GA–A26484

AN INVESTIGATION OF SECURE REMOTE INSTRUMENT CONTROL

by
D.P. SCHISSEL, G. ABLA, T. FREDIAN, M. GREENWALD, B.G. PENAFLOR,
J. STILLERMAN, M.L. WALKER, and D.J. CIARLETTE

JULY 2009

GENERAL ATOMICS

DISCLAIMER

GA–A26484

AN INVESTIGATION OF SECURE REMOTE INSTRUMENT CONTROL

by
D.P. SCHISSEL, G. ABLA, T. FREDIAN,* M. GREENWALD,* B.G. PENAFLOR, J. STILLERMAN,* M.L. WALKER, and D.J. CIARLETTE[†]

This is a preprint of an invited paper to be presented at the Seventh IAEA Technical Meeting on Control, Data Acquisition, and Remote Participation in Fusion Research, June 15–19, 2009, Aix-en-Provence, France, and to be published in the *Proceedings.*

*Massachusetts Institute of Technology, Cambridge, Massachusetts.
[†]US ITER Project Office, Oak Ridge National Laboratory, Oak Ridge, Tennessee.

Work supported in part by
the U.S. Department of Energy under
DE-FC02-04ER54698, DE-FC02-01ER25455, and DE-AC05-00OR22725

GENERAL ATOMICS PROJECT 30200
JULY 2009

GENERAL ATOMICS

Abstract

This paper examines the computer science issues associated with secure remote instrumentation control for magnetic fusion experiments. Computer science research into enhancing the ability to scientifically participate in a fusion experiment remotely has been growing in size in an attempt to better address the needs of fusion scientists worldwide. The natural progression of this research is to examine how to move from remote scientific participation to remote hardware control. The vision is to define a gatekeeper software system that will be the only channel of interaction for incoming requests to the secured area of the experimental site. The role of the gatekeeper is to validate the identification and access privilege of the requestor and to insure the general validity of the proposed request. The vision for the gatekeeper is that it be a modular system that is simple in design and defined in a way that makes its implementation and operation transparent and obvious. The architecture of the module interface is flexible enough that it can easily allow the future addition of new modules. At the same time, it should be transparent to end-users and allow a high volume of activity so as to not provide a work bottleneck. The results of the gatekeeper design and initial implementation are presented as well as a discussion on the implication of this research on the operation of fusion experimental machines such as ITER.

1. Introduction

It has been approximately twenty years since the concept of the worldwide web was first discussed within the high-energy physics community [1]. On a similar time scale, people were realizing that the reduction in the cost-performance ratio of computing and communications combined with the beginnings of these technologies being used in a synergistic fashion had the opportunity to dramatically change the way science is conducted. The term "collaboratory" was used to define [2] a "center without walls in which researchers can perform their research without regard to physical location — interacting with colleagues, accessing instrumentation, sharing data and computational resources and accessing information in digital libraries." Since that time there has been an explosive worldwide growth both in developing collaboratories for a broad range of sciences as well as the beginnings of the science of collaboratories [3].

There are a multitude of facets that comprise typical scientific enquiry or workflow. Therefore, as scientific teams have tried to push part of their workflow beyond the physical boundary, the term collaboratory has been applied to a wide variety of capabilities. The vast data volumes anticipated to be generated by the Large Hadron Collider (LHC) and the Compact Muon Solenoid (CMS) detectors located at the European Center for Nuclear Research (CERN) [4] necessitated a novel design for data storage and analysis that went beyond CERN's physical boundaries. Their solution, to create a computational and data grid that extended worldwide via high-speed networks is an excellent example of a collaboratory. Here, the traditional computer room is replaced by hundreds of distributed computing sites [5]. Present day fusion research does not have the data storage requirements of the high-energy physics community but, nonetheless, it offers an example of virtualizing computational resources (what is today called cloud computing) [6]. In this instance, FusionGrid provided access to large computational codes over the Internet without the scientist having to know where the resource was located nor have specialized knowledge of the supporting infrastructure. To support the debating of ideas that is so critical to the scientific process, collaboratories have investigated a wide variety of methods from commercial solutions to customized creations. Two of the major technologies that have been created out of science-based projects are the Access Grid [7] and EVO [8]. Both attempt to bring together a large number of distributed scientists by making a unified communication and visualization environment.

