Chapter 2
Adapting the Instrument Element to Support a Remote Instrumentation Infrastructure

M. Prica, R. Pugliese, A. Del Linz, and A. Curri

Abstract GRIDCC was a project funded by the European Commission with the goal to extend the grid by integrating instruments and sensors with traditional computing and storage resources. This chapter describes how the GRIDCC's instrument element concepts have been implemented to support the challenging requirements of the DORII project and of a usable remote instrumentation infrastructure. The new Instrument Element has been designed in order to be easy to use for both the application engineers and the scientists who are the final users of both the instruments and the grid infrastructure.

Keywords Grid · Middleware · Remote operations · Remote instruments · Sensors · Online processing

1 Introduction

Current grid technologies offer unlimited computational power and storage capacity for scientific research and business activities in heterogeneous areas all over the world. Thanks to the grid, different virtual organizations can operate together in order to achieve common goals. As the technology of the classical grid infrastructure (typically composed of computing and storage elements) matured, the attention shifted toward the actual sources of data: the instruments and sensors. A significant effort worldwide is being put in providing a closer interaction between various types of instruments accessible from the grid on the one hand and the traditional grid infrastructure on the other hand. The goal is to provide an end-to-end grid environment for scientific activity, connecting the points of data collection with the points of data analysis. The GRIDCC [13] project proposed and realized a new grid component called the Instrument Element (IE) [20, 10, 21] that provides the traditional grid with an abstraction of real instruments and grid users with an interactive interface to control them.

M. Prica (✉)
Sincrotrone Trieste S.C.p.A., Strada Statale per Basovizza 14, 34012 Trieste, Italy
e-mail: milan.prica@elettra.trieste.it

F. Davoli et al. (eds.), *Remote Instrumentation and Virtual Laboratories*,
DOI 10.1007/978-1-4419-5597-5_2,

A new implementation of the Instrument Element (IE2) will be among the building blocks of DORII [6]. DORII (deployment of remote instrumentation infrastructures) is a 30-month project that started in February 2008 and is funded by the European Commission under the seventh framework program. It seeks to deploy a testbed for remote instrumentation, building on the experience of several preceding projects like RINGrid [27], GRIDCC, g-Eclipse [11], or int.eu.grid [16]. For the Instrument Element, DORII will mark the passing step from the prototype implementation to production quality software. The IE2 integrates easily to the Open Geospatial Consortium (OGC) [24] reference model and particularly to the sensor web enablement principles thanks to its flexible and extensible architecture. Also, the IE2 will be included in the Italian Grid Infrastructure [14].

2 Related Work

In September 2004, the GRIDCC project was launched by the European Union. Its main goal was to exploit grid opportunities for secure and collaborative work of distributed teams to remotely operate and monitor scientific equipment. GRIDCC also focused on adoption of the grid's massive memory and computing resources for storing and processing data generated by such equipment. Moreover, the GRIDCC middleware has been deployed on a few pilot applications with the aim to validate it and show its use and effectiveness, both in real contexts and in testbed environments. These applications have been selected in strategic sectors such as follows: (1) the remote control and monitoring of a large particle physics experiment [5]; (2) the remote operation of an accelerator facility [7, 25]; (3) the remote control and monitoring of a widely sparse network of small, but numerous, power generators (a real power grid); (4) the landslide hazards monitoring; (5) the ensemble limited area of forecasting meteorology; and (6) the device farm for the support of cooperative distributed measurements in telecommunications and networking laboratories.

The instrument element (IE) is a concept unique to the GRIDCC project. It is an abstraction of the instrument (or group of instruments) into a standard interface, which can be used within the rest of the GRIDCC architecture. The term instrument is used in this context to define a piece of equipment that needs to be initialized, configured, operated (start, stop, standby, resume, application-specific commands), monitored, or reset.

2.1 Motivations for the New Implementation

The 3 year long experience of the GRIDCC project indicated a set of requirements for the Instrument Element middleware. First, ease of use is a fundamental feature for the adoption of the IE framework by its target communities. In particular, attaching devices to the middleware must be kept as simple and intuitive as possible. Second, the security issues must be addressed properly. IE should provide an

effective multi-user environment with a locking mechanism that prevents concurrent access to operations that alter the instrument state. The locking mechanism should allow the reservation of an instrument or a group of instruments for the duration of an experiment. IE middleware must support the grid operations in the most transparent way possible for the final users, while taking care of all the necessary steps to allow safe connection to the grid world. Effective handling of the alarms and the events triggered by the instruments is another highly desirable feature. The orginal IE lacked a few of these features, most seriously the concurrency control mechanism. Besides, connecting new devices required a very complex and hardly intuitive procedure. All of the above suggested a radically different design approach for the IE middleware and thus the new implementation.

