Remote Instrument Control with CIMA Web Services and Web 2.0 Technology

Douglas du Boulay,[1] Sandor Brockhauser,[2] Clinton Chee,[1] Kenneth Chiu,[3] Tharaka Devadithya,[4] Richard Leow,[1] Donald F. McMullen,[4] Romain Quilici,[1] Peter Turner[1]

[1]Department of Chemistry, University of Sydney, Sydney, NSW 2006, Australia.
[2] Instrumentation Group, European Molecular Biology Laboratory,Grenoble, France.
[3]Computer Science Department, SUNY Binghamton, Binghamton, NY, USA.
[4]The Pervasive Technology Labs at Indiana University, Bloomington, IN 47404 USA.

***Abstract*—The Common Instrument Middleware Architecture (CIMA) model for Web services based monitoring of remote scientific instruments is being extended and enhanced to provide a capability for remote instrument control. X-ray diffraction has been selected as an ideal domain for prototype development, with the goal being a comprehensive and feature rich portal system for access to remote instruments and their data. The system has two principle components, one of which serves the instrument and data, and the second serves the client user. Plugin modules are used to provide flexibility and re-use, and the notion of plugin control is being developed. The architecture supports remote access to multiple instruments from a single portal. The use of Web 2.0 Pushlet and AJAX technologies has been introduced for push based portlet refresh and updating. An X3D based 3D virtual representation of the instrument provides data collection simulation and (pseudo) real time instrument representation**

***Index Terms*—remote instrument control, Web services, Web 2.0, middleware, CIMA, virtual instrument.**

I. INTRODUCTION

There are obvious financial, functional and educational returns in developing collaborative access services, including instrument control, for remote scientific instrumentation. State of the art high performance laboratory instrumentation, such as high flux X-ray diffraction systems and powerful electron microscopes, is increasingly expensive and too costly to replicate in multiple locations. Not only is there the high initial capital cost, there is the on-going burden of technical staffing and specialised maintenance costs. Remote access services would maximise returns on the high construction and operating costs of landmark national research facilities, such as synchrotrons and neutron sources.

The remote desktop approach to remote instrument access, such as typified by the use of Virtual Network Computing [1] and its many variants, CITRIX [2], Sun Secure Global Desktop [3] and NX NoMachine [4], has the significant advantages of ease of set-up and familiarity. While convenient, these approaches are not ideal and can afford remote instrument users with excessive control over expensive and potentially dangerous instruments.

A significant disincentive to building custom-built remote access systems, is that there is a high coding overhead that may well reproduce functionality already provided by an instrument manufacturer. A significant advantage of the custom built interface approach however, is that the actions of the remote instrument user can be tightly controlled, while at the same time services outside the desktop environment can be provided to offer a richer operating environment.

The use of portal-portlet technology has a number of incentives, including the provision of rich functionality and global accessibility, and no need for any client software other than a suitable Web browser. The performance distinction between stand alone interfaces and browser based interfaces is being eroded by Web 2.0 technologies, such as AJAX [5] and DOJO [6].

Web services offers several compelling benefits:

- Location, platform and language independent. Facilitates legacy code re-use.
- Facilitates integration of the instrument into the Web (and Grid).
- Robust underlying security model (WS-Security supports security tokens and certificates).
- Use of HTTP as the underlying Web transport protocol facilitates firewall passage.
- Supports Service Orientated Architectures (SOAs). Can integrate distributed and heterogeneous services.
- Facilitates the linkage of multiple users and resources for collaborative interactions across the Web.

The use and performance of Web services for the remote control of relatively simple laboratory devices, such as a waveform generator, has been described by Yan et al. [7]. At the other end of the spectrum, the GridCC project to Grid enable large scale facilities, such particle accelerators, is underpinned by Web services [8].

The Common Instrument Middleware Architecture [9-11] started as a NSF Middleware Initiative project researching a consistent and re-usable middleware framework to enable and embed instruments as addressable Web and Grid resources using Web services. The CIMA model is intended to be scalable across the domain of scientific instruments. Thus far CIMA has been developed solely for remote instrument and sensor monitoring, although CIMA mediated instrument control has been outlined [12].

