CAGRID-TAVERNA INTEGRATION PROJECT SUMMARY REPORT

STIAN SOILAND-REYES
ALAN R. WILLIAMS
ALEXANDRA NENADIĆ

UNIVERSITY OF MANCHESTER
OCTOBER 9, 2009

Project Information	
Project Title	caGrid-Taverna Integration
Start Date	September 1, 2008
Original End Date	September 1, 2009
Actual End Date	October 31, 2009
Lead Institution	University of Manchester
Project Director	Prof. Carole A. Goble
Project Managers	Constantinos Astreos (University of Manchester) Musharaf Rashid (SAIC-Frederic)
Stakeholder Institutions	University of Manchester, SAIC-Frederick, NCICB

1 INTRODUCTION

This document is the final project summary report for the *caGrid-Taverna Integration project* defined by the International Subcontract #28XS206 between *SAIC-Frederic Inc*. and the *University of Manchester, UK*.

The project ran for 14 months from September 2008 till the end of October 2009. Originally, the project was planned to last for 12 months, but a two-month extension was introduced in order to allow for the *Silver level compatibility* review process to finish before the official end of the project.

The report is structured as follows: Section 2 provides the project summary; Section 3 gives a brief explanation of each of the phases of the project and its objectives; Sections 4 and 5 describe the accomplished work, issues encountered and their resolutions for each of the project phases, together with lessons learned and recommendations for future enhancements; and finally Section 6 presents the work on dissemination and publicising of the project's results.

More information about the project together with all project deliverables is available from:

- Project's NCICB gForge Web site: https://gforge.nci.nih.gov/projects/taverna-cagrid/
- Project's wiki pages: http://www.mygrid.org.uk/dev/wiki/display/caGrid/Home

2 PROJECT SUMMARY

The US *National Cancer Institute* (NCI[1]) provides an integrated biomedical informatics infrastructure called the *cancer Biomedical Informatics Grid* (caBIG[2]), to expedite the US cancer research community's access to key bioinformatics platforms. In partnership with the cancer research community, the NCI has created a common, extensible informatics platform that integrates diverse data types and supports interoperable analytical tools.

caGrid is the underlying network and security architecture for caBIG; it provides the basis for connectivity between all of the cancer community institutions in the US, allowing research groups to tap into the rich collection of emerging cancer research data while supporting their individual investigations.

One of the focuses of the caBIG programme is to provide the US cancer research community with a tool to orchestrate data and analytical services into workflows in order to automate and expedite the running of scientific experiments. The open source *Taverna Workbench* developed by the *myGrid*[3] team from the *University of Manchester, UK*, has developed a graphical workflow design, management and execution environment that is used by a large number of scientists worldwide. The myGrid team has accumulated years of experience in connecting to various third party analytical and data services and linking them in scientific workflows. Integration of Taverna with caGrid, the backbone of the caBIG infrastructure, is intended to bring this powerful tool and the myGrid team's experience to the caBIG community.

[1] http://www.cancer.gov/
[2] https://cabig.nci.nih.gov/
[3] http://www.mygrid.org.uk/

3 PROJECT PHASES AND KEY OBJECTIVES

The project was realised in two independent phases, objectives of which are described below.

Phase 1 objective: Provide interoperability between caGrid infrastructure and Taverna Workbench in order to enable orchestrating of caGrid services within Taverna workflows, thus automating and speeding up execution of scientific experiments for caGrid user community.

Phase 2 objective: Make selected widely used and proven third party analytical services (already accessible from within Taverna) available on caGrid in order to widen the pool of services available to caBIG community.

Each of the phases is covered in more detailed in the following sections: Section 4 describes the work on the caGrid-Taverna interoperability, while Section 5 explains making third party analytical services caGrid-enabled. Encountered issues, resolutions and lessons learned are included at the end of each of these sections.