Collaboratories have also dealt with instrument control. In the field of microscopy, the DeepView system [9] provided a collaborative framework for distributed virtual microscopy that include instrument control. Most likely one of the most obvious cases of remote control are the two rovers, named Spirit and Opportunity, that are part of the Mars Exploration Rovers (MER) mission [10]. The Interoperable Remote Component Architecture (IRC) is a generalized framework for equipment command, control, and monitoring of remote devices and sensors [11] and has been used and tested in a variety of environments including rover control.

In this paper, we examine specifically the issues associated with equipment control in a nuclear fusion experiment. The requirements here are very different than those mentioned previously whether it be the many more degrees of freedom than are available in microscopy or the real-time control that is not available for the Mars rover (~20 minute signal round-trip-time). Section 2 presents our specific motivation and vision for fusion research while Sec. 3 presents our design. Finally, Sec. 4 discusses the present implementation and ongoing testing while Sec. 5 is a summary discussion.

2. Motivation and Vision

The construction and operation of ITER is substantially supported by the engineering and scientific expertise of the many nations that have joined this large scientific collaborative effort. This geographically dispersed group will deliver plant subsystems, including diagnostics, which will be assembled to create ITER. When operation begins, approximately a decade from now, onsite as well as off-site personnel will substantially support ITER. The more that can be done remotely for ITER's benefit, the greater the level of expertise that can be made available, since it is not reasonable to assume all the world's experts will travel to France to support this experiment. To this end, the ability to perform remote control, whether for a diagnostic or the machine's operation, will increase the probability of success for the ITER mission.

2.1 Recent Experience

For a further examination of motivation, we discuss our recent experience performing remote plasma control. Compared to diagnostic control, plasma control is significantly more complicated and therefore is an excellent test bed for examining the finer nuances of the Gatekeeper design.

The DIII-D Plasma Control System (PCS) is a hardware and software application that is used to control and monitor tokamak plasmas [12]. Since its early success at DIII-D, it has been deployed worldwide [13] including on the EAST tokamak at ASIPP in Hefei, China.

In March of 2009, members of the DIII-D fusion research staff working from their home laboratory in San Diego, USA, participated in the remote operation of the EAST Tokamak in Hefei China. Among the goals was to provide real-time physics and computer support of EAST plasma preparations.

A critical component of this was the ability to operate a local copy of the EAST Plasma Control System graphical user interface (GUI) in San Diego and have it be able to communicate directly with the EAST PCS server processes running on computers at ASIPP. Being able to run a local copy of the EAST PCS GUI was necessary for achieving optimal performance in viewing and updating EAST plasma control parameters for the upcoming discharge. For the most part, the performance observed was close to the same performance obtained by users running the GUI at ASSIP.

Since the PCS had been designed as a client server application the only thing that was needed to enable remote operation was an open TCP/IP connection between the computers in San Diego and EAST. This was achieved through the use of Secure Shell (SSH) TCP/IP tunnels which were made to an EAST portal computer which could be accessed by San Diego computers and provide a link from San Diego to the EAST PCS computers behind the EAST firewall.

Security for protecting the EAST tokamak from unauthorized requests to update plasma discharge parameters was provided through the built in access control facilities of the PCS software. The PCS access control codes automatically screen updates sent to the coordinating "waveserver" process responsible for managing plasma control parameters for the impending discharge, to insure that they have originated from an authorized source.

