caGrid Development Operations and Maintenance (28XS097) Project Summary Report

This report summarizes the activities and overall status of the caGrid Development Operations and Maintenance under Statement of Work *28XS097* per the requirements of section 4.1.4 of that S.O.W.

Work Accomplished against the SOW

The SOW identifies three major areas of work:

1. Identifying, triaging, debugging, developing and deploying bug fixes and minor enhancements of the toolset
2. Providing operational support to the tools and services for the NCI Production deployment of the tool.
3. Providing technical leadership to the community on the toolset.

Additionally, the SOW identifies some program management related deliverables and tasks, such as

1. A wiki space containing all software deliverables, project management documentation, etc.
2. A technical project plan, to be updated and made available via the wiki on a monthly basis.
3. A technical status report, updated monthly and made available via the wiki.
4. This document

To this end, the caGrid development team has accomplished the following:

Bug fixes and enhancements:

A total of 83 issues have been entered in the Jira tracker since the start of the SOW on June 24, 2011, including 7 bugs, 25 new feature requests, and 11 issues identified as tasks, most of which comprise documentation tasks. Of these, two bugs were fixed, and nine new features implemented with four in progress as of this writing.

Major new functionality completed during this SOW's period of performance included the completion of development of the SHA 2 support release of caGrid, implementation of upgrade tooling for services, and integration of support for data services backed by the latest release (4.4) of caCORE SDK.

Operational support for NCI production deployment

The caGrid development team in conjunction with the caGrid Knowledge Center has worked closely with the NCI to ensure high uptime and availability of the production grid deployment.

Typically, this involves answering questions about the grid APIs and underlying functionality, but occasionally diverges into new feature requests. An example of the latter include Jira item CAGRID-795, which is a request for additional functionality to be added to the authentication service for configuring account lockouts.

The development team has also answered numerous "Data Calls", which is the NCI's mechanism for obtaining information about existing or proposed technical functionality that might impact other user communities. The most prominent example of this activity is the development team's ongoing work to inform the NCI regarding the impact of the upgrades and changes to support SHA 2 certificates in the grid.

Community technical leadership

The development team provides front-line support along side the caGrid Knowledge Center for technical aspects of questions and requests about the software. The KC forums provide a "front door" through which questions and requests for support are made from the community, and the development team provides timely technical response.

Documentation of the product is supplied and updated through a community accessible wiki at http://cagrid.org. This wiki contains comprehensive guides, which help adopters install, configure, manage, and interact with caGrid.

Active engagement with the user community in the form of teleconference participation, email communication, and presentations allow the development team to lead the community in adoption and uptake of the caGrid software stack. This includes assistance with installations and integration of the library APIs with application projects.

Other program management related deliverables

The deliverables of sections 4.1.1, 4.1.2, 4.1.3, and 4.1.4 of the S.O.W. are available on the caGrid Development Wiki space of the NCI wiki. The space can be accessed at the URL https://wiki.nci.nih.gov/display/cagridproject/caGrid+Development+Wiki+Home+Page. All related deliverables are organized by name and attached to the space's pages as PDF documents.

Issues and Resolutions

The project encountered a few significant technical issues during this phase of development.

Upgrades of existing services to the SHA 2 support release

Historically, caGrid has supported upgrading services from two releases back of the current release with a minimum of user intervention, if any at all. For example, caGrid 1.4 supports upgrading a service which was created with caGrid 1.2 or caGrid 1.3 to the 1.4 release using out-of-the box tooling. Upgrades to 1.4 are not supported from services generated with caGrid 1.0 or 1.1. This policy has allowed the software to improve over time, while providing a simple migration path for the established user base. As the underlying tech stack of caGrid has not changed substantially since the initial 1.0 release, this has not proven to be a process that introduces significant technical problems for end users.

