Documentation of the Implementation Experiences of an ACGT compliant analytical service

Project Number:	FP6-2005-IST-026996
Project Acronym:	ACGT
Project Title:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Instrument:	Integrated Project
Deliverable name:	Documentation of the Implementation Experiences of an ACGT compliant analytical service
Period covered:	From 1st February 2007 to 31st January 2008
Start date project:	1st February 2006
Duration:	48 Months
Submission Date:	20/5/2008

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	N/A
Document name:	Documentation of the Implementation Experiences of an ACGT compliant analytical service
Document type (PU, INT, RE)	RE
Version:	Pre-Final
Submission date:	20/5/2008
Editor: Organisation: Email:	Vassilis Virvilis Biovista vassilis@biovista.com

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

ABSTRACT:

This technical report presents experience gain in implementing the prototype integrated demonstrator for ACGT. It records the issues and solutions adopted by the 4 main partners involved in the implementation of the prototype and is intended to serve as a guide for third parties aiming to deploy their own applications in the ACGT environment.

KEYWORD LIST: Reference implementation of ACGT services, implementation experiences, issues and difficulties

MODIFICATION CONTROL			
Version	Date	Status	Author
1.0	11/4/08	Pre-Final	Vassilis Virvilis
1.5	19/5/08	Final	Vassilis Virvilis
2.0	20/5/08	Final	Vassilis Virvilis

List of Contributors

All WP leaders have made contributions of material regarding the status and work performed by the WP they are responsible for, have reviewed the pre-final version of the report and provided comments, which assisted in the production of the final version.

As a result the list of contributors is:

- Stelios Sfakianakis (FORTH)
- Stefan Castille (Custodix)
- Erwin Bonsma (Philips)
- Andreas Persidis (Biovista)
- Manolis Tsiknakis (FORTH)

Contents

1 Introduction

The ACGT software environment represents a non-trivial setup that can present developers who wish to create and deploy ACGT-compatible solutions with a number of challenges. These challenges exist at the data integration level, the software set-up level, and the security compliance level to name but a few.

To complicate matters even more, the underling technologies themselves represent a moving target since they too evolve and their integration into a single functioning solution in the manner that is required by ACGT has not been attempted before.

The effort required to successfully deploy an integrated solution that functions as per its specifications is significant. With this in mind, we have put together this document in order to capture the experience of the team involved in the creation of the prototype implementation of a “reference service”, demonstrated at the project review of December 2007. The prototype implementation was primarily based on Biovista’s literature mining services which for the demonstrator were coupled with the security modules developed by Custodix and the Workflow Enactor developed by FORTH. In addition prototype implementation work was carried out in connection with the data access services which aim to transparently connect clinical data from distributed sources in member organizations (hospitals, research institutes and companies).

It is hoped that this experience will help other developers of the consortium but also of third parties successfully navigate the labyrinth of technologies, hardware set-up, security protocol set-up and other related choices that have to be made in order to create a functioning module that works as intended within the ACGT environment.

This document is the first of a series of similar documents that by the end of the project will capture the entire collective experience of the consortium in developing ACGT-compatible modules and will be made freely available to interested parties.

The remainder of the document is organised as follows:

- Section 2 presents the Biovista services that were used to create the functionality of the prototype. It then describes the main technological issues that had to be resolved in order to (a) deploy these services over the Grid and (b) make them compatible with the security infrastructure of ACGT
- Section 3 discusses the main issues that arose in connection with the workflow Enactor.
- Finally, section 4 presents issues that arose in connection with the deployment of the data access services within the grid environment.

2 Biovista Services

Biovista Services Overview

The prototype demonstrator is based on two main classes of Biovista services: the Literature-mining services and the Dictionary services

In order to familiarise the user with these reference services we begin by presenting a brief overview of each.

2.1.1 Literature Mining Services

For the reference implementation we used 2 methods in the literature mining services class. A brief overview of these follows.

2.1.1.1 getBibliography()

Description: This method returns the specified maximum number of pubmed ids. The method searches the most frequent occurrences of the desired search term (entity name). The caller must also specify the type of the search term i.e (name = cancer, type = disease). See below for a list of available types. In case of Invalid type or wrong input the method returns zero hits.

