

Instruments and Sensors as Web Services

Kenneth Chiu,¹ Randall Bramley,¹ John C. Huffman,²
Kia Huffman,² and Donald F. McMullen³

{chiuk,bramley}@cs.indiana.edu,
{huffman,kihuffman,mcmullen}@indiana.edu

¹Computer Science Department, Indiana University

²Indiana University School of Informatics

³Pervasive Technology Laboratories at Indiana University

1 Introduction

Grid computing [37] is proving to be a useful paradigm for organizing and harnessing distributed resources. By provisioning the fruits of fundamental computer science research as services for such as scheduling and authentication, it is bringing distributed computing into the everyday lives of working scientists. As the convergence between Web services and Grid computing continues in standards such as the WS-Resource Framework, we expect to see this trend continue, aided by the complementary nature of Web services and Grid computing.

With this goal, current Grid computing research [36, 28] understandably focuses primarily on the marshalling of computation and data, and their integration at loci of analysis and synthesis. This focus has spawned the notion of computation and data Grids, respectively. Less well investigated, however, are the *sources* themselves of data, such as scientific instruments and sensors. These are still largely off-line to downstream Grid components, and are poorly integrated as Grid entities. This is acceptable if data is viewed as a static resource after whose archival only does scientific activity begin.

In reality, however, the collection of data is as much a part of the scientific process as its analysis. Data collection is not a rote procedure, and often interacts profoundly with interpretation and analysis, whether by human or machine. Ignoring this interaction can lead to inefficient use of computational and human resources, and limits the development of new cyberinfrastructure techniques such as dynamic data-driven application simulations (DDDAS) [23], autonomic computing, and software agents. The disadvantages of keeping instruments off the Grid are further exacerbated by three trends in sci-

entific research: (1) increasing investments in geographically extended, international collaborations organized around large shared instrument resources, (2) increasing real-time use of instruments by remote researchers both for first-look activities and pipelined data handling, and (3) increasing deployments of large-scale sensor networks.

We thus see a need for bringing instruments¹ on the Grid as first-class members, and the Instrument Middleware Project seeks to facilitate this task by researching and developing a set of standards and software components. Together these will form the Common Instrument Middleware Architecture (CIMA), which can be then used to Grid-enable a variety of instruments and sensors, ranging from large shared resources to tiny wireless controllers such as the Berkeley Mote sensor package [58, 44, 35], as well as embedded PC-104- and VME-based controller systems.

By promulgating a common set of concepts and interfaces, we hope to increase interoperability between instruments and software. This interoperability will extend along a number of different axes. For example, data analysis software can be insulated from different versions of functionally similar instruments, thereby increasing the flexibility and durability of instrument software. Common interfaces will also promote interdisciplinary collaboration by facilitating compatibility between applications and instruments developed by different communities. A common instrument middleware will also extend the accessibility of instruments to new classes of users, such as high schools and minority-serving institutions, by reducing the software development and deployment effort.

CIMA will be based on the emerging Open Grid Services Architecture (OGSA) [26] being developed by the Global Grid Forum (GGF) [27]. OGSA includes the Open Grid Services Infrastructure (OGSI) [55], which is used to define a common interface to all OGSA Grid services, and the OGSA Grid Data Service Specification [2], which defines a interface to access data. OGSI uses the Web Service Definition Language (WSDL) [20] to specify interfaces to services.

Along with the Grid, another recent development in scientific computing is the use of software components [57, 10] to enhance code modularity, encapsulation, and reuse by focusing on deployment and packaging. The Department of Energy is developing the Common Component Architecture (CCA) [18] (of which two of the project members are active participants) to produce a software component specification for scientific computing. CIMA will include CCA components where applicable.

The CIMA implementation will be evaluated in three settings representing a spectrum of shared instrument applications: X-ray crystallography at a synchrotron source, real-time acquisition of network performance data with embedded monitors, and small sensor such as the Berkeley Mote wireless sensors. The end product will be a consistent and reusable framework for including shared instrument resources in geographically distributed Grids.

In Section 2 of this whitepaper, we discuss some characteristics of instruments and sensors. In Section 3 we summarize some recent developments in information technology. In Section 4 we suggest how these developments may impact the practice of science.

1. We will henceforth refer to scientific instruments and sensors collectively as instruments.

Section 5 presents our proposed middleware architecture. Some related work is discussed in Section 6. Section 7 presents some case studies in specific application domains.

2 Scientific Instruments and Sensors

Scientific instruments and sensors are crucial to scientific advancement. They provide the raw observations used to develop, verify, and falsify theories. Though instruments vary widely in their architecture and construction, typically a physical detector of some kind originates the data. The data then undergoes a variety of transformations, including physical, numerical, and lexical. Eventually it reaches some kind of general purpose computer. Along the way, the data may pass through a variety of processing units which we refer to as *nodes*. These nodes are situated between the actual media forming the communication links. Nodes may be highly-specialized processors, embedded computers, or general-purpose computers.

The following discussion should be considered vis-à-vis at least four exemplars: a unique large detector such as the ATLAS [7] instrument on CERN's Large Hadron Collider, a set of detectors such as the beamline access points at a national synchrotron source such as diffractometers at Argonne's Advanced Photon Source [1], a few score telescopes or electron microscopes accessed and controlled remotely through computer mediation, and a large number of embedded sensors used to gather environmental data from a wide area.

2.1 Characteristics

Instruments form a diverse category of machines, with widely varying characteristics. As a basis for discussion, we identify the characteristics we believe will significantly impact the design of instrument systems. We divide the characteristics into two groups, intrinsic and extrinsic. Intrinsic characteristics stem from the needs of the science, while extrinsic characteristics are essentially the capabilities of the hardware. Taken together, these two sets constrain the possible designs of the software architecture.

2.1.1 Intrinsic Characteristics

The intrinsic characteristics of an instrument are determined by the scientific requirements. Intrinsic characteristics cannot be changed by improved hardware.

Number of sources. Some instrument consist of a single data source. Others, such as a large seismic array, consist of myriad sources connected together. The number of sources impacts the design of the information system supporting the instrument. A large number taxes not only the scalability of the data transport, but also the management and administration interfaces. For example, manually adding metadata to specify provenance may be acceptable for single-source instruments, but not for one consisting of hundreds of individual sensors. Furthermore, systems comprised of many individual sensors must provide practical means for tasks such as identifying broken sensors and upgrading firmware.

Data rate. Some instruments monitor events that occur at a very high frequency, or perhaps generate images or even video. Such instruments output data at a very high rate. Other instruments provide data at relatively low rates, such as once a minute, hour, or even day.

Timeliness. Some instruments output data that can be analyzed later. Others require real-time processing of data. This might be because researchers are using the instrument interactively; because it is being used to trigger emergency responses; or because the analysis is being used to drive the collection process itself in a control loop. Instruments requiring real-time processing can impose stringent requirements on their information systems.

Value. In some instruments, each datum may be extremely valuable, such as those from interplanetary spacecraft. In others, like a massive sensor network, the loss of some readings is not catastrophic. Systems for handling valuable data may need to incorporate fault-tolerance or redundancy into their design.

2.1.2 Extrinsic Characteristics

Besides intrinsic characteristics, other factors can affect the design of information systems for instruments. These factors are determined by the hardware design, and are termed extrinsic characteristics. Extrinsic characteristics may change as the hardware changes.

Processing power. Instrument systems must perform a variety of computational tasks, ranging from participating in network protocols to numerical analysis. These tasks may be divided among a variety of nodes. Some nodes may consist of little more than an analog-to-digital converter. Others may be supercomputers consisting of thousands of processors.

Memory. Each node may have a varying amounts of memory. High-value, high-rate data may require large amounts of memory to ensure reliable delivery, especially in unreliable communication environments. Nodes with appropriate capacity would be able to buffer data to compensate for communication outages, or for logistical reasons.

Bandwidth. Some instruments have very little bandwidth available in situ, while others have ample. This bandwidth interacts with the intrinsic data rate required by the instrument to affect protocol design. In situations where the required data rate exceeds the available bandwidth, techniques such as compression or decimation may be appropriate, which will in turn require more processing power.