3 Design Approach

Despite the fact that the IE2 is a completely new implementation of the GRIDCC's Instrument Element concept, the existing IE WSDL [15] interface has been kept unchanged in order to assure the compatibility with the rest of the GRIDCC middleware, in particular with the virtual control room (VCR) [26]. VCR is a portal-based client for the GLite [12] Grid and GRIDCC instruments that integrates a set of collaboratory tools in support of team work. Minor WSDL changes will likely occur in the near feature to allow for additional features and will require some modifications on the VCR side as well. Just as the original IE, the IE2 runs in the Apache's Tomcat [3] container as an Axis [2] web service. IE2 is entirely implemented in Java. Unlike the old version, it does not use database back-end for storing configuration data and registering new devices. Instead, the framework uses the local XML files for storing such information. Another major difference is the user interface: the old IE had both a web-service interface for remote clients like the VCR and a proper web interface. The latter, however, could provide only remote access to the instrument, but was lacking the support for the grid security. With the new IE, we kept just the web-service interface. Currently, the VCR is the only front-end to the IE2, but other clients (e.g., g-Eclipse) might follow soon.

The interface to a single instrument is called instrument manager (IM). It is a protocol adapter that allows the middleware to talk to the physical instrument or, more precisely, its control system. The major novelty regards the IM creation process, which is described in detail in the next section.

Two locking mechanisms are provided: implicit locking of the IM access occurs automatically during the execution of commands that trigger state change on the instrument or modify the values of its parameters or attributes. Explicit locking of an instrument for a certain time period is triggered by a user command, e.g., to perform a set of operations on the instrument. The explicit locking is also the base for integration with the reservation service.

Instruments can be grouped in logical units called contexts, and contexts can be further grouped in other contexts creating a tree-like structure. Near future

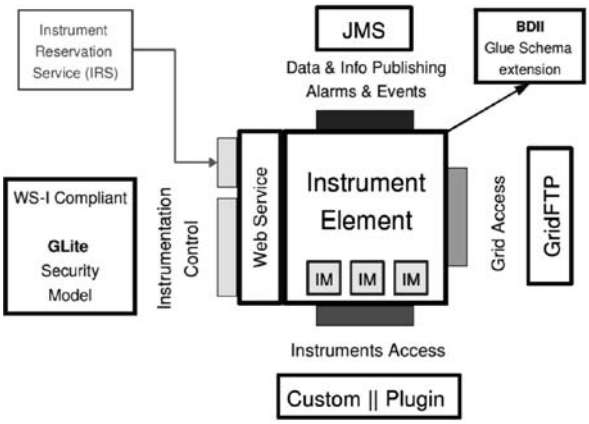

Fig. 2.1 Architecture of the Instrument Element 2

development will certainly include the locking of the entire context in order to support reservation of a set of instruments needed for complex experiments. A simple schema of the IE architecture can be seen in Fig. 2.1.

User authentication is performed using the GLite security model. The IE exports a web-service interface for delegation of the client's VOMS proxy certificates. Certificates are used for both user authentication and authorization. The client, in our case the VCR, delegates the user proxy certificate to the IE where the certificate is used for the user authentication. The VOMS attributes may be used for more fine-grained control of the user authorization. The proxy certificates are further used for accessing the grid, in particular the storage elements. The framework supports the grid operations providing a utility that allows instrument managers to store their output to a grid storage element transparently. All a developer of an instrument manager has to do is to invoke from the command implementing code the utility methods specifying only the basic parameters like the name of the file to be copied and its location on a storage element. The middleware performs the authentication behind the scenes and copies selected files using the GridFTP protocol.