Herein we describe extensions and enhancements to the CIMA model, being undertaken as part of the development of a comprehensive and feature rich remote access portal system that includes remote instrument operation and control [13]. In part two of the paper we present the overall system architecture and the rationale behind that architecture. The two component architecture is explained, and examples are given of XML parcel types that have been introduced to support instrument control. The notion of plug-in control is introduced. The implementation and application of the system is then described in part three, and this includes the nature of the supporting technologies used in the implementation. Finally, a summary is presented in part four.

II. ARCHITECTURE AND PRINCIPLES

A modular service oriented architecture (SOA) model using Web services has been adopted to provide maximum flexibility and capability. The principle components of the portal system are shown in Fig. 1.

As the figure suggests, the use of browsers to provide the user interface is inherently collaborative. While only one person can be authorised to control the instrument, multiple collaborators may access the portal, subject to authorisation, and hence monitor a data collection and inspect the data being generated. The users may thus collaboratively determine the best way to undertake a data collection.

The system architecture is comprised of two primary containers; one located at the instrument site (Source, or Producer) and providing instrument services, and a second that provides user access interface components (Sink, or Consumer) and need not be co-located with the instrument. For instance, several institutions under a common project may want to provide remote access to a shared instrument or set of instruments. In that case a user interface portlet container may be beneficially located at each of the user institutions. Alternatively, container A could be shared between multiple institutions to provide remote access to multiple shared instruments (not necessarily located at one site). The second model may be desirable for large facilities such as a synchrotron, for which a single Container B could serve multiple instruments or beamlines. A third model would have both containers located at the instrument site or facility, but with Container A residing in a DMZ.

Both Source and Sink containers have Web services components that receive SOAP calls from the corresponding partner container. Complementary Web services stubs (not shown in Fig 1) are responsible for assembling and sending the SOAP messages. The CIMA components assemble and provide the XML parcels delivered in the SOAP messages. Communication between the complementary Source and Sink CIMA components is formally described in terms of *channels* that are, in effect, defined by XML parcel exchange endpoints.

Flexibility and a capability for re-use is provided through the use of plugins to the CIMA components of each container. The relationship of the plugins to the CIMA framework is schematically illustrated in Fig. 2. Channel and plugin access requires a registration or subscription process, and registration depends on client authentication and authorization. Registration thus determines which services are made available to the client.

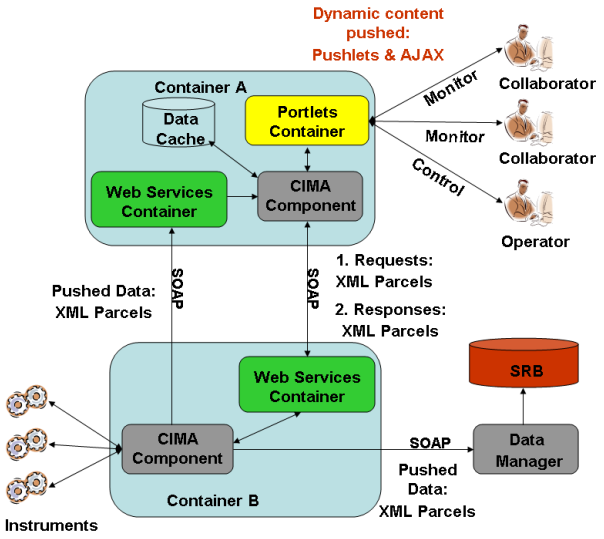

Figure 1. Remote instrument control elements. Only one person at a time can control any given instrument.

The introduction and development of instrument control services has been undertaken in accord with the CIMA channels and plug-in model [12]. Plugins may serve individual sensors and/or actuators, or may serve the aggregation of sensors and actuators that defines an instrument. Instrument control has been implemented through the introduction of an Instrument Instruction Module and its partner plugin, the Instrument Monitor Module. The Instrument Instruction Module serves as an instruction interface to specific instrument software (or device drivers). That is, the module translates CIMA parcels (see below) into instrument specific instructions to be sent to the instrument interface. The instructions may be to get instrument status information, change the state of the instrument or operate the instrument (e.g. collect data).

As Fig. 2 suggests, plugins may also be used for individual sensors and actuators. Utility plugins may also be used and we have, for instance, introduced a plugin type for the conversion of an instrument data image into an image format suitable for display in the client browser interface.

