Chapter XX

WEB 2.0 FOR GRIDS AND E-SCIENCE

Geoffrey C. Fox[1,2,3], Rajarshi Guha[3], Donald F. McMullen[4], Ahmet Fatih Mustacoglu[1,2], Marlon E. Pierce[1], Ahmet E. Topcu[1,2], David J. Wild[3]

[1]Community Grids Laboratory, Indiana University, Bloomington, IN USA 47404, [2]Department of Computer Science, Indiana University, Bloomington, IN USA 47405, [3]School of Informatics, Indiana University, Bloomington, IN USA 47408, [4]Knowledge Acquisition and Projection Laboratory, Indiana University, Bloomington, IN USA 47404

Abstract: Web 2.0-style services and capabilities collectively define a comprehensive distributed computing environment that may be considered an alternative or supplement to existing Grid computing approaches for e-Science. Web 2.0 is briefly summarized as building upon network-enabled, stateless services with simple message formats and message exchange patterns to build rich client interfaces, mash-ups (custom, composite, Web applications), and online communities. In this article, we review several of our activities in these areas: service architectures for Chemical Informatics; Web 2.0 approaches for managing real-time data from online experiments; management and federation of digital entities and their metadata obtained from multiple services; and the use of tagging and social bookmarking to foster scientific networking at minority serving institutions. We conclude with a discussion of further research opportunities in the application of Web 2.0 to e-Science.

Key Words: Grid computing, Web services, Web 2.0, REST,

1. INTRODUCTION: BROADENING THE DEFINITION OF GRID COMPUTING

Distributed computing research has been greatly influenced by developments in Internet computing and e-Commerce. The key concept of these systems is the Service Oriented Architecture [1], in which network accessible services expose well-defined programming interfaces and communicate with well-defined, application-layer network protocols and message formats. The global scale of electronic commerce and online communities provides an unprecedented opportunity to investigate research problems in scalability, reliability, system robustness, system federation, and security.

The general Web Service Architecture [2] concepts that build upon service oriented architecture abstractions are typically specified using XML-based standards and recommendations. The Web Service Description Language (WSDL) is used to describe a service's capabilities. It also is a guide for requestor agents to construct valid messages for interacting with the service. SOAP is the most common network message format and encapsulates the specific communications between a requestor agent and a service (and also possibly routing intermediaries). SOAP is also an extensible format: additional directives for security, reliability, addressing, and similar higher level qualities of service are added as supplemental information to the SOAP header.

Grid computing [3], as it is normally defined, is aligned closely with Web Service Architecture principles. The Open Grid Forum's Open Grid Computing Architecture (OGSA) [4] provides, through a framework of specifications that undergo a community review process, a precise definition of Grid computing. Key capabilities include the management of application execution, data, and information. Security and resource state modeling are examples of cross cutting capabilities in OGSA. Many Grid middleware stacks (Globus, gLite, Unicore, OMII, Willow, Nareji, GOS, and Crown) are available. Web and Grid Services are typically atomic and general purpose. Workflow tools (including languages and execution engines) [5] are used to compose multiple general services into specialized tasks. Collections of users, services, and resources form Virtual Organizations are managed by administrative services. The numerous Web Service specifications that constitute Grids and Web Service systems are commonly called "WS-*".

There are alternatives to this approach. As we have shown in previous discussion [6], Web 2.0 collectively represents a comprehensive distributed computing paradigm that, while compatible with Web Service Architecture principles, is a challenge to the usual WS-* implementations. The initial applications of Web 2.0 to e-Science are surveyed at the workshop, "Web 2.0 and Grids", at Open Grid Forum 19 (see http://www.semanticgrid.org/OGF/ogf19/). See also keynote talks by Hey and Goble and also the panel discussion moderated by Fox at Grid 2007 (http://www.grid2007.org/). As we survey in this article, Web 2.0 approaches can meet the requirements of cyberinfrastructure activities (such as chemical informatics and online laboratories). It furthermore suggests a broader view of cyberinfrastructure to include support for user and community-driven metadata, social networking and online communities of scientists.

Key Web 2.0 Concepts: Web 2.0 is a diverse and uncoordinated activity, but we may characterize it generally as building on the following key concepts.

Representational State Transfer (REST) Services: REST services [7] stringently avoid issues of exposed state and provide a single programming interface (essentially, HTTP GET, PUT, DELETE, and POST) for all services. REST services operate on URLs, which may respond with XML messages as well as other formats. XML payloads may be in any format. News feeds using RSS and Atom formatting are typical examples. These are best known as news feed formats, but they can be used as general-purpose envelopes for conveying sequential data (similar to SOAP). JavaScript Object Notation (JSON) is an alternative to XML for message exchange. The combination of clients to multiple services in a single application is called a mash-up (see www.programmableweb.com). REST services are purposefully simpler and have less sophisticated interaction patterns than WS-* services, relying on human developers rather than tooling to construct applications.

Rich user interfaces based on JavaScript, AJAX, and JSON: Rich user interfaces make use of client-side Web browser scripting coupled with REST-style server calls to enable browsers to provide enhanced interactivity that overcomes the limitations of the HTTP Request/Response invocation pattern. Updated content is delivered from the server to the user based on interactive events such as keystroke and mouse events rather than through HTML FORM actions. Numerous JavaScript libraries and tools exist for creating these applications. These range from high-level JavaScript libraries (Yahoo's YUI, Prototype, Scriptaculous) to JavaScript and HTML generators (Google's GWT and Ruby on Rails, for example). Related commercial environments (Adobe Flash/Flex and Microsoft Silverlight, for example) also are available.

Virtual communities and social networks: Web 2.0 services are often designed to enable users to establish networks and share content (such as uploaded images, movies, and interesting Web sites). Online communities can develop in a number of ways, but a common example is through shared tags of online resources. Tags are single word descriptions of URLs. Collections of tags

that describe a particular online resource or type of resource are known as a "folksonomy", a coined term that indicates the emergent and user-driven nature of the description.

Widgets, gadgets, and badges: The content and metadata of Web 2.0 services such as Flickr and del.icio.us may be accessed through their Web sites, via RSS feeds, and via REST API calls. They may also be accessed via exportable badges (typically XHTML and JavaScript) that may be embedded by a user into other Web pages. Start Pages such as iGoogle and Netvibes aggregate these small, self-contained user interfaces to Web 2.0 services. Such pieces are known as widgets or gadgets and are conceptually similar to Java portlets that are often used to build science gateways [8]. Architecturally, badges and gadgets are very different from portlets, as the former are aggregated on the client side and are (essentially) unrestricted by the Web container, while the latter are aggregated on the server side and are under the control of the portal service provider. Gadgets can be shared and tagged to create user environments.