Electrical power. Some instruments do not have access to the power grid, perhaps due to a remote or inaccessible location. These instruments may need to incorporate power-saving measures, perhaps controlled remotely by data analysis algorithms. For wireless sensors, power can also affect transmission range, which would in turn affect the IT infrastructure.

Uniqueness of design. To complicate the picture even further, many specialized instrument resources are of unique construction either by virtue of intrinsic design, or variability in product characteristics between product generations or among vendors if composed of off-the-shelf components. Some instruments are in effect never finished.

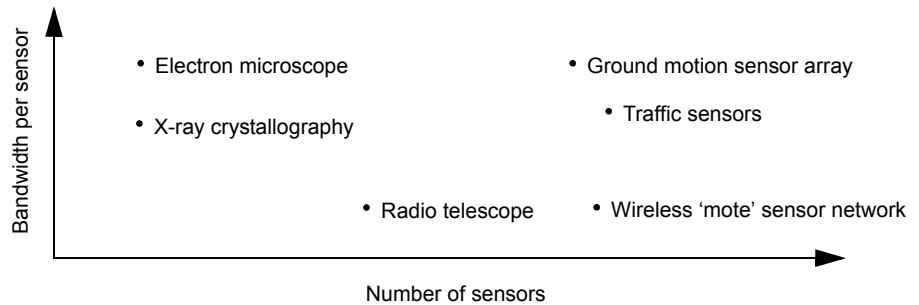
Commonality between instruments simplifies system design, but may be difficult to achieve.

2.1.3 Instrument Space

We take two of the intrinsic characteristics, number of sensors and data rate, and combine them into a scalability space as shown in Figure 1. This illustrates a space of remote instrumentation based on number of sensors per instrument and bandwidth required per sensor. Three groups are shown: those with a few sensors, those with tens of sensors, and those with hundreds or more. This space serves to illustrate the range of data processing requirements of instrument systems. These three groups span several orders of magnitude in numbers of sensors and duty cycle.

FIGURE 1.

Space of typical remotely accessed instruments.



2.2 Operational Modes

Instruments may be operated in a number of different modes. The mode affects many aspects of the software design, since it effectively sets the ground rules for the requirements.

Collection. In this mode, the information flow is strictly from the instrument. This simplifies the communication hardware and software, but limits the space of possible solutions to any issues that may arise. For example, the instrument cannot be placed in a power-saving state by the result of a data analysis.

Collection with maintenance control. In this mode, information flow is primarily from the instrument. However, some control can be asserted on the instrument for tasks such as calibration and installing new firmware.

Human-in-the-loop. In this mode, a scientist evaluates the data during collection. Any adjustments or control operations are communicated to a technician controlling the instrument. This mode is suitable for high-value instruments where a technician is normally on-site anyway.

Remote control. In this mode, the scientist (or software) is in direct control of the instrument. For some instruments, this requires fail-safe systems to prevent damage to the instrument or injury to bystanders.

3 Developments in Information Technology

Though technology has been applied to information well before the digital revolution, it is only recently that we have considered the processing of information and its embodied knowledge as themselves artifacts worthy of independent study.

Much of the recent work in information technology involves some group of computers acting in concert, which is termed distributed computing. Distributed computing has had a long history of both theoretical and applied research. Many of the more powerful results, however, have had limited success moving from the research setting to the real-world setting. This has been due in part to the significant effort necessary to use these results.

In this section we look at some recent developments in information technology, especially in distributed computing, that help manage the consumption and production of information.

3.1 Web Services

The Web services community has not agreed upon a precise definition of a Web service. However, for the purposes of this paper we will use that given in the WS Arch [71].

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Web services provide a platform- and location-independent means to execute code through a remote procedure call or message-oriented interface. This basic idea is arguably no different from those of many other distributed computing efforts. However, previous efforts have generally been aimed at smaller scales, or have been research projects. These efforts have tended to neglect the distasteful ugliness of real-world, large-scale distributed systems, and so have found limited adoption.

The Web services community, on the other hand, is actively committed to the real world, blemishes and all. This has led to a number of design principles, based in part on the experiences with the World Wide Web (WWW). Web services tend to emphasize loose-coupling, openness, interoperability, and simplicity. Performance and conciseness are somewhat secondary. Another characteristic is that Web service standards focus on

network interoperability as opposed to the code portability¹ emphasized by standards such as Common Object Request Broker Architecture (CORBA) [42].

3.1.1 Web Service Standards

Web service interfaces are usually described in Web Service Definition Language [15] (WSDL), which is fundamentally message-based but can be interpreted in terms of remote procedure calls or document exchanges. WSDL is a World Wide Web Consortium [59] (W3C) standards effort supported primarily by IBM and Microsoft.

The service interface is platform and language neutral, and messages can be delivered by any protocol for which a WSDL binding has been defined, such as SOAP over HTTP or SMTP. Services are defined as collections of *ports* or addresses implementing the service, each with an abstract definition, the *port type*, and a concrete definition, the *binding*. Each port is further defined in terms of *operations*, which form the method signatures for the exchange of messages at the ports. Bindings for each port define which protocols can be used to deliver messages to that port.

Any number of protocols can be used to deliver messages to the Web Service but SOAP has become the de facto least common denominator for bootstrapping other protocols which may be necessary for performance, reliability, or security reasons. The application can be written in almost any language and SOAP bindings currently exist for numerous languages. SOAP can use a variety of transports including HTTP, SMTP, raw TCP, and instant messaging protocols such as Jabber [34].

3.2 Grid Computing

The Grid is a grand plan for the future of scientific computation. The term Grid comes from analogy to the electric power grid, and Grid computation likens computer resources to electricity. Computational resources should be dependable, consistent, and ubiquitous. The user of the computational power should be unaware of its exact source. Like the electrical power grid, the Grid requires significant infrastructure, both software and hardware. On the Grid, resource management and scheduling cannot be accomplished by logging-on to the desired machine and manually verifying its idle status. Reliable and pervasive services must be in place to automatically locate computational resources, retrieve data from disparate locations, perform the computation, and return the results to the user. All of this should take place without the user being aware of the details of geographical locations.

The Grid is a form of distributed computing, but takes an approach somewhat different from previous work. Globus, the de facto standard for the Grid, provides a low-entry barrier toolkit, rather than a complete, integrated environment. Users can utilize small parts of Globus, such as job creation, without wholesale changes to their applications, operating systems, or programming languages. In this sense, like Web services, the Grid is an attempt to bring distributed computing out of the research lab and into the real world. Grid computing has brought coherence to much of the development effort for

1. Code portability is the ability to run or compile code written for one platform, such as Microsoft Windows, on a different platform, such as Linux.

large-scale computing projects, and has enhanced the ability to focus distributed computing and storage resources on a single large-scale application.

3.2.1 Open Grid Services Architecture

The essentially complementary approaches of Web services and Grid computing suggest the deployment of Grid services as Web services, and at this confluence is the OGSA standard being developed by the GGF [26].

The OGSA standard describes a Web service architecture for the Grid. Grid services are Web services, and applications interact with the Grid according to Web service precepts. OGSA includes “an integrated set of service definitions that address critical application and system management concerns”, including such services as being defined by the Data Access and Integration Services Working Group [25]. OGSA itself is based on the Open Grid Services Infrastructure (OGSI) [56]. OGSI defines some characteristics and interfaces common to all Grid services.

As a preliminary step in realizing a Web-services based Grid, the Globus organization has rewritten the Globus Toolkit to serve as a reference implementation of the OGSI. This has been released as GT3, and provides secure SOAP messaging based on Globus security. Service lookup at the application level is available by Universal Description, Discovery and Integration (UDDI), with lower level Globus functions still relying on Globus MDS-2 LDAP calls.

3.3 Components

A fundamental approach to any difficult problem is to find abstractions that can be decomposed into less difficult subproblems. In mathematics, for example, a complex theorem is built up from a series of lemmas. In computer science, divide-and-conquer is a fundamental paradigm for algorithm development. The assumption here is that complexity can be localized. This technique has been applied to software engineering in a number of forms, including structured programming and object-oriented programming (OOP).