Network performance (10 to 40 Kbytes per second) between San Diego and EAST was a limiting factor for viewing the archived post shot results from EAST, but did not have any significant impact on the operation of the PCS GUI. This was due to the fact that most of the communication between the local PCS GUIs and remote control server processes involved very small packets of information, such as 32 bit floating point vertex pairs for specifying waveform target values.

One important issue, that remains unresolved, was the communication of PCS status messages originating from the EAST PCS to the locally run GUIs in San Diego. Because of the way in which the SSH tunnels were set up between San Diego and EAST, messages originating from San Diego were able to get through to the EAST computers and allow responses to be communicated back from EAST to San Diego. However, messages initiated at EAST used to send certain status information back to the PCS GUIs were being blocked by the San Diego firewall. In order to get around this limitation, users were required to log directly into the EAST computers from San Diego in order to monitor the status updates to the PCS. A possible redesign of the PCS messaging software is being considered including the ideas presented in the next Section in order to avoid this problem in the future.

The recent experience in the remote plasma control operations support of the EAST tokamak from San Diego proved to be very successful in achieving many of our goals. The lessons learned will be applied to improve the Plasma Control System as well as used in the initial Gatekeeper deployment.

2.2 Gatekeeper Vision

The Operations Request Gatekeeper is responsible for screening all actions, which we define as requests, requiring information entry into the secure machine zone, which is defined as the computer network used for all machine and diagnostic control. This includes requests that are from those parts of the experimental site that sit outside of this control network. The term request is meant to imply anything from a simple command (e.g. change a scalar value) to a complex serious of instructions (a new tokamak PCS file) to perhaps updating software.

The Gatekeeper's decision on whether to pass a request inside the secure zone is based on security verification (authentication and authorization), grammar validation (is the request well formed), and validity checks (is a request within a specified range of values). If all these tests are passed, the request is queued for entry inside the machine zone where a detailed validation check is made before execution.

After detailed examination, the decision has been made to place several elements of the gatekeeper functionality inside the machine zone. These include the authorization and configuration databases, request processing and verification modules. The reasons for this decision are to increase the security of the verification system and to prevent duplication of the request validation code. These checks will need to be performed on requests that originate from in the machine zone as well, so the gatekeeper should use the same verification mechanisms. Further, the current design requires only database reading across the gatekeeper/machine-zone boundary (no database writing) and keeps sensitive information in the most secure zone.

The Gatekeeper system will provide the mechanism to communicate with the correct verification system including a well-defined API. With this design, groups that are supplying machine components will have a guide for their own verification modules that can be tested before placing it into operation.

3. Design

3.1 Overview

Figure 1 is an overview of the Gatekeeper interfaces with external users and the secure machine zone. An external requestor uses a web browser, a command line, or a custom created client employing an API defined as part of this task, to submit a request into the machine zone. Our assumption is that general tokamak data, including relevant plant system status is available through other mechanisms. For example, it is entirely reasonable that the general tokamak data can be replicated outside the machine zone for immediate availability by authorized personnel. Therefore, data requests or machine status requests can be made without interacting with the Gatekeeper.

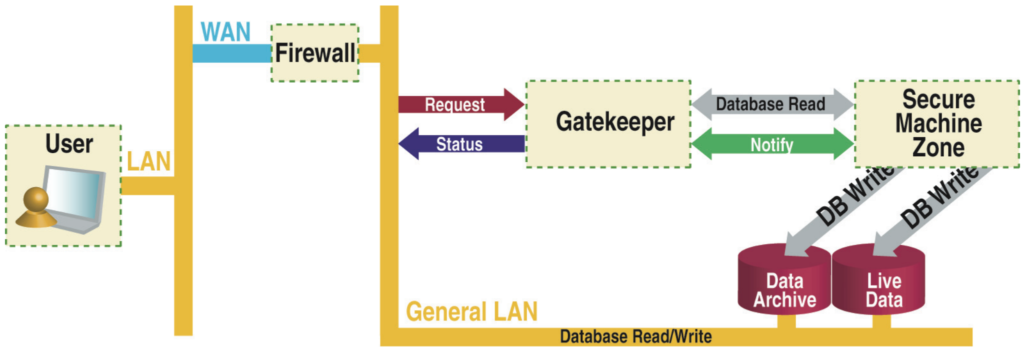

Fig. 1. An overview diagram showing how a user communicates a request into the secure machine zone. All requests entering the secure machine zone must pass through the Gatekeeper.

