Integrative Cancer Research
INTEGRATE
Through Innovative Biomedical Infrastructures

SEVENTH FRAMEWORK PROGRAMME

ICT-2010-270253

INTEGRATE

Driving excellence in Integrative Cancer Research through Innovative Biomedical Infrastructures

STREP
Contract Nr: 270253

Deliverable: 2.2 Inventory of reusable/available relevant solutions and components

Due date of deliverable: (01-31-2012)
Actual submission date: (01-31-2012)

Start date of Project: 01 February 2011 Duration: 36 months

Responsible WP: Philips

Revision: final

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission Service	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (excluding the Commission Services)	

0 DOCUMENT INFO

0.1 Author

Author	Company	E-mail
Gaston Tagni	PHILIPS	gaston.tagni@philips.com
Jasper van Leeuwen	PHILIPS	jasper.van.leeuwen@philips.com
Brecht Claerhout	Custodix	brecht.claerhout@custodix.com
Kristof De Schepper	Custodix	kristof.deschepper@custodix.com
Sergio Paraiso	UPM	sparaiso@infomed.dia.fi.upm.es
Juan Manuel Moratilla	UPM	jmmoratilla@infomed.dia.fi.upm.es
David Pérez del Rey	UPM	dperez@infomed.dia.fi.upm.es
Alexandre Irrthum	IJB	alexandre.irrthum@bordet.be
Maria Stella Dolci	IJB	stella.dolci@bordet.be
Haridimos Kondylakis	FORTH	kondylak@ics.forth.gr
Manolis Tsiknakis	FORTH	tsiknaki@ics.forth.gr

0.2 Documents history

Document version #	Date	Change
V0.1	11/24/2011	Starting version, template
V0.2	11/24/2011	Definition of ToC
V0.3	01/16/2012	First complete draft
V0.4	01/16/2012	Integrated version (send to WP members)
V0.5	01/16/2012	Updated version (send PCP)
V0.6	01/20/2012	Updated version (send to project internal reviewers)
Sign off	01/31/2012	Signed off version (for approval to PMT members)
V1.0	01/31/2012	Approved Version to be submitted to EU

0.3 Document data

Keywords	
Editor Address data	Name: Gaston Tagni Partner: Philips Address: High Tech Campus 34, (6-030) 5656AE Eindhoven, The Netherlands Phone: +31 40 27 49709 Fax: E-mail: gaston.tagni@philips.com
Delivery date	01/31/2012

0.4 Distribution list

Date	Issue	E-mailer

Table of Contents

1 Introduction

One of the objectives of INTEGRATE in delivering the envisioned solution to data integration is the (re)use of standards and existing approaches and to build solutions that extend (when possible) these existing technologies. Achieving the level of data integration, both semantic and syntactic, envisioned in the project requires the (re)use and development of several technologies in different areas such as data privacy and security, storage of semantic data, information retrieval, user consent management, vocabulary management, EHR management, etc.

As mentioned above (re)use of existing technologies and standards is important in building the INTEGRATE platform. This deliverable reports on some relevant technologies that we think will be the basis for building INTEGRATE components. The deliverable extends the work of D2.1 by reporting on benchmark results of different tools and by giving an overview of other relevant solutions not mentioned in the previous deliverable. The aim of this deliverable is to give a summary of the discussions that have been going on in the project since the beginning in what regards the use of existing technologies.

1.1 Outline of this Document

This document is organized as follows: Chapter 1 introduces the motives and goal of this deliverable. Chapter 2 gives an overview of different Data Management and Data Warehouse solutions that are considered relevant for building the INTEGRATE solutions. This chapter covers semantic repositories, clinical research data warehouses and tools for extracting, transforming and loading data. In the case of semantic repositories the chapter reports on the performance evaluation of the different solutions. Chapter 3 reports on reasoning services considered relevant for reasoning over medical ontologies. It gives a short overview of the relevant systems and reports on the latest benchmark results. In Chapter 4 we report on some relevant ontology mediation tools that were not reported in the previous deliverable. Chapter 5 covers the security and privacy standards relevant to the project. In Chapter 6 two software solutions for tracking biological samples in clinical trials are described, namely caTissue and the BrEAST tracking tool. Finally, Chapter 7 concludes the document.

2 Data Management

This chapter gives a brief overview of several Data Management and Data Warehousing solutions that could potentially be used in INTEGRATE for implementing the desired solution. One such family of technologies are semantic repositories, database-like systems that support the storage, management and querying of semantic metadata. Another well-known alternative for storing, managing and querying data are Database Management Systems (DBMSs). Example of these are Oracle RDBMS, Microsoft SQLServer, IBM Informix, among others. RDBMS have become the de-facto standard for data management thanks to their performance and capabilities. For this reason in this document we refrain from reporting and introducing these systems. Another family of technologies relevant to INTEGRATE are Clinical Research Data Warehouses which are particularly tailored to managing clinical research data. Among the solutions we consider in INTEGRATE are the i2b2 platform and transMART. Finally, the chapter will report on several tools for loading, transforming and extracting data.

The aim of this chapter is to provide partners of the INTEGRATE project enough information to make an informed decision about which data management and data warehousing technologies can be used in the project for implementing the desired solution.

2.1 Semantic Repositories

This section gives a short overview of the most relevant semantic repository solutions taken from Deliverable D2.1 and extends such document with a report on the evaluation of the performance of these systems. The systems evaluated and considered are Sesame, Virtuoso, AllegroGraph and the OWLIM family of semantic repositories. Although other solutions exist (see Deliverable 2.1) in this deliverable we decided to focus our attention on the most commonly-used technologies that provide native RDF storage and relational database capabilities.

2.1.1 Sesame

Sesame is an open source, Java-based framework for *storing, processing and querying RDF* data efficiently. It was developed as a prototype application within the European project On-To-Knowledge[1] and is currently maintained by the Dutch software company Aduna[2]. Over the years Sesame has become one of the de facto platforms for managing RDF metadata, a feature that is reflected by the number of other semantic technologies that make use of it. For example, many semantic technologies vendors implement the Sesame's SAIL, a key component of the Sesame's architecture.

In terms of its (re)usability the Sesame server requires Java 5 or newer to run and a Java Servlet Container that supports the Java Servlet API v2.4 and Java Server Pages v2.0 or newer. The most common solution in this case is the Apache Tomcat servlet container[3]. The Sesame 2.0 server comes in the form of two Java web applications: A

[1] http://www.ontoknowledge.org
[2] http://www.openrdf.org/
[3] http://tomcat.apache.org/

Sesame HTTP Server and the OpenRDF Workbench. The first provides HTTP access to Sesame repositories while the second provides a web interface for querying, updating and exploring the repositories of a Sesame Server. Sesame supports four types of repositories: an in-memory repository, a native repository (on disk), an RDBMS repository solution that can store data into RDBMSs, currently only PostgreSQL and MySQL are supported and, an HTTP repository (a proxy) solution that facilitates the access to remote Sesame repositories through HTTP.

Data stored in Sesame repositories can be queried using SPARQL and SeRQL languages.

2.1.2 Virtuoso

OpenLink Virtuoso[4] is a cross-platform universal server developed by OpenLink Software. It offers a series of data and metadata facilities such as Web server capabilities, file and database server functionality, native XML storage and a Universal Data Access Middleware. The Virtuoso Universal Server comes in two flavours: Open Source and Commercial. The open source version, distributed under the GNU General Public License, differs from the commercial version in that it does not include the Virtual Database Engine (a mechanism for accessing multiple databases transparently as if it were a single database). At the time of writing, the latest version is 6.2 which can be installed on multiple platforms including Windows, Solaris, Mac OS X and Linux.

2.1.3 OWLIM

OWLIM (1) is a family of *semantic repository solutions*. It is fundamentally a DBMS that provides functions to manage (store, query and reason over) data structured according to RDF. It is the result of the research efforts partially made within several European projects including SEKT[5], TAO[6], TripCom[7], LarKC[8] and SOA4ALL[9] among others and Ontotext AD[10]. OWLIM has been used in several domains and sectors as a basic data integration platform for heterogeneous data. The domains include telecommunications, life sciences and the publishing sector. OWLIM comes in three flavours: OWLIM Lite, OWLIM Standard Edition (SE) and OWLIM-Enterprise Edition (EE).

As shown in Figure 1 (taken from[11]), OWLIM can be accessed either through Sesame or through ORDI[12], an Ontology Representation and Data Integration middleware. One of the limitations of OWLIM is that it provides its own reasoning system (TRREE) and cannot, in principle, work with third-party reasoners such as Pellet, or the more

[4] http://virtuoso.openlinksw.com/
[5] http://www.sekt-project.com/
[6] http://www.tao-project.eu/
[7] http://www.tripcom.org/
[8] http://www.larkc.eu/
[9] http://www.soa4all.eu/
[10] http://www.ontotext.com/
[11] http://owlim.ontotext.com/display/OWLIMv43/Primer+Introduction+to+OWLIM
[12] http://www.ontotext.com/ordi

efficient classifiers such as CEL, Snorocket and ELK. The easiest way to embed OWLIM in an application is using the Sesame's SAIL.

In what regards to the main differences between the three available versions OWLIM-Lite offers limited scalability (in the order of 100M triples), has no query optimization and is available for free. OWLIM-SE, on the other hand, offers query optimization, better theoretical scalability (in the order of 1 Billion triples), full-text search, RDF Rank and better (theoretical) processing speed. OWLIM-SE is distributed under commercial and free-for-academic-purposes licenses.

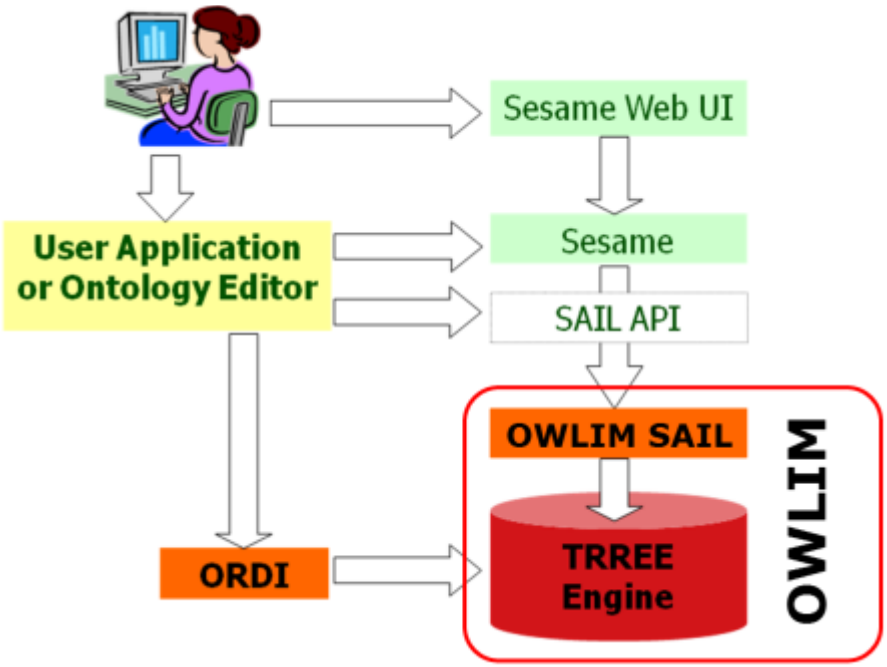

Figure 1 OWLIM and its relation to the Sesame framework

2.1.4 AllegroGraph

AllegroGraph[13] is a high-performance, persistent RDF (graph) database that supports storing, querying and reasoning over RDF-based data. More generally speaking AllegroGraph is a graph database for managing graph-based data such as for example, RDF data, social networks and, in general, any kind of data that can be structured as a graph. It was developed by Franz Inc. and is distributed as a closed-source application under a commercial licence. At the time of writing this report the latest version of the system is 4.3. AllegroGraph is available for both 32 and 64-bit Windows, Linux and Mac OS configurations. In addition, it can be deployed on the Amzaon EC2 infrastructure.

2.1.5 Comparison and Benchmarks

[13] http://www.franz.com/agraph/

The literature on benchmarking of RDF semantic repositories contains a myriad of benchmark suites, results, publications on different approaches and systems, use cases and data sets. A number of benchmark suites exist for assessing the performance of these systems in terms of various parameters such as triple loading time, query processing/answering performance, scalability, capacity to handle concurrent requests, etc. In addition, some benchmarks focus on assessing the performance of a given system based on the expressiveness of the ontological language they support for reasoning. These include benchmark analysis of RDFS as well as different fragments of the OWL family of ontology languages such as OWL-DL, OWL Lite and the OWL 2 profiles EL and RL.

Although the details and results of the benchmarks differ from one another the general approach is the same across all of them. A typical benchmark consists of a set of systems (RDF stores in this case) whose performance needs to be assessed, a series of criteria based on which performance will be measured and a suite of testing data that may include data from a given domain, metadata and queries, e.g. SPARQL queries. Typically, when benchmarking semantic repositories two tasks are evaluated: *data loading* and *query processing/answering*.