- Input parameter: search_term, the entity name to look for
- Input parameter: the type of the search_term
- Input parameter: The max_results output is the upper bound of the size of the returning list
- Output: Returns a list of pubmed_ids containing the specified entity

2.1.1.2 getCoocurance()

Description: This method returns the specified maximum number of most frequent co-occurred terms with the specified search term. The called must specify both the search term, and the destination type. See below for a list of available type. In case of non-matching or invalid input the method returns zero hits

- Input parameter: search_term, the entity name to look for
- Input parameter: the type of the search_term
- Input parameter: the type of the of co-occurring terms
- Input parameter: The max_results output is the upper bound of the size of the returning list
- Output: Returns a list of entities that co-occur with the specified search term.

2.1.1.3 getAvailableTypes()

Description: This is a convenient method for the caller in order to gain insight to the valid types, the above services know about.

- Input parameter: None
- Output: Returns a list of all the available types the previous methods know about .
-

2.1.2 Dictionary Services

The main objective of the dictionary service is to help a user to select the proper search term to query the system. The usefulness of the dictionary is best illustrated in the following scenario.

Let's assume that the user doesn't know the exact spelling of the desired term or that the user mistypes the term in the query input. With the dictionary available the 10 closest matches available to the system are presented to the user in order to correct the desired search term.

Currently the primary client of the Dictionary services is the Biovista Ontology viewer.

General Issues

In deploying the literature mining services described in section 1, there were 3 kinds of issues that had to be dealt with:

- Set-up issues
- IDE issues
- Security infrastructure issues

However one should note that all the problems that are presented in detail below stem mainly from a common source. The technology stack (namely the combination of web services, grid services etc) that is used is new, evolving and as a consequence somewhat immature with a lot of overlapping and contradicting features and implementations. On the other hand since the chosen underlying platform is constantly gaining in momentum and is actually attracting new developers and other interested parties, it has been selected by the consortium as a suitable base.

2.1.3 Setup Complexity

The required software stack in order to build webservices (ws) with the globus toolkit is:

1 java

2 tomcat

3 globus-toolkit

The globus-toolkit builds the WSRF functionality on top of Axis1 (not the newer and non deprecated Axis2) thus effectively introducing yet another layer of indirection. The problems with these multiple layers are threefold:

- Configuration complexity
- Problem discovery
- Installation complexity

The whole software stack is difficult to configure properly since the configuration elements are scattered among the system files of each part of the stack.

When the software stack is not properly configured the error report is a) obfuscated and non descriptive b) hidden in a multitude of log files located at three different places. This effectively inhibits the development cycle which is largely based on trial and error.

The installation is non obvious since the resulting webservice should be distributed within the *WSRF* directory of the Globus toolkit and not directly as a tomcat webapp. Furthermore the installation procedure resets any previously saved configuration setup in the Globus toolkit.

The Globus Toolkit employs the Axis1 web service framework to perform the main tasks required for WS integration such as communication, deployment, error handling etc. The Axis1 framework itself acts as a container for the developed webservices. Normally Axis1 expects the web services to publish in the form of a packed ".aar" file. At this point one needs to remember that *tomcat* acts as a container requiring ".war" files. Of course the Globus

Toolkit is also a container requiring ".gar" files. This means that the developer needs to understand the order of expansion of subcontainer modules.

The order of expansion goes as follows. First the ".war" file is unpacked by tomcat. The globus toolkit webapp starts up, initializes itself and starts listening for incoming connections. All "*.gar" files inside are unpacked and these are actually the grid services offered. The unpacking of the "*.aar" files does not take place, since this happens implicitly during the unpacking of ".gar" files. This non-standard use of Axis[1] by the Globus Toolkit means that some knowledge of Axis1 is not directly usable which places an additional burden for the selective use of this knowledge on the developer.

In connection with the testing of modules, current browsers are not able to invoke WSRF web services like normal Axis1 web services, thus making light testing impossible and effectively requiring a functional client for every service deployed in order to validate that the web service is working properly.

Finally a very important parameter of the setup is the specific location of the error and activity logs of java, tomcat, grid services and web services. This becomes even more important in complex java/tomcat setups where multiple loggers (commons-logging,) are available and quite probably overlapping. The problems in this instance are:

- it is not obvious from which resource each of these gets its configuration data,
- this configuration data can be partially hardwired and partially existing in a configuration file
- different loggers (e.g. commons-logging) may use other loggers as a backend which means that it is very difficult to identify which is the active configuration and the solution can only be found by detailed source code inspection or by trial and error.)