In the remainder of this chapter, we examine a range of applications and case studies using Web 2.0 for e-Science. Section II describes the application of service architectures to chemical informatics, in which we are supplementing WS-* services with Web 2.0 approaches. Section III describes the application of Web 2.0 approaches to instruments in online laboratories. Section IV discusses our work in federating multiple online services for searching scholarly journals. This research addresses important gaps (federation and synchronization) in unrelated services. Section V describes our efforts to adapt the concepts of tagging and online bookmarking to the problem of helping researchers at minority serving institutions identify collaborators. Finally, Section VI summarizes the work and describes further research opportunities.

2. CHEMICAL INFORMATICS AND WEB 2.0

The field of chemical informatics, or cheminformatics, has its origins in the 1960's with the development of the first representations of 2D and 3D chemical structures and structure-activity models [9] to relate chemical features to biological activity. Since then the field has increased significantly in terms of scope and techniques. The field is highly multidisciplinary - many methods used in cheminformatics have first appeared in the statistical, mathematical, text searching, machine learning and artificial intelligence literature. In addition, in recent years, there has also been an increased intersection of biology and chemistry (chemical biology [10, 11], chemogenomics [12], etc.).

Commercial tools and private, proprietary data controlled by industry have historically dominated cheminformatics. However, the growing field of academic cheminformatics has brought more open source tools and libraries (such as Openeye, Daylight, CDK [13], and JOELib) to the community, and the National Institutes of Health's open research (PubMed) and open community data (PubChem) online services are spearheading a greater amount of open information. Thus, as we review in this section, the time is ripe for the application of service-oriented computing architectures and distributed infrastructure to the field.

To this end, we have developed the *Chemical Informatics Cyberinfrastructure Collaboratory (CICC)* to develop state of the art cheminformatics tools and techniques and provide access to these via cyberinfrastructure [14, 15]. In this context cyberinfrastructure includes facilities such as computational grids, databases, and Web Services. The goal is to make cheminformatics functionality available in such a manner that non-expert users can use the infrastructure as prepackaged tools but will also be able to combine different tools and methods to create applications that are personalized for their specific problem. In the following sections we discuss aspects of the infrastructure that allow us to achieve these goals.

One of our primary activities has been to build a service-oriented computing framework, implementing Web Services using the CDK and other libraries. We highlight below one of these services, which wraps the R statistical package, and discuss the natural extension of this service into a Web 2.0 approach.

R Statistical Services and Community Models: A common task in cheminfomatics is the development of predictive models. These may be simple linear regression models or more complex models such as support vector machines or random forests. The traditional approach has been to develop the model and report its statistics in a publication. Such a model of dissemination precludes any form of machine (or even human) interaction with the model. An alternative approach is to provide a web page front end to the software that can evaluate the model. Such an approach is certainly an improvement, but still faces the restriction that one must manually navigate to the web page to make use of the model for predictive purposes. Another aspect of such models is that they may require calculation of features that are not publicly available. Thus even if one did have access to the underlying model, one could not calculate the features required as input to the model.

Our approach to this problem is the development of an R based web service infrastructure, in which one can deposit a previously built model (currently a R binary file representation of the model) that is then made available as a web service. The infrastructure is sufficiently general that any model support by R can be deployed. Once deployed, one can access the model via SOAP based web services to obtain predictions and various statistics. However the current limitations include the fact that it is difficult to interrogate the model without actually loading the model in R, models are restricted to R and that the feature calculation problem is not addressed.

Future work aims to address these issues by way of developing an XML-based model specification document. Such a document will include information regarding the type of the model (linear regression, SVM, etc) as well as various model statistics. In addition the model must include provenance data such as author, data of development and so on. As we discuss in more detail in Section V, these descriptions are suitable for keyword tagging. An important component of such a description will be a specification of the input features. This is especially challenging due to the wide variety of features that one may calculate using different software.

A key feature of any solution to this problem will involve community consensus on how one specifies features. This may be done using a top-down ontology approach, but it is also a good test case for tagging and folksonomy-style approaches. We envisage folksonomy-based approach using keyword tagging, where features will be characterized by what they represent, the software (and version of the software) required to calculate them and so on. Such an approach has the advantage of being open-ended and usage-driven. Given such a model description, one need not load models into an R session (thus saving time) but more importantly, models can now be searched. This represents a significant advantage over the current state of the art. In addition to model specification, we also plan to investigate the issue of model exchange. As noted above, our infrastructure is based on R. But there are many other popular statistical platforms and models built in R cannot be run on them. We plan to use Predictive Modeling Markup Language (PMML) as a means of converting our R models to a platform-agnostic XML format. A number of platforms such as (SPSS, DB2 Miner etc.) support this format.

With a combination of model specification and model exchange, we believe that the R based computational infrastructure will provide a flexible and open approach to the deployment and searchabilty of predictive models in cheminformatics

Databases and Data Handling: Though various computational services are very useful a key component of cheminformatics development is easy access to data as well as methods to manipulate datasets. To this end we have developed a number of databases that are derivatives of PubChem. The PubChem database is a collection of nearly 10 million molecular structures along

with bioassay results for 549 assays. Along with developing our infrastructure we have built databases that provide add-on functionality to the data in PubChem. Examples include performing docking with the PubChem structures and generating 3D structures for 99% of PubChem. All our databases are relational databases based on the PostgreSQL platform. Though we have provided direct SQL access to these databases, we strove to provide a mode of access that is similar to our other services. Thus each database can be accessed via SOAP based Web services. In addition to providing access to databases we also provide services that allow one to manipulate datasets. One example is the VOTables services. VOTables are an XML specification for handling tabular data. Our services allow one to convert various forms of tabular data (plain text and Excel formats are supported) to a VOTables document and vice versa. Many of the statistical Web services can accept VOTables data directly.

The above discussion has focused on accessing the various services via SOAP calls using standard Web Service techniques, which constituted the majority of our initial work. However, we have subsequently examined the use of Web 2.0 techniques for building services. Given that the services are effectively remote function calls, we can consider other ways to expose their functionality. One example is the use of RSS feeds and REST (specifically HTTP GET) invocations. Traditionally, news sites have used RSS feeds to syndicate news items. More generally, RSS feeds can be used to syndicate any type of "new" item or state change in an existing item. In our case we have made available a number of database searches in the form of RSS feeds. For example the user can specify a query for the PubDock database, asking for docking results for molecules that have a score above certain specified value. The query is executed and the return value is an RSS feed (in RSS 2.0) format that can be viewed with any RSS reader. Furthermore our RSS feeds are able to embed chemistry within them by the use of Chemical Markup Language. Thus, if a given RSS item has an associated 2D or 3D structure, the structure can be embedded as a CML [16] fragment within the item. Such combinations are termed CML-RSS [17] feeds and can be viewed in a variety of environments such as Jmol or Bioclipse. Example of such CML-RSS feeds can be found at http://www.chembiogrid.org/cheminfo/pcrss/dockrss/form and http://www.chembiogrid.org/cheminfo/rssint.html.