The W3C maintains a list of benchmarks and results of RDF stores. Among them we find the following:

2.1.5.1 The Berlin SPARQL Benchmark (BSBM)

This benchmark (2) provides a suite of benchmarks for comparing the performance of RDF storage systems that expose their data through SPARQL [14] endpoints and the SPARQL protocol [15]. One of the design goals of this benchmark was to facilitate the performance evaluation of semantic storage systems focusing on the query performance over large scale RDF data and not on the reasoning performance. This is based on the observation that many semantic Web applications and use cases do not require complex reasoning but efficient query processing. The benchmark have been updated since its conception in 2008. Many of the extensions made stemmed from critics from the community in regards to several aspects such as the scope of the queries with respect to the data, i.e. the queries refer to small parts of the data and the linearity of the queries that affect the metrics in large scale settings. When considering the complete picture of the Semantic Web that requires both scalable reasoning and efficient query answering methods this benchmark has an important limitation. The benchmark does not consider the reasoning performance of the systems under evaluation, although this limitation is actually by design. Nevertheless, the benchmark can be used to draw important conclusions about the load and query performance of the systems under scrutiny and can serve as a complement to other benchmarks tailored to measure the reasoning performance of these systems.

The BSBM consists of the following elements [16]:

- **Dataset**: the benchmark consist of a dataset that describes an e-commerce application where products are offered by different vendors and users search and post comments or reviews about the products. One of the limitations of this

[14] http://www.w3.org/TR/rdf-sparql-query/
[15] http://www.w3.org/TR/rdf-sparql-protocol/
[16] http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/spec/index.html

dataset, and the benchmark in general is that by fixing the application domain the results of the benchmark are somehow not general enough to reflect the performance of the systems being evaluated in a more general setting, for instance by considering the type of data and queries that other domains would impose. In other words, it is not clear how representative the set of queries is of the complexity of other domains. The dataset is represented according to three different data models. One according to the RDF triple data model, another one according to the Named Graph data model and the third follows a relational data model representation. Representing the dataset in three data models enables the analysis of systems that store RDF data according to these three data models, namely, native RDF storage systems that store data in RDF format (triples), systems that store the data directly in relational format, i.e. in relational databases and systems that provide RDF views over relational data.

- **Three query mixes**: queries of different complexity that have their roots in three use cases namely, the explore use case, the explore and update use case and the business intelligence use case. Queries are represented in three data formats, namely, RDF data model, Named Graphs and Relational Data model.
- **Performance measures**: these are used for assessing the performance of the systems under evaluation.
- A data generator and test driver tools.

The latest results [17] (of February 2011) of this benchmark show the performance comparison among the following five semantic data stores:

- BigOWLIM, now called OWLIM Standard Edition, (version 3.4.3129)
- Virtuoso (version 7.00.3200)
- BigData (version rev. 4169)
- 4store (version 1.1.2)
- Jena TDB (version 0.8.9).

It evaluates the load performance and query performance of these systems. In the following we present a brief overview of the results obtained from this benchmark. For details of the benchmark results the reader is referred to 17[17].

The **load performance** of these systems was evaluated by loading 100M and 200M triples into the stores, including both explicit data and implicit information derived off-line through forward-chaining, i.e. the systems did not perform any type of reasoning. The results of the *explore use case* show that BigOWLIM performs considerably better than any of the other four systems both when loading 100M and 200M.

Query performance was measured by running 500 query mixes and computing the number of query mixes per hour (QMpH) processed by each system both for single users and multiple users accessing the system. The results show that when dealing with a single user the query performance of all systems degrade as the size of the data increases, in this case from 100M to 200M triples. On the one hand, in the context of the *explore use case* Virtuoso shows the best performance of all the systems evaluated as it managed to process more query mixes per hour (QMpH) than any of the other 4 systems. 4Store was shown to be the second best option when it comes to

[17] http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/results/V6/index.html

query performance. On the other hand, in the context of the explore and update use case (where only three systems were evaluated) 4store shows the best performance followed by BigOWLIM.

Regarding the **scalability of the systems** the results are not complete enough to make an informed conclusion. Only two dataset sizes where considered, namely 100M and 200M triples. We think that in order to be able to infer the scalability of these systems more dataset sizes should have been used, for example, 10 different group sizes. However, from these results it is possible to see that the system that scales the best is 4store, which shows a 17.82% decrease in performance compared to 49.20% for BigOWLIM and 36.49% for Virtuoso. Although Virtuoso has the highest QMpH of all the systems in both 100M and 200M triples settings the difference in QMpH between Virtuoso and 4store is not that big which makes 4store the best of the two in terms of scalability. The numbers are as follow:

- 4store: decrease in performance (QMpH) is 17.82%
- BigData: decrease in performance is 26.07%
- Virtuoso: decrease in performance is 36.49%
- Jena TDB: decrease in performance is 36.54%
- BigOWLIM: decrease in performance is 49.20%

In regards to the **query performance of these systems when multiple users are accessing the system** in the context of the *explore use case* Virtuoso displayed the best query performance in both cases, i.e. with 100M and 200M triples. BigOWLIM was the second best system while 4store did not manage to perform with more than 1 client. This is a important result because it shows the limitations of this particular system when there is a need for serving multiple users. As in the case of single users all systems showed a natural decrease in performance as the size of the data increases.

In terms of **scalability when considering number of clients handled**, BigOWLIM appears as the system that scale the best when considering the 100M triples dataset. It shows more than 160% increase in performance when four clients are issuing queries, more than 260% when 8 clients are connected and more than 330% increase with 64 clients. To notice is how Virtuoso performs better than BigOWLIM with 4 and 8 clients and then drops its performance with 64 clients. In the case of 200M triples Virtuoso is the system that best scales w.r.t. the number of clients capable of handling and the QMpH. In addition, when considering how each system scales w.r.t. to the dataset size with a given number of clients, BigOWLIM shows the biggest decrease in performance while, although BigData has the smallest decrease in performance with 1, 4 and 8 clients (only worse than Virtuoso handling 64 clients) the absolute QMpH for each case makes Virtuoso a better performer than BigData; even with a bigger decrease in performance Virtuoso manages to handle more queries than any other system. The same can be said about BigOWLIM which, although having the highest decrease in performance still is capable of handling more QMpH than BigData and Jena TDB.

2.1.5.2 The Lehigh University Benchmark (LUBM)

The LUBM benchmark was created by researchers at Lehigh University with the aim of *"facilitating the evaluation of semantic repositories in a standard and systematic way"*. It was designed for assessing the performance of large scale repositories when

queries are evaluated against a single ontology. The benchmark is comprised of an ontology describing the domain of an university, a data generator that creates synthetic data and a set of queries expressed in a language similar to KIF. The semantics of the ontology is covered by OWL Horst.

- **Benchmark of Virtuoso**: OpenLink Software presents the results of a non-independent experiment carried out by OpenLink Software to assess the performance of Virtuoso (version 5.0.4) when loading data and processing queries using the single-server configuration and the clustered version of Virtuoso. The results [18] show that the data size does not affect the query processing time as long as the data fits in memory. The current version of Virtuoso goes even further by storing in memory double the amount of triples stored in previous versions. This is as expected as the performance decreases due to disk access when the data needs to be stored partially or completely on disk.
- **Benchmark of BigOWLIM**: Ontotext, the makers of the OWLIM family of semantic repositories have conducted experiments using the LUBM benchmark to evaluate the load and query performance of BigOWLIM. For the experiments they used different sizes of the dataset up to 90K universities. The results show the following [19]:
    - Loading Performance: The results show that BigOWLIM (version 3.1) is capable of performing reasoning (OWL Horst) over 12 Billion explicit triples at a load rate of 11K triples per second. It can also store 20B triples in a server of about 10,000 dollars. They also show that BigOWLIM is capable of handling 1.1B triples in a consumer PC worth 2,000 dollars and can load the dataset at a rate of 66K triples per second. Loading and reasoning takes in this case about 14hrs at a rate of 21K triples per second.
    - Query Processing: In what concerns query processing performance, the version of BigOWLIM tested is able to process queries with an average time of 200ms. per query.
- **Non-independent Benchmark of BigOWLIM**: The same experiments performed by the OWLIM team to evaluate the performance of BigOWLIM with different deployment configurations was conducted using the LUBM. The results [20] show that for the **loading task** BigOWLIM performs better when deployed through Sesame or Jena. In terms of query evaluation BigOWLIM performs better than Jena and when used in combination with Jena the performance of answering queries that return large results slows down.
- **Benchmark of AllegroGraph**: AllegroGraph has been evaluated using LUBM in a non-independent benchmark [21]. The system evaluated is AllegroGraph v4.0. The results of LUBM(8000) show that AllegroGraph needed a total time of 2:37:46 to load the dataset which comprises 1.1B triples; without performing static materialization as AllegroGraph uses a different approach, namely Dynamic Materialization. These results were obtained on a system with 2 x 6 core AMD Opteron Processors, 2439 SE 2.8 GHz, with 64 GB RAM, running Fedora 10. It's interesting to see how this loading time compares to BigOWLIM

[18] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSArticleLUBMBenchmark
[19] http://www.ontotext.com/owlim/benchmark-results/lubm
[20] http://www.ontotext.com/owlim/benchmark-results/owlim-jena-performance
[21] http://www.franz.com/agraph/allegrograph/agraph_bench_lubm50.lhtml

version 3.3 that in order to load the same dataset required 3.8Hrs running on a system with 1 x (Core i7 940, 2.93GHz, quad-core, HT processor), with 12GB RAM and running Solaris. This is interesting because even though the test systems are considerably different BigOWLIM requires about 50% more time to load the dataset including OWL Horst reasoning at loading time. This speaks well of BigOWLIM. In addition the query performance of AllegroGraph is better than that of BigOWLIM v3.1.

2.1.6 Summary of Semantic Repositories

Choosing a semantic repository solution is a difficult task. Several factors need to be considered including licensing, scalability (measured according to different variables), performance (measured according to different variables), need for advanced reasoning capabilities (temporal, geo-spatial, approximate, anytime, etc.), the size of the data to be managed by the repository, type of queries, number of users, among others.

One of disadvantages of systems like OWLIM is the lack of flexibility w.r.t. using a third-party reasoner. OWLIM uses a proprietary reasoner called TRREE which cannot be easily replaced by other solutions. This is important for many reasons. In particular, given the type of ontologies relevant for INTEGRATE (SNOMED CT, MedDRA, MeSH, etc.) and their characteristics classifiers such as ELK, Snorocket and CEL have proven to be highly efficient in reasoning over these ontologies. However, none of these can easily be plugged into the OWLIM family. Any solution that needs to combine the performance of these classifiers with the broader set of features offered by other reasoners such as Pellet, Racer and HermiT needs to implement the reasoning service on top of OWLIM, a task that can be daunting.

Sesame, on the other hand, provides a more flexible approach as it allows users to plug-in their own reasoning solutions by implementing the repository's SAIL. Also, Sesame has proven to be a very efficient repository for handling RDF-based data natively. Moreover, given that the type of reasoning required in INTEGRATE could be covered only by using RDFS Sesame seems like a more appropriate solution than OWLIM. However, if data size becomes an issue in INTEGRATE switching to OWLIM would be the best option given its scalability properties in terms of number of users, number of queries and number of RDF triples.

Virtuoso offers the possibility of storing the data in relational databases which allows users to rely on well-established database technology for querying and storage. Although Virtuoso does not support reasoning natively reasoning capabilities can be implemented on top of the repository through query re-writing at query time or entailment and materialization through SPARQL. The drawback of these approaches is that implementing an efficient solution for complex reasoning is a hard task.

2.2 Clinical Research Data Warehouses

This sections gives a short overview of the i2b2 (version 1.5) and transMART platform in the context of the INTEGRATE project.

2.2.1 I2b2

The i2b2 Center (Informatics for Integrating Biology and the Bedside) [22] is developing a scalable informatics framework that will enable clinical researchers to use existing clinical data for discovery research and, when combined with IRB-approved genomic data, facilitate the design of targeted therapies for individual patients with diseases having genetic origins.

The i2b2 software is designed as a set of SOA web services (called “cells” in the ib2b nomenclature, with the complete environment named a “hive”). The core services[23] are:

- **Project management**, used to provide user authentication and manage group and role information. It also keeps track of what cells are part of the hive
- **File repository**, holding large files of data including radiological images and genetic sequences. The files are generally referenced from the Data Repository Cells.
- **Identity management**, used to manage a patient's protected health information in a manner consistent with the HIPAA privacy rule. Patient data is available only as a HIPAA defined "Limited Data Set" to most of the i2b2 services.
- **Workflow framework**, used to process information in steps through various parts of the hive. Most processed information will come to reside in the Data Repository Cell or as a display to the user
- **Ontology management**, managing the terminology and knowledge information.
- **Data repository** (CRC), containing the clinical data (phenotypic and genotypic data) in a structured format. Data queries and visualizations are available through this service.

Of particular interest for INTEGRATE are (the design of) the clinical data warehouse and the ontology. Therefore, we look in this document at the data repository and the oncology service, in addition to technical considerations.