4 PHASE 1: INTEROPERABILITY BETWEEN CAGRID INFRASTRUCTURE AND TAVERNA WORKBENCH

At the moment, the vast pool of available caGrid services are not being used to their full potential as they exist independently and are not "aware" of each other, despite being designed with semantic interoperability in mind. This phase of the project was aimed at providing connectors from Taverna Workbench to caGrid services available in the caBIG infrastructure to enable linking such services into meaningful workflows and help automate the execution of scientific experiments for users in the caBIG community.

4.1 TASKS

This project phase was realised through the following tasks (corresponding deliverables from the Project Management Plan have been indicated where appropriate).

(1) **Deliverable 2.1.1.1:** Develop semantic search plugin for Taverna that enables discovery and search for caGrid services based on various criteria (i.e. metadata available about caGrid services through the *Index Service* repository).

(2) **Deliverable 2.1.1.2:** Develop a plugin for Taverna that supports invocation of caGrid-enabled services. All caGrid services are implemented as WSRF Web services and the developed plugin is capable of discovering the inputs and outputs of such Web services on the fly, and, after letting the user specify the input values, invoking them using SOAP over HTTP. The outputs of services are collected and either displayed to the user or passed downstream to another service in the workflow.

(3) **Deliverable 2.1.1.3:** Provide support for invoking secure caGrid services. Such services typically require a secure HTTPS connection and user authentication prior to allowing access. The plugin from deliverable 2.1.1.2 was enhanced to support secure services by enabling users to log in to their affiliated institutions and obtain short-term proxy certificates for accessing the grid (by interacting with caGrid's GAARDS security infrastructure and integrating with caGrid's fabric of trusted CAs). Enhancements were also added to Taverna Workbench to securely manage usernames and passwords as well as proxy certificates so as to provide single-sign-on experience to users.

(4) **Deliverable 2.1.2:** Packaging of the integrated product, release notes and installation guide. The release of the final product was made available for download from http://www.mygrid.org.uk/taverna/cagrid/2.1-beta-2/.

(5) **Deliverable 2.2:** Provide tutorials, training material and end-user support. An important aspect of the project was providing the caBIG community with the knowledge and support on how to create workflows using Taverna. In order to do this, we have prepared the user guide and tutorials, available for download from Taverna's web site http://www.mygrid.org.uk/tools/taverna/tutorials/.

We have been providing continuous end user support through our mailing lists *taverna-users*[4] and *taverna-hackers*[5].

We have also held presentations and demonstrated the project software deliverables at various occasions and caBIG events (see Section 6 for more details).

4.2 OUTPUTS

The developed enhancements were made available as plugins to the current Taverna Workbench 2.1 b2 release, and the installation and user guides for building workflows have been prepared and made available on-line. Taverna's plugin installation mechanism meant that end users did not have to do the download and install anything manually – all they had to do is point to the URL of the released product (consisting of two plugins) from inside Taverna Workbench.

To demonstrate the success of this phase of the project, we have created a workflow that performs lymphoma prediction by combining existing caBIG services - data service *caArray* and the analytical service *GenePattern*. This demonstrates how complete analysis experiments based on published scientific methods can be conducted over the caBIG infrastructure by orchestrating caGrid services in workflows using Taverna Workbench.

4.3 ISSUES, RESOLUTIONS AND LESSONS LEARNED

4.3.1 CLASHING AXIS VERSIONS

When attempting to use caGrid's Globus *security handlers* to develop security support for caGrid services, we encountered a clash of two different versions of the Axis library being used by our plugin. Basically, Globus used the *Axis 1.2 Release Candidate 1* (Axis 1.2 RC1), whereas Taverna's Web service support used the newer *Axis 1.4*. Due to various interface changes between the two Axis versions, they have been rendered incompatible. So, instead of our initial plan to extend the Taverna's original Web service support plugin, we had to create an *alternative Web service plugin* for communicating with caGrid services, which was using Axis 1.2 RC1 instead of Axis 1.4.

This means that we will have to face maintenance issues in keeping these plugins synchronized when enhancing Web services support in Taverna in the future. This problem will go away once Taverna moves to a new plugin management system that will allow for two alternative Axis implementations to coexist in the same plugin. An alternative would be to make the security handlers of caGrid and Globus work with Axis 1.4, which would require changes to the caGrid software.