Workflows and Mashups: Workflow tools and mash-ups offer alternate approaches to aggregating multiple sources of information and computation into new and ad-hoc applications. Workflow tools are generally restricted in the kinds of links that can be made between services (for example, one can usually follow a linear path through a set of services, the output of one serving as the input to another, but more complex constructs like looping and non-sequential interaction fit less well into the paradigm), although they usually are built with user-friendly interfaces that allow development by non-programmers. The Yahoo! Pipes service is an attempt to allow such workflow development on the Web, specifically for processing RSS and Atom feeds. Completed workflows can usually be shared with others. Mash-ups permit a much higher degree of flexibility in the intercommunication of services, but generally require scripting or programming using API's to achieve this (although there is no intrinsic reason why mash-ups could not be created in a graphical user interface, and indeed the latest version of blogging programs like BlogSpot permit the use of computational components in a page). In the pharmaceutical industry, one particular workflow tool called Pipeline Pilot (www.scitegic.com) has been widely used in cheminformatics. This tool uses a graphical interface for workflow development, and includes extensive pre-packaged computation and database functionality for cheminformatics and bioinformatics. In the non-commercial sector, Taverna (taverna.sourceforge.net) and Knime (www.knime.org) have been used. While mash-ups have been developed for cheminformatics, they have not revolutionized or commoditized the field in the same way that has happened with online mapping. The reason for this could be lack of

availability of simple, easy to use API's or the underlying complexity and proliferation of the data structures in the field. However, given the growing use of distributed computation in both cheminformatics and bioinformatics, mash-ups, as well as workflow tools, seem like an excellent way of integrating these two fields.

User Interfaces: RSS feeds are one way to access and view information provided and generated by our cyberinfrastructure. Another mode of access to our functionality is the use of Userscripts, which are examples of rich client interface techniques. Userscripts are small scripts that can run in browsers when particular URLs or groups of URLs are visited in the browser. What makes them unique is that they can modify the content of a Web page on the fly, by altering the page components in the script. They can therefore be used to make user-customized variants of Web pages. We have applied userscripts, particularly Greasemonkey scripts for Firefox (https://addons.mozilla.org/en-US/firefox/addon/748) that use our Web service infrastructure to enhance existing chemical and cheminformatics resources on the Web [18]. These scripts include, inter alia, the visualization of 3D representations of molecules when visiting PubChem (using our Pub3D database), and the automatic mark-up and hyperlinking of chemical structures in journal articles.

3. SCIENTIFIC INSTRUMENTS AND WEB 2.0

Online Instruments and Services: Instruments and sensors provide observational data needed to drive the process of discovery in science. Along with simulation, experimental observation forms the basis for forming and testing theories. Availability and accessibility of appropriate instrumentation for a given research program can be a rate limiting factor in discovery. Furthermore sensors and sensor networks are playing an ever-increasing role in longitudinal studies in ecological, earth and biological sciences. Remote access to instruments and sensors shared by a research community is a highly desirable or critical capability in many fields of research and is becoming requirement for many new large-scale instrument installations such as synchrotron light source beam lines. [19]

There are numerous solutions to the problem of remote access to instruments, ranging from extending an instrument's console to users via screen sharing (e.g. VNC [20] or NX, http://www.nomachine.com/) to specifications for embedded "smart sensors" such as the IEEE 1451 suite [21]. There are also domain-specific solutions such as the Antelope messaging system (http://www.brtt.com) used in seismology to network ground motion sensors and the Monterey Bay Aquarium Research Institute "plug and work" instrument, Puck (http://www.mbari.org/pw/puck.htm). Each of these solutions provides part of the solution for remote access but individually is only suitable for a narrow range of applications.

We are developing the Common Instrument Middleware Architecture (CIMA) [22], a broad, middleware-based approach that can be used to solve remote access and control problems in many areas and on many scales. Use of a common middleware specification enables instrument users to build, test and deploy software to implement their experiments as needed, and to continue to use existing software with minimal modification when instruments are redesigned or upgraded. [23] Such middleware also allows hardware developers to open up their products to take advantage of community driven application development efforts. CIMA provides application-level access to remote instruments and sensors through interface and protocol standards based on Web Services and a Service Oriented Architecture (SOA) approach.

We are now investigating approaches for the next generation of CIMA. As discussed in Section I, Web 2.0 is a counterpoint to Web Services that shows great promise for increasing the diversity and rate of production of new services as well as the accessibility and reuse of existing

services. The immediate challenges for CIMA middleware are to re-implement a Web Services product as a URL-based RESTful service and to develop gateways to existing CIMA instrument services. CIMA in its current form can potentially leverage WS-* layered products such as WS-Security, WS-Addressing, and WS-Attachment, so in migrating to a service based on a Web 2.0 approach, the functionality provided by these WS-* specifications need to be considered.

The advantages for adopting the Web 2.0 approach are clear with hundreds of new services being created and made available as Web 2.0 APIs (see for example www.progammableweb.com). As a matter of development cost and capability, the pool of skilled developers and integrators of these services is very large and growing rapidly, representing a potentially huge marketplace of developers worldwide, contrasting sharply with the availability of developers for more traditional Web Services. In e-Science applications, where specifications can be fluid, development time horizons are close, and funding is often aimed at science, not system development, using Web 2.0 approaches may provide a path to high quality, low cost software systems.

In sections below we will discuss the impact of Web 2.0 technologies and applications on the evolution of the CIMA architecture and the design of RESTful services for instruments, sensors and other real-time data sources.

CIMA Service Architecture: An instance of a CIMA instrument service has three main components: the service code consisting of instrument independent communications and management functionality and plug-ins for specific instruments, sensors and actuators; the Channel protocol and Parcel XML Schema that define operations on instruments available to clients; and the instrument description which allows clients to understand the functionality and data provided by the instrument.

The service component provides a Web Services (WS) endpoint for clients to communicate with the instrument using the Channel protocol. Service life-cycle management functions include loading and execution of hardware device drivers (plug-ins), registration of the service instance with directory services, and communications transport start-up and shutdown.