2.2.1.1 Data Repository (CRC) Cell

The data repository cell (also called the Clinical Research Chart (CRC) repository cell) is one of the core cells. Its main requirements are:

- It must be able to hold healthcare information from many different venues and allow it to be queried rapidly even if there are hundreds of millions of rows
- It must be easily combined with other project repositories to form large unified repositories
- Finally, it must allow objects to be stored that are present in the genomic data.

The CRC is designed as a data warehouse, focusing on clinical data (a patient’s phenotype and genotype information). Data in the CRC is de-identified, except for encrypted patient notes. The CRC relies on the Project Management cell for authentication and on the Ontology cell for meta-data management. The CRC provides two services, a setFinder webservice, and a PDO (Patient Data Object) webservice. *setFinder* manages a user’s setFinder queries. The queries are used to create a set of

[22] https://www.i2b2.org/
[23] https://www.i2b2.org/software/index.html

patients that satisfy the specified criteria. The API closely mimics the way the graphical user interface is designed, and setFinder queries are composed of query constraints, a list of panels and its items. The criteria put constraints on concepts from the i2b2 ontology in order to select instances from the data mart
The *PDO* webservice exposes the clinical data, providing access to patient information such as clinical observations, demographics and provider data.

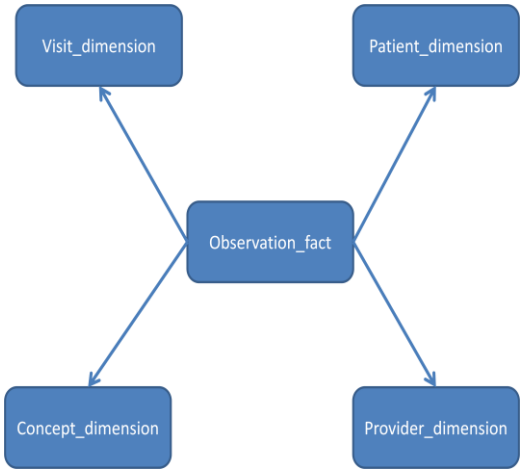

Figure 2 i2b2 star scheme overview

The data mart uses a star schema[24] as information model (see Figure 2), with patient observations as fact table, and visits, patient, concept and provider tables as dimensions. Each observation has an associated visit (a particular patient encounter), patient, concept and provider instance. An observation contains various attributes to store details such as the (measured) value of the observation, its units, the confidence interval, the begin date and end date of the observation, etc. The type of event is indicated by using the concept table combined with a modifier code (found in the Observation table). A modifier code can be used to give “context” to the indicated concept (e.g. a drug concept can be modified to indicate dose, route, and frequency). The observations made during the visit of a patient are aggregated using the visit table. The patient table uniquely identifies the patient and contains patient specific data such as demographics. And finally the provider table represents the physician or provider at the institution, involved in this observation. The PDO returns data according this information model (in xml).

2.2.1.2 The Ontology Management Cell

The Ontology Management cell manages the i2b2 ontology. The ontology follows a tree[25] structure (see Figure 3). External vocabularies / terminologies can be imported into this tree structure. The tree structure implies a subsumption relationship. There is no support for other/additional types of relationships, which has its implications when importing an ontology which supports different/multiple relationship types.

[24] http://en.wikipedia.org/wiki/Star_schema
[25] http://en.wikipedia.org/wiki/Tree_(data_structure): an acyclic connected graph where each node has zero or more children nodes and at most one parent node

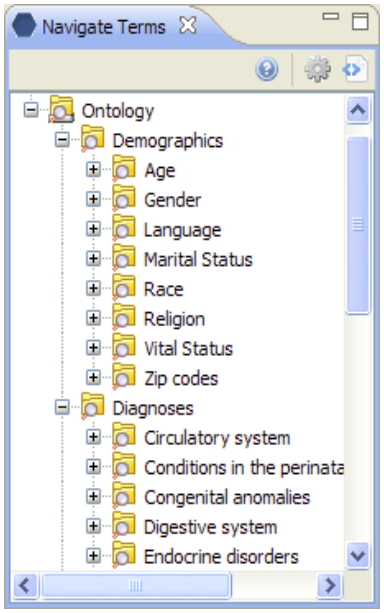

Figure 3 i2b2 ontology (screenshot)

2.2.1.3 Architectural Considerations

The upside of the i2b2 data model is that it is very simple. With relative little effort, data can be imported and queried. When looking at the Integrate use cases, it should be investigated whether the information model is not a little bit too simplistic, in a sense that we possibly lose information that we are interested in. In i2b2, relationships are not modelled explicitly. This precludes the possibility of (explicitly) relating observations (except through the "Visit" dimension). HL7 v3 RIM[26] , for example, includes compositional, reference and succeeds relationships. The RIM also includes a much more extensive "Role" object, whereas i2b2 combines persons and roles in Patient and Provider.

When looking at the ontology, there are four attention points. The first attention point is that the ontology cell only allows one type of relationship (hierarchical relationship). When importing different types of ontologies using a different type of relationship, these ontologies are "unnaturally" homogenized. The second attention point is that some (relevant) ontologies use multiple relationship types (like SNOMED CT). This information cannot be recorded in the current i2b2 implementation. In addition, some ontologies (again with SNOMED CT as example) allow for the subsumption of multiple parent nodes, which is also not supported. And finally, the i2b2 data model only references to pre-coordinated terms in an ontology, whereas (again) SNOMED CT allows for the use of post-coordinated terms, making it difficult to import data when post-coordination is used.

2.2.1.4 Technical Considerations

I2b2 uses a http REST/SOA- style integration, where XML documents are used to exchange messages between applications, and XML Schema documents define the

[26] http://www.hl7.org/implement/standards/rim.cfm

application interfaces (as opposed to the more conventional SOAP/WSDL approach). REST is “simpler” than a soap/wsdl approach. However, a lot of the complexity of SOAP/WSDL has been moved to the xml document specification. Therefore, REST is not necessarily simpler and might lack tooling support as the (content of the) xml schema definitions are not standardized.

There is quite some documentation available, but most of the documentation is rather shallow, requiring diving into the source code repository in order to understand the functionality provided by the components. Also the community does not seem very active, with a low volume mailing list and small wiki.

Another observation is that the CRC provides plug-in support. Surprisingly enough, the various plug-ins do not use the cell’s API for interaction with the CRC data, but connect directly to the database.

2.2.2 transMART

tranSMART is a knowledge management platform that enables scientists to develop and refine research hypotheses by investigating correlations between genetic and phenotypic data, and assessing their analytical results in the context of published literature and other work.

It went live at J&J at June 2009 and it is a full translational medicine warehouse. The basic components of the tranSMART, shown in Fig. 1 (taken from (3)) are the following:

- Data Repository
- Dataset Explorer

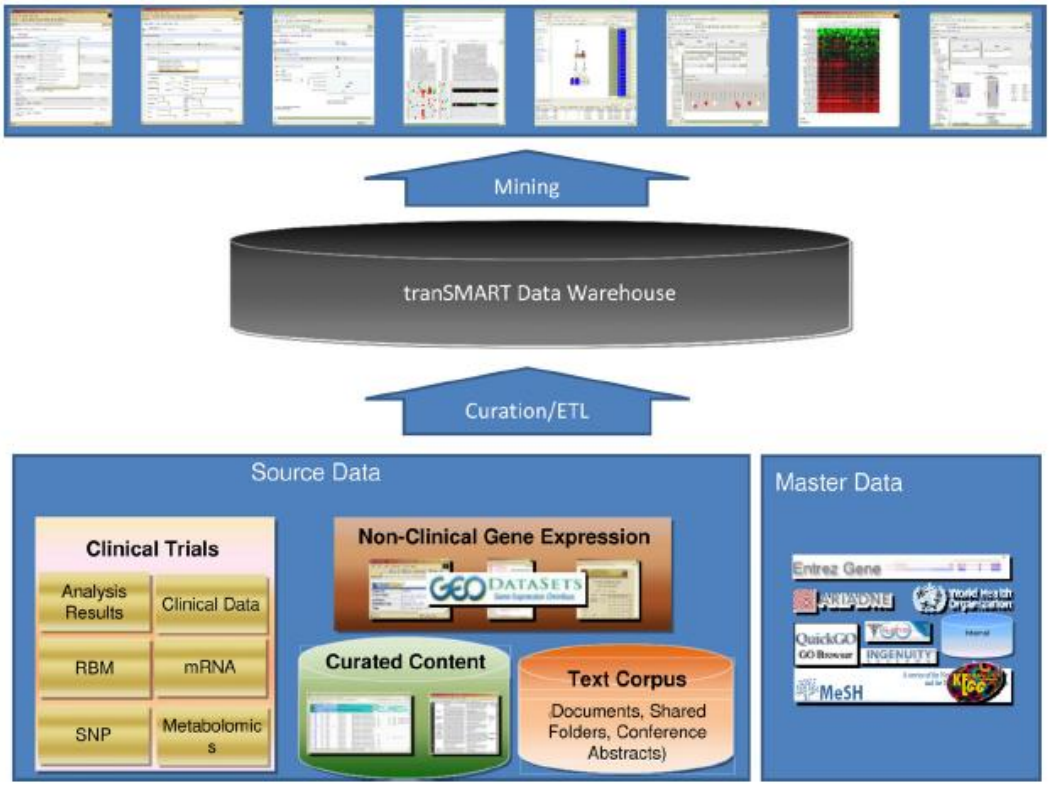

Figure 4 The tranSMART System

2.2.2.1 Data Repository

The tranSMART data repository combines a data warehouse with access to federated sources of open and commercial databases. The data that can be stored in the warehouse include:

- Phenotypic data, such as demographics, clinical observations, clinical trial outcomes, and adverse events
- High content biomarker data, such as gene expression, genotyping, pharmacokinetic and pharmaco-dynamics markers, metabolomics data, and proteomics data
- Unstructured text-data, such as published journal articles, conference abstracts and proceedings, and internal studies and white papers
- Reference data from sources such as MeSH, UMLS, Entrez, GeneGo, Ingenuity, etc.
- Metadata providing context about datasets, allowing users to assess the relevance of results delivered by tranSMART.

The data is normalized to conform with CDISC and other standards to facilitate search and retrieval. Moreover the published literature can be accessed as well to evaluate each specific analysis in the context of a broader universe of reported research. Finally external databases can be integrated as well using automated ETL tools to leverage curation in manual curation.

2.2.2.2 Dataset Explorer

The transMart Dataset Explorer provides powerful search and analysis capabilities. The core of the Dataset Explorer extends the i2b2 application, Lucene text indexing and GenePattern analytical tools. The design of the transMart allows organizations in selecting the already provided analytical tools accessible through Dataset Explorer or connect their own open source or commercial tools to expand tranSMART's capabilities.

2.2.2.3 Other Key Features

The system also incorporates role-based security mechanisms and can be integrated into an organization's existing infrastructure to simplify user management. The security model adopted allows an organization to control data access according to internal policies as well as HIPPA, IRB, FDA, EMEA and other regulatory requirements. Moreover, the solution is hosted on Amazon Elastic Compute Cloud in other to have access to unlimited disk storage and scalable computational resources.

2.3 ETL Tools: Extract, Transform and Load

Extract, Transform and Load (*ETL*) is a process that allows moving data from different sources to a new data storage system. During this process the data can be transformed according to the new data storage system structure. This process is also known as data integration process. Nowadays there are several software solutions for an ETL mapping both proprietary and open source solutions. In the next sections we describe some of the most used.

2.3.1 Proprietary Solutions

2.3.1.1 Oracle Data Integrator

It is a software focused on data transformation and merging processes proposed by Oracle, one of the most important object-relational database management systems. This system is called to replace the Oracle Warehouse Builder, a wide solution that also allows data integration and ETL processes. It has a structure called E-LT (extract-load, transform) which performs the transformation process after loading the data. The advantages of this system are more performance, efficiency and scalability.

2.3.1.2 Adeptia ETL Suite

Adeptia ETL is another proprietary solution that, though an intuitive and easy to use graphical interface, provides powerful data conversion capability, supporting several formats. It is divided in three components. The first one is a *"web-based Design Studio that provides wizard-driven, graphical ability to document data rules as they relate to validations, mapping and edits"*. The second component *"is the Central Repository where all the rules and mapping objects are saved"*. And the last one *"is the Run-time Execution Engine where the mapping rules and data flow transactions are executed on*

incoming data files and messages". Figure 5 (taken from [27]) depicts the Adeptia ETL structure.

Figure 5 Adeptia ETL structure

2.3.1.3 Other proprietary solutions

Other proprietary solutions exist. Among them we find the following:

- Pervasive Software
- IBM Websphere DataStage
- Informatica
- Adeptia ETL

2.3.2 Open Source Solutions

2.3.2.1 Pentaho Data Integration

Pentaho Data Integration (PDI), also known as Kettle, is an open source ETL software. As it can be read on its web page *"Pentaho Data Integration delivers powerful Extraction, Transformation and Loading (ETL) capabilities using an innovative, metadata-driven approach. With an intuitive, graphical, drag and drop design environment, and a proven, scalable, standards-based architecture, Pentaho Data Integration is increasingly the choice for organizations over traditional, proprietary ETL or data integration tools"*. As described, Pentaho uses a graphic interface where the transformations can be easily designed. This design tool is called Spoon. After the design, PDI allows to run the transformations in processes that it calls Jobs. The transformations and the jobs are stored in XML format. It is compatible with Windows, UNIX and Linux. Figure 6 depicts an example of the PDI Kettle interface [28].