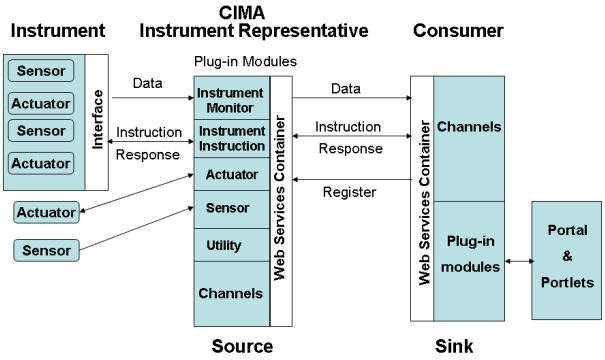

Figure 2. CIMA based components: Source components reside in Container B and the Sink components are located in Container A (see also Fig 1).

Data management may also be provided through an additional plugin or via Web services access to a separate management service [13].

Plugin selection and configuration is currently static, in the sense that the specification is provided in an XML configuration file loaded on container start-up. The same process provides for service registration and hence channel definition. Registration details are persistent in that they are stored in a file that can be read if Container B needs to be re-started.

There are two primary modes of communication between the two containers. One mode is synchronous and involves a request/response pair in which a request is sent from container A to container B, and for which a response is then expected in return. The CIMA component in Container B extracts data from the XML parcel included in the request from A, builds instrument-specific instructions and finally returns a response from the instrument as an XML parcel. The response may simply be an acknowledgement, or data/metadata associated with the instruction. New parcel types have been introduced to support the request/response mode (see below).

Requests are of two kinds; those that affect the instrument state (e.g. SET and OPERATE requests for instrument control) and those that retrieve instrument information (e.g. GET for instrument status data).

The second type of interaction occurs when an instrument pushes information. The push may be the result of an earlier asynchronous request, state changes in the instrument, or because the instrument otherwise sends data on a regular basis. In this case the CIMA component of Container B (Source) sends an XML parcel via SOAP to the CIMA component of Container A (Sink). The parcel content is then extracted and relevant data is transferred into a temporary store, or Data Cache, and the portlet content is updated.

The parcel structure is simple and flexible, and has an associated schema. The following example illustrates a GET request to retrieve an instrument parameter (a goniometer angle in this case):

```
<Parcel>
  <Type>http://www.cima.usyd.edu.au/2006/Get</Type>
  <Body>
      <Source>BIS</Source>
      <Variable>OMEGA</Variable>
  </Body>
</Parcel>
```

The parcel returned by the Source has a similar structure:

```
<Parcel>
  <Type>http://www.cima.usyd.edu.au/2006/Command_Response</Type>
  <Body>
      <Value>-295.41</Value>
      <TimeStamp>2006-10-23 06:03:58 UTC</TimeStamp>
  </Body>
</Parcel>
```

A further example illustrates the simple nature of an OPERATE instruction parcel

```
<Parcel>
  <Type>http://www.cima.usyd.edu.au/2006/Operate</Type>
  <Body>
      <Source>BISControl</Source>
      <Command>DRIVE</Command>
      <Parameters> … </Parameters>
  </Body>
</Parcel>
```

Although the skeletal parcel structure is simple, quite complex parcels can nonetheless be assembled, and this inherent parcel flexibility facilitates the introduction of new parcel content and new control and monitoring system modules. Currently for instance, we are developing a plugin module for TANGO device servers [14] (Fig. 3). TANGO is an object oriented and distributed control system using CORBA [15], and is being developed as an open source collaboration between the Alba, Elettra, ESRF and Soleil synchrotron facilities.

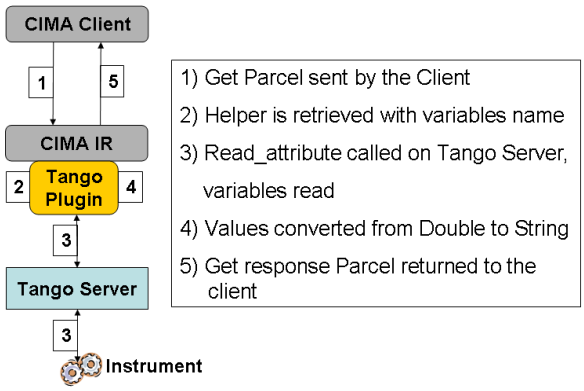

Figure 3. TANGO instrument control system as a CIMA plugin.

The flexibility of the parcel schema means that only formal Web services method required for the Source and for the Sink, and is a method for the receipt of parcels.