2.1.4 Lack of IDE (Integrated Development Environment)

Since Globus and WSRF is a relatively new technology, the integration with popular IDEs is almost non existent. The normal workflow imposed by the toolkit is different from a common java project, since it involves parsing the WSDL and auto generating new java code which must be compiled alongside with manually written code.

The above procedure either requires custom ant scripts that cannot be generic because they should have a intimate knowledge of the system (e.g. paths.) or help of an IDE that could streamline and speedup the development cycle.

The solution employed in the case of the prototype was to use a plugin for the eclipse IDE[1] with the proviso that since the technology is not really mature it is not recommended for production use.

[1] ihttp://mage.uni-marburg.de/trac/gdt/wiki

Security Integration

The security integration of the grid infrastructure is designed to be transparent to the user assuming the user in this case is the application developer of the services offered. In order to start with the security integration one must have an already working setup (see section 2.2).

From that point on the security integration requires the addition of foreign libraries in a tomcat setup. This is directly translated to extra cost for the maintainers of the system that tomcat runs since subsequent updates may render the working setup non-functional.

2.1.5 Testing Procedure

During the early stages of the development the validation and the testing required 2 people in two different organizations (Custodix and Biovista) to synchronize their activities. The difficulty to achieve local testing impacted the speed of the development cycle which would have been less, if the involved partners had begun with local testing first and then continued with remote authentication.

Local testing of the authorization framework is possible by using a gridmap file. A gridmap file is a local file describing the access policy for each published service. When a gridmap file is present the testing of the fine grained security can be done in place, bypassing the network based security option. This testing is the most critical part because if done properly it can save some effort and help streamline the final deployment. Once it has been confirmed that local authorization works, it is then a matter of replacing the local authorization with the Grid Authorization Service (GAS).

With local authorization complete it is then essential to ensure that the local firewall (if this exists), is appropriately set up to allow remote connections to the port that GAS is listening.

In conclusion, while the collaboration of developers in distinct organizations and at a distance can delay the development process, the use of instant messengers (MSN, skype, ...) can reduce this to a minimal inconvenience and can even help debug other problems that are only discovered thanks to external review/testing.

3 Integration in the Workflow Environment

The workflow environment is the place where the integration of the ACGT platform as a whole and its interoperability capabilities are put under scrutiny. In a workflow different ACGT services and tools are joined and connected by passing data and information to one another and therefore building and executing such a workflow is actually a “litmus test” for the ability of the ACGT components to work together.

In this context the difficulties we have encountered can be classified into the following categories:

- Web Service interoperability issues
- Semantics based incompatibilities
- Operational and error propagation issues

These aspects are further explained in the following paragraphs.

Web Services Interoperability

The middleware technology chosen for the implementation of the ACGT platform is Web Services with a Grid substrate. Even if we confine ourselves into the “pure” Web Service technologies there are many options with no “a priori” certainty for interoperability. The basic technologies used to implement Web Services are:

- XML, as a common serialization format
- SOAP, as the XML-based messaging protocol
- WSDL, as the service description language

But there are also a number of additional specifications and standards that cater for different aspects of web services, like:

- Security: WS-Security[2], WS-SecureConversation[3], WS-SecurityPolicy[4] (an addendum to WS-Security), WS-Trust[5].
- Messaging and Addressing: WS-Addressing[6], WS-Notification[7], WS-Eventing[8]
- Metadata: WS-Policy[9], WS-MetadataExchange[10], Semantic Annotations for WSDL[11]

---

2 http://www-106.ibm.com/developerworks/webservices/library/ws-secure/
3 http://www-106.ibm.com/developerworks/webservices/library/ws-secon/
4 http://www-106.ibm.com/developerworks/webservices/library/ws-secpol/
5 http://www-106.ibm.com/developerworks/webservices/library/ws-trust/
6 http://www.w3.org/TR/ws-addr-core
7 http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn
8 http://www-128.ibm.com/developerworks/webservices/library/specification/ws-eventing/
9 http://www.w3.org/TR/ws-policy/
10 http://www-128.ibm.com/developerworks/library/specification/ws-mex/
11 http://www.w3.org/TR/sawsdl/

- And a lot more[12]

A lot of these specifications are not "standards" at all, many of them are not widely implemented, and so on, therefore it is thought that in order to be always on the safe side one should just stick to the basic functionality offered by SOAP and WSDL. These problems have been identified and the Web Service Interoperability (WS-I, http://www.ws-i.org/) organization has been founded to offer guidance to WS implementers and users on how to increase interoperability. The most important deliverable of WS-I so far is the Basic Profile that consists of "*a set of non-proprietary Web services specifications, along with clarifications, refinements, interpretations and amplifications of those specifications which promote interoperability*." There is also the Basic Security Profile that provides guidelines on the use of WS-Security and the REL, Kerberos, SAML, UserName and X.509 security token formats.