[27] http://www.adeptia.com/products/data_transformation.html
[28] http://www.sfero.net/incuriosando/immagini_articolo/kettle05.jpg

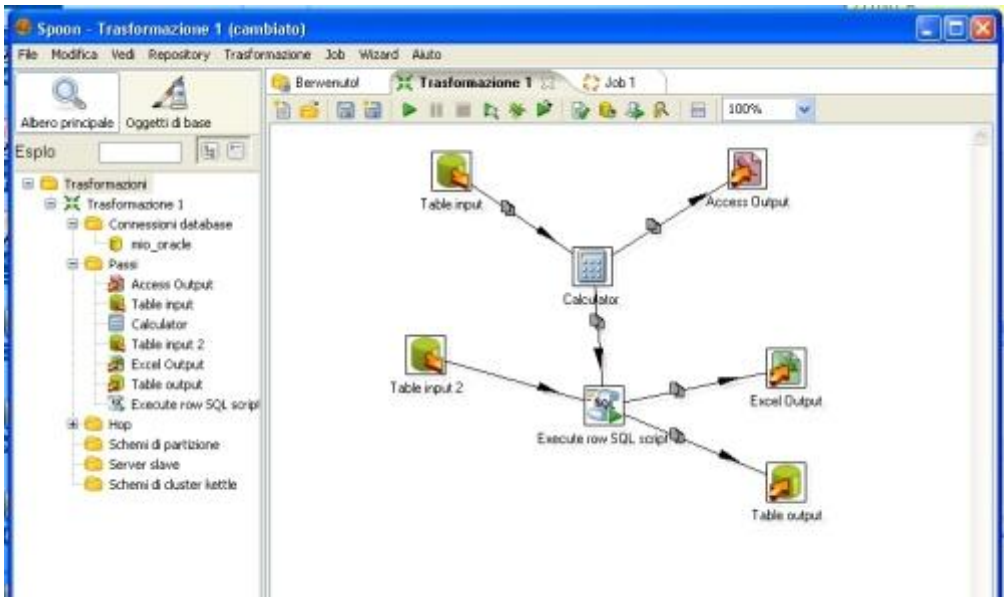

Figure 6 Example of the PDI Kettle interface

2.3.2.2 Talend Open Studio for Data Integration

Talend Open Studio is another open source ETL solution. It allows for creation and monitoring of the design for the data integration processes. Similar to Pentaho Kettle, it uses a graphic interface for designing all the ETL processes. The main difference with Pentaho Kettle is that in Talend the procedures are translated to Java or Perl before compilation and execution. This Java or Perl programs that it generates can be executed in different tools. It is compatible with Windows, UNIX and Linux. Figure 7 (taken from [29]) depicts an example of the Talend Open Studio interface.

[29] http://i1-linux.softpedia-static.com/screenshots/Talend-Open-Studio_1.png

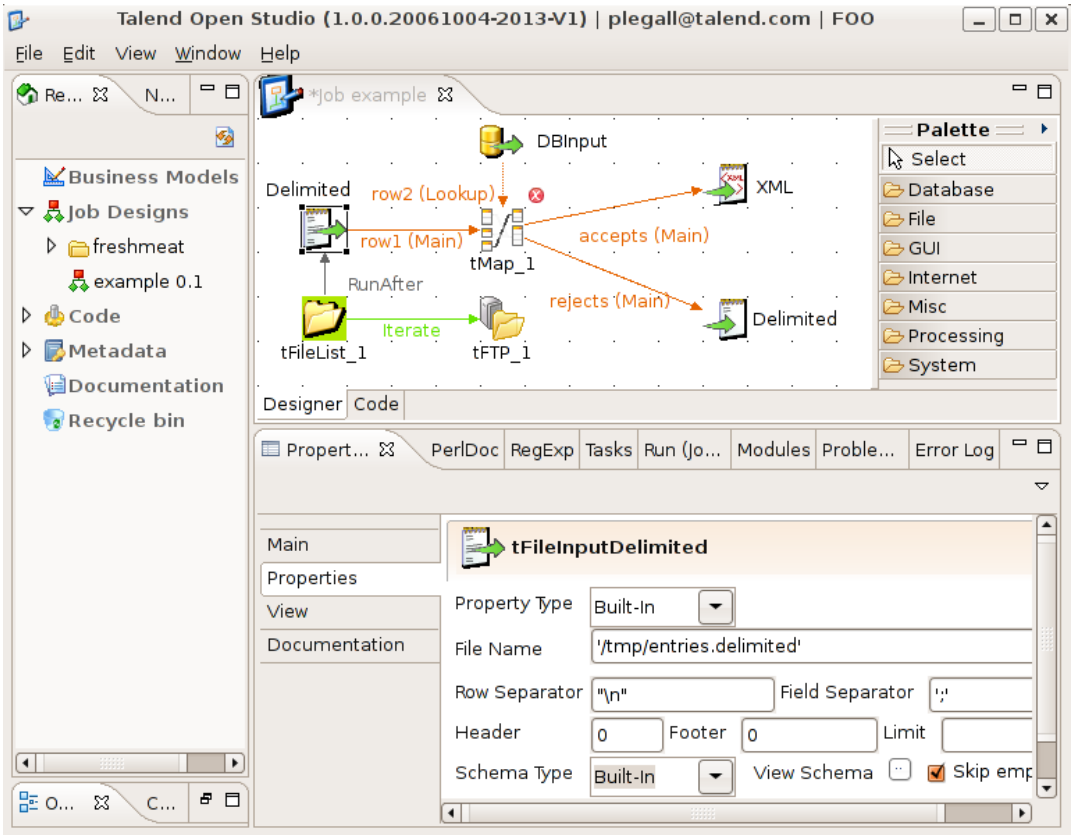

Figure 7 Example of the Talend Open Studio interface

In addition to these there exist other open source solutions such as the following:

- CloverETL
- Scriptella

2.3.3 Comparison and Benchmarks

As comparison, the most commonly used ETL solutions are Kettle (Pentaho) and Talend Open Studio. Between both of them, the GUI of Pentaho is easier than Talend's although the latter offers more options/possibilities. Talend Open Studio also has more components that allow transformation between much more different systems, on the other hand PDI has enough components that allows to cover most parts of the ETL process.

There are several benchmarks comparing Pentaho Data Integration: Kettle and Talend Open Studio with other software. The interested reader is referred to [30] [31] [32].

[30] http://www.cloveretl.com/sites/applicationcraft/files/files/case-studies/Comparison_CloverETL_vs_Talend_Pentaho.pdf
[31] http://marcrussel.files.wordpress.com/2007/08/benchmark-tos-vs-kettle.pdf
[32] http://it.toolbox.com/blogs/infosphere/etl-benchmark-favours-datastage-and-talend-28695

3 Reasoners

This chapter gives a brief overview of the relevant reasoning systems available in the Semantic Web community to support reasoning over ontologies. For a detailed overview of these systems the interested reader is referred to Deliverable D2.1. This chapter extends the report published in such deliverable by reporting on the results of several benchmarks aimed at evaluating and comparing the performance of the different solutions. The aim of the chapter is to provide partners of the INTEGRATE project enough information to make an informed decision about which reasoning system(s) can be used in the project for implementing the desired solution.

3.1 Pellet

Pellet (4) is an OWL reasoner developed by Clark & Parsia[33] that provides reasoning services for OWL ontologies with support for OWL 2.0. It is distributed under the terms of the AGPL v3 license for open source applications and under alternative license terms for proprietary, commercial closed-source applications.

Pellet is implemented in Java and its reasoning services can be accessed programmatically through its own Java API or by using one of the many bindings to common programming toolkits such as Jena and the OWL API [34]. Pellet has also been integrated in the Protégé ontology editor [35]. Additionally, Pellet implements the DIG interface which allows users to access the reasoner's services through HTTP requests. Queries can be specified in SPARQL or RDQL.

3.2 Racer Pro

RACER stands for Renamed *ABox and Concept Expression Reasoner*. It is a Description Logics reasoning system that supports reasoning over OWL Lite and OWL DL ontologies. RACER Pro is the commercial version of RACER. The system is maintained and released by Racer Systems GmbH & Co. KG [36].

RACER is distributed free for research purposes while RACER Pro is the commercial version of the system. It can be accessed programmatically through Java, C and C++ APIs and through the TCP/IP protocol. Queries in RACER can be specified in the nRQL and OWL -QL query languages. Reasoning services are provided through standard APIs including the OWL API, the DIG interface and the OWLink API.

3.3 FaCT++

FaCT++[37] (5) is a DL *tableau-based reasoning platform* that supports reasoning over OWL DL ontologies and, in its latest version, it provides limited reasoning support for OWL 2 ontologies. It is an open source project distributed under GNU LGPL.

From the (re)usability perspective FaCT++ is available as a Protégé plug-in and, additionally, its services can be accessed through the DIG and OWL API v3.2

[33] http://clarkparsia.com/pellet/
[34] http://owlapi.sourceforge.net/
[35] http://protege.stanford.edu/
[36] http://www.racer-systems.com
[37] http://owl.man.ac.uk/factplusplus/

interfaces. One of the limitations of FaCT++ is its lack of a native SPARQL query engine to support information retrieval.

3.4 HermiT

HermiT [38] (6) is a Description Logics reasoning system that is able to reason over OWL Lite and OWL 2 ontologies and provides support for OWL 2 data types. It is an open source project written in Java and distributed under the GNU LPGL[39].

HermiT's reasoning services can be accessed through a command line interface or programmatically from Java-based applications using the OWL API v3.2.2 (it is not backward compatible with the OWL API v3). HermiT can also be used from within Protégé through its plug-in implementation. It can process ontologies serialized using RDF/XML syntax, the OWL Functional Syntax, KRSS and OBO. One of the limitations of HemiT is its lack of native support for SPARQL queries.

3.5 CEL

CEL (7), *Classifier for $\mathcal{EL}+$*, is an OWL 2 EL classifier that is able to provide polynomial-time classification capabilities for types of knowledge typically found in Life Science ontologies such as SNOMED CT. The *main reasoning task supported by CEL is ontology classification*. CEL was developed by the Technical University of Dresden.

From the (re)usability perspective CEL is distributed as an open source project[40] and its reasoning services can be accessed through Protégé (as plug-in), programmatically through the OWL API and DIG interfaces and through a command line interface. What differentiate CEL from similar systems such as Snorocket and ELK is its API for querying the underlying ontology.

3.6 Snorocket

Snorocket[41] (8) is a high-performance, Java-based implementation of the same algorithm for classifying $\mathcal{EL}+$ ontologies implemented by the CEL classifier (7). It is distributed as a Protégé plug-in [42] and provides a simple API to enable third party applications have access to a fast classification service for SNOMED CT. Snorocket is specifically tailored for (incrementally) classifying the SNOMED CT clinical terminology. It was developed by the Australian E-Health Research Center and CSIRO.

From the (re)usability perspective Snorocket lacks all the additional functionalities of traditional (full-fleshed) DL reasoners, namely, instance retrieval (query capability), consistency checking, satisfiability checking, etc. This means that any solution built around Snorocket must necessarily implement all the extra functionality. Snorocket supports ontologies written in two variants of the KRSS language, the SNOMED CT distribution format and the OWL 2 functional syntax.

[38] http://hermit-reasoner.com/
[39] http://www.gnu.org/licenses/lgpl.html
[40] http://lat.inf.tu-dresden.de/systems/cel/
[41] http://aehrc.com/hie/snorocket.html
[42] http://protegewiki.stanford.edu/wiki/Snorocket

3.7 ELK

ELK [43] (9) is an ontology reasoner (classifier) developed by the Knowledge Representation and Reasoning group at the Department of Computer Science of the University of Oxford in collaboration with the University of Ulm under the context of the European project *ConDOR: Consequence-Driven Ontology Reasoning*. ELK supports reasoning over the $\mathcal{ELHR}+$ fragment of OWL 2 EL. The main feature that distinguishes this reasoner form other EL classifiers and reasoners is its use of parallel computing techniques to spread the computation over multiple CPUs or cores.

From the (re)usability perspective, ELK is implemented in Java and distributed as an open source project under an Apache 2.0 license. It can be accessed through its command line interface and programmatically through the OWL API. Additionally, two plug-ins, one for Protégé and the other for the Snow Owl [44] ontology editor are available. Compared to Snorocket, ELK supports two reasoning tasks, namely classification and consistency checking. Similar to Snorocket it supports the OWL Functional Syntax and does not provide native query capabilities. The main disadvantage of ELK is that it does not provide a native GUI, although its services can be accessed through ontology editors such as Protégé and Snow Owl.

3.8 Comparison and Benchmarks

In this section we report on the results of several benchmarks designed to evaluate the performance along different criteria of the reasoners introduced in the previous section. The goal is to highlight the relative performance of these systems with the aim of helping users in choosing among the different alternatives.