The second component, the Channel protocol, is used by clients to interact with remote CIMA-enabled instruments by sending and receiving chunks of XML based on the Parcel XML Schema. SOAP over HTTP is used as the primary transport for parcels, the encoding and style of the SOAP messages is document/literal rather than RPC/encoded to reduce dependence on Web Services and to make asynchronous communications possible between clients and the service. A complete client-server Channel consists of the server's WS endpoint for commands and data requests in Parcel XML chunks to the instrument, and a second WS endpoint at the client to receive replies asynchronously from the instrument. The parcel schema defines a tag that enumerates operations that clients can perform on the instrument and data types that can be returned by an instrument. Additional elements provide source and destination information for routing, and data plus metadata.

The third CIMA component is the instrument description, provided in RDF as OWL-DL instances. The description is based on a controlled vocabulary that provides information about instrument capabilities, access methods and input and output data formats [24].

CIMA is intended to be a component in a larger service oriented architecture in which it provides real-time instrument or sensor data to downstream components. SOAs tend to be "pull" oriented where clients make RPC-like requests to the service. Grid systems also tend to be stateful in which clients use operations to query or change the state of an instance of the target service or associated resource dedicated to their use. CIMA is stateful in that clients can have a relationship with an instrument service that can last many transactions, but it also supports a "push" model in which information can be provided asynchronously to clients as it becomes available. This push capability requires Web Services-aware clients, with an instance of a Web

Service built in to or associated with the client to provide a sink for the service to call back with new instrument data. A publish/subscribe approach is an alternative to this WS-based callback mechanism and we have implemented callbacks based on Java Messaging Service and Antelope.

As an alternative to WS callbacks and publish/subscribe approaches, a Web 2.0 approach could use Ajax on the user's Web browser to poll for new data continuously. If supported by the server and the client data could also be streamed directly to code running on the browser via server push. Since the CIMA protocol is entirely contained in the Parcel XML Schema a parcel message could be delivered by GET or POST HTTP transaction without the additional overhead of transporting, parsing and handling the SOAP envelope. Message delivery latency would now also depend on the frequency of polling by the client in addition to factors present in the server push scenario.

REST Services for Instruments: Our objective is to re-implement the SOA version of CIMA services so that it can coexist and work well with Web 2.0 applications and services. We expect that a "resource oriented architecture" (ROA) approach using RESTful interfaces will provide a suitable mapping from our current SOA architecture to a form that can leverage existing REST-based services and Web 2.0 technologies [25]. The current callback scheme can be preserved if the client provides a Web server to which callbacks can be made but in general the client's role must change from message sink to polling the data source. The server must also add buffering for each client to compensate for polling latency and additional semantics for requesting buffered data must be added to the polling sequence but these are relatively minor changes and suitable to an SOA version of the polled service as well

There are other considerations in moving from SOA to ROA for real-time data services beyond the basic shift from event-based callbacks to polling. In particular, there is a stark contrast between the processes used to create WS-* specifications and Web 2.0 APIs: WS-* specifications are usually aimed at solving a more general enterprise-level process problem and are carefully vetted through a standards body such as OASIS (http:// www.oasis-open.org/) and receive much consideration before the first evaluation implementations are available. Web 2.0 APIs on the other hand tend to be focused on providing specific functionality rapidly and in a somewhat more ad hoc manner. The result then is that classic SOAs are stable, evolve very slowly, tend to require considerable developer knowledge, and are often difficult to develop without supporting development tools.

In theory Web Services specifications can also be layered and functionality added incrementally to an application by adding the appropriate handler and client code for a given Web Services function to the client's code and server's handler stack. Development and testing of Web Services generally requires both client and server test harnesses and a complex development and test environment. The hallmark of Web 2.0 is rapid development and implementation of APIs that conform to a simple (i.e. REST) interaction method. This approach allows developers to prototype and refine a service quickly and the results to be tested and evaluated using only a Web browser or program implementing HTTP GET calls.

The economic tradeoffs are stark. It seems clear that developing REST services that can be integrated with other Web 2.0 offerings is a winning strategy where timeliness or development cost are at issue, and Enterprise Web Services are appropriate where use of WS-* is required or the client or service under development must interact with other Web Services.

Portals for CIMA Instruments: Although clients that acquire data to storage media do most of the work when it comes to performing an experiment with a CIMA-enabled instrument, on-line instruments and sensors are more interesting and useful when provided with user interfaces. We also provide access to instruments and data through a GridSphere portal which hosts JSR 168 portlets and provide portlets with a user log-in context for identity management and access control [26, 27]. Individual user interface functions in the portal are implemented as portlets

accessing back-end CIMA services with Web Services interfaces. Although GridSphere is very functional and suits the current requirements for the CIMA project, the community of GridSphere and JSR 168 users and developers is limited and further development of both environments is in question. A further issue is that development for GridSphere is nontrivial and combining this with the need for portlets to bind data from Web Services and to provide Web Service sinks, the complexity of the result is high and maintainability is consequently a problem.

If we look to Web 2.0 technologies though, the situation seems brighter. A CIMA service that operates in a RESTful manner can be composed with other Web 2.0 offerings such as Google desktop, NetVibes or Yahoo widgets to rapidly assemble e-Science portals using a mix of community developed and commercial offerings. As an experiment we were able to reproduce some of the functionality of our GridSphere portal in iGoogle using off-the-shelf widgets in a matter of a few minutes using RESTful versions of CIMA services.

Research Opportunities for Real Time Instruments: There are further opportunities for accessing real-time data from instruments and sensors using Web 2.0 technologies. In a recent experiment we used the news distribution protocol Atom (http://tools.ietf.org/html/rfc4287) as a method for sending CIMA parcel "events" to interested users through common applications for reading Atom feeds. These systems provide a well supported, almost zero development approach for real-time condition monitoring and alerting. This approach further enables us to integrate Atom-based services into mashup tools. As described in Section II, highly functional signal processing applications that consume and process the feed data streams from CIMA-enabled sensors can be developed using Yahoo Pipes (http://pipes.yahoo.com/pipes/) and shared broadly both as "code" and as functioning service instances. Yahoo Pipes offers a compelling model for future community-based distributed programming efforts and particularly for creating value-added real-time data products from instruments and sensors embedded in our environment.

Emerging Web 2.0 technologies and APIs offer a flexible rapidly evolving environment for developing real-time applications for sensors and instruments. Moving from a Enterprise Web Service approach to REST for instrument services presents some challenges with respect to client design but doing so makes a broad range of Web 2.0 components and environments available for rapid and cost-effective development of software systems for instrument driven e-Science.

4. FEDERATING ONLINE DIGITAL ENTITIES

Many Web 2.0 services such as social bookmarking sites and scholarly journal search services provide overlapping capabilities. Unfortunately there has been little work (and little incentive from the service providers) on integrating or federating these services. We describe in this section a federating architecture for such services with a specific application to managing online digital entities. In Web 2.0 terminology, this may be viewed as a server-side mash-up.

