Robert Macura M.D. Ph.D.

Background:

The cancer Biomedical Informatics Grid (caBIG) is a voluntary, virtual informatics infrastructure that connects data, research tools, scientists, and organizations. Its goal is to speed the delivery of innovative approaches for the prevention and treatment of cancer. By combining shared vocabulary, data elements, and data models caBIG is expected to become a common-standard informatics platform used for all cancer research supported by NIH grants.

Microsoft Office Excel's functionality, especially in terms of statistical analysis and visualization, has contributed to its wide popularity in the scientific community. For over a decade, Excel's spreadsheet has been the primary tool used by biologists and biomedical scientists to analyze their cancer research data.

As part of the **Microsoft Excel caBIG Smart Client** (xl-caBIG) project we are developing extensions to Excel that allow users to access caBIG data-services. We strongly believe that xl-caBIG smart-clients, by leveraging the scientists' intimate familiarity with MS Excel and the wealth of cancer data available on the caBIG network, will become very useful to the caBIG community.

Core Functionality:

xl-caBIG's core functionality we consider centrally important to the user's ability to access caBIG data.

I. Scientists will browse and search, using xl-caBIG's Windows Form Contextual GUIs, the published Index Service in order to find caBIG data-services relevant to their research.

   a. The GUI will be able to display all available data-services. This will initially be an extremely useful view as there are few data-services currently on the caBIG network. As the caBIG grows, search will significantly increase in importance (see ***I.b& I.c***). [ALPHA]
   b. Users will be able to filter data-services by data-service metadata descriptions (e.g. all data-services originating from a particular cancer research center). [BETA]
   c. Users will also be able to filter data-services based on service-type (e.g. which data-services provide data of a particular type: 'gene') [1st RELEASE]

II. After selecting a relevant data-service's data-type, users will be able to consume its objects as cells in their xl-caBIG smart-clients workbook.

   a. get-all [ALPHA]
   b. filter data by limiting the fields of the data-type that will be retrieved [BETA]
   c. filter data by limiting certain fields of the data-type to specific values (For numerical fields xl-caBIG will support operators: ==, !=, <., and >. For string fields: == and !=) [1st RELEASE]
   d. expand on II.c by supporting boolean operators (&& and ||) to construct more complex filters [2nd RELEASE]

III. Since some (currently a small percentage) data-services are only available to authenticated and authorized users, xl-caBIG smart clients should support User Credential Management via Grid User Management Service (GUMS).

   a. The smart-client must interface with *GUMS's User Interface* so users can access their credentials, create and destroy proxies, and use them to invoke grid services that require them. Security issues pertaining to transmitting the password and persistently storing credentials will have to be solved. [1st RELEASE]
   b. xl-caBIG should additionally provide access to *GUMS's Registration Interface* process so new users can apply for caBIG credentials. [2nd RELEASE]

Additional Functionality:

We have identified features that improve xl-caBIG's ease of use but don't affect smart-client's core functionality.

1. Use a "bookmarking" interface to store discovered caBIG data-services for the duration of the open xl-caBIG workbook.
2. Store user credentials for the duration of the open workbook (i.e. so they don't have to be re-typed every time access to a secured caBIG data-service is initiated)
3. Devise a mechanism so that bookmarks and user-credentials are stored when the document is saved. Security ramifications will need to be thought through.
4. What sort (if any) caching mechanism is appropriate to optimize bandwidth and load on the caBIG network? We are considering a system for updating data between the workbook and caBIG data resource where we only download data that was created or updated since the last time data was downloaded to the workbook.

xl-caBIG Client-Server Architecture:

The current test architecture of caBIG is ***caGrid 0.5***.

caGrid confirms to Open Grid Services Architecture (OGSA) grid infrastructure standards. It is implemented using OGSA reference implementations: Globus Toolkit 3.2 and the OGSA Data Access Integration (OGSA-DAI) framework version 5.0.

caGrid leverages the Globus Toolkit by providing the required core services (e.g. caDSR and EVS), toolkits and wizards for the development and deployment of community provided services and APIs for building client applications.

The Globus Toolkit is written in Java 1.4 as are the caGrid high-level APIs and toolkits. Visual Studio Tools for Office provides the ability to integrate .NET managed code, in Visual Basic .NET or C#, with Excel 2003.

Since caBIG data-services are simply web applications that respond to XML formatted requests, smart-clients could essentially be constructing appropriate XML queries and reformatting the serialized XML responses as cells in the Excel Worksheet. Developing xl-caBIG in this way would mean reimplementing caBIG high-level Java API's in a .NET language such as C#. We considered and rejected this potential approach because it would be essentially recreating the work of the caBIG community and would be very brittle to changes in the caBIG or the Globus Toolkit APIs.

An alternative approach to implementing xl-caBIG smart-clients would involve using client/server architecture and intermediary middleware to bridge the gap between the caBIG Java API hooks and the .NET managed code in which the Excel smart-clients will be written. The server would be developed in Java as a thin wrapper around the caBIG high level API and would use middleware to service requests from the .NET Excel smart-clients.

We are using the open-source middleware platform ICE (Internet Communication Engine) from ZeroC (http://www.zeroc.com/). ICE is suitable because the client and server can be written in different programming languages including .NET and Java.

Interfaces, operations, and the types of data that are exchanged between the client and server are defined using SLICE (Specification Language for ICE). The client-server contract defined in the Slice language is independent of a specific programming language. The Slice definitions are compiled for specific programming languages by ICE into an API of generated code.

Since the xl-caBIG smart-clients will be implemented in the .NET framework, they will be leveraging .NET's features for managed deployment. Managed .NET code allows updates (in the form of

DLLs) to be detected and downloaded from servers but also allows the user to exercise strong control over how the code will be executed. The document-centric model means updates will not be a new version of the document, just a version of the DLLs.

The Java implemented xl-caBIG servers, because of their strong dependencies on caGRID 0.5 and underlying toolkits, are anticipated to be much more difficult to install and configure than the xl-caBIG Excel client.

xl-caBIG .NET smart-client source-code, xl-caBIG Java server source-code, and the intermediary SLICE definitions will be made available under a BSD open source license on our SourceForge website: (http://xl-cabig-client.sourceforge.net/). We will be hosting an 'official' xl-caBIG Java server that will be the default server to xl-caBIG smart-clients. Advance users with special needs in terms of performance or availability will be able to roll-out their own xl-caBIG servers their smart-clients can connect to.

We aim that the overhead to using caBIG through MS Excel be negligible. ICE uses a binary protocol that has turned out for other applications to be very efficient in network bandwidth, memory use, and CPU overhead. Since ICE supports replication and load-balancing, as the xl-caBIG smart-clients user base grows the xl-caBIG server supporting them will easily scale by dedicating more hardware.

Releases:

We are planning on developing xl-caBIG smart-client/servers using the Extreme Programming methodology-- regular releases and incremental development. We have identified the features (see Core Functionality/Additional Functionality Sections) that we aim to include for each of the xl-caBIG releases: Alpha (March 2006), Beta (April 2006), Release (August 2006), Follow-up (October 2006).