One of the common benchmark suites used for evaluating the performance of reasoners is the University Ontology Benchmark suite (UOBM) (10) which extends the well-known LUBM suite with new axioms to create two new variants, namely an OWL Lite ontology and an OWL DL ontology. This benchmark aims at evaluating the scalability and inference capabilities of OWL reasoners with respect to various reasoning tasks and the expressiveness of ontologies. Both LUBM and UOBM are based on synthetically-generated ontologies.

A more recent evaluation initiative has been proposed by the European SEALS project [45]. This project aims at providing infrastructure to evaluate semantic technologies. One of its campaign, the Storage and Reasoning Systems Evaluation Campaign includes test cases for standard reasoning services.

3.8.1 Comparing HermiT, Pellet and FaCT++

In early experiments with HermiT (6) the authors evaluate and compare the performance of the reasoner against two other state-of-the-art DL reasoners, Pellet (version 1.5.1) and FaCT++ (version 1.1.10). Both, Pellet and FaCT++ reason using tableau calculi and implement many of the standard optimization techniques for DL

[43] http://code.google.com/p/elk-reasoner/
[44] http://www.b2international.com/portal/snow-owl
[45] http://www.seals-project.eu/

and tableau-based reasoning. HermiT differs from these two reasoners in the type of inference, more specifically the completion rules, used for building the tableau and on the specific blocking techniques used.

The benchmark was performed using ontologies used in real-life applications in the medical domain. The experiments were conducted over several variations of the GALEN ontology (11), ontologies from the Gardiner ontology suite (12) and using the Biological Ontologies (OBO) foundry[46]. The performance of the reasoners was measured in terms of the classification time. The experiments showed that *HermiT is capable of performing as good as or better than the other DL reasoners* tested, specially for complex ontologies. Another important point to highlight from these experiments is the fact that many of the reasoners were not able to classify some of the ontologies. This behavior was attributed to the characteristics of the ontology and in particular to the presence of cyclic axioms that cause an explosion on the size of the models which eventually leads to memory shortage. Another thing to note is the fact that *HermiT performed relatively better than Pellet and FaCT++* when *classifying simplified versions of the GALEN* medical ontology. For details of the classification times the reader is referred to (6).

During the 2010's SEALS Storage and Reasoning Systems Evaluation Campaign FaCT++, HermiT and jCel[47] were compared in terms of their classification, class and ontology satisfiability times and entailment results. With respect to classification time jCel has a better average reasoning time (ART) than FaCT++ and HermiT although with less correct results. The same pattern was observed w.r.t. the ontology classification task.

3.8.2 Comparing HermiT, Racer Pro and OWLIM

The benchmark of OWL reasoners conducted in 2008 by (13) measures the performance of reasoners w.r.t. several reasoning tasks in terms of their *total response time or, more accurately, performance time*, i.e. the time it takes a reasoner to carry out the entire given task; which in turn is divided into loading time and the actual response time (i.e. *total response - loading time*). Implicitly, every task is reduced to solving a query. For example, when reasoners are evaluated w.r.t. the Abox consistency task the response time is the time it takes the reasoner to solve a query that triggers the Abox consistency check. The data sets and ontologies used for the evaluation are representative of their respective OWL fragments although relatively less complex, in general, than real-life ontologies such as those used in the medical domain, e.g. GALEN, SNOMED CT, the Gene ontology, etc. Nevertheless the ontologies used in the experiments allow for assessing how well reasoners handle different OWL fragments w.r.t. several reasoning tasks.

The results of this experiment point at *HermiT as the reasoner with the best classification time* while *RacerPro outperforms the rest in loading time*. This suggests that *for applications where the ontologies are loaded off-line HermiT should be the*

[46] http://obofoundry.org
[47] http://jcel.sourceforge.net/index.html

right choice. However, *in applications where the distinction between loading time and classification time is not important then RacerPro would be the best choice*. With respect to conjunctive query answering OWLIM has the best overall performance for OWL DLP and RDFS(DL) while KAON2 has the best overall performance for OWL DLP. Sesame, on the other hand, has the best query answering time for RDFS(DL).

The results also show that OWLIM performs very well in answering conjunctive queries over lightweight ontologies (RDFS(DL) and OWL DLP). However, for more expressive ontologies such as those expressed in the OWL DL and Lite fragments KAON2 has been shown to outperform tableau-based methods. These experiments have shown that in general for *very expressive ontologies Racer Pro and KAON2 represent a better choice* with *KAON2 being a better choice for large Aboxes. OWLIM is a good choice for less expressive ontologies*. The final conclusion of this experiment is that to choose the right reasoner for a given task one needs to take into consideration the specifics of each reasoner and the characteristics of the ontology and reasoning tasks. Also, rule-based reasoners scale well for large Aboxes but are limited in terms of the expressivity they can handle. On the other hand, tableau-based methods can handle very expressive logics but do not scale well to large Aboxes (instance data). For the actual experimental results the interested reader is referred to (13).

3.8.3 Performance on Large Datasets

A more recent benchmark of various OWL reasoning systems have been conducted by (14) (see also (15)). The goal was to evaluate the performance of OWL reasoning platforms with respect to large volumes of data, to measure the correctness of the reasoners in terms of soundness and completeness and to assess the performance of the systems depending on the communication interface used to interact with them. The experiments were conducted using the UOBM and the systems tested were Pellet (v2.0.0RC5), RacerPro (v1.9.3b), FaCT++ (v1.2.3) and KAON2 (version of June, 2008). HermiT was initially considered but later discarded as it failed to process the UOBM data sets.

In tune with the observations made from other benchmarks the results are mixed without a clear performer across all settings (data sets, tasks and queries). More specifically, when *loading data the results showed fairly similar performance for all systems*. In terms of *realizing the Abox* (finding the most specific concept each individual is an instance of) *RacerPro outperformed every other system in almost all cases*. In terms of correctness the results showed that almost every reasoning system failed the tests by providing incorrect answers or consuming more resources than the imposed limit. Some systems showed incompleteness of results. *The system that performed best was RacerPro*. Also important to notice from these results is how many of the systems failed to handle even simple OWL Lite ontologies. Incorrect results (unsound behavior) were traced back to implementation bugs and/or ontology features, such as inverse properties, that made reasoning hard for all the systems. In some test cases for example, *Pellet and FaCT++ produced incomplete results*.

The results also showed that some systems behave better when accessed through certain types of interfaces such as using RacerPro through the OWLLink interface. In general, the results indicate that the performance of a reasoning platform may be influenced by the interface used for accessing the reasoning services. This could be attributed to the maturity of the interfaces implemented by the reasoners and/or the

complexity of each interface. On the positive side the results highlight the gradual improvement made by different implementations of some of the reasoners over time, especially in the capacity to handle ontologies of increasing expressiveness. The bottom line is, however, that reasoning technology is still rather immature and that improvements must still be made both in terms of new reasoning techniques and optimizations and in the development process as well to avoid errors.

3.8.4 Performance with respect to OWL 2 EL

The work in (16) addresses the question of which class of reasoners perform better when dealing with biomedical ontologies such as SNOMED CT. It investigates through a series of experiments whether reasoners built for very expressive ontology languages such as Pellet, FaCT++, Racer, etc. perform better than the family of reasoners (classifiers) specifically designed for the type of ontology language underlying the type of ontologies found in the life sciences and biomedical domain. In particular it focuses on the comparison between reasoners and classifiers for ontologies specified using the OWL 2 EL fragment.

The experiments focus on terminological reasoning over OWL 2 EL ontologies and thus are concerned with reasoning tasks that manipulate the terminological part of ontologies. Reasoners are evaluated with respect to the following tasks: classification, Tbox consistency check, concept satisfiability and concept subsumption. These tasks were evaluated in the context of well-established medical ontologies, namely the Gene Ontology, SNOMED CT and NCI, the National Cancer Institute thesaurus.

The results obtained in these experiments highlight the strengths and weaknesses of the reasoning systems considered. First of all, an important result of these experiments covers the completeness and soundness exhibited by the systems. Although they implement methods that have been proven to be sound and complete in theory their implementation reflects the opposite. In particular, the versions of Pellet and Snorocket tested were shown to be unsound and not complete when evaluated against SNOMED CT, although subsequent versions solved the issue.

A second result, regarding the classification time the results vary depending on the ontology used although the clear winner is CB, the *consequence-based reasoner* implemented by the University of Oxford that implements a decision procedure for Horn SHIQ ontologies (17). Snorocket exhibited relatively good results as well. Also important to highlight is how well specialized (lightweight) reasoners (i.e. those designed for less expressive logics that exploit the structure found in many life science ontologies) such as Snorocket, TrOWL, CB and in less manner CEL perform compared to reasoners built for more expressive logics such as Pellet, Racer Pro, FaCT++ and HermiT. As expected classification time increases for all reasoner as the expressiveness of the ontology language increases.

Third, an interesting result of these experiments is the behavior of the reasoners when computing the sub classes of a given class (in the SNOMED CT ontology). Here, a discrepancy in the number of returned answers is observed for some reasoners that depends on whether concepts are referred to by its name or by the anonymous class that denotes the concepts. This seems to indicate a bug in some of the reasoners.

The results of these experiments also reflect the advances in the area of automated reasoning in the biomedical and life sciences domain. All the reasoners evaluated in these experiments managed to classify SNOMED CT, a well-known ontology that not so long ago many reasoners failed to classify (and many other still struggle with).

3.8.5 Highly-expressive and Lightweight Reasoners

The first experimental results of the performance of the CEL system was reported in 2006 (7). In this work CEL was benchmarked and compared against three state-of-the-art DL reasoners based on tableau methods, namely Pellet, RacerMaster and FaCT++. The results showed that CEL is capable of performing at the same level as the other three reasoners and in many situations even above the performance levels of these. This is an important result because it highlights the difference in performance between tableau-based methods and other methods for classifying ontologies and the suitability of CEL for classifying real-world ontologies in the life sciences domain. As put by the creators of CEL these results also serve as "*a strong argument for the use of tractable DLs based on extensions of EL*".

In the work of (18), in 2010, the performance of Snorocket is assessed and compared against that of Pellet, FaCT++ and CEL using ontologies from the life sciences domain. The ontologies used in the experiments are the Gene Ontology (GO), the Foundational Model of Anatomy (FMA), the GALEN medical knowledge base and the SNOMED CT ontology. In the experiments Snorocket managed to outperform the other reasoners in the context of a classification task, especially when the size of the ontology increases, in which case many of the other systems failed to finish. This work also evaluates the performance of Snorocket in incremental classification. Although the experiments show a very interesting performance of Snorocket when classifying SNOMED CT incrementally, the experiments are rather incomplete as no comparison is made with other reasoners and only one ontology is tested. Further analysis should be done to provide a more robust argument in favor of this reasoner. This, however, does not hide the merits of the reasoner.

In a recent work (9), the performance of ELK was evaluated and compared against that of other classifiers and highly-expressive reasoners, in particular against CB, FaCT++, Pellet and Snorocket. The experiments also show how the performance of ELK (measured in terms of classification time) improves as the number of processing cores increases. The experiments were conducted using SNOMED CT, GALEN, FMA and the Gene Ontology (GO). The results show that even with one core (worker) ELK's performance is equal or better than that of the other systems. When more than one core is used ELK outperforms every other system by a factor of at least 50%. As the number of workers increases from 3 to 4 the gain in performance starts to stabilize. These results make ELK the preferred alternative for dealing with medical ontologies, specially with SNOMED CT.

3.8.6 Summary of Experimental Evaluations

None of the existing reasoners are able to successfully classify either GALEN or FMA, two complex and large medical ontologies used in practice. In general, experiments have shown that reasoning performance is affected by the combination of several

factors. For example, although more expressive ontologies tend to greatly affect the performance of reasoners sometimes less expressive ontologies demand more computational resources, e.g. time, than more expressive ontologies with a larger number of axioms or classes. In other words, the expressiveness of an ontology does not necessarily mean bad news for a reasoner. Another general finding from these experiments is that no reasoner outperforms every other reasoner across ontologies and tasks.

In the end, the choice of which reasoner to use depends very much on the characteristics of the ontology, including expressiveness, size and the reasoning task one is interested in. For conjunctive query answering, for example, a determinant factor is the expressiveness of the ontology. However, since the expressiveness of an ontology is determined by a combination of features such as the number of existential quantifications, the number of *owl:sameAs* axioms and the number of inverse roles, among others and, to the best of our knowledge, no studies have shown that a single of these features affects reasoning performance alone the choice of which reasoner to use is in practice rather difficult. In the end the selection must be made taking into account a combination of factors that not only include the characteristics of the ontology but also the type of queries most commonly used in the application domain, the available resources, etc.

For biomedical ontologies CB, Snorocket, CEL and TrOWL exhibit considerably better performance than other traditional reasoners with respect to the classification task. Although no independent evaluation of the performance of ELK is yet available, naturally due to the fact that the classifier is relatively new, a few experimental evaluations conducted by the ELK's team show that the reasoner is able to outperform similar systems in classifying biomedical ontologies such as SNOMED CT. This has also been verified within INTEGRATE by conducting similar experiments and comparing its performance with that of CEL and Snorocket.