The Data Cache contains status information about the instrument as well as temporary files (such as data frame images for portlet display and review), and is populated when data pushed from the instrument arrives, or when a response containing instrument information is received. The Cache is used to minimise SOAP calls to Container B, when a GET request is issued and the desired data is already available in the cache. The cache has the same lifecycle as Container A, and might not be applicable for all instruments.

X-ray diffraction instruments are typically equipped with CCD based detectors that return data as binary images, with ASCII headers. The size of these images, or frames, depends on the nature of the detector and may vary from less than a Mb to several Mb. Several thousand such images will be accumulated during a data collection. The use of Web services to transport data images may then be problematic. In addition to the default push of base64 encoded images via SOAP, we have introduced an optional push/pull mode to avoid bottlenecks arising form the accumulation of images in transport and out of memory exceptions. In the push/pull mode a Binary Ready parcel containing the URL of the image is sent to the registered Sink; the image itself is not sent. The Sink can then retrieve the file when ready, using what ever means appropriate (SCP, FTP, gridFTP). Recent evaluations suggest that the use of SOAP with attachments provides better performance than the direct incorporation of base64 encoded images in XML parcels.

Recently we have introduced a capability for plugin control by the on-site administrator and, to a lesser extent,

by the remote user. The currently rudimentary capability (Fig. 4) provides a basis for exploring the potential utility and benefit of such control. The START instruction starts the plugin, and cannot be called remotely. STOP stops the plugin, but keeps the plugin registered and remotely accessible. RESTART stops the plugin and then starts it again, and it remains remotely callable. TERMINATE removes or cleans the plugin from the application. After a TERMINATE instruction, the plugin cannot be accessed remotely. Further plugin handling instructions planned but not yet implemented include SUSPEND and RESUME for thread control.

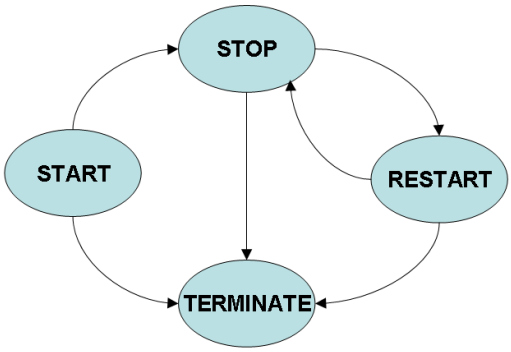

Figure 4. Schematic depiction of current capability for the plugin 'state machine'.

III. IMPLEMENTATION AND TECHNOLOGIES

Crystallographic molecular structure determination instrumentation has been selected as a particularly attractive system application development domain, with well defined work-flows and data structures, and relatively common (if not standardised) instrument types. Crystallographic data collections are accordingly well suited to remote control.

The two primary system components (Source and Sink) are deployed in a Servlet container, and currently we use either Tomcat [16] or Jetty [17]. Tomcat may used for both containers, however there is also the option of using Jetty for Container B. The use of Jetty allows multiple instruments to be served through multiple 'stand-alone' instances of the Instrument Representative (IR) of Container B (see Fig 2). Perhaps a drawback in some contexts, if each instance resides on the same machine then each must be assigned a distinct port number. The location of a Container A in a DMZ would mitigate this drawback.

Multiple Tomcat instances could be used, however we find Jetty has the advantage of being relatively lightweight. While suitable for development, we have yet to determine if Jetty would be suitable in a production setting.

Alternatively, and perhaps more attractive, the introduction of plugin control facilitates multiple instrument access through the loading of multiple instrument modules in the Instrument Representative within a single Tomcat instance of Container B.

We have used both Axis [18] and CXF [19] for the provision of Web services and currently use embedded CXF as this allows programmatic initiation. Any JSR 168 compliant portlet container may be used for Container A, and we currently use GridSphere [20].

The goal of real-time updating of instrument status and data displays for effective and safe instrument control has driven our introduction of Pushlet [21] and AJAX Web 2.0 technologies to enable (pseudo) real-time data push from the instrument to the client. It is then possible for instance to view CCD based X-ray detector images, along with metadata, without polling. The utilization of these technologies has in turn improved the functionality of our instrument monitoring portlets. AJAX is attractive in allowing the updating of the content of a particular portlet, without the need to reload all of the other portlets. DWR [22] was used to facilitate the use of AJAX.