[4] https://lists.sourceforge.net/lists/listinfo/taverna-users
[5] https://lists.sourceforge.net/lists/listinfo/taverna-hackers

4.3.2 CLASSLOADING AND CYCLIC DEPENDENCY PROBLEM

The Taverna plugin for caGrid initially included dependencies to two separate libraries, *caGrid* and *Globus*. In Taverna's plugin management system *Raven*, each library gets its own Java *classloader*, with links to the classloaders of its dependencies. As we did not know the interdependency graph between the various caGrid and Globus JAR files (which were not built using Maven), we simply merged the individual JARs of Globus into a single `globus.jar`, and the individual JAR files of caGrid into a single `caGrid.jar`. This meant that the plugin would have two dependencies (to Globus and caGrid libraries) and that Raven would effectively make one classloaders for each of them.

However, we have now faced a cyclic dependency problem. The caGrid library depends on the Globus library and that was fine until we realised that the Globus library needed a dependency back to the caGrid library. The reason for this, as we discovered later, was because Axis client configuration files define *caGrid security handlers* by classname. The handlers are located in the caGrid library but are needed by Globus to handle sending of secure messages and have to be discovered and loaded by the classname. This means that the Globus library classloader will attempt to load the handler classes from the Globus library and its dependencies. Since they are not located there (i.e. they are in the caGrid library), Globus requires a dependency to the caGrid library in order to find the required classes. This introduced a cyclic dependency problem where caGrid library depended on Globus that in turn depended on caGrid.

To avoid this dependency loop, we had to make a single library by merging the Globus and caGrid libraries into a *mega-library*. This means that if any of the classes in the caGrid library has to be upgraded at some point in the future (which is highly likely), a new mega-library would also have to be assembled even though Globus library has not changed at all.

This problem is mainly down to *limitations in Taverna's plugin management system*. It allows two different Taverna plugins to use two different version of Axis, but will not allow it for the same plugin.

The work on enabling invocation of secure caGrid services was initially pushed back for later in the project as we anticipated performing various changes to Taverna's plugin management system to support cases like this one. However, this has not happened as the decision has been made to move directly towards using a standard plugin framework called *OSGi* as a replacement for Taverna's plugin management. This work is scheduled for 2010 and was outside the timeframe of this project. Thus, we went for the intermediate solution described above (with the mega-library) in order to provide support for invoking secure caGrid services. However, the work on this project has helped us identify the issues with the current and clarify the requirements for the future Taverna plugin management system and presented an excellent use case.

4.3.3 ISSUES WITH CERTIFICATE AUTHORITY AND SERVERS' CERTIFICATES

Although caGrid's security handlers allow secure invocation of Web services, Taverna also need to inspect the *WSDL definitions* of such services prior to actual invocation. As this inspection is not performed using the Globus library, and the WSDL definitions of such services are typically accessible through HTTPS, we needed to install the caGrid *root CA certificates* and any *intermediate CA certificates* into Java's *trust store*. The reason for this is because Java will only open an HTTPS connection to a trusted site and placing the CAs' certificates into Java's trust store implicitly establishes this chain of trust.

We enabled this by having the caGrid-Taverna plugin insert pre-bundled caGrid CAs' certificates into Java's trust store. Initially, we also wanted to enable automatic update of the CAs' certificates from

caGrid's trust fabric by using caGrid's *SyncGTS* service (in the case certificates are revoked or expire). However, SyncGTS seems to be specialised for service containers and not Web service clients like Taverna. Although we believe this is doable from Taverna, time constraints prevented us from working on this issue any further.

Another issue we encountered involved certificates of some servers running caGrid services. Such services have a host certificate issued by a caGrid CA where the hostname in the certificate is specified as `host/pc23.domain.com` instead of the regular `pc23.domain.com`. We later discovered that this was the case with caGrid v1.2 services and that in caGrid v1.3 this issue was resolved but there was still a number of old-style services running. Other services had their host certificates issued with a name that would not match the actual server name, e.g. the host name in the certificate is `pc23.domain.com` where the actual host server name would be my.pc23.org or, even worse, 176.23.45.78. In all of these cases, the mismatch between the actual server name and the name in the certificate would cause HTTPS connections from Java to fail.