Managing State in Distributed Digital Entities: We have been working on providing consistent service based architecture based on the event-based model for reconciling the dynamic and possibly inconsistent multiple sources of metadata information obtainable from Web 2.0 services for resource annotation and tagging. As part of our research, we have designed, developed and implemented the following modules to our prototype project, the Semantic Research Grid (SRG). The system architecture is shown in Figure1.

Session and Event Management: This module provides a mechanism for storing data about the current user. These user specific data include user authentication credentials, any modifications to a Digital Entity (called minor events), and the selected "view options", which control the level of detail with respect to the metadata fields displayed for each Digital Entity. Once the user has logged in the SRG system, the user's authentication credentials, all minor events for each Digital

Entity, and view options for metadata fields of a Digital Entity are all maintained in the user session. When a user logs out from the SRG system, all unused minor events (modifications to a Digital Entity) for a dataset creation are removed from the current user's session. To solve the consistency problem in the SRG system, we have designed a novel event-based consistency model based on the concepts of *event* and *dataset* [28]. In our model, we adopt the view of an event as a time-stamped action on a digital entity, which only maintains the modifications to an object. We distinguish between *minor* and *major* events: insertion of a new digital entity into the system or deletion of an existing digital entity from the system is considered a major event; modifications to existing digital entities are considered minor events. Each Digital Entity has a reproducible state that is the sequence of the events that have been applied to it.

Digital Entity Management and Update Model: This module integrates PubsOnline software (an open source tool for management and presentation of databases of citations via the Web) into the SRG system and provides an interface for searching the local/remote databases of SRG. It also provides a user with an interface: (1) to manually insert a Digital Entity into one of the local/remote SRG databases; (2) to access to the history of a Digital Entity, from its entry into SRG system to present, and ability to rollback to a previous state; (3) to view detailed information about a Digital Entity; (4) to update any metadata fields of a Digital Entity, which is saved into session as a minor event for this Digital Entity.

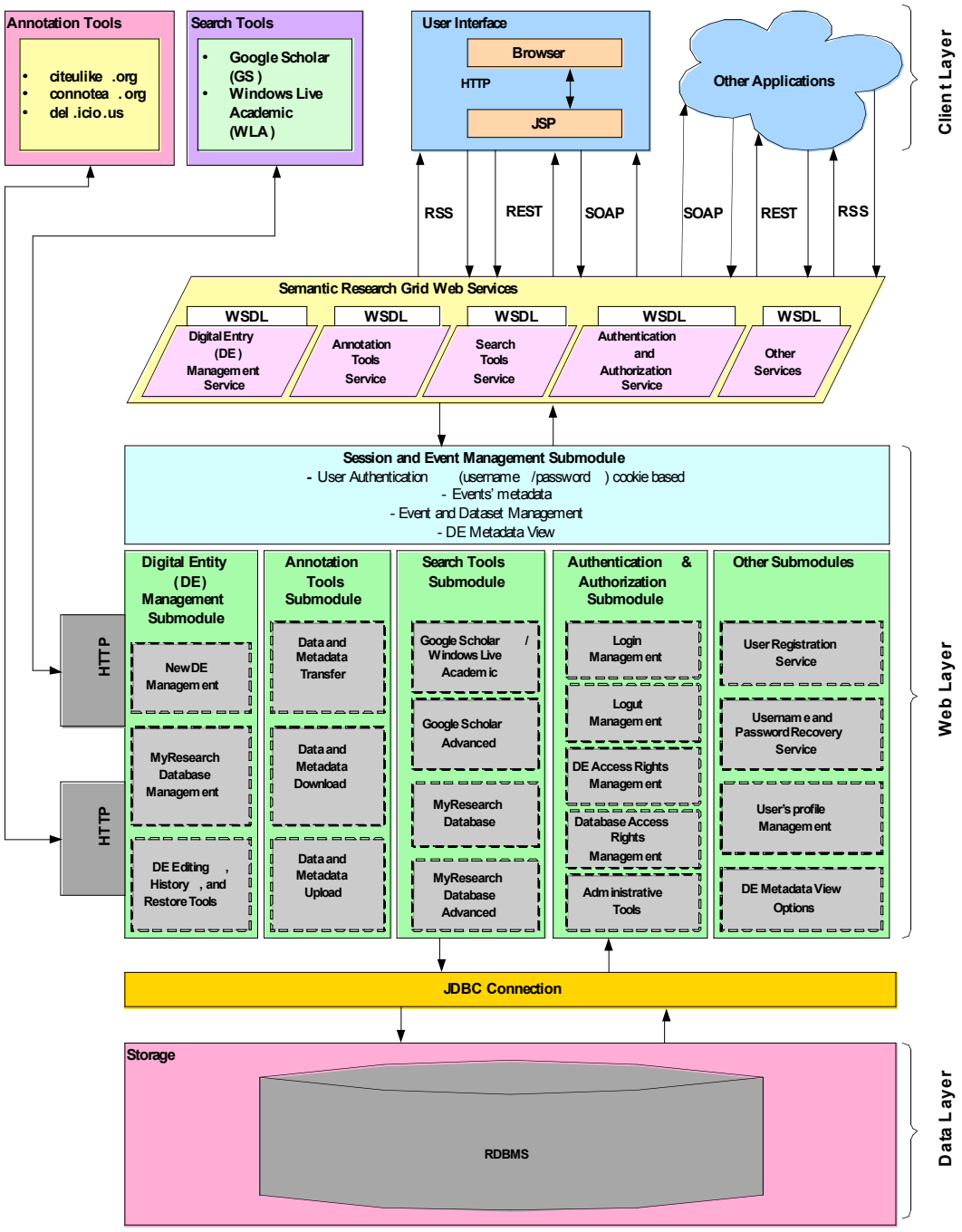

Figure Chapter XX-1. Semanitc Research Grid architecture.

In our system, we need to have a well-defined update model, which is built on top of an event-based model, to provide the user with flexible choices to manage their Digital Entities. Our update model uses events for updating digital entries and it is based on the following concepts: a) keep the existing version; b) replace the existing version with the new one; and c) merge the existing and the new version.

In our update model, we can provide the user with an ability to select an option from the above update model to be applied for all matching digital entries or an each individual digital entry. In this manner, updates can be applied to an each individual or all digital entries as a

default based on the selected choice. We have also designed and developed "Periodic Search for Updates" module that allows a user to search and retrieve updates for Digital Entities in the system.