The current form of the instrument access portal system provides an instrument control pane (Fig. 5) allowing the user to define and initiate simple data collections. Options for the provision of more complex data collections are currently being evaluated. The pane provides for data collection parameter input, webcam monitoring of the instrument and crystal sample, and a display of the current CCD detector data image. As mentioned, images and data are dynamically updated through the use of AJAX and Pushlet technologies.

Another tabbed pane augments the browser interface with instrument status information and, for example, displays the X-ray generator voltage and current settings, the cooling water temperature, the CCD based X-ray camera temperature, and laboratory temperature and humidity (Fig. 6). There are also tabbed panes for diffraction image inspection. An individual image may be selected for display in the portlet, or a range of images may be selected and viewed at a user selected display rate (image set animation). In this manner the quality of the data may be quickly assessed, and a decision may then be made to continue or to abort the data collection. A Web services driven tool for multi-user collaborative image inspection is also being developed to enhance the collaborative capabilities of the portal system. Students or less experiences researchers may thus consult with remote experts, and jointly determine the merits of a sample.

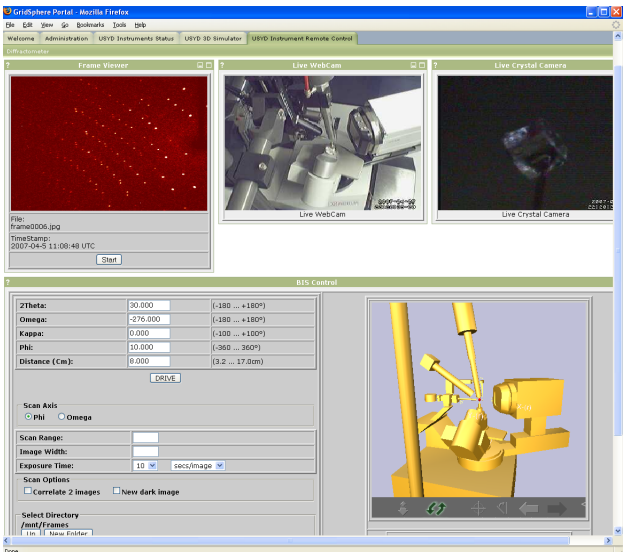

Figure 5. GridSphere portlet for X-ray diffraction instrument control. The current diffraction data image is shown in the top left portlet.

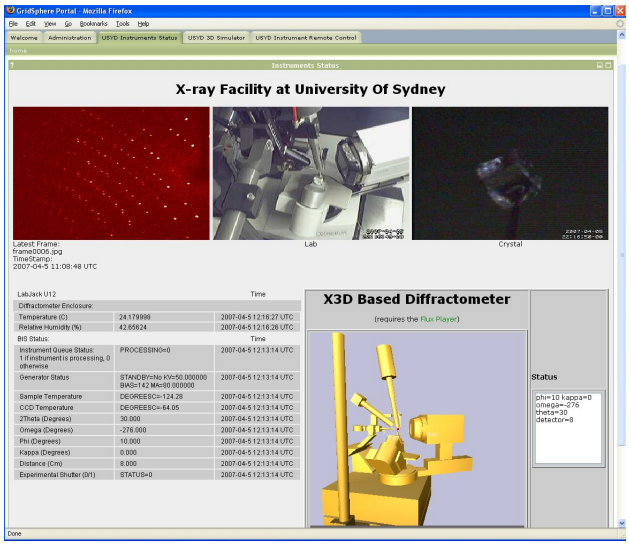

Figure 6. Portlet providing instrument status information.

An important element of the control portlet is an X3D [23] based virtual representation of the instrument. This provides an easy to assess visual representation of the current state of the instrument and has low bandwidth requirements, as only a small number of (pushed) instrument parameters are required to update the model in the client browser. The virtual representation at least partially solves the 'dark laboratory' problem that arises when laboratory lights are turned off, or when a webcams fails.

The use of X3D has several attractions, including being ISO standards compliant, independent of the source of the XML and extensible such that the component architecture is easily extended and use can be made of XML schema and, conceivably, an instrument ontology. The XML for an X3D model can be read or written using standard XML tools, and can be integrated seamlessly into any XML enabled application or Web service.