Programming language objects are not exposed to the end-user, however, and so software components were introduced to apply some of principles of OOP to entities that can be manipulated by the end-user. Components do not replace objects, but are an application of the ideas behind OOP, such as abstraction and encapsulation, to the broader issues of packaging, deployment, and coarse-grained composition. No agreed upon definition of software components exists, so rather than attempt a definition, we will list some of the characteristics we include in the notion of a software component:

Independently deployable. A component can depend on another component for correct operation, but it still must be possible to install and upgrade it individually. Furthermore, this deployment should not require programming skills.

Reusable by third-parties. Components can be reused in different applications by the end user. Such interchangeability is crucial to making them effective at coping with rapid change.

Connectable by third-parties. Components can be connected to one another in a fashion similar to stereo components, or hardware modules. This allows the end user to solve completely new problems from existing components.

Large granularity. A component generally encapsulates more functionality than an object. Most components will be implemented internally with many objects.

Distributed. This characteristic is not usually included in the notion of components, but is important to our purposes. Varying hardware restrictions may impose varying locations for the software pieces composing an instrument system, so collocation cannot be assumed. Two components which might reside in the same process under one hardware configuration might need to be separated under a different configuration. Note that a distributed component framework does not preclude collocation optimizations for when the components do reside in the same machine or even process [49].

A number of different efforts have developed component standards. The OMG has developed the CORBA component model as part of the CORBA suite. Sun's Enterprise JavaBeans is also considered a component model, as well as Microsoft's COM. The DOE CCA is developing a component model tailored for scientific computing. This standard includes aspects of parallel computing and FORTRAN interoperability that the other standards do not address.

3.4 Knowledge Engineering

As pace of interactions increases, interoperability becomes ever more difficult to achieve. Computer systems are still brittle. Seemingly minor changes can bring large systems to a grinding halt. Frustratingly, often the changed information is semantically equivalent, but merely with minor syntactic variations.

The area of knowledge engineering seeks to address these problems by providing techniques for processing "knowledge" about the world. By providing languages for specifying ontologies, and systems for manipulating them, knowledge can be handled more effectively.

Application ontologies provide descriptions of a domain. Problem-solving methods (PSMs) are a knowledge engineering artifact for describing tasks at high-level [53]. A PSM can be used reused with different application ontologies.

4 The Cyberinfrastructure Revolution

Information is the driver of scientific inquiry and insight. From observations and measurements, we validate, refine, and formulate models and theories. From the literature we extend previous work, and cross-fertilize between disciplines to discover new ways of applying existing ideas. From personal communications we discuss, argue, and brainstorm, fomenting new paradigms to challenge existing modes of thought.

Scientific progress results from the gradual accumulation of knowledge, enabled by the exchange of information. In prehistoric times, information technology consisted of our human oral, aural, and mnemonic abilities. Our sum knowledge was limited by what we

could collectively remember. The invention of writing extended our total knowledge beyond our limited memories, and exploited our large visual cortex to increase our I/O bandwidth. For the first time, we could accumulate knowledge in a permanent form, greatly amplifying what we could accomplish otherwise. The invention of the printing press caused another exponential leap in knowledge. If we view the dissemination of information as data traveling through a graph, the printing press allowed the branching factor at each nexus to be much greater than what could be accomplished through manual transcription.

Each of these developments profoundly altered the frequency, depth, and pervasiveness of human discourse, which eventually led to disruptive changes to society and the nature of scientific inquiry. We believe that we are now undergoing a similar revolution, brought on by the enormous increase in processing, communication, and storage capabilities of the last few decades. By speeding the acquisition, processing, and analysis of data, information technology also speeds the intellectual processes crucial to scientific progress. This acceleration occurs at all levels, ranging from the cognitive process of the individual scientist to the social interactions between disciplines. The result is a qualitative change in the way scientists work.

In the information network model mentioned above, information technology not only increases the branching factor, but also the frequency of mass exchanges of information. With electronic publishing in the form of such things as digital libraries and even weblogs, no longer does a two-way mass exchange take a significant amount of time and effort. Furthermore, search engines such as Google effectively increase the fan-in factor as well. Search engines allow us to quickly process a large amount of incoming data. In the future, software agents may effect another leap in our fan-in factor.

The advances in electro-optical technologies compose the foundation for these developments. Effective utilization of these advances, however, requires a layer of hardware, software, institutions, and personnel, which has been termed *cyberinfrastructure*. This layer bridges the gap from the switching capabilities of the base IC technologies to the problem-specific applications of each discipline. It transforms the raw functionality of integrated electro-optics into services and abstractions directly usable by domain specialists.

Cyberinfrastructure comprises a diverse set of elements. Among them are integrated services for knowledge management, observation and measurement, visualization, interaction, and collaboration.

4.1 Bringing Instruments Online

Instruments and sensors form a crucial part of cyberinfrastructure, and realizing its full benefits depends critically on sophisticated, dynamic interactions between producers and consumers of scientific data. These interactions require the integration of instruments within the cyberinfrastructure.

4.1.1 A New Approach

Data from instruments typically has an extensive lifecycle, which includes corrections and calibrations, annotations with metadata to indicate its semantic meaning, and then

storage in a database or filesystem on some computer system. Further stages in the life-cycle can include combining the data with other data, creating new data artifacts by performing analyses and simulations, and ultimately curation as it is transferred to new storage media.

Standards bodies, such as the Global Grid Forum, are working on the later transitional phases listed above, but have yet to adequately address the earlier stages near the instrument, where the data is still relatively “live”. Instead, researchers currently treat observed data as primarily a computational commodity or raw material, to be turned into knowledge through reduction, fusion, and analysis. As such, information technology has been applied primarily to the latter parts of this value-add chain, in the reduction and analysis phase. These can be performed off-line with respect to the instrument that collected the data and outside any possible interaction between the analyzer of the data and the instrument itself. Large data projects organized around instruments such as detectors at the Large Hadron Collider have yielded useful application testbeds such as GriPhyN [29] and iVDGL [8] that focus on data somewhat after it has been acquired from the instrument. These projects by design largely ignore the instrument from whence the data comes.

This focus on the later stages of the data lifecycle is understandable; while database systems, for example, have had large scale convergence on a few API standards (such as SQL and XQuery), instruments vary widely in their architecture, construction, external interfaces, and usage modes. However, this single-minded concept of data as distinct from its source leaves a critical hole in the Grid that we believe must be filled to realize the full benefits of the new cyberinfrastructure. For some contexts, such as real-time analysis, the data and the instrument are so closely coupled that any separation simply interposes additional latencies and barriers to effective utilization. For other contexts, a closer association of the data with the control of the instrument will foster a tighter feedback loop between the instrument and the domain experts. Among other benefits, this will allow experts to analyze and adjust the data gathering process itself while it is still in the critical early stages, which will avoid collecting poor quality data, saving time and resources.

4.2 Trends and Benefits

A confluence of trends are energizing the development of cyberinfrastructure. These trends are both opportunities and pitfalls. In each of these trends, online instruments are a significant contributor to realizing the benefits.

Rapid growth in hardware performance. Researchers today have much greater access to powerful hardware. Computations that used to be performed at large centers can now be performed on desktop workstations. A recent NSF survey reported that 74% of scientists perform analysis on machine different from that which generated the data [5].

This decentralization of computer power allows scientists to directly access instruments for various purposes. Some scientists may have wish to merely perform preliminary analysis to verify that the experiment is preceding correctly. Others may perform the complete analysis on their desktop. But in each case, fully utilizing this capability

requires standards to ensure interoperability between the desktop software and the instrument.

The increasing power of desktops also grants groups without access to high-performance computers the ability to realistically benefit from direct access to instruments. Such access will be facilitated by standards to promote the widespread availability of software.

Rapid growth in digital information and data. As more information comes online, the benefits of bringing instruments online are magnified synergistically. Having the information and the instrument online means that the analysis and comparison process can be performed more rapidly, which allows the scientist to be more selective of the quality of data that she collects.

Furthermore, Grid-enabled instruments will allow the automatic assimilation and cataloging of the huge amounts of collected data. A recent NSF survey found that 50% of scientists reported that ineffective data cataloging and/or difficulty locating required data was the primary impediment to the use of digital or federated data repositories. An additional 26% reported inadequate or missing information on quality control as the primary impediment [6].