Annotation Tools: This module implements an interface that allows a user to manage the social bookmarking tools such as Delicious, CiteULike, and Connotea. Through this module a user can: (1) upload Digital Entity data and metadata to one of these social bookmarking Web sites; (2) download Digital Entity data and metadata from one of the social bookmarking Websites into one of the SRG research databases; (3) transfer Digital Entity data and metadata between these social bookmarking Websites.

Search: This module provides an integration layer that allows searching over multiple databases. This is an extensible module that includes implementations for searching Google Scholar, Microsoft Windows Live Search, and multiple local databases.

Consistency Model: We have been also working on the consistency mechanism of the system. We have two mechanisms to provide consistency: push and pull. We push updates to annotation Websites from our system once they occur by using the remote sites' Web API. We also periodically check annotation Web sites for updates, and pull them into our system if there is any. Our consistency model is a data centric consistency model. It implements strict consistency: once updates occur in the system, they are propagated right away. Our model is based on the primary copy consistency protocol. Since updates are treated as events, a user can always roll back to a previous state at any time by undoing events.

User Management: The SRG system has the following modules for managing user interactions: (1) user registration module; (2) username and password recovery module; (3) user profile management module, where a user can edit personal information, modify system password, and request subscription to available SRG system groups; (4) Digital Entity metadata "view options" mechanism, which allows a user to define the metadata fields of a Digital Entity to be displayed or hidden.

Figure 1 shows the overall architecture of the SRG system. This system consists of three main layers: (a) the *client* layer; (b) the *Web* layer; and (c) the *data* layer. The client layer is implemented using Java Server Pages (JSP) but can be built with any Web Service client. The client layer communicates with the Web layer over the HTTP protocol through SOAP messages encapsulating WSDL-formatted objects. The Web layer consists of several Web services that handle communication with the existing online tools. As with other projects described in this paper, we are investigating alternative protocols and message formats (such as REST APIs and RSS/Atom feeds) for managing interactions with the Web layer. Our goal is provide all major Web 2.0 access modes (see Section I) in the Web layer. The Web layer communicates with the data layer through JDBC connections. Finally, the data layer is composed of several local and remote databases.

Authentication and Authorization Issues: In collaboration systems such as SRG, users may access group-editable metadata. We must provide mechanisms for establishing privileges to view and possibly edit portions of Digital Entities within the SRG system.

Access Rights: We have adopted an access-control matrix model that supports multiple groups and multiple users for each object. The model should support any changes for group of people. There are set of objects (Digital Entity or database, access rights) pairs for users. Digital Entities may be created by the users of the SRG system in several ways: (a) using annotation tools (Delicious, CiteULike, and Connotea); (b) using search tools; (c) manually, through the "Insert New Citation" Web interface. The access policy has two basic modes. *Public:* Digital Entity creation must be associated with at least one group, including possibly universal membership groups. *Private:* the Digital Entity can only be assessed by its owner. The owner can modify the

access mode. For both public and private operation, read and write (i.e. modify or edit) permission are given to user or group for the inserted Digital Entity.

The SRG system distinguishes between three kinds of users: *Owner* is initiator of the Digital Entity and database creation; *Group* is any available groups in SRG system; and finally *Other* users are all remaining users. The owner of Digital Entity and database can specify the Digital Entity and database permissions for all three kinds of users mentioned above.

We have used a permission model similar to UNIX file system. In SRG system, Digital Entity corresponds to the file element, and its home database corresponds to the directory element. For each Digital Entity and database, there are three types of access rights defined in the systems. Access rights, summarized in Table 1, are used in the implementation of the authorization module. Authenticated users can create databases dynamically. A user needs to have write permission for the database in order to put Digital Entities into the database.

Table 1. Access rights summary

Access Right	**Digital Entity (DE)**	**Database**
Read	Read DE	Access to database for viewing Digital Entities
Write	Modify DE	Insert Digital Entities into database
Delete	Delete DE	Remove entity from system

The access matrix describes data protection and permissions that each subject holds for each object . As an example in Table 2, users and groups are shown in the rows and Digital Entities in the columns. In this example, Bob is the owner of DE_1, DE_2, and DE_3. He has read(R), write (W) and delete (D) accesses for these Digital Entities. $Group_1$ has write access for DE_1, and read access for DE_3. Alice is another user and has only read access to DE_2. The system allows having only one owner of a Digital Entity and database. However, there might be more than one group for Digital Entity and database. Bob can modify DE_1, DE_2, and DE_3 user access rights or give them permissions to groups and other users.

Table 2. Access control matrix for digital entities

Name	**DE_1**	**DE_2**	**DE_3**	**User**
Bob	R,W,D	R,W,D	R,W,D	Owner
$Group_1$	W		R	Group
$Group_2$	R,W	R,W	R	Group
Alice		R		Other
Henry	R,W			Other

The owner of the Digital Entities may give access rights to other users. We have defined two methods for this operation. First, the owner may give permissions for all Digital Entities stored in the database. A user can give permissions to any user or groups that can be accessed by all users of that group for selected database. However, users and groups need to have required permissions for owner's selected database. Second, owners can modify their Digital Entity permissions for users and groups.

Group Administration: Having permissions for Digital Entities and databases for users and groups does not provide a complete authorization scheme. We need to have levels of control of the users and groups to complete the authorization module. We have defined the following levels of administration control: Super Administrator and Group Administrator. The system allows having more than one Super Administrator. An existing Super Administrator can add other Super

Administrators to the system. A Super Administrator can assign any user to become a Group Administrator, and remove a Group Administrator from group. Each group has at least one Group Administrator. If a user creates a new group, that user automatically becomes Group Administrator. Users are allowed to belong to more than one group. User can make a request to member of any group in the system.

5. RESEARCHER TAGGING AND MATCHMAKING

As we discussed in the introduction and expanded in Section IV, online bookmarking services provide a very simple way for users to share interesting URLs with each other. This sharing may be amongst friends and collaborators in networks, or it may be anonymous and democratic: by selecting a particular site, one "votes" for it. Bookmarked content is commonly described with keyword tags as well as free-form comments provided by the user. Bookmarking services often provide tag recommendations based on previous tagging to encourage the user to avoid synonyms and alternative spellings.