Finally and as a summary of the experiments and results found in the literature it is clear that there is no silver bullet for choosing the right reasoning system for an arbitrary application domain. Ultimately, the decision should be made based on an analysis of several characteristics, such as the ones mentioned above, and on the specific requirements of the application at hand which include the required reasoning tasks, the available resources and the complexity, type and size of the target ontologies among others. In particular, for INTEGRATE, it is necessary to carefully identify the expressiveness of the ontologies that will be used and that of the core data model and, the reasoning tasks involved in the different use cases.

4 Ontology Mediation

A mapping process consists in the transformation of data from a source to a target. More specifically, ontology mapping consists of mapping components, classes or attributes from one ontology (source) to the components, classes or attributes of another (target) ontology.

In deliverable 2.1 we reported on several technologies for ontology mediation. One such type of technology not reported in that document are ontology editors which supports ontology mediation. Three such tools are the following:

- **Snoggle**: Snoggle [48] *"is a graphical, SWRL-based ontology mapper to assist in the task of OWL ontology alignment. It allows users to visualize ontologies and then draw mappings from one to another on a graphical canvas. Users draw mappings as they see them in their head, and then Snoggle turns these mappings into SWRL/RDF or SWRL/XML for use in a knowledge base"*.
- **OntoEdit**: OntoEdit is a tool for the edition of ontologies. It does though an open graphical interface in a web environment. Due to the extensibility, it allows to the users adjust it to their requirements. The main purposes of OntoEdit are, offering a tool to graphically represent the ontologies that can store and manipulate them in a relational data base.
- **Visual Ontology Modeler**: As it can be read on its website, *"Visual Ontology Modeler is a visual application for building component-based ontologies. It is a UML-based modeling tool that enables ontology development and management for use in collaborative applications and interoperability solutions"*.

Other similar systems and tools include Protégé [49], Optima [50] (19) and Onion (20).

[48] http://snoggle.semwebcentral.org/
[49] http://protege.stanford.edu/
[50] http://cs.uga.edu/~uthayasa/Optima/Optima.html

5 Security and Privacy Standards

5.1 PIMS

PIMS (Personal Information Management System) offers a central service for tracking patient identifiers originating from different data sources. By assigning pseudonyms, the system supports unique identification of a patient in different administrative domains. Re-identification of pseudonyms to identifying information and cross-domain linkage of identities are controlled by a role based access control system. This makes them fully auditable.

PIMS incorporates a configurable probabilistic matching engine based on the standard Fellegi-Sunter statistical matching algorithm. This algorithm makes it possible to link records representing the same real world patient, hereby compensating for incomplete or differently structured information. For pair-wise comparison of fields, PIMS makes use of a variety of fuzzy matching algorithms, most of which are based on the Jaro-Winkler metric. This allows dealing with various types of data entry errors (typos, misspelling, ocr scanning, etc.). Altogether, these algorithms provide a good trade-off between efficiency and efficacy.

PIMS will be used in the INTEGRATE platform as identity manager; guarding and linking the different domain ids (hospital id, screening id and research id) from a patient. In this way the research domain is kept separated from the screening domain.

The currently implemented functionality of PIMS is probably sufficient for the INTEGRATE platform. This means that the "biggest" cost for using this component in INTEGRATE will be the integration in the platform.

5.2 CATS

CATS (Custodix Anonymisation Tool Service) is a service, developed by Custodix, that meets the de-identification requirements of different types of projects by providing a central anonymisation engine. More Specifically, in CATS, users can transform given input files (clear text, CSV, XML, DICOM, etc.) based on a fully configurable set of transformation rules (called a privacy profile). The main transformation functions are clearing of data in a file, pseudonymising identifying information, encryption of sensitive data, string replacement, etc. CATS evolved from CAT which offered a local standalone anonymisation tool instead of a service.

Because of the modular nature of the CATS platform, the set of already supported standard data formats can be extended with new formats. The CATS service can be invoked by using one of the provided interfaces:

- A web interface: An end-user can upload files for transformation through a web front end.
- A Web service interface: CATS is equipped with a web service layer (SOAP, WSDL), secured with SAML tokens.
- CATS client tool: A user can launch a client side CATS tool (Java web start) to process files locally before uploading to CATS.

In INTEGRATE, CATS is responsible for de-identifying the different data that will be used in the research domain of the platform. Each data file that enters this domain is transformed by the CATS service before it can be stored in the data warehouse. Using this approach, the research data warehouses contain only anonymous data, meeting the legal requirements for INTEGRATE.

The currently implemented functionality of CATS is probably sufficient for the INTEGRATE platform. This means that the "biggest" cost for using this component in INTEGRATE will be the integration in the platform.

5.3 Shibboleth

Shibboleth[51] is an architecture and implementation of a federated Single Sign-On authentication and authorisation infrastructure heavily coupled with SAML. The primary function of the Shibboleth system is to support identity federation between multiple sites using the SAML protocol standard. Shibboleth's added value lies in support for privacy, business process improvement via user attributes, extensive policy controls, and large-scale federation support via metadata. Hence Shibboleth accommodates richer and more complex metadata distributed by a federated operator. It has more refined capabilities for managing trust implicit in larger communities. It allows users and enterprises to manage attribute releases, reflecting the greater number and variety of participants.

The implementation part of Shibboleth offers an implementation of three main components[52] meeting the SSO profile and protocol requirements:

- The **Identity Provider** (IdP) is an entity that authenticates principals and produces assertions of authentication and attribute information.
- A **Service Provider** (SP) is an entity that gives access to resources.
- Next to these two components there is also an optional **Discovery Service** component. This component can keep track (in case of multiple IdPs) of the IdP that was selected by a user, using a browser cookie.

Shibboleth is developed by Internet 2, a networking consortium containing people of different domains (communities, industry and government). The main objective of Internet 2 is to develop and maintain a leading-edge network.

In the INTEGRATE platform, Shibboleth will be the central authentication component. When a user wants to use a service of the INTEGRATE platform that is protected, he is redirected to the Shibboleth identity provider where he can authenticate him/herself. This IdP invokes and sends a security token that is validated by the services. Next to the security token, the IdP can also send additional authorisation attributes in the responses to the services.

The current version of Shibboleth is focused on web browser applications, meaning that there is limited support for web service standards like the WS-* specifications. For the INTEGRATE platform this will mean that Shibboleth needs to be extended with

[51] https://wiki.shibboleth.net/confluence/display/SHIB2/Home
[52] http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf

WS-* functionality to provide an IdP that can handle web service requests. The Single Logout profile is also not implemented in the Shibboleth framework, this because there are no technical solutions that meet the requirements of this profile.

5.4 A XACML Engine

The XACML Decision Engine provides an authorisation service by making access control decisions for incoming access requests. These decisions are the result of evaluating user defined policies with the incoming requests.

The engine is an implementation of the OASIS XACML de-facto standard (see XACML section Deliverable 2.1), meaning it provides complete support for all the mandatory features of XACML. Specifically, there is support for parsing policy and request/response files, decision making for incoming requests using the policies and determining applicability of policies. The engine will also support some specific features like contextual attributes and the role based access control profile (21) of XACML.

The XACML Decision Engine will be the central component for authorisation in the INTEGRATE platform. All access control requests are evaluated by this engine using the policies that contain the INTEGRATE access rules determined in the security model (see Deliverable 2.3/2.4).

The standard XACML functionality of the engine can be provided by a third-party implementation like Sun[53] or JBOSS[54] XACML engine. Next to this standard functionality, the engine will probably need extensions in order to provide support for contextual attributes.

5.5 LDAP

LDAP (Lightweight Directory Access Protocol) is an application protocol that can be used to access and maintain distributed directory information services over an internet protocol network. It is commonly used to manage large groups of users. The core of the protocol is defined by the Internet Engineering Task Force (IETF) in RFC4510 (22). The directory information services contain information that is organized in a hierarchical directory structure. This information can be queried and filtered so that only the required information is returned. LDAP directory is a "write once, read many times" service.

The LDAP hierarchy is similar to a directory structure on a file system. The data in the directory is organized similar to DNS domain components. A domain fp7-integrate.eu becomes *"DC=eu, DC=fp7-integrate"*, where DC stands for domain component. This makes the merging of data easier. Further separation of the content is achieved using logical separations called Organizational Units (OU). The most important aspect of LDAP is the possibility to have fine-grained control over the use of passwords. This means that users with a higher degree of access can be forced to have more secure passwords. The passwords itself are managed with password policies.

[53] http://sunxacml.sourceforge.net/
[54] http://community.jboss.org/wiki/PicketBoxXACMLJBossXACML

LDAP is a good solution to use as user credential store as part of the user management services in INTEGRATE (mainly due to the presence of The flexible password policies, which take a high implementation effort when building from scratch). For this the INTEGRATE platform will include an already existing implementation of the LDAP protocol like OpenDS[55] or OpenLDAP[56].

As far as we can see now the server part of the LDAP solution will not need a lot of modifications. The client side will have a higher contribution cost. Modifications and extensions will be done in the API and GUI in order to fully support the LDAP protocol stack. Next to these implementation costs, there is also a standard integration and deployment cost.

[55] http://www.opends.org/
[56] http://www.openldap.org/

6 BioTracking

Tracking of biological samples in clinical trials and their traceability on location and availability are essential features in the changing landscape of translational research associated with clinical outcomes.

Moreover, when any analytical results are associated to the collected samples, the strict compliance with regulation on data privacy protection is mandatory.

Existing software solutions are focusing mainly on the tracking of samples in repositories for large analytical laboratories or bio banks. One example is the Mosaic Sample Management Software[57] that has the following features:

Remote Ordering

The customers of the sample bank can place orders in terms they understand – a list of the samples they require and the form in which the samples should be supplied. Mosaic determines whether the samples are available in the required form for immediate despatch and, if not, initiates the necessary workflows to create them.

Inventory Tracking

Inventory information is robustly tracked in the inventory database, so you always know exactly what you have and where it is.

Workflow Management and Integration of Robotic Workstations

Mosaic's workflow manager orchestrates all manual and automated processes that handle the samples. Staff are guided through the individual operations, and data is automatically passed to robotic workstations. This eliminates human error and ensures that the correct methods and procedures are executed on the workstations, through every step of every sample preparation process.

Software like Mosaic are managing samples that are already in a bio bank but do not handle the workflow of samples from the set-up of the study protocol to the actual delivery of the sample to a bio bank, what we could call the upfront workflow.

The upfront workflow should consider the following aspects:

- Which samples are required to be collected by a study protocol? Tumour tissue, whole blood, serum, plasma, DNA, etc.
- The attributes of each sample, as defined by the study protocol, like the conditioning of a tumour tissue (frozen or formalin fixed paraffin embedded), the optimal storage temperature (-20°C or -70°C), the quantity of each sample in different degrees (how many frozen samples, or how many aliquots of serum, or, probing down, how many ml of serum for each aliquot).
- The time points for the collection of each sample (baseline, week 2, Month 3, etc.).
- For some studies the type of samples and their collection time points could be dependent on the treatment arm assigned to the patient (example: 3 time points or sample types for patients in arm A and 4 time points or sample types for patients in arm B), or to the type of treatment the patient is receiving prior to the experimental treatment (example: in a study testing a new biological agent,

[57] http://www.titian.co.uk/products/mosaic-management-software/

Arm A, against a standard one, Arm B, following a chemotherapy treatment with or without anthracyclines, the collection time points could be different if the patient received prior anthracyclines or not, regardless of the treatment arm assigned at randomisation, A or B).

- Test expected to be performed on the samples. This is a continuous aspect, some of the tests can be requested by the study protocol but others are stored for future undefined analysis.
- The location of the samples and the ordering of transfer from one location to another (from participating site to a central lab, or from the central lab to the bio bank).
- Laboratories that will perform the analytical assessments, in order to defined the transfer workflow.
- The availability of the samples (example: 10 serum sample aliquots collected for a patient of which 2 were used for IGF1 determination, 1 accidentally destroyed, 7 still available) to provide summary reports and to quantify if the number of patients with available samples is sufficient to test a hypothesis or is sufficient to satisfy the needs of all the research proposals.

Other features include the possibility to report analytical results with customised entry forms, to download the results in an excel or CSV format and to give to the participating sites the possibility to print-out test results produced by a central laboratory.