The following is a listing of the main X3D code used for the model shown in Figs. 5, 6 and 7:

```
<Transform rotation="0 0 1 0" scale="1.0 1.0 1.0"
                              translation="0.0 0.0 0.0">
  <ProtoInstance name="Machine_axes"/>
  <ProtoInstance name="BL19b-baseplate"/>
  <ProtoInstance name="ColdStream"/>
  <ProtoInstance name="Telescope"/>
  <Group DEF="Omega">
    <Transform DEF="omega_angle" rotation="0 0 1   0.00">
     <ProtoInstance name="Omega_Block"/>
     <Group DEF="Kappa">
      <Transform DEF="alpha_plus" rotation="0 1 0  -0.872665">
       <Transform DEF="kappa_angle" rotation="0 0 1 0.0">
        <Transform DEF="alpha_minus" rotation="0 1 0 0.872665">
         <Group DEF="Kappa-circle">
          <ProtoInstance name="Kappa_Base"/>
          <Transform DEF="phi_angle">
           <ProtoInstance name="Phi_Block"/>
           <ProtoInstance name="Crystal"/>
          </Transform>
         </Group>
        </Transform>
       </Transform>
      </Transform>
    </Group>
   </Transform>
  </Group>
  <ProtoInstance name="Collimator"/>
  <Group DEF="detector_block">
   <Transform DEF="twotheta_angle" rotation="0 0 1 0.0">
    <Group>
     <Transform DEF="detector_distance" translation=".170 0 0">
      <ProtoInstance name="Detector"/>
     </Transform>
     <ProtoInstance name="KBtn-TThetaCircle"/>
    </Group>
   </Transform>
  </Group>
</Transform>
```

X3D scenes can be dynamically updated using Javascript and the X3D Scene Authoring Interface (SAI). Currently we use FluxPlayer [24] as an SAI capable browser plug-in to display our virtual instruments. The user can 'zoom in' on the virtual instrument and adopt any viewing position around the instrument, including preset positions.

A strong disincentive to providing remote client control of a physical device is that unskilled operators, or simple data entry errors, may lead to costly damage to the instrument. Ideally the instrument control software installed at the remote site would include a collision map to prevent accidental damage. In practice however collision map software is not always provided, and when such software is available it may have weaknesses or bugs. The risk of collision damage can be reduced by limiting the functionality of the remote instrument access interface. The risk of damage can also be mitigated through the use of the virtual instrument as a simulator to test the viability of a data collection strategy (see Fig. 7). We are also exploring the use of the virtual model to automatically generate a collision map by exhaustively iterating through the range of possible component movements.

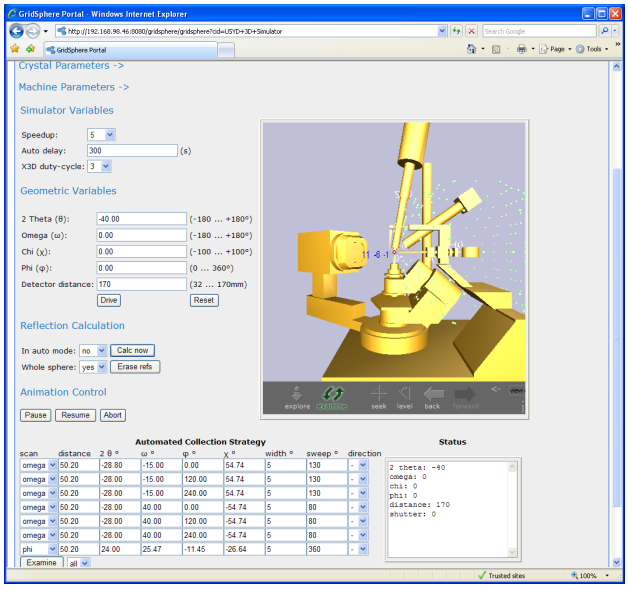

Figure 7. Portlet to operate a X3D based diffraction instrument simulator.

Given sufficient preliminary scanning information (sufficient to provide the crystallographic orientation

matrix), the simulator now has a capability to display the expected location of all of the diffraction data for the sample. The viability and efficiency of the data collection strategy can then be assessed.