As the request leaves the WAN, it passes through the experimental facility's border firewall, it enters the site and encounters the Gatekeeper. The first action of the Gatekeeper is to perform initial security checking that includes authentication (you are who you say you are) and authorization (you are allowed to make the request). We note that this security mechanism will most likely not be unique to the Gatekeeper. Such a mechanism is needed to cover all external actions, including the request for data mentioned above. In that sense, the Gatekeepers first action will be to call this central security mechanism. There can be a variety of methods to solve this general security problem. Today, many of the collaboratory projects provide single-sign-on in a distributed environment through the usage of a Public Key Infrastructure (PKI) [14]. The X.509 certificate standard along with a trusted Certificate Authority is one method to implement PKI for secure communications [15]. Of course, the method chosen today may be very different than the one implemented a decade from now. Therefore, a module design that allows changing of the security mechanism as technology evolves is the most prudent choice. For the remainder of this paper we do not investigate further authentication and authorization but instead go deeper in the Gatekeeper system specifically.

Our design also adheres to the principle where no system writes to another system's database. This is done for tighter security; remote reading is allowed but writing is not. Therefore, the Gatekeeper can read information from the secure machine zone but not write. A request, that the Gatekeeper deems worthy of passing on, is queued up for transfer and a Gatekeeper module inside the machine zone is notified of the queued request. It is up to the request processor in the machine zone to reach out and read the requested information from the Gatekeeper.

3.2 Gatekeeper Details

Figure 2 shows the details of the Gatekeeper system. As stated in the previous section, the first Gatekeeper action is checking authentication and authorization. After that is accomplished, the Gatekeeper checks that the request is properly formatted and has the proper grammar. As an example, if there is an accompanying file with the request, the file type and format are checked. Or, if the request is in the form of XML it is checked for proper syntax. To be able to make this check, the Gatekeeper reads out of the machine zone from the Configuration Database what is the appropriate syntax (e.g. XML schema). Once this check is satisfied, the Gatekeeper will perform an initial simple check on allowable values (range validation). Simple values and commands must be of proper type and format for entry into the request database. The Gatekeeper determines the allowable values by reading from the Configuration Database.

If both of these checks pass, the Gatekeeper writes into a local request database and sends a notification to the Request Processor that is inside the machine zone. The Request Processor at an appropriate time reads out of the request database all the required information, combines this with information from the configuration database, and then sends the request to the appropriate validation module(s). Validation modules are written by the most appropriate organization, those with the most detailed domain-specific knowledge. Part of the detailed Gatekeeper design is to specify the validation module interface.

Finally, if the validation module accepts the request, the Request Processor passes the request to the appropriate Execution Module (often within a particular machine sub-system) to be acted upon.