With the advent of the SHA 2 release, significant changes had to be made to the underlying tech stack of caGrid, which made upgrades substantially more complicated. For most cases, the development team was able to automate these changes and render the upgrade experience every bit as seamless as it was for previous releases. An interesting edge case came to light, however, in the form of custom-modified grid services. The team worked through various scenarios of replacing libraries where possible, implementing wrapper APIs to emulate the previous functionality, code generation tools, and runtime bytecode injection techniques in an effort to upgrade such services without significant developer interaction and rework. Unfortunately, such efforts proved the issue to be complicated to the point of being impractical to implement in a manner that would work in all – or even most – cases. The issue was resolved by sticking to the historical upgrade policy, which indicates that custom-modified services will be upgraded to the greatest extent possible, but leaving the onus on the developer of the service to ensure those customizations are compatible with the new technology.

Supporting data services backed by caCORE SDK version 4.4

The artifacts and API of caCORE SDK 4.4 is somewhat different from the functionality found in version 4.3, which led to issues for users who attempted to build caGrid services backed by caCORE SDK 4.4. A new data service style was built to facilitate this support, and released as a community project and update to the Introduce toolkit.

To support upgrades from one version of caGrid to another, and ensure a more seamless user experience, this project was rolled in to the main caGrid codebase and ships out-of-the box.

Community validation of the changes made by the SHA 2 support release

The changes required to support SHA 2 certificates in caGrid require some work on the part of developers of grid services and, to a lesser extent, application developers. Since the changes made involved substantial reworking of the security code behind caGrid, validation of its correctness is essential. Facilitating both processes requires some ramp up for the developers, as well as a convenient means by which the code can be tested.

To kick-start community testing of the SHA 2 support grid, we created a development grid instance and made it accessible to the public. This gives us a sandbox environment in which we can publish the latest core services, and allows downstream developers a chance to integrate with the new release.

Recommendations for future enhancements

The caGrid development team is in a unique technical position in which user requests, use cases, and ideas are often discussed and considered. The team also balances this against the technical direction the NCI wants to move in, and the prevailing trends in industry. With this background, the development team makes the following recommendations for future enhancements to caGrid:

Migrate the system to the latest and final WS-* specifications

The caGrid system is built on a version of the Globus toolkit which predates many of the final WS-* specifications. Migrating to these final specs would increase interoperability with other standards-based platforms.

Leverage WS-Security

A move to WS-Security would increase interoperability with other standards-based platforms, but must be weighed against the risk of breaking backwards compatibility with the existing implementation of caGrid. WS-Security and the related WS-Trust would provide additional standard mechanisms for authentication and authorization not currently available in caGrid.

Replace Globus 4.0 with a modern web services framework

The Globus toolkit does not take advantage of many of the built-in networking features of the Java platform, which leads to scenarios where proxy configuration becomes difficult. The age and relative obscurity of the Globus 4.0

toolkit also reduces the pool of developers who are familiar with the technology and ready to work on the platform.

Streamline the workflow to build model driven services

The process and timeline to create a UML model, move through the NCI toolchain, and arrive at a caGrid data service is daunting enough to significantly impact uptake of the system. A simplified, fast, and iterative process with a very low learning curve is imperative for continued success.

Potential implementation strategies

As part of the development team's technical due diligence, we make recommendations on potential implementation strategies for new features. For the new feature recommendations above, we've outlined some methods of implementation:

Migrate the system to the latest and final WS-* specifications

A web services toolkit which supports the file WS-* is a requirement, but the choice of toolkit has implications for the rest of the grid tools and services. There is essentially no way to make this move without breaking backwards compatibility, however some options exist that would make migration more straightforward.

The path of least resistance and fastest implementation may be to move to Globus 4.2. This version supports the final WS-* specs, but maintains the core WSRF functionality on which most of the grid services rely. From an API perspective, it is substantially similar to Globus 4.0.3, and so it shouldn't require an extensive rebuilding of the grid services to utilize. This might also make adoption easier for users of the current grid infrastructure.