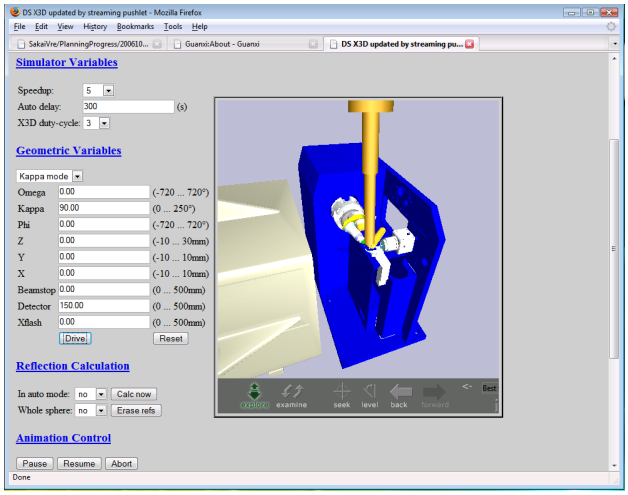

Figure 8. Portlet to simulate a synchrotron beamline X-ray diffraction instrument .

The virtual instrument offers a visual test-bed for developing further instrument control plug-ins. As mentioned above, the simulator is currently being linked to a TANGO device server, and the same could be done for the EPICS [25] control system. The development of a TANGO plugin for CIMA is being undertaken in conjunction of a virtual model for a synchrotron beamline X-ray diffraction system (Fig. 8). Synchrotron instruments are complex and pose a considerable modelling and remote access challenge.

The virtual instrument may also serve as an indestructible training tool that can be accessed via the internet from anywhere. Major facilities such as synchrotrons are expensive to operate and use, and the use of a virtual instrument for user training has obvious attraction.

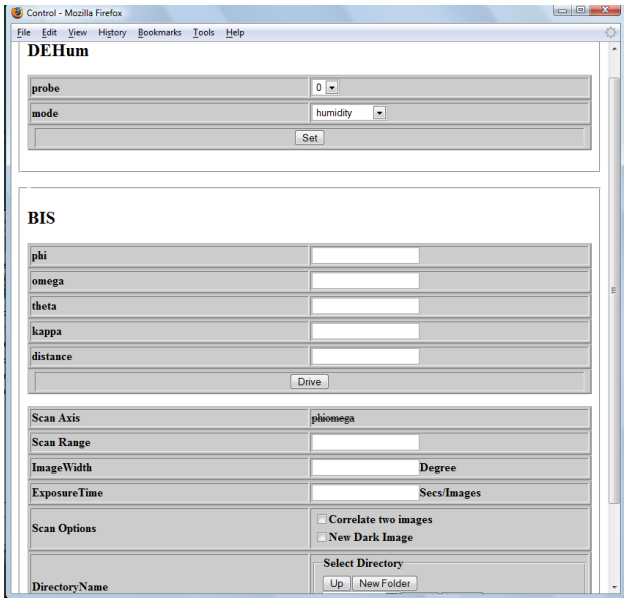

Figure 9. A crude browser based interface for instrument control generated with XSLT from an XML description file.

Recently we have begun to explore the possibility of using XML instrument and component plugin descriptions as a means of automatically generating browser interfaces via XSLT (see Fig. 9). This possibility has the attraction of facilitating the introduction of new plugins with minimal associated portlet development. The XML instrument description could further include an X3D representation of the instrument. There is then the intriguing prospect of a comprehensive and visualisable instrument description that could be used in developing an ontology for instruments. Preliminary work has been undertaken to investigate the potential of OWL and RDF representations of an instrument ontology.

IV. CONCLUSION

The Source and Sink components of the CIMA model have been augmented with components providing instrument and plugin control. These components form the core of a portal system for collaborative remote instrument control and monitoring, and which includes portlets for data inspection and experiment simulation. Multiple users may access the portal services, but only one user may control the instrument. Data collection can be simulated with an X3D based virtual instrument representation that also provides at least a partial solution for the dark laboratory problem. The virtual instrument may also be used to asses the viability of a data collection strategy, and can be used for instrument user training purposes. A TANGO device server client plugin module is being developed to provide a capability for remote monitoring and operation of synchrotron beamline instrumentation.

ACKNOWLEDGEMENTS

The support of the ARC e-Research SRI, GrangeNet and the ARC Molecular and Materials Structure Network (mmsn.net.au) is gratefully acknowledged. CIMA is supported by National Science Foundation cooperative agreements and grants SCI 0330568 and MRI CDA-0116050. The assistance of Kia L Huffman (IU) and Alexandre Grobbo (EMBL) is gratefully acknowledged.

REFERENCES

[1] VNC: Virtual Network Computing; http://www.realvnc.com/. Accessed 27 January 2008. Variants include TightVNC, RealVNC, UltraVNC, and TridiaVNC.

[2] CITRIX: http://www.citrix.com/. Accessed 27 January 2008.