6.1 CaTissue

6.1.1 Overview of the CaTissue Suite

The cancer Biomedical Informatics Grid (caBIG®)[58] is an initiative launched by the American National Cancer Institute (NCI) to "*create a virtual network of interconnected data, individuals and organizations that work together to redefine how cancer research is conducted.*" As part of the initiative a toolset of software component to support fundamental and clinical cancer research have been developed. Table 1 shows a sample of these software solutions.

caBIG tool name	Brief description
caAdapter	Mapping and transformation among data sources, e.g. from HL7 v2 messages to HL7 v3 messages
caIntegrator	Integration clinical and experimental data across studies
caArray	Microarray management system
C3D	Clinical trial data management system
caTissue	Bio specimen management

Table 1 caBIG software solutions

Here, we focus on the caTissue Suite, the bio specimen management solution from caBIG, as it is deemed particularly relevant for the INTEGRATE project and incorporates most of the features that one expects from a sophisticated bio specimen management system. A very extensive documentation of caTissue (including user,

[58] https://cabig.nci.nih.gov/

technical and deployment guides, data models, presentations and tutorials) can be found on the caBIG website, and we will only give an overview of this tool here.

caTissue is a web application that includes the following functionalities:

- Management of information pertaining to collection, storage, annotation, labelling, tracking and quality assurance of multiple types of samples (tissue, biological fluids, DNA, RNA and protein)
- Management of sample processing events, including creation of aliquots and derivative samples
- Management of sample movements between repositories, shipment, requisitions and requests
- Linking of pathology annotations and textual pathology reports
- Specialized interfaces for bio repository staff and research scientists
- Multi-step and multi-arm sample collection workflows, including skip-logic form generation
- Patient study registration and tracking of informed consent status at the sample level
- Powerful queries, including queries based on temporal relationships, parameterized queries, and a query “wizard”.
- Bulk download of sample data from spreadsheets
- Scalability to large, multi-site repositories
- Sophisticated access control based on user roles and privileges, and support for SSO (Single Sign On)
- Integration with existing IDPs (identity providers)

6.1.2 Technical Characteristics

caTissue Suite is a web-based J2EE application with a database backend. As of March 2011 (version 1.2), it can be deployed on 32- or 64-bit Windows servers (certified for Windows 2000 and XP) or on Linux servers (certified for Red Hat 9 or Red Hat Enterprise ES/AS 2.1 or higher). It has the following software pre-requisites:

- The Java Development Kit JDK, version 1.5
- The application server JBOSS, version 4.2.2 GA
- A relational database management system MySQL 5.0.45 or Oracle 9i or 10g
- The Ant build tool, version 1.7

The supported browsers are Internet Explorer 6/7, Firefox 2.0 and Safari 3.1. Note that caBIG recommends deploying the web application and the database server on two different machines. Moreover, integration with external software components is possible through the provided Java API (Application Programming Interface), which allows remote creation, modification and querying of the data. caTissue Suite is freely accessible, with a liberal software license[59] and an open source approach.

59

https://wiki.nci.nih.gov/display/caTissuedoc/caTissue+Suite+User's+Guide+Copyright+and+License+v1.2

6.1.3 Potential Limitations of caTissue Suite for INTEGRATE

caTissue Suite is a powerful and flexible application. It is able to scale to large numbers of samples. For example, at Washington University at St Louis, more than half a million samples were managed across repositories by a single installation (as of September 2010).

However, because it is powerful and flexible, caTissue Suite is also complex. Customization to suit the needs of INTEGRATE might thus require considerable investment in studying the underlying code and data models, and considerable additional development.

caTissue Suite uses its own data models for sites, users, patients, trials, etc. which might make integration with the core data model of INTEGRATE difficult. In particular, integration of patient and sample identifiers will have to be evaluated more thoroughly.

Finally, security in caTissue Suite is managed through a dedicated module developed as part of the caBIG tools (CSM, the Common security Module) and which provides a unified solution for user authentication, authorization, and user provisioning. It will be necessary to evaluate if this solution is compatible with the sophisticated security solutions that will be implemented for the INTEGRATE platform.

6.2 BrEAST Biotracking Tool

The BrEAST Biotracking was developed in 2008 to facilitate the screening of patients to assess their eligibility for a large multicentre international trial in adjuvant breast cancer with a sample size of over 8000 patients (the ALTTO trial) and a smaller neoadjuvant trial with 455 patients (the NeoALTTO trial), to track the biological samples for translational research and their transfer to laboratories for analysis or to a bio bank for storage for future use.

Since then, the application has been upgraded and it is now in use for another adjuvant breast cancer trial.

Biotracking specification requirements (server):

- Apache web server version 2 or newer
- PHP 5.1 or newer
- Oracle database version 9 or newer
- SFTP server
- Windows 2003 server or newer

Biotracking specification requirements (client):

- Google Chrome 12 or above
- Mozilla Firefox 4 or above
- Internet Explorer 7.0 or above
- Safari 4.0 or above
- JavaScript must be activated
- Popup must be allowed

The system depends also on external sources of data:

- The study sites contact details, provided by the Sponsor, used to populate the site contacts table. Set-up at study start and maintained/updated along the study.
- The Sponsor's sample ID: some pharmaceutical companies have their own logistic specifications, used for the kits preparation, which include a sample ID specific to the type of sample and the collection time point (example: serum at baseline=201, serum at week 2=202, etc.). Set-up at study start.
- The screening and randomisation files, provided in CSV format by the entity performing this service. Depending on the study, there can be one file including the patient ID, date of birth, the screening date, the randomisation date and stratification factors or two files, one including the patient ID, date of birth and screening date and the other the patient ID, randomisation date and screening factors. The arm is present or not depending whether it is a blind or unblended study.

For this last aspect, it means that the transfer of data from the randomisation system must be implemented and tested in advance. The data are transferred to a SFTP server and a specific program is automatically uploading the data in an oracle table.

The access roles are:

- Study site: could be a research nurse, or a pathologist, or an investigator or sub-investigator, no distinction on the roles at the sites (edit, update and delete).
- Central lab: to acknowledge receipt of the samples and report analysis results. Central Lab users will have access to specific entry forms as per type of analysis.
- Monitor: only browsing privilege and restricted access to sample list and related PDF forms (central lab results form or translational sample tracking form)
- Translational research coordinator: full browsing privilege to samples list and edit/update to samples shipment logistic forms.

The users request the access via the BrEAST web portal[60] and are re-directed to a registration form where they have to report their contact details and specify their role. A BrEAST control module is used to define the creation of user accounts. The BrEAST control module may be extended in order to be compliant with specific study access requirement.

[60] https://www.br-e-a-s-t.org/

The structure of the underlying database is a mix between relational and EAV tables, see screenshots of the schemas below (Figure 8 and Figure 9).

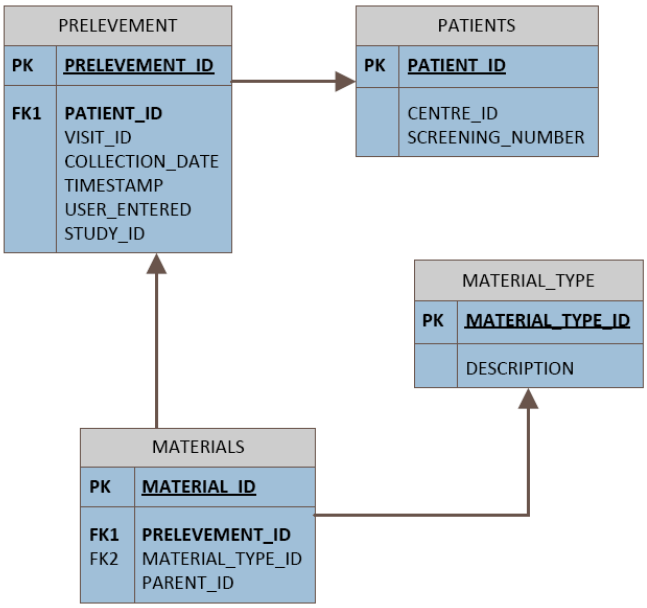

Figure 8 Sample of relational tables

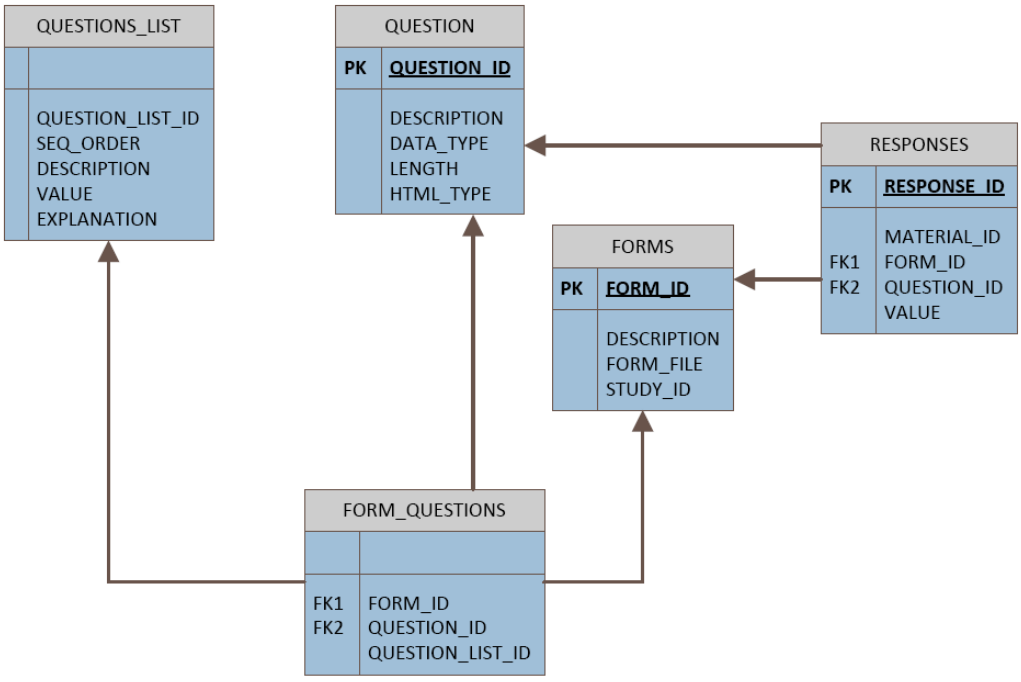

Figure 9 Example of EAV tables

Specific functionality requirements within the context of the workflow for a trial requiring a screening assessment of the HER2 status are given below.

6.2.1 Examples of Functionality Requirements

The central lab assessments can be done by more than one lab. The collection of samples can be dependent of the type of treatment received. For example, a study randomising patients to receive either a new monoclonal antibody (Arm A) or a standard one (Arm B) following either an athracyclines based chemotherapy (A-b) or a non-anthracyclines based chemotherapy (n-A-b), has collection time points dependent on the type of chemotherapy used (n-A-b or A-b) prior to the biological treatment, regardless of the biological treatment assigned at randomisation (Arm A or Arm B).

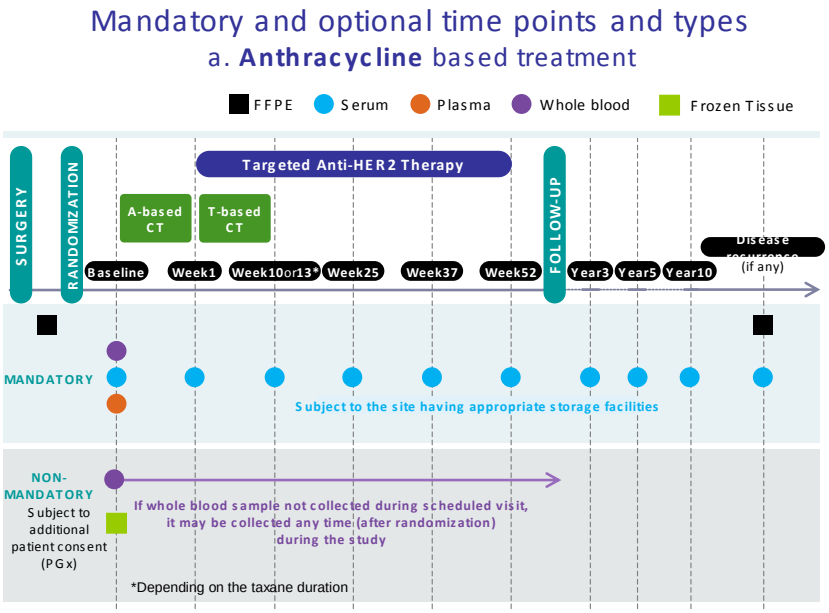

Figure 10 samples flow-chart for patients receiving anthracyclines (screenshot)

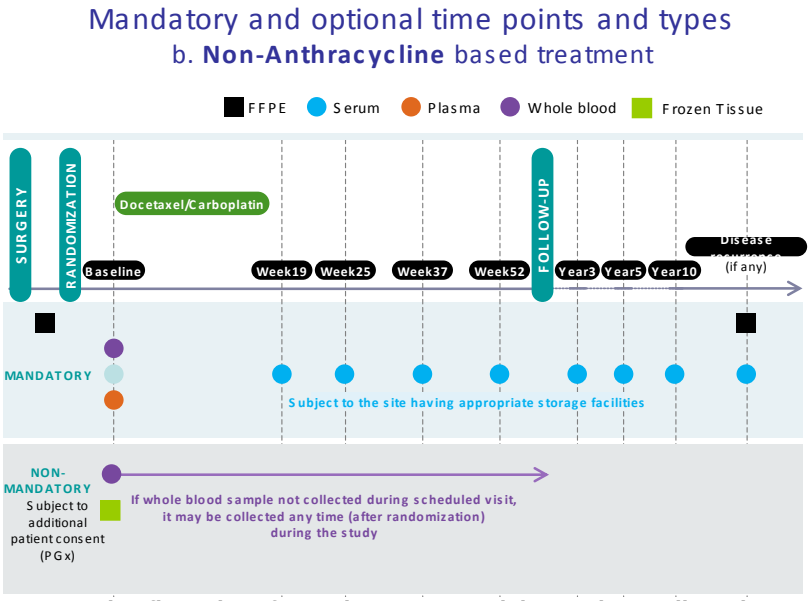

Figure 11 samples flow-chart for patients not receiving anthracyclines (screenshot)

Samples for PK analysis have more than one withdrawal time point on the same day and different time points in one country in respect to all other participating countries. This is possibly due to gain additional data (ADME) on a specific population to avoid future bridging studies, as described by the ICH topic E5 (R1) regulation (Note for guidance on ethnic factors in the acceptability of foreign clinical data (CPMP/ICH/289/95)). The Samples attributes are specific for each type of sample. Skip-logic functions to show or hide entry fields are depending on the sample type. For example, if the sample type selected for tracking is “Tissue”, the “Conditioning” will be displayed and the drop-down list will have the options “Paraffin embedded” or “Frozen”, while if the samples is “Serum”, the “Conditioning” field will be hidden, same for the field “laterality”, which is applicable only for the sample “Tissue” but not for any of the blood samples.