In order to resolve the `host/pc23.domain.com` vs. `pc23.domain.com` issue, we had to extend Java's standard SSL handling to allow for these exceptions. In cases where there was a total name mismatch, we could not have done anything and connecting to such services from Taverna would fail.

This is not strictly a caBIG issue as these services are provided by the member institutions and caBIG do not have control on how they name their servers. However, since caGrid's CAs issue certificates to such servers, some guidance should be provided on the naming conventions (i.e. host name in the certificate must match the server name).

5 PHASE 2: CAGRID-ENABLING OF ANALYTICAL SERVICES

A large number of excellent third party analytical services is available on the Internet to scientists free of charge. They are not accessible to caBIG users through the tools they are normally using as these services do not follow caBIG-prescribed rules on how to build Web services, nor do they have their inputs and outputs semantically annotated or registered in repositories such as caBIG's *caDSR*, *Index* or *Global Model Exchange* (*GME*) service. This means that such services are neither easily discoverable nor accessible for a typical caBIG user. For this reason, caGrid-enabling of third party analytical services means increasing the pool of services available to caBIG users.

5.1 TASKS

This project phase was realised through the following tasks (corresponding deliverables from the Project Management Plan have been indicated where appropriate).

(1) **Deliverable 3.1:** Selection and describing of analytical services for caGrid-enablement. The services we have chosen were InterProScan[6] and NCBI BLAST[7], provided by the European Bioinformatics Institute[8] (EBI). InterProScan service allows users to query their *protein sequences* against member databases in InterPro warehouse and combines different *protein signature (domain) recognition methods* native to the InterPro member databases into one resource with look-up of the corresponding InterPro and GO (*Gene Ontology*) annotations.

[6] http://www.ebi.ac.uk/Tools/webservices/services/interproscan
[7] http://www.ebi.ac.uk/Tools/webservices/services/ncbiblast
[8] http://www.ebi.ac.uk/

NCBI BLAST is a *sequence similarity search* service, emphasis of which is on finding regions of sequence similarity that could yield to functional and evolutionary clues about the structure and function of the query protein or nucleotide sequence.

Both services are freely accessible as Web services, are highly reliable and widely used by the scientific community worldwide, and can be easily integrated with existing caBIG tools (such as *caBIO*, *GridPIR*, etc.) in biologically meaningful workflows.

(2) **Deliverable 3.2:** Design the architecture of the caGrid-enabled analytical services and the UML model of the system that meets the *caBIG Silver level compatibility guidelines* and is aimed at reusing the components already present in the caBIG infrastructure, namely *Common Data Elements* (CDEs) from *caDSR*.

(3) **Deliverable 3.3:** Develop an implementation plan for the designed system.

(4) **Deliverable 3.4:** Develop a detailed test plan documenting how the designed system (i.e. each of the selected analytical services) is going to be tested.

(5) **Deliverable 3.5:** Submit the Silver level compatibility package for the review by the *Cross-Cutting Workspaces* for the final review and approval, which included:

a. Semantically annotating the UML model of the system with special emphasis on CDE reuse. We have managed to reuse 8 existing CDEs from the caBIG's *Information Representation Working Group* (IRWG)[9] and *Protein Information Resource* (PIR)[10] projects in our UML model.

b. Loading the annotated model into the caDSR production database and reviewing the model using *CDE Browser* and *UML Model Browser* tools. This involved closely working with the caDSR curation team and loading the model first to the sandbox database and reviewing it before moving it to the *production caDSR*.

(6) **Deliverable 3.6:** Implement the designed system as caBIG Silver level compliant and deploy the services on the caGrid Production infrastructure. This task involved exporting the UML model from the *Enterprise Architect* tool to a set of XML Schemas and applying the caGrid tool *Introduce* to generate the service skeletons and then implementing the services in Java.

(7) **Deliverable 3.7:** Conduct testing of the implemented and deployed analytical services as described in task (4).