Another option is a wholesale replacement of Globus with a framework like Apache CXF. This would have the simultaneous effect of expanding the potential developer base, as CXF is a fairly common framework, and of ensuring all grid services are stateless services. The obvious drawback is that removing WSRF from the grid will require extensive reengineering of the core services and of any applications that rely on functionality provided by WSRF.

Leverage WS-Security

Implementation of WS-Security should happen concurrently with the move to a final WS-* spec-compliant system. Globus 4.0.3 does not natively support WS-

Security, but a toolkit like CXF does, or can be made to do so much more readily than Globus 4.0.3. The WS-Security specs reference the final WS-* specs internally as well, which makes the move to WS-* a requirement.

Replace Globus 4.0 with a modern web services framework

Most of the underlying code in the core grid services could be ported to work inside a modern web services framework, albeit with significant considerations given to design in areas where WSRF resources and stateful services are currently used. For example, the Federated Query Processor (FQP) is implemented first as an engine and an API, which is then wrapped by a grid service. The API can be used standalone, or incorporated inside a web service generated with the framework of choice.

A framework could be as simple as Apache Axis 2 if the only requirement is for SOAP services. CXF is a more natural choice if different sorts of endpoints, like REST are a requirement. Consideration should be given to advanced platforms like WSO2 as well, with an analysis of the tradeoff in terms of overhead and licensing weighed against the additional out-of-the-box functionality being made before a decision is rendered.

Streamline the workflow to build model driven services

One way to gain a substantial improvement to the overhead of developing services based on a model driven architecture would be to leverage Enterprise Architect's plugin and scripting system. Simply removing the overhead and time required to export models out of EA and into tools like SIW and caCORE SDK would vastly improve the user experience and turnaround time to build services.

Meeting the mission, goals and objectives of this effort

The caGrid development team has delivered on the objectives of providing continued support for the caGrid core services and users of the system during the execution of this Statement of Work. Several new features have been identified from interaction with the user community, and many have been implemented to support their scientific use cases. The level of technical support has remained very high on the knowledge center forums, which facilitates adoption of the caGrid system. The development team has also integrated support for the latest versions of NCI tools like the caCORE SDK, which keeps the caGrid platform current and relevant.

Additionally, the team has provided support to the NCI directly for the production installation of caGrid, and offered insight into and level of effort estimates for feature requests driven by the adopter community and the NCI itself.

Lessons Learned

In the course of executing this Statement of Work, the caGrid development team encountered some learning opportunities.

1. The general cutbacks and shutdown of many caBIG development projects impairs the ability of the caGrid team to implement and release new features. Since caGrid is an "upstream" dependency of many other caBIG projects, features like the SHA 2 support can't be released until there are developers available to work on the other projects and incorporate the requisite changes.
2. Somewhat related to the previous item, backwards compatibility is a major concern for the NCI when changes are proposed in caGrid. This leads to architectural decisions, which are often a compromise against the optimal design in the interest of supporting users, and services that don't have the resources to perform updates and new releases.
3. Barrier to entry, whether real or perceived, is a key factor to adoption and uptake of caGrid. Concurrent with this, a simple and clearly described explanation of the benefits of using the grid and an overview of the underlying processes it abstracts away is essential to driving early interest.

Outstanding Issues

At the conclusion of this SOW and as of the time of this writing, the following are the remaining issues not yet closed out that were surfaced during the period of the SOW.

Bugs

Key	Summary	Priority	Affects Version/s
CAGRID-788	Website column of cat_entry table stores URL as serialized java.net.url	Minor	
CAGRID-787	Catalog editing issues in Portal 3.5	Minor	
CAGRID-784	Introduce shouldn't display extensions which are flagged as "deprecated" and "should be removed"	Minor	caGrid 1.5, caGrid 1.6

CAGRID-737	WEBSSO incorrectly configures Authentication and Dorian clients	Minor	caGrid 1.4
CAGRID-735	WEBSSO hides exceptions returned from Dorian	Minor	caGrid 1.4