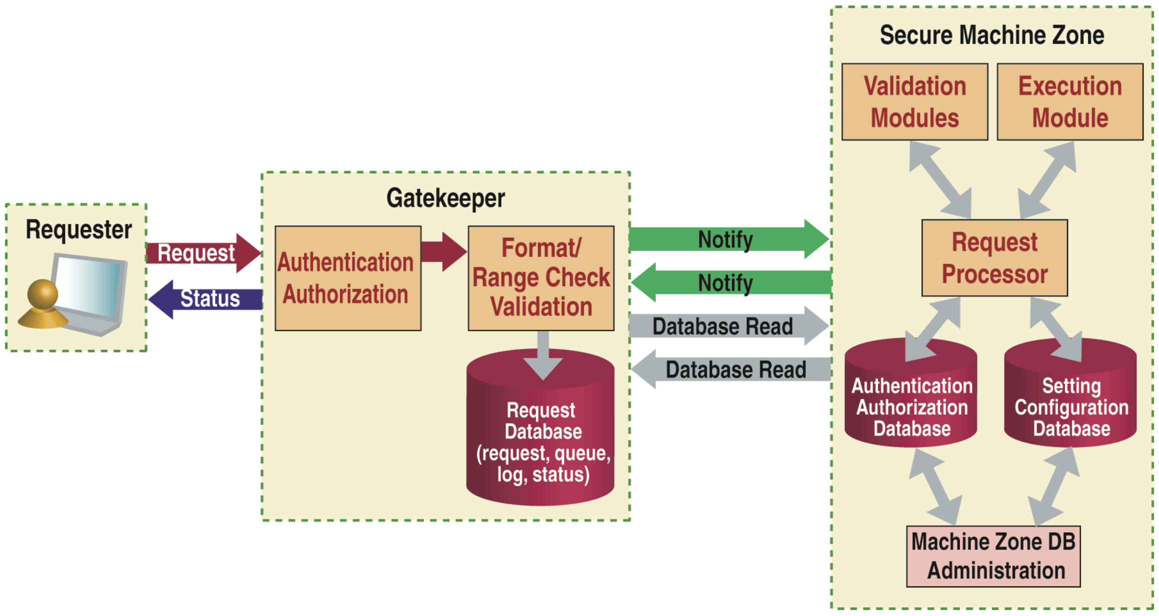
Secure Machine Zone
Validation Modules
Execution Module
Gatekeeper
Requester
Request
Status
Authentication Authorization
Format/ Range Check Validation
Notify
Notify
Database Read
Database Read
Request Processor
Request Database (request, queue, log, status)
Authentication Authorization Database
Setting Configuration Database
Machine Zone DB Administration

Fig. 2. A detailed diagram showing the interaction of the Gatekeeper with the secure machine zone. For security reasons, all database interactions between the Gatekeeper and the machine zone are read only. The administration of the databases inside the machine zone is only allowed from inside the zone.

The status of each submitted request is available to the requester. All steps that a request traverses, namely authentication/authorization, grammar check and range validation, whether successful or not, are logged and can be obtained by the requester or by another authorized user. The aim will be to provide users with sufficient information to correct errors in their requests without exposing unnecessary internal information.

4. Ongoing Implementation and Planned Testing

A prototype software system for the Gatekeeper is presently being implemented. A web browser is acting as the client and Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) with username and password is being used as the security mechanism. Web server applications, running on two different machines, are being used to mimic the Gatekeeper and the secure machine zone. A methodology to support requests to the Gatekeeper based on XML has been developed.

The software implementation design is modular structure so that each functional component of the Gatekeeper is independent of other components. The core functional components are: 1) Authorization and Authentication, 2) Request Format Verification and Initial Range Validation, and 3) Request Processor. The Request Processor includes hardware specific validation modules and corresponding execution modules. All three components run independently from each other and exchange information via simple sessions, event messages, and database I/O. The modular design makes future upgrades easier since one component can be changed without affecting the other components. The easy upgrade is a requirement since new authentication schemes might be implemented, new devices and corresponding Request Processors might be added, over a machine's lifetime. The modular design also makes it possible for the Gatekeeper to accept functional changes of a component while other parts remain the same.

Another benefit of modular design is that each developer is responsible to just those parts that are pertinent to the work being performed. This type of specialization fosters the creativity and excellence within each developer's domain of specialty without being worried about the development of other functionalities contained within the system.

The prototype Gatekeeper development is being done via the Python [16] programming language and the Django high-level Python Web Framework [17]. Django was originally implemented for the daily needs of an online news website and mainly used for rapid development a variety of web-based applications. However, after careful examination, it was realized that the Django framework can be utilized in the development of the Gatekeeper prototype system.