To resolve these issues and prevent data archives from becoming data mortuaries requires accurate, reliable metadata and data format standards. The most logical place to bind the data to its metadata is at its source, the instrument itself, and doing so requires online instruments.

Another application of high-quality metadata is data mining.

Large shared resources. As many of our scientific disciplines mature, they place ever greater demands on our ability to measure the natural world. Further breakthroughs require measurements at the extremes of scale, resolution, and energy; which requires ever more expensive instruments. Placing these unique instruments on the Grid simplifies the sharing of these valueable resources.

Collaboration. As our ability to investigate ever more complex systems increases, we are finding that the behavior of these systems often involve complex interactions that span multiple areas of expertise. This requires interdisciplinary, often international, collaboration.

This interdisciplinary collaboration often involves using data from other fields. In fact, at a recent meeting of the environmental research and education community it was reported that some scientists are spending 75% of their time finding and converting data from other fields [4]. This is obviously an enormous waste of precious human capital.

Format differences will always exist, for historical reasons if nothing else. But by developing the proper knowledge management, formats can be converted automatically when needed.

As an example of how a common instrument middleware can greatly increase the effectiveness of collaboration, consider the National Virtual Observatory. Currently astronomers can query the database for images of a specific wavelength. The quality of

astronomical images can be affected by atmospheric conditions and seismic conditions, among others. By correlating the image metadata with readings from weather, pollution, and seismic monitors, additional useful information can be gleaned from the imagery.

Real-time usage. Current scientific research requires increasingly detailed knowledge about our natural world. Often, acquiring this knowledge requires real-time interaction. For example, numerical weather models perform better when they can control meteorological sensors in real-time feedback loops to optimize data collection. Integrating these instruments into such systems is simplified if they can participate fully in the Grid.

Predictive monitoring of forest fires is another application driving the push towards real-time interaction with sensor arrays.

In silico simulation. As computational power increases, computer simulations are being used increasingly in tandem with observed data. The comparative analysis of the simulated results with instrument measurements is greatly facilitated if Grid interfaces are available to bridge between the simulated and observed, thereby automating the analysis and verification. For example, changing a parameter in the simulation may require that observed data be renormalized. If the sensor data is well-described with an ontology, this renormalization can be automated.

Education. As our economy matures, we are seeing an accelerated shift to an information-based society. Education becomes increasingly paramount. On-line instruments can make direct sensor data much more available to primary and secondary schools, which ordinarily might not have the budget to access specialized instruments nor the specialists to interpret complicated, legacy data formats.

Accessibility. Computer technology has been shown to be of great benefit in allowing the physically challenged to contribute to science. Some tasks, however, such as converting speech-to-text, still require processing power beyond that of the average desktop. For these tasks, one can envision a service running on a shared cluster. In this context, the microphone then becomes an instrument. Some desktops might have sufficient power to perform some of the computations locally, thus improving response-time and reducing the load on the shared cluster. This scenario closely resembles those for which the Grid was designed, and thus such systems would benefit from instruments as Grid services.

Diversity. Minority serving institutions have historically lagged in the adoption of computer technology. Network connectivity has emerged as one of the most significant impediments. High bandwidth instrument data can always be filtered and aggregated to reduce bandwidth requirements, but such processing would have to be separately implemented for each data format. Standard Grid interfaces for instruments can facilitate automation by providing a common base for describing data in a manner than generic problem solving methods (from knowledge engineering) can then act upon.

5 Meeting the Challenge

The preceding section has described how Grid-enabled instruments can benefit the conduct of scientific research and spread its impact to segments of society that have had

limited access to scientific instruments. In this section we outline how we might actually bring instruments online through the development of a common middleware architecture.

Unfortunately, bringing instruments online is not as simple as connecting them to a network. For instruments to participate, they must be represented via portions of the cyberinfrastructure layer. These portions, known as instrument middleware, allow instruments to be active participants in the Grid by allowing them to interact with Grid services.

5.1 A Common Instrument Middleware

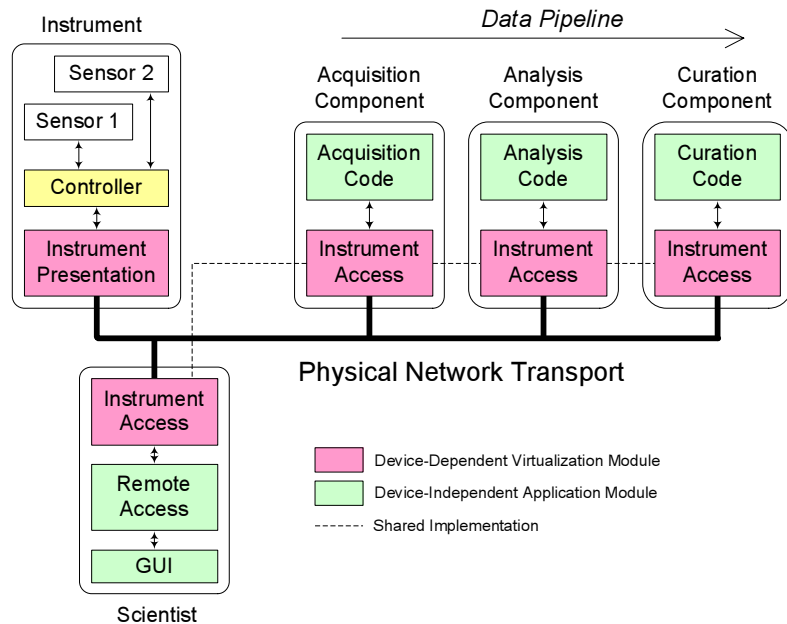
As part of the cyberinfrastructure, the NSF envisions thousands of online instruments along with thousands of collaboratories [3]. Distinct middleware could be implemented ad hoc for each instrument. However, this would be wastefully redundant and, more importantly, risks balkanization, which would lead to numerous compatibility and interoperability problems. The benefits offered by cyberinfrastructure would be severely limited.

Furthermore many of these instruments are unique, or one of only a few. They may in fact be essentially never “finished,” undergoing constant revision to meet the particular dictates of each experiment. This causes continual re-invention of the controlling software and frequently results in redesign of data output formats from the instrument. As a consequence, software at later stages in the reduction and analysis pipeline is potentially subject to rewriting when any changes are made to the instrument, even if changes are remedial with no new capabilities being added.

These concerns suggest the development of a common middleware architecture, utilizing modern approaches to distributed computing. Such an architecture would seek to discover commonalities among the different disciplines, and build a common set of paradigms, interfaces, and ontologies. Complex differences between disciplines would be hidden where possible by encapsulation within adaptation layers, or ameliorated through the use of knowledge engineering—presenting a relatively simple but flexible Web service interface to the rest of the data pipeline. To maximize reuse, we believe that this layer should be placed as close as possible to the instrument, which would shield the bulk of the cyberinfrastructure from changes to the instrument and mediate access to downstream components. This middleware layer need not be colocated with the controller, and can even be elsewhere on the network. Figure 2 illustrates how this middleware would provide an abstract interface to the bulk of the system.

FIGURE 2.

This diagram shows how a common middleware infrastructure can improve the interoperability and resilience of instrument software systems. Without an encapsulation layer, some—or perhaps even all—components that access the instrument must have complete knowledge of the instrument explicitly built-in. In contrast, with a common instrument middleware, changes to the instrument can be isolated.



The core activity of the CIMA project is to research and develop a standard middleware and interface methodology for instruments that can be used with a variety of instrument types and interconnect technologies, and which can be embedded in the hardware package of the instrument.

5.2 Requirements

To meet the challenge, the instrument middleware will need to satisfy the following requirements:

- Support processing of data in real-time across a variety of timescales.
- Support different rates of data production under varying available bandwidth. This will probably require the ability to integrate multiple protocols within one system.
- Provide end-to-end reliability and fault tolerance.
- Support large numbers of sensors.
- Support integrated representations of instruments comprising disparate sensor types, such as a climate-monitoring instrument consisting of multiple sensors.
- Represent and support the different operational modes described in Section 2.2.
- Support real-time fusion of heterogeneous information from many sources to create one data stream.

- Support for pipelined data reduction and analysis.
- Provide functional transparency. Each function of the instrument must be completely and accurately accessible.