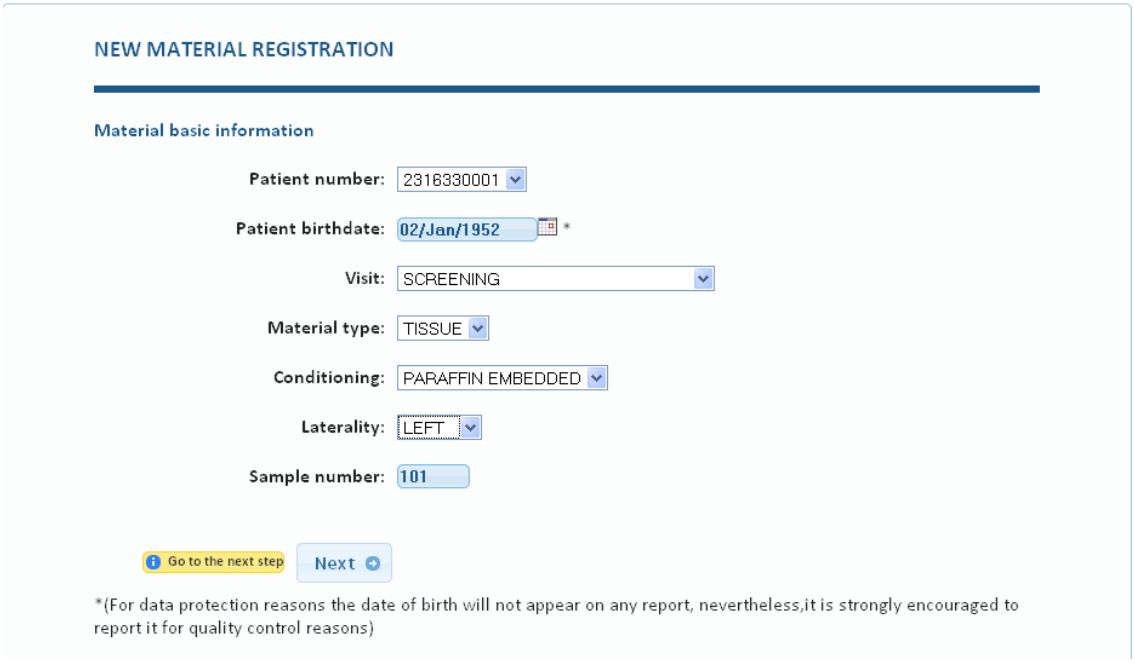

Figure 12 Sample tracking form 1 (screenshot)

The second form is for the completion of the sample's attributes that goes more into specific items like collection date, storage temperature or number of aliquots. The storage temperature is particularly important because sites that do not have a -80°C freezer and store samples at -20°C, should be monitored for a quicker turn-around samples transport to the bio bank.

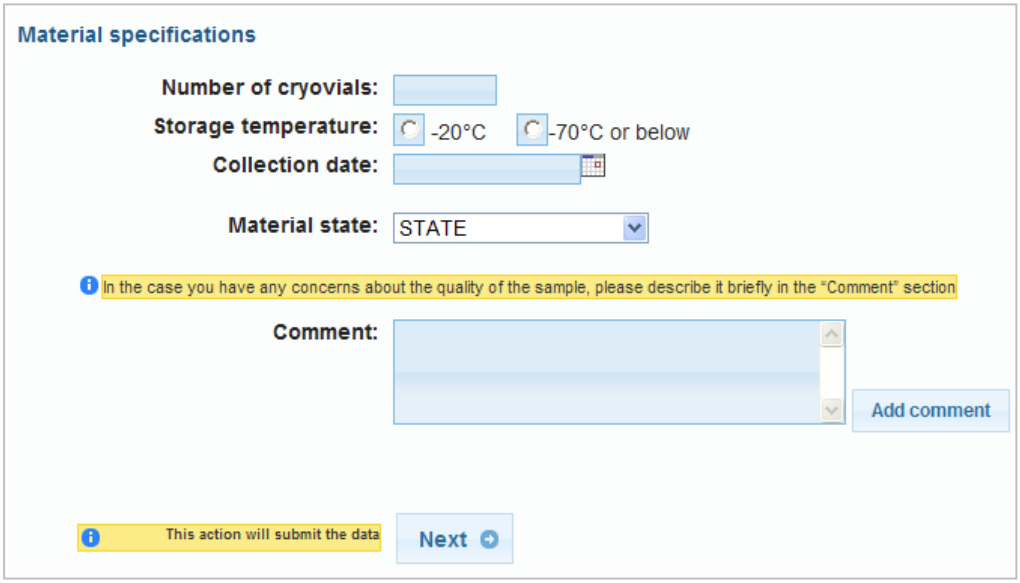

Figure 13 tracking step 2 (screenshot)

Pathologist contacts are also collected to facilitate the communication between central and local laboratory. Transport from one location to another is regulated by the application. The 3 steps are:

- Translational research (TR) supervisor requests samples to site
- Site confirm availability
- TR send request of shipment to currier

6.2.2 Biotracking Limitations

The BrEAST biotracking has two main limitations that might make its re-use in the context of INTEGRATE difficult:

- Lack of standardized semantics. While some of the concepts used in the BrEAST biotracking are coded according to standardized terminologies (e.g. sample types encoded with CDISC codes), some other important concepts are not. For example, pathology results such as HER2 status are encoded with proprietary codes, which cannot be automatically mapped to the standardized (SNOMED-CT) codes that will be used for the rest of the INTEGRATE platform.
- Fine optimization to specific clinical trials. The BrEAST biotracking has been used so far to support the operations of three clinical trials (ALTTO, Neo-ALTTO and APHINITY), the data model and software have been highly optimized for these trials. Consequently, it might be difficult to generalize it in order to accommodate clinical trials with different characteristics.

7 Conclusions

Regarding the (re)use of semantic repositories, Sesame and OWLIM seem to be the appropriate choices for the project given the flexibility of Sesame in building extensions and the performance delivered by OWLIM when handling large volumes of data. The bottom line is that any choice of a repository will have to be made based on the reported empirical results and in-house evaluation of these technologies to assess their performance against the type and volume of data used in the project. In addition to this, the reasoning requirements must be taken into account given that most repositories provide a reasoning service. Concerning ETL tools Kettle (Pentaho) and Talend Open Studio seem to be the appropriate solutions for this type of tools as they offer a balanced set of features and performance and efficiency.

In terms of reasoning systems ELK, Snorocket and to a lesser extend CEL, have a clear advantage over highly-expressive solutions such as FaCT++, HermiT, Pellet and Racer. This is due to the fact that their design is tailored to the specific features of the kind of medical ontologies that are likely to be used in the project.

In regards to privacy and security, the currently implemented functionality of PIMS and CATS is probably sufficient for the INTEGRATE platform. This means that the "biggest" cost for using this component in INTEGRATE will be the integration in the platform. On the other hand, Shibboleth will need to be extended with WS-* functionality to provide an IdP that can handle web service requests. The Single Logout profile is also not implemented in the Shibboleth framework, this is because there are no technical solutions that meet the requirements of this profile. The same holds for LDAP and XACML where some work to extend these solutions are needed in order to fully integrate them within the INTEGRATE platform.

Regarding biotracking solutions the BrEAST biotracking tool’s limitations, namely its lack of standardized semantics and its optimization tailored to specific clinical trials, may hinder the possibility of using such tool within the project. As for th ecaTissue Suite its customization to suit the needs of INTEGRATE might require considerable investment and additional development. Moreover, because caTissue uses its own data model the integration with INTEGRATE’s data model is an issue that will require considerable work. On the positive side caTissue Suite is able to scale to large numbers of samples.

8 References

1. *OWLIM: A family of scalable semantic repositories.* **B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, R. Velkov.** 1, 2011, Semantic Web – Interoperability, Usability, Applicability, Vol. 2, pp. 33-42.
2. *The Berlin SPARQL Benchmark.* **C. Bizer, A. Schultz.** 2, 2009, International Journal on Semantic Web & Information Systems, Vol. 5, pp. 1-24.
3. *Knowledge Management in Translational Medicine.* **S. Szalma, V. Koka, T. Khasanova, E. D. Perakslis.** 8, 2010, Journal of Translational Medicine, Vol. 68.
4. *Pellet: A practical OWL-DL reasoner.* **E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, Y. Katz.** 2, 2007, Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 5, pp. 51-53.
5. *FaCT++ description logic reasoner: System description.* **D. Tsarkov, I. Horrocks.** 2006. Int. Joint Conference on Automated Reasoning (IJCAR). pp. 292-297.
6. *HermiT: A Highly-Efficient OWL Reasoner.* **R. Shearer, B. Motik, I. Horrocks.** 2008. 5th Int. Workshop on OWL: Experiences and Directions.
7. *CEL: A Polynomial-time Reasoner for Life Science Ontologies.* **F. Baader, C. Lutz, B. Suntisrivaraporn.** [ed.] N. Shankar U. Furbach. s.l. : Springer-Verlag, 2006. IJCAR. Vol. LNAI 4130, pp. 287-291.
8. *Fast Classification in Protege: Snorocket as an OWL2 EL Reasoner.* **M. Lawley, C. Bousquet.** 2010. Australasian Ontology Workshop.
9. *Concurrent Classification of EL Ontologies.* **Y. Kazakov, M. Krotzsch, F. Simancik.** s.l. : Springer, 2011. 10th International Semantic Web Conference (ISWC11). Vol. LNCS 7032.
10. *Towards a Complete OWL Ontology Benchmark.* **L. Ma, Y. Yang, Z. Qiu, G. Xie, Y. Pan, S. Liu.** [ed.] J. Domingue Y. Sure. 2006. The Semantic Web: Research and Applications. Vol. LNCS 4011, pp. 125-139.
11. *Goals for concept representation in the GALEN project.* **A. L. Rector, W. A. Nowlan, A. Glowinski.** 1993. SCAMC.
12. *Automated Benchmarking of Description Logic Reasoners.* **T. Gardiner, I. Horrocks, D. Tsarkov.** s.l. : CEUR-WS.org, 2006. International Workshop on Description Logics. Vol. 189.
13. *Benchmarking OWL Reasoners.* **J. Bock, P. Haase, Q. Ji, R. Volz.** 2008. ARea2008 Workshop on Advancing Reasoning on the Web: Scalability and Commonsense.
14. *Who the Heck is the Father of Bob? - A Survey of the OWL Reasoning Infrastructure for Expressive Real-World Applications.* **M. Luther, T. Liebig, S. Bohm, O. Noppens.** 2009. ESWC.
15. **Liebig, T.** *Reasoning with OWL - System Support and Insights.* Ulm University. 2006.
16. *Comparison of Reasoners for large ontologies in the OWL 2 EL Profile.* **K. Dentler, R. Cornet, A. ten Teije, N. de Keizer.** 2, 2011, Semantic Web, Vol. 2, pp. 71-87.
17. *Consequence-driven Reasoning for Horn SHIQ Ontologies.* **Kazakov, Y.** 2009. 21st International Conference on Artificial Intelligence (IJCAI). pp. 2040-2045.
18. *Fast Classification in Protege: Snorocket as an OWL2 EL Reasoner.* **M. Lawley, C. Bousquet.** 2010. The Sixth Australasian Ontology Workshop.
19. *OPTIMA: Tool for Ontology Alignment with Application to Semantic Reconciliation of Sensor Metadata for Publication in SensorMap.* **R. Kolli, P. Doshi.** 2008. IEEE International Conference on Semantic Computing. pp. 484-485 .

20. *An algebra for semantic interoperability of information sources.* **P. Mitra, G. Wiederhold.** 2001. IEEE 2nd International Symposium on Bioinformatics and Bioengineering Conference. pp. 174-182.
21. **Anderson, A.** *Core and hierarchical role based access control (RBAC) profile of XACML v2.0.* OASIS. s.l. : OASIS, 2005. p. 23.
22. **Zeilenga, K.** *Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map.* OpenLDAP Foundation. 2006. RFC 4510.
23. **S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. F. Patel-Schneider, L.A. Stein.** OWL Web Ontology Language Reference. *World Wide Web Consortium.* [Online] 10 February 2004. http://www.w3.org/TR/owl-ref/.