Feature Requests

Key	Summary	Priority	Status	Affects Version/s
CAGRID-797	Group Search	Minor	Open	
CAGRID-795	Update authentication Service to include configurable lockout settings	Minor	In Progress	caGrid 1.4
CAGRID-794	Upgraders for SDK 4.2 data services from 1.5 and 1.4 to 1.6	Minor	Open	caGrid 1.3, caGrid 1.4, caGrid 1.5, caGrid 1.6
CAGRID-793	Upgraders for SDK 4.3 data services from 1.5 and 1.4 to 1.6	Minor	Open	caGrid 1.3, caGrid 1.4, caGrid 1.5, caGrid 1.6
CAGRID-790	Data Service Upgrader for 1.4 and 1.5 to 1.6	Minor	In Progress	caGrid 1.6
CAGRID-781	Add exposed validateQuery operation to Introduce generated Data Services	Minor	Open	caGrid 1.4
CAGRID-775	Support for JBoss 5.1 in caGrid 1.4	Minor	Open	caGrid 1.4
CAGRID-768	Support for JBoss 5.1.x running wsrf grid services	Minor	Open	caGrid 1.6
CAGRID-767	WS-Enumeration extension upgraders from 1.4 and 1.5 to 1.6	Minor	Open	caGrid 1.4, caGrid 1.5, caGrid 1.6
CAGRID-763	xmiToDomainModel generates invalid domain model when primitive types are used in XMI	Minor	Open	caGrid 1.4
CAGRID-762	Add ability to completely remove a user from Grid Grouper	Minor	Open	caGrid 1.4
CAGRID-761	Add ability to specify that a Group can only allow members from a specific authentication Service	Minor	Open	caGrid 1.4
CAGRID-760	Allow Dorian Administrator to manually add an IFS user	Major	Open	caGrid 1.4
CAGRID-755	Merge caCORE SDK 4.4 data service style project into caGrid 1.5 and trunk	Minor	In Progress	caGrid 1.5, caGrid 1.6
CAGRID-754	Upgraders for SDK 4.4 data services from 1.4 to 1.5	Minor	In Progress	caGrid 1.3, caGrid 1.4, caGrid 1.5, caGrid 1.6
CAGRID-717	Give IdP administrators a way to unlock locked accounts	Minor	Open	caGrid 1.6

Dependencies and Risks

Volatile external dependencies of the caGrid system are few from a technology perspective, however a few have the ability to impact caGrid in a negative way.

1. caCORE SDK. The SDK utilizes code from caGrid to handle CQL internally. After much conversation with the caGrid project lead, it was decided to remove this circular dependency by dropping support for native CQL processing out of the caCORE SDK. This will happen in some future release of caCORE SDK, but caGrid won't see the benefit until a data service style is created for that version.
    a. The data service style can easily be released independently of the rest of caGrid and distributed through Introduce's standard update tools.
2. Changes to, or bugs found in the CSM API may impact the way caGrid's authentication mechanisms interact with systems like LDAP for identity provisioning.
    a. caGrid has recently pulled a copy of the CSM API in to its codebase for the purpose of maintaining the changes and bugfixes required. Since the version of the API required by the authentication providers is no longer actively maintained, this was necessary to solve ongoing issues such as account lockouts.
3. The NCI approved tech stack calls for specific versions of tools such as Hibernate and Spring. caGrid cannot always move to the approved versions due to reliance on other tools which depend on older versions, such as the caCORE SDK.
    a. The tech stack similarly calls for specific versions of Tomcat and JBoss. While caGrid itself is migrating, and has largely migrated, to the recommended versions, it's important to note the impact this has on backwards compatibility, and that some existing tools may rely on functionality that is different or no longer provided in the new versions.

Due to the unique and central position of caGrid in the NCI technology portfolio, it is also subject to non-technical risks and dependencies, such as:

1. Conflicting requirements for backwards compatibility and advances in technology.
2. Lack of funding for other project development teams to adopt changes in caGrid makes it difficult to implement features that would require changes in their applications as well. This makes high priority features such as the SHA 2 support release impossible to roll out.