[3] SSGD: Sun Secure Global Desktop: http://www.sun.com/software/products/sgd. Accessed 27 January 2008. Formerly Tarantella.

[4] NX NoMachine; http://www.nomachine.com/. Accessed 27 January 2008.

[5] AJAX: Asynchronous JavaScript Technology and XML http://java.sun.com/developer/technicalArticles/J2EE/AJAX. Accessed 27 January 2008.

[6] DOJO: http://dojotoolkit.org/. Accessed 27 January 2008.

[7] Y. Yan, Y. Liang, X. Du, "Controlling remote instruments using Web services for online experiment systems. ", *Proceedings. 2005 IEEE International Conference on Web Services, 2005 (ICWS 2005).*

[8] GridCC: www.gridcc.org. Accessed 27 January 2008.

[9] R. Bramley, K. Chiu, J.C. Huffman, K.L. Huffman, and D.F> McMullen, "Instruments and Sensors as Network Services: Making Instruments First Class Members of the Grid." *Indiana University Computer Science Department Technical Report 588,* December 2003.

[10] T. Devadithya, K. Chiu, K.L. Huffman, D.F. McMullen, "The Common Instrument Middleware Architecture: Overview of Goals and Implementation." *Proceedings of the First IEEE International Conference on e-Science and Grid Computing (e-Science 2005)*, Melbourne, Australia, Dec. 5-8, 2005: 578-585, IEEE Computer Society.

[11] R. Bramley, K. Chiu, T. Devadithya, N. Gupta, C. Hart, J.C. Huffman, K.L. Huffman, Y. Ma, D.F. McMullen, "Instrument Monitoring, Data Sharing and Archiving Using Common Instrument Middleware Architecture (CIMA)." *Journal of Chemical Information and Modeling*, **46**(3):1017-25, 2006.

[12] D.F. McMullen, T. Devadithya, K. Chiu, "Integrating Instruments and Sensors into the Grid with CIMA Web Services." *Proceedings of the Third APAC Conference on Advanced Computing, Grid Applications and e-Research (APAC05)*. Gold Coast, Australia, September 25-30, 2005. http://grid.cs.binghamton.edu/projects/publications/integrate-APAC05/

[13] I.M. Atkinson, D.J. du Boulay, C. Chee, K. Chiu, T. King, D.F. McMullen, R. Quilici, N.G.D. Sim, P. Turner, M. Wyatt, "CIMA Based Remote Instrument and Data Access: An Extension into the Australian e-Science Environment." *Proceedings of IEEE International Conference on e-Science and Grid Computing (e-Science 2006).* December 2006. Amsterdam, The Netherlands.

[14] TANGO: http://www.tango-controls.org/. Accessed 27 January 2008.

[15] CORBA: http:/www.omg.org/. Accessed 27 January 2008.

[16] Tomcat: http://tomcat.apache.org/.Accessed 27 January 2008.

[17] Jetty: http://www.mortbay.org/. Accessed 27 January 2008.

[18] Axis: http://ws.apache.org/axis/. Accessed 27 January 2008.

[19] CXF: http://incubator.apache.org/cxf/. Accessed 27 January 2008.

[20] GridSphere: www.gridsphere.org/gridsphere/gridsphere. Accessed 27 January 2008.

[21] Pushlets: http://www.pushlets.com/. Accessed 27 January 2008.

[22] DWR, Dynamic Web Remoting: http:/dwr.dev.java.net/. Accessed 27 January 2008.

[23] X3D: http://www.web3d.org/about/overview/. Accessed 27 January 2008.

[24] Flux Player: http://www.mediamachines.com/. Accessed 27 January 2008.

[25] EPICS: http://www.aps.anl.gov/epics/about.php/. Accessed 27 January 2008.

AUTHORS

Douglas du Boulay, Clinton Chee, Richard Leow, Romain Quilici and **Peter Turner** are with the Department of Chemistry, University of Sydney, Sydney, NSW 2006, Australia.

Sandor Brockhauser is with the Instrumentation Group, European Molecular Biology Laboratory, Grenoble, France.

Kenneth Chiu is with the Computer Science Department, SUNY Binghamton, Binghamton, NY, USA.

Tharaka Devadithya and **Donald F. McMullen** are with the the Pervasive Technology Labs at Indiana University, Bloomington, IN 47404 USA.

Manuscript received 28 January 2008. Published as submitted by the authors.