protocols for the semantic integration of clinical trials in ACGT".

Security of patient's data is a very strong requirement imposed by the European Commission. In ACGT, a complete framework focused on ensuring secure data access and data transactions has been developed—the so called GAS Security framework. This framework has been incorporated to the Semantic Mediator, enabling a credential-based data access system—only users owning the necessary credentials will be allowed to access the data. More information on the GAS Security framework can be found in deliverable "D11.2: Implementation of the ACGT core security services & Initial implementation of the Pseudonymisation tool".

Finally, the Semantic Mediator offers its services to its client layers in the platform. These include a workflow editor. This module allows building complex workflows whose actions can include the access to the data offered by the Semantic Mediator. These accesses are SPARQL queries built through the Query Tool—this tool allows uploading the designed queries to a query repository for subsequent use in the workflow editor. Through these workflows, users will be able to mix data accesses through the Semantic Mediator with data processing actions, through the available data analysis tools. More information on the workflow editor can be found in deliverable "D9.2: Report on the implementation of the integrated ACGT environment and workflows".

8 Conclusions

During the four and a half years that the ACGT project lasted, we devoted to the development of a semantic mediation layer to offer heterogeneous database integration services to clinicians. The requirements included homogeneous and transparent access to multilevel data sources (including clinical patient data and genomic data). The highly evolving nature of the biomedical domain (and in concrete of the domain of clinical trials on cancer) forced us to adopt a flexible design for the mediation layer. Syntactic heterogeneities were dealt with database wrappers, so we focused on semantic heterogeneities. The development of the ACGT Master Ontology provided us the necessary semantic framework to deal with these heterogeneities, as we adopted it as the common "vocabulary" for the underlying data sources.

From the beginning, we tried to find the optimal compromise between features and performance. Query translation approaches based on Local as View often present performance problems, thus we opted for limiting the space of possible queries, ensuring that this would not affect the end user experience. Thorough tests and demos proved that this approach fitted perfectly the user necessities, as every needed query was successfully performed while maintaining reasonable response times (in the range of tens of seconds).

Another important part in the development of the semantic mediation layer was the end-user oriented tools. During initial meetings with our end-users, we understood that, together with a feature-rich system for performing integrated access to heterogeneous databases, it was equally important to offer tools that ease the access to this system. The goal was to facilitate the access to the Semantic Mediator by hiding the inherent complexity of the SPARLQ query language. To do so, we developed the ACGT Query Tool, a web-based graphical tool that allows building and submitting SPARQL queries to users lacking any knowledge on this query language. Again, demos showed that this tool achieved its goals successfully.

9 Bibliography

[Brochhausen et al. 2010] Brochhausen M, Spear AD, Cocos C, Weiler G, Martín L, Anguita A, Stenzhorn H, Daskalaki E, Schera F, Schwarz U, Sfakianakis S, Kiefer S, Dörr M, Graf N, Tsiknakis M (2010) The ACGT Master Ontology and its Applications – Towards an Ontology-Driven Cancer Research and Management System. J Biomed Inform 2010 (e-published ahead of print, DOI: 10.1016/j.jbi.2010.04.008).

[Martín et al. 2008] Martín L, Anguita A, Jiménez A, Crespo J (2008) Enabling Cross Constraint Satisfaction in RDF-Based Heterogeneous Database Integration. Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence, 2008.

[OBO] The Open Biological and Biomedical Ontologies. Available at: http://www.obofoundry.org/

[OGSADAI] The OGSADAI Project. Available at: http://www.ogsadai.org.uk/

[OWL] OWL Web Ontology Language Semantics and Abstract Syntax. Available from http://www.w3.org/TR/owl-semantics/; last visited: 7-18-2010.

[RDF] RDF Resource Description Framework. Available at: http://www.w3.org/RDF/

[SIOP] International Society of Paediatric Oncology: Nephroblastoma (Wilms Tumour) - Clinical Trial and Study SIOP 2001. Final version January 2002, ammended 2004 and 2007, EUDRACT No.: 2007-004591-39.

[SPARQL] SPARQL Query Language for RDF. Available from http://www.w3.org/TR/rdf-sparql-query/; last visited: 7-18-2010.

[TOP] http://clinicaltrials.gov/ct2/show/NCT00162812; last visited: 7-18-2010.