Even with only the SOAP and WSDL basic elements for implementing Service Oriented Architectures with Web Services there can be interoperability problems. An example for such problems is the “style” of the WSDL used: this can be RPC / encoded, Document / Literal, Document / Literal / Wrapped, or Document / Literal / Bare and it basically defines how the service interactions defined in the WSDL at a high level map to XML messages sent over the “wire” [WSDL]. So it was early found out that BPEL and the Apache ODE Workflow Enactor specifically do not support Web Services implemented in accordance to the RPC/Encoded style. The WS-I consortium also recommends against the use of SOAP Encoding rules. The services implemented following the Document/Literal style in WSDL are the ones that are fully compliant with BPEL and ODE.

Another problem that we encountered was related to the Java Axis toolkit used so that the Web Services comply with the corresponding Grid software (Globus Toolkit). In particular Java Axis version 1 seems that by default does not create a "proper" WS-I compliant WSDL in the case of arrays. More specifically it does not create "wrappers" for arrays which can be the cause of interoperability problems. It seems that you can either enrich the WSDD with some additional typing information or you can use the -w (or --wrapArrays) undocumented parameter to the wsdl2java tool to produce the correct serialization of arrays (and also the correct WSDL will be produced at runtime).

Things become even worse when we add the Grid technologies in the mix. The state of the art Grid architecture and standards are collectively identified as the Open Grid Services Architecture (OGSA) and it was first presented in the paper The Physiology of the Grid by Ian Foster, et.al [OGSA]. OGSA is based on SOAP and WSDL and therefore should be to a large extent compliant with the web service technologies. But the statelessness of the web services was considered to be inappropriate for many Grid applications so the Web Service Resource Framework [WSRF] was proposed as a set of specifications to provide guidelines and "standardized" interfaces for building stateful services. In WSRF a resource is the concept used to associate state with and it is identified with WS-Addressing endpoint reference (a simple URI or more complex XML document). Communication with a WSRF service requires the inclusion of the identifier of the specific resource inside the SOAP Header so that the service could locate the state that it can match the request with. The OGSA WSRF Basic Profile is a similar effort to WS-I to define a profile for WSRF implementations. Unfortunately Globus Toolkit 4 (GT4), which is the most popular implementation of WSRF is not compliant with it in its current version (as of 4.0.3). For example GT4 uses a deprecated (pre-standardization) version of WS-Addressing. In addition to this, WSRF and its use of endpoint references in general requires special machinery and handling from web service frameworks and toolkits that are WS-I compliant. The requirement

[12] See http://roadmap.cbdiforum.com/reports/protocols/ for an overview

for the identification of the resource's endpoint reference in the SOAP Header rather than in the operations' parameters puts hurdles in tools that inspect the WSDLs of services and dynamically build code to invoke the services' operations.

Semantics based incompatibilities

When a user during the design of a new workflow is presented with a number of services, she should be facilitated to filter out the ones that are not useful to her for the problem at hand. In other words the “system” should present only the most appropriate and pertinent service information. This requirement relates to the service metadata: the description of the input and output parameters, the services functionality, the constraints under which the services function as expected etc. There’s the need therefore to have up-to-date information in the Metadata Repository that can be easily searched for and discovered.

Additionally the services should be made as much compatible to each other as possible. This requires the adoption of a common set of data types and data formats to be used throughout the ACGT platform and these data types should be presented explicitly and manipulated in a user friendly manner in the Workflow Editor. As an example we had come across a number of technical hitches in designing workflows that contain a service that accept “by value” a single value in their input but the “previous” service produces many values and additionally stores them in a file, effectively using a “by file reference” communication. A number of custom services were needed to parse files, custom extract values from them, passing them in a “loop” to the next service, etc.

We envisage that these problems will be eliminated as soon as we have a clean and sound data type ontology and all the services comply with it and be annotated accordingly in the Metadata Repository. The Workflow Editor needs to be extended to be familiar with these data types and support the relevant operators for their manipulation.