(8) **Deliverable 3.8:** Prepare installation and user documentation. This included documentation on compiling the source code for the analytical services, documentation on the example invocation of the analytical services using the implemented caGrid client code, and documentation on installing the analytical services and deploying then on caGrid.

(9) **Deliverable 3.9:** Provide feedback on caBIG tools used and semantic infrastructure. We have prepared two separate documents – one non-technical executive summary and a separate document with technical details of the issues we encountered while performing the tasks above. Portions of these two documents are included in this report.

[9] https://wiki.nci.nih.gov/display/ICR/Information+Representation+Working+Group+(IRWG)
[10] http://pir.georgetown.edu/

5.2 OUTPUTS

The main outputs of this phase of the project are the implementation of the two analytical services and the documentation on how to install, test and use them. The services are live and running on caGrid Production infrastructure and available for use by caBIG users.

As a result of the Silver level compatibility process, the UML model of the services has been loaded in caDSR and, due to reuse of existing CDEs, the services are semantically interoperable with other caGrid services using the same CDEs. Note that, at the time of writing this report, the review process has not been concluded yet (it has been kicked off on September 24, 2009).

5.3 ISSUES, RESOLUTIONS AND LESSONS LEARNED

5.3.1 OVERALL PROCESS

As we started this project, our team was new to caBIG infrastructure, technologies and services. We relied on tutorials, documentation and our VCDE and Architecture mentors to get an understanding of the processes and practices.

5.3.2 SERVICE SELECTION AND DESCRIPTION

The project started with *selecting two existing analytical services* available from third parties. We achieved a general understanding of the services, how they worked and their APIs. This later formed the basis for building the *UML model* of the data types for the services' inputs and outputs. This bit of the process worked fairly well, with our bioinformatician working with Juli Klemm and the caBIG *ICR Workflow Working Group* in finding suitable use cases where the chosen services could be combined with existing caBIG services.

5.3.3 UML DESIGN

As we moved on to the design phase, we realised we have underestimated the importance of *designing UML models* for caBIG services. Therefore, we have not fully anticipated the amount of effort needed to do the modelling, several rounds of reviews of the model and enabling reuse of existing data elements.

The caBIG *template models* gave us a good starting point for how to structure the model, but as they were mainly oriented towards data services instead of *analytical services*, we needed to do additional structuring to be able to generate the XML schemas from the model. Additionally, all caBIG tutorials on making services Silver level compatible are oriented towards data services, and not analytical services, which initially caused some confusion on our part.

Overall, tutorials on how to make caBIG Silver level compatible services do not fully capture the *know-how* and *best practices* needed to make a Silver level service, but our mentors (in particular Robert Freimuth from the *Mayo Clinic, Rochester*) were very helpful in providing us with guidance on this matter. Additionally, the information presented in different tutorials is sometimes not consistent and in sync. On the positive side, caBIG provides a large number of good and professionally done tutorials and few mistakes we have spotted in them have been promptly dealt with by Claire Wolfe, a member of the curation team.

There is a need for more documentation on how to *reuse data elements at the time of designing the UML model*, as the focus on reuse is currently during the later annotation phase. Encouraging reuse at an earlier stage could make it easier to build the model correctly and implicitly help designers learn the caBIG modelling practices.

The initial UML model was very close to the original analytical services' data formats. It was then fully annotated, loaded into caDSR and implemented as caGrid services. After the review by the mentors, it became clear that the original service's data were inconsistent in several places, for instance the protein identifiers used as input and returned by the service as output were not in the same format. This caused the model and implementations to be *reworked* to use several newly registered data types (CDEs) from the *PIR* project and the *ICR Workflow Working Group* that were not available at the time we originally did our UML model. This *CDE reuse* resulted in a further streamlining of the model and eventually resulted in service interfaces being more compatible with other caBIG services and between themselves. However, is also introduced additional work for us as we did the whole design-annotation-implementation process twice.