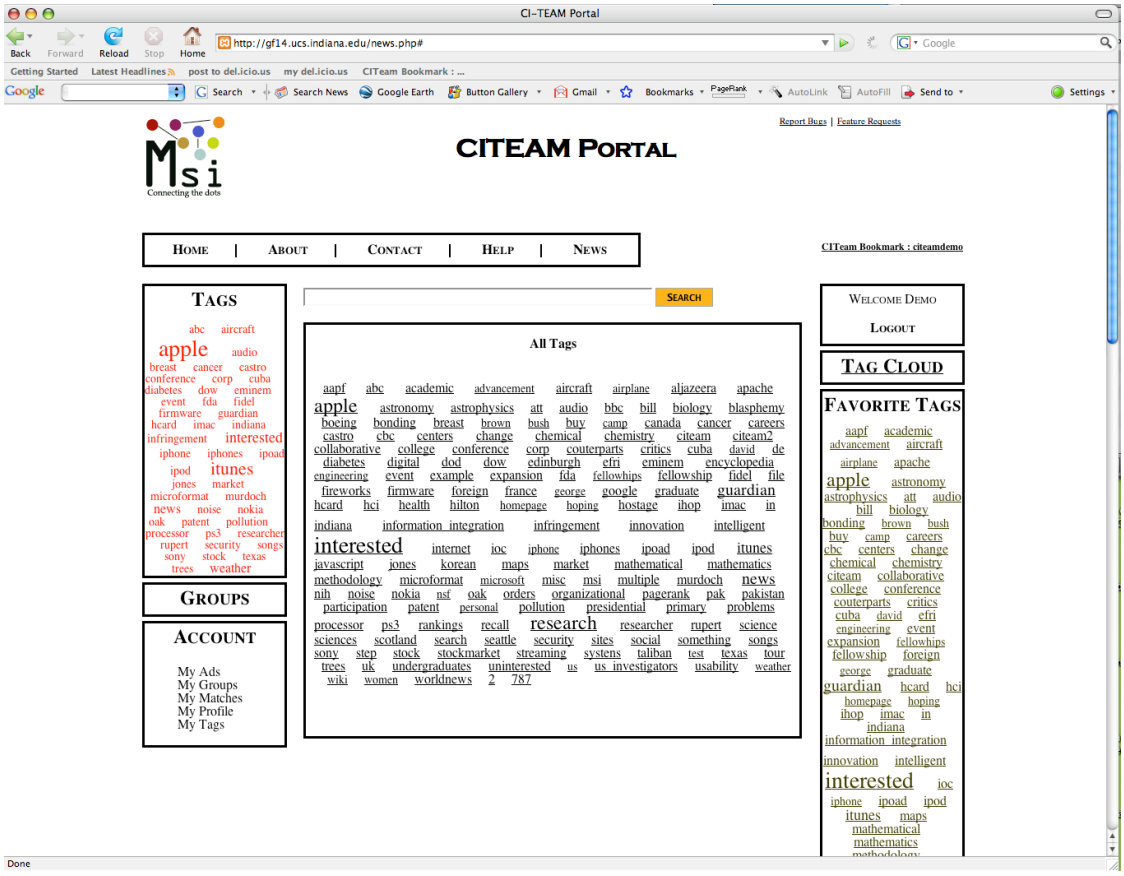

Figure Chapter XX-2. The MSI-CIEC Portal is designed to help users manage online bookmarks and identify researchers with similar tagging profiles.

Although simple, tagging systems provide a very fundamental capability: they enable community driven metadata descriptions of URL-identified Web resources. One may envision the extension of this to URIs generally. Seen from this point of view, we may readily recognize

that tags and their associated URLs can be represented as graphs and are a variation on the graph-based Semantic Web relationships more commonly represented with RDF and OWL. We may treat either all tags as nodes and URLs as arcs, or vice versa. This allows us (with sufficient amounts of data) to apply interesting computational graph techniques such as cluster identification and ranking.

To seed these research issues as well as provide a useful service, we are developing the MSI-CIEC Portal (Figure 2), a bookmarking and social networking service to help faculty at minority serving institutions find interesting research opportunities and collaborators in their fields.

The MSI-CIEC Portal's basic function is to provide a user interface to bookmarks. We use Connotea as a backend service for storing bookmarks: users have accounts created simultaneously in the MSI-CIEC Portal and in Connotea when they register. While Web browsing (particularly journal browsing as described in Section II), users may choose to bookmark and tag a particular journal article or NSF calls for proposals (available via RSS feed). The portal provides a browser toolbar button (installable by dragging the installation link from the portal interface to the browser's toolbar area) to allow sites to be tagged in a process that is not disruptive to browsing.

Figure 2 illustrates various "tag cloud" views. Other portal views include displays of selected RSS news feeds, such as NSF funding opportunities. To further simplify the tagging process for these particular feeds, we are developing a "click tagging" process: a user can indicate interest, disinterest, or ambivalence towards a particular proposal call by clicking an icon. This allows users to quickly create a set of tags interesting to them. Other users, examining the same link, can see other users interested in the same proposals.

To further populate the system, we are incorporating public data obtained from NSF and TeraGrid user databases. We represent data such as project award number, principal investigator name, NSF division, and directorate to tags. In addition, principal investigators' public information is used to generate profile pages within the system. This allows the user to treat other users as tags for navigating through the system (to see for example someone's list of collaborators or funded projects in various NSF directorates). Alternatively, the user may want to view the profile entries of others in the system.

The MSI-CIEC portal may be used for bookmarking, managing proposal call links from the NSF RSS feed, or navigating through databases. However, its full purpose (when sufficiently populated with data) is to investigate social networking strategies. These strategies are intended to help researchers find other researchers with similar or complementary interests. We are basing this functionality on tag profiles that users build up through normal use of the portal as well as seeded information from NSF databases. As we have observed, tag profiles and their associated resources may be treated as graphs. This allows us to examine techniques of varying sophistication for determining similarity between users. These may range from simple link counting to clustering, machine learning, and Page Rank-style algorithms.

6. CONCLUSIONS AND RESEARCH CHALLENGES

We have reviewed the application of Web 2.0 technologies to e-Science. Sections II and III reviewed the evolution of two Web Service-based cyberinfrastructure projects (chemical informatics and instruments) into Web 2.0-style distributed systems. Common themes include the use of REST-style services communicating with message formats adapted from news feed standards. We also discussed the use of Web 2.0 for less traditional cyberinfrastructure: the federation of multiple bookmarking and scholarly search services (Section IV) and the use of tagging to build up scientific communities (Section V). By adopting Web 2.0 approaches, we

hope that these systems may leverage the considerable investment by major software and service providers. Mashup building tools (which work primarily with RSS and JSON) are an important example.

Challenges facing Web 2.0 for e-Science include the lack of standards to enable portability of user identity and metadata between services. Typical Web 2.0 services such as Flickr, Facebook, and del.icio.us provide portable capabilities (such as JavaScript badges and RSS feeds) but they do not provide mechanisms for sharing or federating identity, preferences, and other metadata. We have made initial efforts to synchronize digital entity representations in multiple, unrelated services, but this needs further investigation.