Operational and error propagation issues

A number of issues were identified during the workflow enactment as well. For example in the ODE Enactor it has been observed that long running services (typically more than 1 minute) result in Socket timeouts. No workaround has been found yet. This of course is unacceptable because in a real world setting it is typical that some “heavy” Grid processing “jobs” run for a couple of hours. Nevertheless the correct approach to handle such an issue is not have interactions through the network that take so long. If some request can last very long then the service's interface should be transformed from a synchronous into an asynchronous one. An asynchronous interaction will separate the submission of a request with the harvest of the results, so that each interaction is downsized in acceptable latencies. This of course will make the workflows more complex (because instead of a single request, two or three should be intermixed in a loop until the high level “operation” completes) but the benefit will be more sensible use of the IO resources (open sockets, etc.) and elimination of the timeout errors provided that all client - server interactions complete in a timely fashion. (See also why “Starbucks Does Not Use Two-Phase Commit” [ASYN])

It is also evident that the user should become aware in case of failures in one or more of the services participating in the workflow. The propagation of errors to the workflow enactor and further back to the user should be supported both by the services and the workflow execution layer. It has happened before that a service had failed *silently* with no propagation of its failure back to the enactor and hence the workflow continued without the rest of computation steps making any sense. BPEL provides features for dealing or even diagnosing failures so we should try to make good use of them both in the services implementation and in the construction of workflows.

4 Data Access services

Data Access Services Overview

We have implemented data access service for three types of data source: relational databases, DICOM medical image databases [13], and BASE [15] which stores bio assay data. The data access services have been implemented as OGSA-DAI services. OGSA-DAI is a web services framework for providing data access [8, 7]. One of the benefits of using OGSA-DAI is its activity framework. It enables a modular way of initiating multiple dependent activities using a single web service request. For example, a single message may trigger the execution of queries (e.g. identifiers of particular images), subsequent retrieval of data (the corresponding images), transformation of the data (e.g. compression into a single archive), and delivery (e.g. to a specified FTP server). This is very convenient as these are operation that you typically need to carry out when accessing data. However, use of OGSA-DAI does introduce additional integration challenges as will be discussed next. More information about the data access services can be found in D5.2 [17].

General issues

Most of the issues discussed in Section 0 also hold for the Data Access Services, as it based on the same software stack. In fact, for the Data Access Services two more layers are introduced. The first layer is the OGSA-DAI framework, which runs on top of Globus. On top of that is a layer of data service resources; for each data source that is made accessible, there needs to be a data service resource. So the software stack becomes:

- Java
- Tomcat
- Globus
- OGSA-DAI
- data service resources

There can be multiple data source services that provide access to different data sources of the same type (e.g. each to a different relational database). In this case, their configuration differs, but the implementation is the same. However, it is also possible to host multiple data service resources that each provides access to a different type of data source. This is the case in ACGT, where there are services for accessing relational data, DICOM medical image data, and bio assay data stored in BASE.

The implementations of each of these different data service resources each depend on various third-party libraries for their implementation. For example, the relational data access service uses D2RQ [11, 12] to translate SPARQL queries to SQL, which in turn uses Jena [10, 9]. The DICOM data access service uses Jena as well, together with dcm4che [14] for accessing DICOM servers. The BASE data access service uses client-code provided by the BASE developers for accessing their BASE web service. Finally, the different types of data access service can also use generic activities. For example, we have implemented an activity for delivering data to Gridge DMS, the grid data management system used within ACGT (see [16] for more details about DMS).

Given this set-up, one of the biggest problems is managing the dependencies between all different third-party software packages that are used. This is especially challenging because all data service resources are deployed within the same OGSA-DAI instantiation[13] and third-party packages (deployed as Java jar files) that are needed by only one or a few of the data service resources are visible to all. This can lead to dependency conflicts between data service resources that are otherwise independent. When data service resource A and B can both individually be deployed successfully inside an OGSA-DAI instantiation, this does not mean that they can be deployed alongside each other.

Two concrete issues that we bumped into may help to illustrate the types of integration problems this gives. Firstly, in implementing the DICOM data access service, we encountered a bug in Jena. After we reported it, the bug was fixed in the next release of Jena. Although we would have liked to, we could not immediately upgrade to this version of Jena. The reason was that another of our data access services, the one for relational databases, was using D2RQ. D2RQ was also using Jena, but an older version. It was not compatible with the latest release. Only after a new D2RQ version was released, which was upgraded to use a more recent version of Jena, could we upgrade the Jena library.