5.3.4 CAGRID AND INTRODUCE

The Introduce tool assumes *existing knowledge of caGrid terminology and concepts*, so although the user interface seems straightforward to get started with, there is still a learning curve. Tutorials helped us a bit along the way, however it was not always easy to know which documentation was the most appropriate for our purpose. Our mentors helped us by providing links to useful tutorials.

5.3.5 INTERACTIONS WITH THE KNOWLEDGE CENTER

The users of the forum in the *Knowledge Center* were quite *responsive* and *helpful* when we came with concrete questions on issues we were stuck on.

The Knowledge Center community was also helpful to respond to our questions on *technical best practices* for how to design caBIG WSRF-services, for instance how to do asynchronous services or WSRF resource properties.

However, it can be difficult for end users to deliver *bug reports* through this channel, as an intended bug report can be interpreted as a request for support.

5.3.6 CDE ANNOTATION

When adding the annotations to the model, we relied on guidance from both our VCDE and architecture mentors, in addition to our assigned NCI curation team. They were both knowledgeable and quick to respond to our queries.

When mapping CDEs for reusability in the *Semantic Infrastructure Workbench* (SIW), it is unclear if it is *attributes* or *classes* that are to be mapped. Restrictions operate on class-level, but the mapping is still done on attribute-level.

We encountered some difficulties in using the *CDE browser* and *UML model browser*. In particular for new users it would be very helpful to have *hyperlinks* connecting in both directions between classes, attributes and concepts. Instead, we tended to click through the concepts and classes to find the public identifier for a CDE and then use the *advanced search* in the CDE Browser with the public identifier from the previous page to find the relevant information for that CDE.

Public identifiers are numeric and different types of such identifiers look similar, but are not resolved in the same way. There are public identifiers for CDEs, concepts, object classes and properties – sometimes we would be searching using the wrong type of public identifier. Our experience from the *semantic web* suggests using *URIs* in the style of `http://cagrid.org/2009/dsr/class/12312312` is more appropriate, as it would not only convey what type of object is identified and make the identifier universal but also implicitly introduce a hyperlink for getting more details.

A useful learning tool for us was to inspect *other projects' UML models*. We were not able find these through the UML Model Browser easily; instead we downloaded the `XMI` and `EAP` files from the individual projects' *gForge* pages.

There was an issue early in the project where the CDE Browser was not working correctly with Mozilla Firefox – this issue was reported to Claire Wolfe from the curation team who escalated this promptly with the developer team and the issue was subsequently resolved.

Overall, we believe that attending a *tutorial in person* on the CDE Browser and UML Model browser would have made it much easier for us to use these tools.

5.3.7 LOADING TO CADSR

Loading of models into the *caDSR* seems to involve manual curation work. We had to manually review that the *CDE*s had been loaded correctly to the repository by clicking through each of them, and then report back to the curation team on CDEs that had not been loaded properly.

Even though loading of the model to the *sandbox caDSR* was successful, moving the model from the sandbox into the *production* caDSR could fail. This was a bit of surprise for us as we expected this process to be automatic. Failure in loading to production meant we again needed to manually check and click through each of the loaded CDEs using the CDE/UML Model Browers and report back for further curation, even though we have already done it once for the sandbox.

5.3.8 REUSE OF DATA ELEMENTS AND INTEROPERABILITY

Our first impression of the data modelling is that it is focused around *semantic* descriptions. Two services can be caBIG compatible in the sense that they agree on what they say a *protein* is, but that does not necessarily imply *syntactic compatibility*. As our team is coming from a workflow point of view, we are traditionally be more interested in services interoperating on a transport level.

caBIG has the *Global Model Exchange* (GME) service that should be able to help in this case by being a registry of XML schema types. At the moment, this service seems underutilised as few projects' models are actually represented there.

5.3.9 WRAPPING SERVICES AS SILVER LEVEL COMPATIBILE

The effort of *making and refining the UML model* by far outweighs the effort needed to actually implement the code to create the services.

About two months' worth of effort were needed just to get started with caBIG concepts and terminology. This implies that the *start-up costs* are relatively high and it can be quite hard to get started with making the very first Silver level compatible caGrid service. However, once done one would know most of the technologies and processes, making it easier to create subsequent services.