There are several reasons to choose the Django framework. First, Django's design uses the Model-Template-View (MTV), which is based on Model-View-Controller (MVC) software design pattern. It allows clean separation of tasks and responsibilities among the prominent aspects of the Gatekeeper software. There are three primary code divisions in Django applications: Model, View, and Template. The Django Models exist independent of rest of the system and it provides easy access to databases. The View accepts and processes requests to the system. The Template is responsible for presentation of the data to users based on a specified template language. The presented data can be in HTML format, XML or any other customized data format.

Another reason to choose the Django framework is its Object-Relational Mapper (ORM) support. Django allows Python classes and their instances to access Relational Databases without using Structured Query Language (SQL) or dealing with specific relational database implementation. Currently, it supports numerous major relational databases including MySQL, PostgressSQL, Oracle, MS SQL, and several storage engines. Changing a database is as simple as changing the settings of the application. This flexibility is very valuable since the Gatekeeper can access any major database that a machine will deploy and there is no need to rewrite the application if a database changes. Additionally, Django's ORM is written with database security concerns in mind. For example, it handles SQL injection attacks [18], a malicious behavior which tries to access or even modify data without proper authorization.

The last reason to use Django framework is its built-in signaling and dispatching capability. As mentioned in Sec. 3, for the sake of tighter security, the Gatekeeper design adheres to the principle "where no system writes to another system's database". The Gatekeeper and the secure machine zone exchange data via database I/O. But the status of a queued request is transmitted by sending and receiving a signal. The Django framework contains a mechanism that enables components to broadcast fast signals to other components. One or multiple components can be registered as signal listeners and act accordingly. This capability enables the prototype gatekeeper to implement the dispatching in a convenient manner.

The current prototype gatekeeper is simulated with two Linux servers. One server is located in General Atomics/DIII-D and acts as a gatekeeper component. An Apache web server is used as the Gatekeeper front end and MySQL as the main database. It is responsible for authorizing and authenticating users, as well as receiving requests. It also validates the request format and performs a range validation check. Another Linux server is located at Plasma Science and Fusion Center (PSFC/C-Mod) at MIT. It runs with a similar setup, but runs the secure machine zone component. It is responsible for waiting for the request queue notifications and then validating the request and passing the request onward to the proper simulated hardware executers.

The prototype system will be tested in several steps. The development process is utilizing the agile software development process so that each component can be tested just after its implementation. The validation and execution modules of the secure machine zone are tested with simulators, which can act as hardware components during the development.