5.3 The Common Instrument Middleware Architecture

To realize these requirements, we propose a Common Instrument Middleware Architecture (CIMA) based on concepts from Grid computing and Web services. CIMA has three main aspects.

1. A model and ontology for describing instruments. The ontology should be sufficient to fully use the instrument.
2. A model and ontology for describing scientific data. This model should be sufficient to allow the data to be used in ways that may not have been previously foreseen.
3. A reusable component toolkit for building interoperable instrument systems.

Web services provide a diverse set of standard protocols with which to build distributed systems. Though our requirements could arguably be met with CORBA or some other technology, much of the current work on cyberinfrastructure is based on Web services. Therefore, as much as possible, we base our work on existing and emerging Web service standards, in particular the Open Grid Services Architecture (OGSA).

In addition we will use current work in component systems for scientific programming and existing efforts to develop a component architecture for data acquisition and instrument control in X-ray crystallography.

5.3.1 Instrument Model

The benefits of a common instrument middleware rely critically on encapsulating each instrument within an abstraction layer, which presents an device-independent interface to the rest of the system. This commonality between instruments facilitates code reuse between different instruments, and also eliminates the need to rewrite such code when the underlying instrument and controller are redesigned.

Abstraction. Fundamentally, abstraction is a mapping from a set of interfaces to a single common interface. The mapping must hide unimportant distinctions between interfaces, while preserving the important ones. Thus, if the common interface is too small, important functionality will be lost. On the other hand, if the single interface is simply the union of all the interfaces, then little will have been gained.

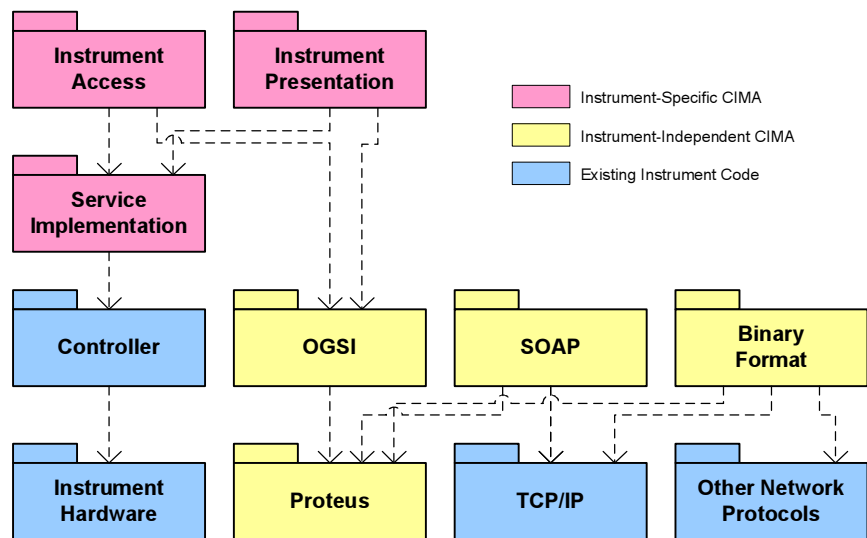
When specified appropriately, the abstraction layer will mitigate the dependence of acquisition and analysis software on the details of the hardware design, and thus facilitate the integration of instruments into the Grid infrastructure. Instrument users will transparently reuse existing software when instruments are modified or upgraded. By opening up their products, instrument manufacturers can take advantage of a broader range of control application and data sinks (data fusion and analysis software). Providing a Grid-aware middleware will facilitate interaction between research groups working at different places in the value chain of scientific inquiry.

Because finding a single unified interface for all instruments is unrealistic, CIMA will use stratified abstractions. At the bottom will be one interface shared by all instruments. The next layer above this will comprise multiple interfaces, each grouping broad categories of instruments. Each higher layer will consist of ever more specific interfaces.

Service Implementation. The actual code implementing the abstractions is the service implementation. This set of routines is responsible for mapping from the instrument-specific functions to the CIMA instrument model. They would perform operations such as providing a description of the instrument, providing calibration data, and exposing control and data functionality. Where appropriate, operations will be provided to establish bi-directional streams for control and data, possibly through WSDL extensions. Figure 3 shows how the various software modules might be layered.

FIGURE 3.

This diagram illustrates the layering dependencies of the various software modules that will together implement the instrument model. Some CIMA modules will be instrument-independent, and can be shared between different instrument models. Others will need to be developed specifically for a given instrument. Both of these modules will use existing instrument and controller modules. In the initial implementation the instrument's existing functionality will be preserved as completely as possible to minimize impact to existing systems, and to compare performance between the the CIMA stack and the existing native implementation. Note that not all layers are appropriate for all instruments.



Additional functionality that could be built into instruments at the service layer include automatically generated metadata, access to diagnostic software, and functional simulation for backward compatibility with older hardware.

Ontology. A central design requirement is that CIMA applications must be able to develop an operational model of the instrument from a minimum of external knowledge. This will reduce the burden of managing and administering a large variety of instru-

ments. Each function of the instrument must be completely and accurately represented by the Grid interfaces provided by the middleware.

The OGSi specification provides a simple mechanism for bootstrapping this functionality. As a Grid service, the instrument will implement the GridService port type, which provides standard operations to query information about the instrument. Given the appropriate query, the instrument will return an ontological description of itself, in a language such as the RDF. The application will then parse the instrument description to learn how to interact with the instrument's control and data ports.

For example, a thermocouple is used to measure temperature, but through a voltage measurement and a calibration curve. This calibration, and hence the measured temperature, may be derived from standard tables for the type of thermocouple used, or through a carefully performed manual calibration against a primary standard. The description of such a thermocouple in RDF might look something like this:

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!-- Default namespace is now "http://cima.org/" -->
<rdf:RDF
  xmlns="http://cima.org/"
  xmlns:rdf="&rdf;"
  xmlns:rdfs="&rdfs;">
  <Description
    my_key="CIMA:A7DB-2287-E7DB-6745"
    my_name="TC7443"/>
  <!-- either a URI and namespace pointing to the
    service definition (CIMA:uri and uri_namespace) or
    the text of the service definition as a CDATA block
    (CIMA:binding_template) can be used -->
  <WSDL
    uri="http://kaplab.iu.edu/TC_service.wsdl"
    uri_namespace="http://kaplab.iu.edu/ns/TC_service">
    <binding_template xml:space='preserve'>
      <!-- CDATA BLOCK OF WSDL -->
    </binding_template>
  </WSDL>
  <Characteristics
    idesc="K-type thermocouple on 16 bit A/D"/>
  <Calibration
    Port_name="thermocouple_reading"
    type="vector">
    <value>
      30, 1.179, 35, 1.397, 40, 1.606, 45, 1.813,
      50, 2.020, 55, 2.225
    </value>
  </Calibration>
  <port
    data_format="u16"
    name="thermocouple_reading"
```