Our project needed to wrap two analytical services and make them Silver compatible – for getting used to the development tools and techniques we initially focused on only the first service. After the first service was fully functional, implementing the second service was done in a few extra weeks.

5.3.9.1 SUGGESTIONS

In our view, if caBIG aims to make it easy to create services that can interoperate, the focus should be more on the XML schemas used for the actual WSRF/WSDL communication with the services, encouraging reuse of schemas and making the schemas structured and easy to understand.

If caBIG wants to encourage reuse of CDEs and classes from the existing projects, it should be made easier to locate the projects, browse the existing data elements and find them in existing projects, with clear links to the UML model and XMI files (and preferably XML schemas).

The various restrictions on CDE modelling and mapping seems to have caused many semi-duplicated CDEs. The existence of several classes for almost the same purpose also makes it more difficult to determine which one to reuse.

5.3.10 ENFORCING SERVICE DESIGN VS. DESCRIBING EXISTING SERVICES

In the *myGrid* team's past experience with describing biological Web services for the use by Taverna Workbench and other Web service clients (e.g. in Web service repositories *Feta* and *BioCatalogue*), the tendency was towards being able to *describe the services as they are and what they do*, without requiring any wrapping or prescribing a formal way to make such services. This meant that these descriptions had to be able to cope with a large range of different service behaviour. Unfortunately, this also meant shifting the *effort of making services interoperate to the workflow designer*, who would use these service descriptions as a guide for creating a plumbing code that would allow two incompatible services to be connected.

From our point of view, it is encouraging to see that a rigid set of requirements for building caBIG services can create services that are compatible out of the box, making the *interoperability effort the responsibility of the service designer* rather than that of the workflow designer.

However, this project has shown that a *considerable amount of effort* is required in order to make an existing third party service Silver level compatible. The initial effort is directed along *understanding the tools and processes of caBIG*, followed by the considerable effort centred along *creating and reviewing the UML model* and its semantic annotations. This requires deep knowledge of both the service internals as well as that of the existing concepts, data elements and caBIG's Silver level design rules. Actually *implementing* the services became trivial in comparison. The main effort needed in this area was getting to know the existing best practices on building WSRF services "the caBIG way", something the *Knowledge Center* users were happy to share with us.

To summarise, it can be *demanding for an outsider* to come into the practice of building a caBIG service – the threshold for any third party service providers to join the caBIG community is high. In our case, we have been contracted to build such services, but the concern is that many independent service providers might not be willing to put in a similar amount of effort to get the Silver level stamp of approval.

5.3.10.1 SUGGESTIONS

Our recommendation would be to, in the short run, set up a *specialised caBIG working group* that would be responsible for selecting and describing useful *third party services* and wrapping them as Silver level compatible. They would also host the service wrappers and provide support to the caBIG community for using them. As the effort needed for newcomers seems to be significantly higher than for experienced caBIG developers, it would be advisable to let the same team do most of the service wrapping, probably in cooperation with the third party service providers who have an understanding of the service internals.

In the long run, we would recommend caBIG to review and simplify its processes to make them more accessible for third party service providers, by requiring less effort in getting to know the tools and architecture, and getting the described services reviewed and accepted as official caBIG services.

We would also advise looking at *describing existing services as they are* instead of requiring third party services to be wrapped or reworked to be usable by the caBIG community. caBIG could try to find a more fine-grained way of mapping/describing the data types and formats of a third party service, for instance by bundling a set of XML transformations with the description.

To allow reuse of existing data types already available in the caBIG infrastructure, one should perhaps focus more on the *actual data and formats* used (e.g. by showing example data), rather than the UML models, data elements and concept codes in isolation. Instead of having to change the service to fit a restrictive model, the description model could be changed to fit a wider range of services. *Browsing and searching* over existing types should be made more accessible for first time service developers, simplifying and encouraging reuse of data elements when deemed appropriate.