As the initial test, the Gatekeeper prototype software will be tested with the DIII-D PCS. The current XML-based request scheme is being implemented to describe PCS commands such as updated the discharge shape algorithm or the position of waveform vertices. The schema not only defines the primitive data types such as string, integer, float, and double, but also has the capability to provide metadata about the filename or location of data in binary format. Figure 3 shows a request in XML format that makes updates on several PCS parameters. First it requests that the shape algorithm be changed to the “Double Null w/PID” algorithm and then sets the test shot to 910006. Finally it requests that a new vertex be added to the end of the WVSP1P signal. The web-based client also accepts PCS request files. The PCS system will be placed in the secure machine zone and sending requests via a web browser will test all the capabilities of the Gatekeeper.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<PCS
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://gateway.gat.com/xml/xmlns/gk/pcs.xsd">
      <action>
            <name> top </name>
      </action>
      <action>
            <name> select </name>
             <param> Discharge Shape </param>
      </action>
      <action>
            <name> update </name>
             <param> algorithm =” Double Null w/PID” </param>
       </action>
      <action>
            <name> update </name>
             <param> testshot = “910006” </param>
       </action>
      <action>
            <name> select </name>
             <param> Discharge Shape/Shotstart/Flux Ratios/WVSP1P </param>
       </action>
      <action>
            <name> add </name>
             <param> -1.224564  2.111239 </param>
       </action>
</PCS>
```

Fig. 3. An XML file that is used to make changes to the Plasma Control System.

5. Discussion

Given the positive results of our initial implementation, the Gatekeeper software will be fully assembled and tested during subsequent remote plasma control experiments with the EAST tokamak. It is anticipated that the initial usage during actual plasma operation will be completed by the end of this year. The SSH tunnels that were previously required will not be necessary. The Gatekeeper will handle some of the tasks previously done by the DIII-D PCS such as the unauthorized request security. Utilizing HTTPS, the Firewall difficulties previously encountered regarding monitoring should also be alleviated by the Gatekeeper.

Although we are hopeful that the Gatekeeper will be a positive contribution towards supporting remote PCS usage on EAST, it is expected that new issues will arrive given the newness of the endeavor. It is our view that continued testing and refinement of the Gatekeeper concept on existing devices is the best way to insure that such software is mature enough to be used routinely. With continued work, such software can be ready for usage on ITER when operations commence a decade from now.

References

[1] http://www.w3.org/Proposal

[2] W.A. Wulf, The collaboratory opportunity, Science **261** 854-855 (1993).

[3] G.M. Olson, A. Zimmerman, and N. Bos, Scientific collaboration on the internet, MIT Press (2008).

[4] See http://www.cern.ch/lhc and http://cms.cern.ch .

[5] J. Bunn and H. Newman, Data-intensive grids for high-energy physics, in grid computing: making the global infrastructure a reality, Ed. F. Berman, G.E. Fox, and A.J.C. Hey, pp. 859–906, New York, Wiley.

[6] D.P. Schissel, *et al.,* Grid computing and collaboration technology in support of fusion energy sciences, Phys. Plasmas **12**, 058104 (2005).

[7] L. Childers, T. Disz, R. Olson, M.E. Papka, R. Stevens, T. Udeshi, Access grid: immersive group-to-group collaborative visualization, Proceedings of the 4th Intl Immersive Projection Technology Workshop, Ames, Iowa, 2000.

[8] D. Adarnczyk, D. Collados, G. Deris, *et al.,* Global platform for rich media conferencing and collaboration, Proceedings of Computing in High Energy and Nuclear Physics Conf., La Jolla, California, 2003.

[9] B. Parvin, J. Taylor, G. Cong, DeepView: A collaborative framework for distributed microscopy, Proceedings of ACM/IEEE Conf. on High Performance Computing and Networks (1998).

[10] J. Wright, F. Hartman, B. Cooper, S. Maxwell, J. Yen, J. Morrison, Driving on the surface of mars with the rover sequencing and visualization program, IEEE Society of Instrumentation and Control Engineers (SICE) Conf., 2005, Okayama, Japan.

[11] T.J. Ames, C.F. Hostetter, Instrument remote control application framework, SpaceOps Conference, AIAA 2006-5521, Rome, Italy.

[12] J.R. Ferron, *et al.,* A flexible software architecture for tokamak discharge control systems, Proceedings of the 16th IEEE/NPSS Symp. on Fusion Engineering, 1996, vol. 2, p. 870.

[13] B.G. Penaflor, *et al.,* Worldwide collaborative efforts in plasma control software development, Fusion Eng. Design **83**, 176–180 (2008).

[14] W. Diffie, *et al.,* IEEE Transactions on Information Theory, Vol. IT-22 (1976).

[15] ITU-T X.509 (03/00), ITU, Recommendation, 2002.

[16] Python Programming Language — Official Website, www.python.org/, visited on June 7, 2009.

[17] Django project web site, http://www.djangoproject.com, visited on May 13, 2009.

[18] SQL injection, en.wikipedia.org/wiki/SQL_injection, visited on June 7, 2009.

Acknowledgment

This work was supported in part by the US Department of Energy under DE-FC02-04ER54698, DE-FC02-01ER25455, and DE-AC05-00OR22725.