```
    signal="voltage"  
    port_direction="OUT"  
    port_type="INTEGER"/>  
</rdf:RDF>
```

The schema consists of four main parts: a description that identifies the specific instrument platform, information about how to use the service (a WSDL document for the service or a URI pointer to it), calibration information for this instrument, and the characteristics of the data produced by each channel. The latter may seem redundant to the WSDL document but it provides a place to put semantic information about the instrument's control and data channels and additional type information which may be necessary if WSDL types are too opaque for application code to parse.

After the application has queried the instrument for a description, it can parse the description to extract an operational model of the underlying instrument and information about how to interact with the instrument's service ports. In this example the application finds that there is one data (OUT) port that provides a voltage signal as a 16 bit unsigned integer. Furthermore the application can consult the CIMA RDF Schema [13] to determine what voltage means and how best to handle this data in a computational or user interface context.

Though this message-exchange is relatively simple, additional research will be necessary to develop rich, yet controlled vocabularies for representing sensors and instruments. A number of XML standards for instruments have been developed, but they have been ad hoc and thus difficult to analyze and verify. Instead, we will use RDF Schema [13] to represent instrument capabilities, metrics, and calibration in terms of signal types and control functions. These representation will be directly associated with the instrument, so applications can then use this schema to develop strategies for acquiring data from the instrument and processing it. Currently this information is usually implicit in acquisition, processing codes, and intermediate file formats; resulting in a system fragile with respect to modifications, upgrades, or new features.

The challenge, then, is to provide a uniform means of representing the identity and functionality of instruments that may span several orders of magnitude in numbers of sensors and duty cycle.

5.3.2 Data Model

Interoperability between cross-disciplinary sources of data is one of the goals of the Instrument Middleware Project. To achieve this, applications will need to share data models.

Similarly to the stratified instrument model, we plan to develop a stratified data model. Some degree of commonality will be developed across all instruments. This will form the lowest layer. Higher layers will be developed for instruments that share the requisite properties.

5.3.3 Component Toolkit

The Grid will be a complex system composed of interoperable, distributed parts. To allow maintenance and changes to the Grid without interrupting operation, the parts will need to be independently deployable. These requirements suggest components as the paradigm for building the Grid.

Our component toolkit will require capabilities in a number of areas. The challenge will be to provide these capabilities within a software base that can accommodate the different scales and requirements for the different classes of instruments.

Communication. WSDL provides a high-level protocol for message exchanges, but does not specify how these abstract messages are actually transmitted on the wire. Instead it fully supports multiple low-level protocols by providing an extensible binding framework to map WSDL messages to the actual wire format.

SOAP has gained wide acceptance as the standard binding for Web services, and thus CIMA will support SOAP. Performance is modest[17], however, especially for floating-point numbers, and thus CIMA will also support other protocols. SOAP will be used as the lingua franca for an initial handshaking, after which more efficient protocols will be used if available. Proteus is a multiprotocol library which will be used to provide a protocol-independent API to the bulk of the CIMA code. It supports the use of plug-ins for run-time protocol selection.

Proteus includes an efficient binary protocol, but another alternative is the use of a notion sometimes referred to as binary XML. XML is a textual format for information. But implicit in its specification is abstract model of document content. This model is captured by the XML Infoset specification [62]. Infosets suggest that we treat XML as merely one representation of an Infoset. We can thus encode an XML Infoset in a much more efficient wire format that avoids some of the performance limitations of normal XML.

Also, it may be the case that our middleware is connected to the instrument via a wire protocol of some kind. In which case, we need some standard way of incorporating this vendor-specific protocol into our acquisition software without making the acquisition software vendor-specific.

Interaction. CIMA will also support different interaction modalities. Remote procedure calls¹ [9] are convenient and intuitive to most programmers. However, they require a request-response message exchange, and typically require as many concurrent threads on the server as outstanding concurrent calls. Their blocking behavior can also be cumbersome when used for interactions that are inherently asynchronous. Thus, for situations requiring high throughput, other message-exchange patterns, such as one-way messages, will be more appropriate. Furthermore, for the most demanding applications, some kind of stream paradigm will be provided to minimize any per message overheads, such as connection set-up and tear-down. Messages are already a part of WSDL, but not streams, indicating a need to research how to integrate streams with Web services.

1. By RPC here, we are referring to a synchronous, blocking, request-response message exchange that mimics a traditional, local procedure call.

Security. Since a primary design goal of CIMA is to integrate instruments into a Grid of computing and storage resources, authentication and authorization is of some importance. We will track OGSF working group activities closely for emerging authentication strategies. Initially we will provide an external validation service for each instrument that the instrument can use to authenticate users based on Globus proxy certificates. In addition to authenticating potential users the validation service will provide authorization information and a limited life key that applications can use to make future calls to the instrument. All communications between the application, validation service and instrument could be encrypted via SSL or IPSec for transport level security.

Ultimately the security model for an instrument depends heavily on how the instrument is used, and a significant aspect of this project is to develop use models. Both authorization and authentication are problematic and in some cases multiple independent schemes may be required at the same time. Allocation of an instrument's resources may imply time dependent changes in the authorization scheme, as, for example, when a specific group of researchers and no others are using a telescope, and within this group specific individuals may be allowed to control the telescope while others may only observe.

6 Related Work

A Grid-based Collaboratory for Macromolecular X-Ray Crystallography Using Synchrotron Light Sources. The XPort [39] project, a DOE NGI program, targeted revolutionary improvements in telepresence for major scientific instrumentation systems. The goal was to exploit a combination of advanced networking technology, Grid services, and remote instrumentation technologies to achieve interactive “better-than-being-there” capabilities for remote experiment planning, instrument operation, data reconstruction, and data analysis. Some of these capabilities were deployed and demonstrated at major DOE facilities, including the Advanced Photon Source and the Advanced Light Source. Building on work from the Globus group and the DOE 2000 Common Component Architecture Forum, the project addressed several issues related to NGI network-based instrumentation including high-speed data collection, reduction, storage and visualization, and real-time instrument monitoring for the acquisition of X-ray crystallographic data from the DND-CAT beamline sector at the Advanced Photon Source and for a similar sealed-beam X-ray diffraction system at the Indiana University Molecular Structure Center. Each component in the system was implemented as a CCAT [10] component written in C, Python or Java. CCD frame data flows from the detector to a cache at the beamline, then to a network archive (currently HPSS servers at IU or ANL) where data reduction and visualization applications that may be distributed across a computational Grid can retrieve individual frames and intermediate data sets. Each component is a CCAT instance running on a machine appropriate for its function. Currently these components include data acquisition, local cache management and instrument monitoring for Bruker goniostats and CCD detectors. We also experimented with the then-new SOAP protocol as a run-time system to replace the relatively heavy-weight Globus mechanisms for services such as instantiating components, connecting them, sending control signals among them, and more generally for data transfers which do not require high-speed, low latency communications

Distributed real-time systems using Tao CORBA [63]. Considerable work has been done to make CORBA suitable for real-time applications. The leading effort is the TAO

object broker, a CORBA implementation designed to support avionics applications. As such TAO emphasizes bounded (low) latency communications and fault tolerance.