If one moves towards annotating elements of an *XML schema* instead of UML models, and allow for reused data attributes independently of reused classes (but encouraging reusing schemas), one could work closer with the data in question and adapt the service annotations instead of moulding the service around an existing service that just happened to be annotated and registered first.

5.3.11 SILVER LEVEL COMPATIBILITY REVIEW PROCESS

As far as we have observed, the process for reviewing a model for Silver level compliance includes the following steps. Initially, the model is reviewed and refined in cooperation with the mentors and the curation team, followed by reviews and refinements of the whole review submission package with the metors. As the package is officially submitted, the review board comes together; they then have to go through the following meetings: *Review kick off*, *Initial review*, *Interim review*, *Final review*, *VCDE presentation*, *Architecture presentation* and finally the *VCDE/Architecture vote*. The board found it difficult to schedule all these meetings within the remaining 30 days of our contract.

There is a total of seven review meetings in order to determine if two services can be considered Silver level compatible – perhaps the review process is intended for more heavyweight data services with a potentially much larger overlap with existing data elements. At times it seemed like the review procedures focus mainly on *building the data model* and the structured descriptions rather than what is needed to *enable actual interoperability*. As a result, the models that pass the review are only semantically compatible without necessarily being *syntactically compatible*, which would probably be of more use to an end user who is trying to combine data sources and services to perform his virtual experiments.

5.3.11.1 SUGGESTIONS

We believe that if caBIG is to *encourage reuse* and welcome more data and analytical service providers to join the community, caBIG should *simplify these processes* and *lower the threshold* for introducing a caBIG service. A more *pragmatic approach* is advisable, having a key focus on how the *end users can achieve their goals* and how *tools and data can more easily be shared* within the community with the least amount of effort and the greatest amount of gain.

We would recommend moving towards a *community-driven review and refinement process* that is involved continuously from the initial design, following through service creation and annotation. This could be inspired by the way the open communities around RESTful Web services and the semantic Web discuss design ideas and best practices without requiring rigid standard frameworks or lengthy review processes. In this way the service developer will not only learn the "caBIG way", but also be in a better position to influence those best practices.

6 DISSEMINATION ACTIVITIES

As part of the project, we attended various caBIG community events where we have given presentations and demonstrated the developed software, as listed below.

2009-12-09 UK e-Science AHM 2009 conference (to be held), Oxford, UK:

- Presentation: *Bringing caBIG services together using Taverna*

2009-09-25 Silver level review kick off meeting:

- Presentation: *Wrapping third party analytical services for caBIG*

2009-09-10 NCI/NCRI joint conference, London, UK:

- Poster: *Building and Running caGrid Workflows in Taverna*
- Poster: *Taverna based workflow infrastructure in caGrid*

2009-07-20 caBIG annual meeting in Washington DC, US:

- Slides for the demonstration stand: *caBig Workflow*
- Software demonstration
- Poster: *Building and Running caGrid Workflows in Taverna*
- Poster: *Taverna based workflow infrastructure in caGrid*
- Poster: *Bringing third party analytical Web services to caBIG*

2009-05-11 caGrid architecture face-to-face meeting, Cambridge, MA, US:

- Presentation: *Taverna workflows in caGrid*

2009-03-19 caGrid user group meeting:

- Presentation: *An Introduction to Taverna*
- Presentation: *Technology behind using Taverna in caGrid*
- Presentation: *caGrid workflow examples*

2009-01-23 caGrid technical design review meeting for analytical services:

- Presentation: *Wrapping analytical services for caBIG*

The work on the project has also generated the following publications in collaboration with our project partners from Argonne Laboratory, University of Chicago, US:

- Alexandra Nenadic, Stian Soiland-Reyes, Carole Goble. *Bringing caBIG services together using Taverna*, UK e-Science All Hands (AHM) 2009 Conference, Oxford, UK, 2009.
- Wei Tan, Kyle Chard, Dinanath Sulakhe, Ravi Madduri, Ian Foster, Stian Soiland-Reyes, Carole Goble. *Scientific workflows as services in caGrid: a Taverna and gRAVI approach*, IEEE International Conference on Web Services (ICWS 2009), 2009.