Web 2.0 parallels Grid and Web Service systems in most capabilities (see for example [6]). Major exceptions include security and notification. OpenID is one important Web 2.0 security standard for creating distributed identity for enabling single sign-on systems, but the problem of distributed access controls (for example) are open issues. Notification and messaging systems are another fundamental problem, since they do not match well with the request/response messaging style used by REST services. One can simulate event pushing with client polling, at the cost of either increased network traffic or increased message latency. These issues are particularly important to instrument and real time grids (Section III).

7. ACKNOWLEDGEMENTS

The Chemical Informatics and Cyberinfrastructure Collaboratory is funded by the National Institutes of Health. CIMA is supported by the National Science Foundation's Middleware Initiative program. The Semantic Research Grid is supported by Microsoft Research. The MSI-CIEC portal is supported by the National Science Foundation's CITEAM project, Award Number SCI-0537498.

8. REFERENCES

[1] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz (eds), "A Reference Model for Service Oriented Architecture." Reference Model for Service Oriented Architecture 1.0 (2006).

[2] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web Service Architecture", W3C Working Group Note (2004).

[3] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." International J. Supercomputer Applications, 15(3), 2001.

[4] I. Foster et al, "The Open Grid Services Architecture, Version 1.5". Open Grid Forum GFD-1.080.

[5] Geoffrey Charles Fox, Dennis Gannon: Special Issue: Workflow in Grid Systems. Concurrency and Computation: Practice and Experience 18(10): 1009-1019 (2006).

[6] Marlon E. Pierce, Geoffrey Fox, Huapeng Yuan, and Yu Deng Cyberinfrastructure and Web 2.0 Proceedings of HPC2006. Available from http://grids.ucs.indiana.edu/ptliupages/publications/CIWeb20Chapter.doc.

[7] Roy T. Fielding, Richard N. Taylor: Principled design of the modern Web architecture. ACM Trans. Internet Techn. 2(2): 115-150 (2002).

[8] Nancy Wilkins-Diehr, "Special Issue: Science Gateways - Common Community Interfaces to Grid Resources." Concurrency and Computation: Practice and Experience, Volume 19, Issue 6, Date: 25 April 2007, Pages: 743-749.

[9] Hansch, C., Maloney, P.P., Fujita, T., Muir, R.M., Correlation of Biological Activity of Phenoxyacetic Acids with Hammett Substituent Constants and Partition Coefficients, *Nature*, 194, 178-180.

[10] Nicolaou, K.C., Joys of Molecules. 2. Endeavours in Chemical Biology and Medicinal Chemistry, *Journal of Medicinal Chemistry*, 2005, 48, 5613-5638.

[11] Schreiber, S.L., Kapoor, T.M., Gunther, W., Eds., *Chemical Biology: From Small Molecules to Systems Biology and Drug Design*, Wiley-VCH, 2007, Wienheim, Germany.

[12] Harris, C.J., Stevens, A.P., Chemogenomics: Structuring Drug Discovery to Gene Families, *Drug Discovery Today*, 2006, 11, 880-888.

[13] Steinbeck, C., Hoppe, C., Kuhn, S., Floris, M., Guha, R., Willighagen, E.L., Recent Developments of the Chemistry Development Kit (CDK) - An Open-Source Java Library for Chemo- and Bioinformatics, *Current Pharmaceutical Design*, 2006, 12, 2110-2120.

[14] Dong, X., Gilbert, K.E., Guha, R., Heiland, R., Kim, J., Pierce, M.E. Pierce, Fox, G.C. and Wild, D.J. Web service infrastructure for chemoinformatics, *Journal of Chemical Information and Modeling*, 2007; 47(4) pp 1303-1307

[15] Guha, R., Wiggins, G.D., Wild, D.J., Baik, M.H., Pierce, M.E., and Fox, G.C. Improving usability and accessibility of cheminformatics tools for chemists through cyberinfrastructure and education, *Cheminformatics*, Accepted September 2007.

[16] Murray-Rust, P., Rzepa, H., Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles, *Journal of Chemical Information and Computer Science,* 1999, 39, 928-942.

[17] Murray-Rust, P., Rzepa, H., Williamson, M.J., Willighagen, E.L., Chemical Markup, XML, and the World Wide Web. 5. Applications of Chemical Metadata in RSS Aggregators, *Journal of Chemical Information and Computer Science,* 2004, 44, 462-469.

[18] Willighagen, E., O'Boyle, N.M., Gopalakrishnan H., Jiao, D., Guha, R., Steinbeck, C. and Wild, D.J. Improving Dissemination of Scientific Literature and Data with Userscripts, *BMC Bioinformatics*, submitted September 2007.

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

[20] Richardson, T., Stafford-Fraser, Q., Wood K.R., and Hopper, A., Virtual Network Computing, *IEEE Internet Computing* 2(1), January/February 1998.

[21] Lee, K.B., Schneeman, R.D., Distributed measurement and control based on the IEEE 1451 smart transducer interface standards. *IEEE Transactions on Instrumentation and Measurement* 49(3), June 2000. p.621-627.

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

[23] McMullen, D.F., Atkinson, I.M., Chiu, K., Turner, P., Huffman, K., Quilici, R., Wyatt, M., Toward Standards for Integration of Instruments into Grid Computing Environments. In *Proceedings of IEEE International Conference on e-Science and Grid Computing* (e-Science 2006). December 2006. Amsterdam, The Netherlands.

[24] McMullen, D. and Reichherzer. T. Identity and Functionality in the Common Instrument Middleware Architecture. *Applied Ontology* 3 (2006), IOS Press.

[25] Richardson, L., and Ruby, S., RESTful Web Services. O'Reilly Media, Inc. May 8, 2007.

[26] McMullen, D.F. and Huffman, K., Connecting users to instruments and sensors: portals as multi-user GUIs for instrument and sensor facilities. *Concurrency Computat.: Pract. Exper.* 2006; 18:1-11.

[27] Yin, H., McMullen, D.F. , Nacar, M.A., Pierce, M., Huffman, K., Fox, G., Ma, Y., Providing Portlet-Based Client Access to CIMA Enabled Crystallographic Instruments, Sensors, and Data. *Proceedings of The 7th IEEE/ACM International Conference on Grid Computing* (Grid2006), Barcelona, September 28th-29th 2006.

[28] Ahmet Fatih Mustacoglu, Ahmet E. Topcu, Aurel Cami, and Geoffrey Fox, "A Novel Event-Based Consistency Model for Supporting Collaborative Cyberinfrastructure Based Scientific Research," in Collaborative Technologies and Systems CTS 2007 Orlando, 2007.