Architecture for Accessing Data Streams on the Grid [45]. Plale has developed a flexible architecture for real-time access to streaming data. She develops a taxonomy for data streams which can be used to determine when a data stream system can be characterized as a data resource accessible through a Grid Data Service. She then realizes such a service through the dQUOB [46] real-time query system. We envision that CIMA can be used to shield such a system from the peculiarities of individual instruments.

EPICS [64]. The Experimental Physics and Industrial Control System (EPICS), developed at the Accelerator Technology (AT-8) group at Los Alamos National Lab) and the Advanced Photon Source (APS) at Argonne National Lab, consists of an architecture for building scalable control systems and a collection of code and documentation comprising a software toolkit. EPICS is based on the idea of virtual channels between acquisition code and the underlying hardware. Although well designed for high data rate applications its complexity has limited acceptance and broader use.

Astronomical Instrument Markup Language (AIML) [11]. AIML is a NASA project to create an XML DTD for the HAWC airborne camera and related systems [12]. The aim was to create a representation of the control and data systems primarily as a specifications document to coordinate work between hardware and software engineering groups. AIML was also used to develop simulations of the hardware and to generate user interfaces. The vocabulary used in the AIML DTD was drawn primarily from the hardware engineering effort and included a large percentage of project-specific terminology, but the project has laid some useful foundations for developing general ontologies for instruments.

Universal Plug and Play (UPnP) [65, 66] is a Microsoft standard to allow devices to interact over an IP network using a zero configuration approach. Basics include using DHCP to acquire an address, a network registry scheme for service discovery based on pre-assigned device codes, and an on-line documentation system to allow users to map device codes to capabilities.

The Salutation Consortium [67] is a competing pervasive computing effort by Japanese gadget makers and NIST in the U.S. offering zero configuration networked device management and service discovery. Salutation code is proprietary although there is a Lite version available for experimentation and evaluation.

7 Case Studies

To illustrate how CIMA can be used in different domains, we present a number of case studies in this section.

7.1 X-Ray Crystallography

The class of sensor represented here is a custom design, single, large shared instrument that produces gigabytes of data per day.

Building on work already done at the Argonne Advanced Photon Source (APS) DND-CAT beamline [24] we will provide a CIMA-enabled version of the XPort [39] platform. This work is expected to initially support the efforts of two independent research teams, one located at IU Molecular Structure Center[50] and the other at the University of Sydney in Australia [51].

Researchers from both of these groups travel frequently to APS, to perform experiments and collect data at ChemMatCARS beamline [52] and DND-CAT beamline[24]. Furthermore, some researchers from IU will be utilizing the recently built MB-CAT beamline at Advanced Light Source (ALS) at Berkeley Lab [38].

Presently, the major disadvantages for researchers collecting data at remote synchrotron sites are, the travel, extra expenses and restricted accessibility for certain international users.

To address these issues we propose that synchrotron sites by adapting to CIMA and providing a small amount of assistance by beamline personnel, will be able to through Web based technologies, allow researchers to remotely perform a X-ray diffraction experiment from their “home laboratories”. The researcher can have the samples to be studied, shipped to the beamline by e.g. a commercial shipping company. At the beam line the samples can be unpacked by the beamline personnel and mounted on the instrument using crystal mounting robots (or mounted manually). The entire process from unpacking, crystal mounting and data collection can be supervised by the researcher in the home laboratory, using Web based tools and designated video/audio conferencing tools specially developed for this purpose.

A large body of data reduction and analysis codes is associated with diffraction experiments and these software can be modified to use CIMA to interact directly with e.g. the DND-CAT and ChemMatCARS detectors. Since a significant effort has already been made in developing the web-based tools for the X-ray crystallography [39], we will also make efforts to extend the existing data evaluation and analysis tools. The analysis tools will allow rapid evaluation of the initial data from the APS by the remote collaboration team, optimizing the use of these over-subscribed resources.

Other aspects, in which adaptation to CIMA will greatly benefit the crystallographic community is that CIMA will provide a unified data access format for the raw unprocessed data. Currently there are very few options for storing raw data from X-ray diffraction experiments in a standard format that can later be read by a wide range of data processing software.

The variety of CCD detectors used at the various beamlines and their different output format is creating several issues for crystallographers. Usually the raw data collected at a certain beamline is processed using the CCD vendor's supported software. Once the crystallographer is back in the home laboratory a reprocessing of the raw data can be tedious unless a copy of the data processing software is purchased from the CCD vendor. The problem is compounded by the fact that many researchers utilize multiple beamlines with different CCD detectors.

To resolve this issue, CIMA will seamlessly interface to a wide variety of makes and models of CCD detectors used at each beamline. Additionally to the CCD vendors native data output format, CIMA will be able to support a standardized metadata for-

mat/output format (img-CIF/CBF) [32] for each CCD detector, regardless of its make and model.

The recently operational NSF funded Analyzing and Visualizing Instrument-Drive Data (AVIDD) facility at Indiana University will be used to store and evaluate the data being generated by the CCD detectors located at the APS. Researchers at other universities will be able to monitor the experiment at APS while processing and evaluating the data being transferred to the IU's AVIDD system.

The availability of an CCD equipped instrument in the Indiana University Molecular Structure Center identical to that at ChemMatCARS will facilitate the development and testing of the specialized crystallographic analysis code that will be accessing the CIMA API from the application side. A major emphasis will be put on the development of software that will allow the remote collaboration team to convert the CCD frame images into a representation of reciprocal space that can be viewed using volume rendering techniques. This will allow the researchers to rapidly examine the initial data to insure that the mosaic character, twinning, and splitting can be handled by the data processing software.

7.2 Embedded Network Monitoring

End-to-end performance analysis for distributed applications and for facility-level monitoring of Grids (e.g., the iVDGL Grid Operations Center [61]) is increasingly dependent on measuring devices embedded in the network (e.g., Network Weather Service nodes (modest data flows, real-time acquisition, moderate number of data sources)). These monitoring nodes provide real-time network performance data needed to detect and correct network problems and represent a class of sensors used in groups of 10 to 100 that produce moderate data flows. Network embedded nodes are being developed for application performance monitoring and analysis under the NSF grant STI-0129592 Network Management Tools for End-to-end Performance Measurement. As a part of this project the interface for these nodes will be modified to include a CIMA interface and monitoring software will be modified to evaluate CIMA in this context.

7.3 Wireless Networked Sensors

The extreme test for CIMA as a unifying technology for ubiquitous computing is embedding it in wireless networked sensors such as the Berkeley Mote. These would explore small data flows, real-time acquisition, and a large number of data sources. The current mote controller and radio transmitter implementation is approximately 1 cm², with a stated aim of reaching sub-mm² sizes co-fabricated with MEMS-based sensors and actuators. This class of sensor explores the use of CIMA in ubiquitous computing applications.

8 Acronyms

API

Application Programming Interface

APS

Advanced Photon Source

ATLAS

A Toroidal LHC ApparatuS

CCA

Common Component Architecture

CERN

European Organisation for Nuclear Research

CIMA

Common Instrument Middleware Architecture

COM

Component Object Model

CORBA

Common Object Request Broker Architecture

DDDAS

Dynamic Data-Drive Application Simulation

DOE

Department of Energy

FTP

File Transfer Protocol

GGF

Global Grid Forum

GriPhyN

Grid Physics Network

HTTP

HyperText Transfer Protocol

iVDGL

International Virtual Data Grid Laboratory

LDAP

Lightweight Directory Access Protocol

MDS

Monitoring and Discovery Service

MEMS

Microelectromechanical Systems

OGSA

Open Grid Services Architecture

OGSI

Open Grid Services Infrastructure

OOP

Object-Oriented Programming

PSM

Problem Solving Method

RDF

Resource Description Framework

SMTP

Simple Mail Transport Protocol