We encountered another problem when developing the activity for delivering data to Gridge DMS. For interacting with the DMS web service, we initially tried to use the compiled and packaged client-code provided by PSNC. This, however, did not work and failed with a "NoSuchFieldError" exception. After careful investigation it turned out that the problem was due to slight incompatibilities in the Axis libraries that were used. The version of Globus that we were using was based on a version of Axis that was slightly different from the one used to generate the compiled DMS client software. Fortunately, the incompatibility did not exist at the source code level, so rebuilding the DMS client code with the Axis libraries used by Globus fixed the problem.

Security Integration

Developing a secure version of our data access services has proved difficult as well. To integrate a web service into the ACGT security infrastructure [16], it needs to use GSI for authentication and use the GAS for authorization. For OGSA-DAI based web services a special plug-in is needed, as authorization should not take place at the OGSA-DAI web service level, but at the level of the data resource services deployed inside OGSA-DAI (authorization decisions need to take into account the particular database that is to be accessed). This OGSA-DAI plug-in is being developed by PSNC, but - at the time of writing this technical note - is not available.

In order to already gain some experience with the security framework, we decided to implement a non-OGSA-DAI based version of one of our data access services (the one for relational databases). For this we initially chose Axis2 as a web services platform, as it is more convenient to use than Axis. We (as it turned out, naively) assumed that it would be possible to integrate the Axis2 web service into the security infrastructure. We soon discovered it was not, as the use of Globus is essential to do so, and all available Globus software is based on Axis, the predecessor of Axis2.

So we had to base our data access service on Axis in order to make it secure. This we did, and we indeed managed to fully integrate it into the ACGT security infrastructure. However,

[13] It is not strictly necessary to host all data service resources inside the same OGSA-DAI instantiation. It is possible to host multiple OGSA-DAI instantiations inside the same web container, or even to run multiple web service containers on the same machine, each with their own OGSA-DAI instantiation. However, such set-ups have drawbacks with respect to scalability and manageability that they are generally best avoided.

this secure service only provides basic query functionality. It for example lacks the choice of delivery options offered by the OGSA-DAI based version. Furthermore, our other data access services, in particular the one for DICOM data, use even more OGSA-DAI features. Changing these to plain Axis-based web services would require much effort, so we are still keenly awaiting the plug-in for OGSA-DAI to secure our OGSA-DAI based web services.

5 References

- [1] http://people.cs.uchicago.edu/~borja/gw06/devtools/GDT_GlobusWorld_2006.pdf
- [2] Gregor Hohpe , "*Starbucks Does Not Use Two-Phase Commit*," November 19, 2004, http://www.eaipatterns.com/ramblings/18_starbucks.html
- [3] I. Foster, C. Kesselman, J. Nick, S. Tuecke "*The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*," [4] Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002, http://www.globus.org/alliance/publications/papers/ogsa.pdf
- [5] Russell Butek, "*Which style of WSDL should I use?*" IBM DeveloperWorks, 31 Oct 2003, http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/
- [6] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, "*The WS-Resource Framework*," March 5, 2004, http://www.globus.org/wsrf/specs/ws-wsrf.pdf
- [7] The OGSA-DAI Project, http://www.ogsadai.org.uk/
- [8] Antonioletti, M., et al: The Design and Implementation of Grid Database Services in OGSA-DAI. In: Concurrency and Computation: Practice and Experience, Volume 17, Issue 2-4, pp. 357-376, February 2005
- [9] Jena Semantic Web Framework, http://jena.sourceforge.net/
- [10] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. "Jena: Implementing the semantic web recommendations". Technical Report HPL-2003
- [11] Bizer, C., Seaborne, A.: D2RQ – Treating Non-RDF Databases as Virtual RDF Graphs. In: Proc. of the 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan, 2004. Poster presentation
- [12] D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs, http://sites.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/
- [13] National Electrical Manufacturers Association: "Digital Imaging and Communications in Medicine (DICOM)", 2004
- [14] Open Source Clinical Image and Object Management, http://www.dcm4che.org/
- [15] BASE – BioArray Software Environment, http://base.thep.lu.se/
- [16] B. Claerhout and S. Castille (Editors), Deliverable 11.1: "Consolidation of security requirements of ACGT and initial security architecture", 1 February 2007.
- [17] Erwin Bonsma (Editor), Deliverable 5.2: "Heterogeneous data access services", 19 December 2007
- [18] Juliusz Pukacki (Editor), Deliverable 3.1: "The ACGT Initial Architecture", 12 February 2007