SOAP

In the original specification, SOAP was an acronym for Simple Object Access Protocol. Later versions stated that SOAP was not an acronym for anything.

UDDI

Universal Description, Discovery, and Integration

W3C

World Wide Web Consortium

WSDL

Web Service Definition Language

XML

Extensible Markup Language

9 References

- [1] *Advanced Photon Source*. Retrieved on November 26, 2003 from <http://www.aps.anl.gov>.
- [2] Mario Antonioletti. *Grid Data Service Specification*. Retrieved on November 26, 2003 from https://forge.gridforum.org/projects/dais-wg/document/Grid_Data_Service_Specification-ggf9/en/1.
- [3] Atkins, Daniel E., et. al. *Revolutionizing Science and Engineering Through Cyberinfrastructure: Report of the National Science Foundation Blue-Ribbon Advisory Panel on Cyberinfrastructure*. National Science Foundation, 2003. Page 1.4.
- [4] Ibid, Page 3-10.
- [5] Ibid, Page B-4.
- [6] Ibid, Page B-11.
- [7] *Atlas*. Retrieved on November 26, 2003 from <http://atlas.web.cern.ch>.
- [8] P. Avery and I. Foster, "The International Virtual Data Grid Laboratory (iVDGL)," 2003, <http://www.ivdgl.org>.
- [9] A. D. Birrell and B. J. Nelson, "Implementing Remote Procedure Calls," *ACM Transactions on Computer Systems*, vol. 2, pp. 39-59, 1984.
- [10] R. Bramley, K. Chiu, S. Diwan, D. Gannon, M. Govindaraju, N. Mukhi, B. Temko, and M. Yechuri, A component based services architecture for building distributed applications. *High Performance Distributed Computing Conference*, 2000.
- [11] J. Breed, "Astronomical Instrument Markup Language," NASA GSFC, 2000, <http://pioneer.gsfc.nasa.gov/public/aiml/>.

- [12] J. Breed, "High Resolution Airborne Wideband Camera," NASA GSFC, 2000, <http://pioneer.gsfc.nasa.gov/public/hawc/>.
- [13] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," World Wide Web Consortium, 2003, <http://www.w3.org/TR/rdf-schema/>.
- [14] M. Champion, C. Ferris, E. Newcomer, and D. Orchard, "Web Services Architecture," W3C Consortium, 2002, <http://www.w3.org/TR/ws-arch/>.
- [15] R. Chinnici, M. Gudgin, J.-J. Moreau, and S. Weerawarana, "Web Services Description Language (WSDL) Version 1.2," World Wide Web Consortium, 2002, <http://www.w3.org/TR/wsdl12>.
- [16] K. Chiu, M. Govindaraju, and D. Gannon, "The Proteus Multiprotocol Library," presented at Proceedings of the 2002 Conference on Supercomputing, 2002.
- [17] K. Chiu, M. Govindaraju, and R. Bramley, "Investigating the Limits of SOAP Performance for Scientific Computing," presented at Proceedings of the Eleventh IEEE International Symposium on High Performance Distributed Computing (HPDC'02), 2002.
- [18] CCA Forum, *The Common Component Architecture*. Retrieved on November 26, 2003 from <http://www.cca-forum.org>.
- [19] Common Component Architecture Forum," 2000, <http://www.cca-forum.org/>.
- [20] Erik Christensen et al. *Web Services Description Language (WSDL) 1.1*. Retrieved on November 26, 2003 from <http://www.w3.org/TR/wsdl>.
- [21] F. Curbera, Y. Golland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana, "Business Process Execution Language for Web Services, Version 1.0," IBM, 2003, <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>.
- [22] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," University of California at Irvine, 2000,
- [23] Frederica Darema, *Dynamic Data-Drive Application Simulation*, Retrieved November 26 from <http://www.dddas.org>.
- [24] "DuPont - Northwestern - Dow Collaborative Access Team," 2003, <http://tomato.dnd.aps.anl.gov/index.shtml>.
- [25] I. Foster, D. Gannon, J. Nick, and W. Johnston. *Open Grid Services Architecture: A Roadmap*. Retrieved in 2003 from http://www.ggf.org/ogsa-wg/ogsa_roadmap.0.4.pdf.

References

- [26] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration (Draft)," 2002, <http://www.globus.org/research/papers/ogsa.pdf>.
- [27] Global Grid Forum. *Global Grid Forum*. Retrieved on November 26, 2003 from <http://www.gridforum.org>.
- [28] A. Grimshaw, *Legion*. <http://www.cs.virginia.edu/~legion>.
- [29] "GriPhyN - Grid Physics Network," 2003, <http://www.griphyn.org/index.php>.
- [30] H. Haas and D. Orchard, *Web Services Architecture Usage Scenarios*. W3C Consortium, 2002. <http://www.w3c.org/TR/ws-arch-scenarios/>.
- [31] "iGrid 2000: Empowering Global Research Community Networking," 2000, <http://www.startap.net/igrd/>.
- [32] img-CIF/CBF, <http://www.iucr.org/iucr-top/cif/imgcif/index.html>.
- [33] "Indiana University Molecular Structure Center," 2003, <http://www.iumsc.indiana.edu/>.
- [34] "Jabber Software Foundation," Jabber, Inc., 2002, <http://www.jabber.org/>.
- [35] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Mobile Networking for Smart Dust," presented at Proc. of ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, 1999.
- [36] C. Kesselman and I. Foster, *Globus Project*. <http://www.globus.org>.
- [37] C. Kesselman and I. Foster, *The GRID: Blueprint for a New Computing Infrastructure*: Morgan Kaufmann, 1999.
- [38] MB-CAT at the Advanced Light Source, <http://www.mbc-als.org>.
- [39] D. F. McMullen and R. Bramley, "The XPort Project," 2000, <http://www.cs.indiana.edu/ngi>.
- [40] D. F. McMullen, "The iGrid Project: Enabling International Research and Education Collaborations through High Performance Networking," presented at Proceedings of the IEEE Internet Workshop, Osaka, Japan, 1999.
- [41] "DCOM," Microsoft, 1998, <http://www.microsoft.com/com/tech/DCOM.asp>.
- [42] Object Management Group. *Common Object Request Broker Architecture : Core Specification, Version 3.0.2*. Retrieved on November 26, 2003 form <http://www.omg.org/cgi-bin/doc?formal/02-12-06>.
- [43] D. Pearson, K. Adams, D. Gannon, D. McMullen, M. McRobbie, A. Robel, S. Wallace, and J. Williams, "TransPAC: A High Performance Network Connection

- for Research and Education between the vBNS and the Asia-Pacific Advanced Network (APAN)," presented at Proceedings of The International Workshop on Asia-Pacific Area Advanced Research Information Sharing Technology, Internet Workshop '98 (IWS'98), Tsukuba, Ibaraki, Japan, 1998.
- [44] K. S. J. Pister, "SMART DUST: Autonomous sensing and communication in a cubic millimeter," 2003, <http://robotics.eecs.berkeley.edu/~pister/SmartDust/>.
- [45] Beth Plale. Architecture for Accessing Data Streams on the Grid. *2nd EUROPEAN ACROSS GRIDS CONFERENCE (AxGrids 2004)*. January 2004.
- [46] B. Plale and K. Schwan. Dynamic querying of streaming data with the dQUOB system. *IEEE Transactions in Parallel and Distributed Systems*. 14(4):422–432. April 2003.
- [47] "PRAGMA, the Pacific Rim Applications and Grid Middleware Assembly," San Diego Supercomputer Center, 2003, <http://pragma.sdsc.edu/>.
- [48] "Resource Description Framework (RDF)," World Wide Web Consortium, 2003, <http://www.w3.org/RDF/>.
- [49] Douglas C. Schmidt, Nanbor Wang, and Steve Vinoski. Object Interconnections: Collocation Optimizations for CORBA. *C++ Report*. September 1999.
- [50] "Service Crystallography at Advanced Photon Sources," Indiana University Molecular Structure Center, 2003, <http://www.iuisc.indiana.edu/synchrotron/scraps.html>.
- [51] Specialist Crystallography at the Advanced Photon Source (SCrAPS), <http://www.ansto.gov.au/natfac/scraps.html>.
- [52] A. S. R. Program, "Specialist Crystallography at the Advanced Photon Source," 2001, <http://www.ansto.gov.au/natfac/scraps.html>.
- [53] M. Stefik. *Introduction to Knowledge Systems*. Morgan Kaufman, San Francisco. 1995.
- [54] "TransPAC," 2003.
- [55] Steve Tuecke et al. *Open Grid Services Infrastructure Version 1.0 (Draft 29, April 5, 2003)*. Retrieved on November 26, 2003 from http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-29_2003-04-05.pdf
- [56] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, and P. Vanderbilt, "Open Grid Services Infrastructure (OGSI) (draft)," Global Grid Forum, 2003, http://www.gridforum.org/ogsi-wg/drafts/draft-ggf-ogsi-gridservice-23_2003-02-17.pdf.
- [57] J. Villacis, M. Govindaraju, D. Stern, A. Whitaker, F. Breg, P. Deuskar, B. Temko, D. Gannon, and R. Bramley, CAT: A high performance distributed com-

References

- ponent architecture toolkit for the grid. *High Performance Distributed Computing Conference*. 1999.
- [58] A. Woo, "Mote Documentation and Development Information," 2000, <http://www.cs.berkeley.edu/~awoo/smartdust/>.
- [59] "World Wide Web Consortium," <http://www.w3.org/>.
- [60] "World Wide Web Consortium," 2003, <http://www.w3c.org>.
- [61] "iGOC - iVDGL Grid Operations Center," International Virtual Data Grid Laboratory, 2003, <http://igoc.iu.edu/>.
- [62] J. Cowan and R. Tobin, "XML Information Set," World Wide Web Consortium, 2001, <http://www.w3.org/TR/xml-infoset/>.
- [63] D. C. Schmidt, D. L. Levine, and S. Mungee, "The Design of the TAO Real-Time Object Request Broker," *Computer Communications, Elsevier Science*, vol. 21, 1998.
- [64] S. A. Lewis, "Overview of the Experimental Physics and Industrial Control System: EPICS," 2000: Lawrence Berkeley National Laboratory, 2000, <http://csg.lbl.gov/EPICS/OverView.pdf>.
- [65] "Universal Plug and Play Device Architecture," 2000: Microsoft Corporation, 2000, http://www.upnp.org/download/UPnPDA10_20000613.htm.
- [66] "Understanding Universal Plug and Play: A White Paper," 2000: Microsoft Corporation, 2000, http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc.
- [67] "The Salutation Consortium," 2003, <http://www.salutation.org/>.
- [68] "Extreme! Computing," IU Extreme Computing Group, 2003, <http://www.extreme.indiana.edu/>.
- [69] "Global Grid Forum," 2003, <http://www.ggf.org>.
- [70] U. S. Navy, "Distributed Engineering Plant," 2000: US Navy, Naval Surface Warfare Center, 2000, <http://www.nswc.navy.mil/dep/>.
- [71] <http://www.w3.org/TR/ws-arch/>