Java Message Service (JMS) [17] allows for asynchronous reception of the messages from the instrument thus eliminating the need for the client polling. In Fig. 2.2 the difference between the JMS monitoring and regular time-interval polling for an attribute can be seen. All four graphs show the output (distance measure) from a proximity sensor. JMS monitoring shows its advantage over client polling both when the variable changes its value frequently (top graph) and in the opposite case when the variable value change occurs seldom (third graph). In the same situations, the regular time-interval client polling either looses intermediate values (second graph) or keeps polling the constant value (bottom graph), thus wasting resources. Another important use of the JMS in the IE2 framework is for signaling alarms and events to the users. IE2 has been tested with the Narada Brokering [23] and the IBM's proprietary implementation, Reliable Multicast Messaging (RMM) [28], but it should work with any provider conforming to the JMS standard.

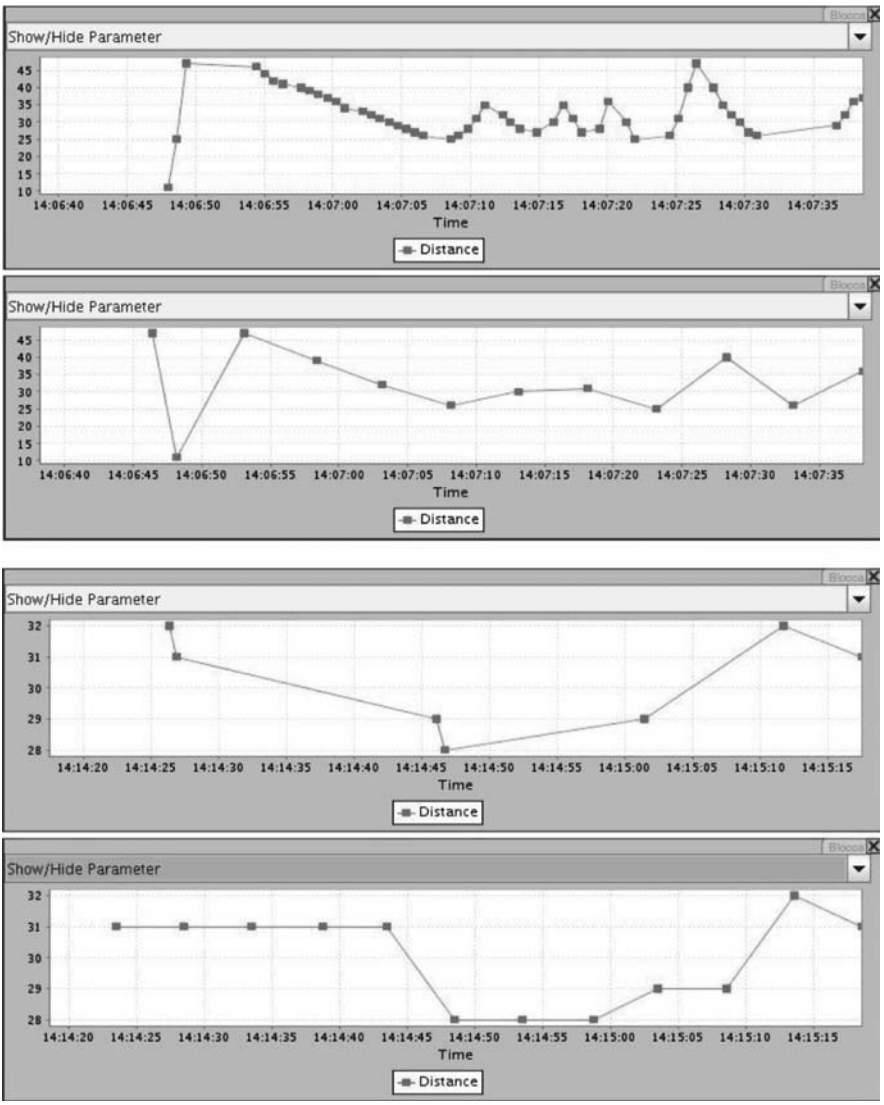

Fig. 2.2 Attribute value polling vs. the JMS monitoring

A centralized information system stores the end point addresses of the published Instrument Elements together with some additional information like status and operative parameters regarding those IEs. Information is kept consistent using the BDII [4] information system adopted by the LCG [18].

The new IE architecture makes the development of the generic plug-ins for control systems much simpler. We provide such a plug-in for TANGO [30], an object-oriented distributed control system based on CORBA, born at ESRF [8] and being

actively developed as a collaborative effort among four institutes: ESRF, SOLEIL [29], ALBA [1], and ELETTRA.

4 Implementation Issues

4.1 Implementing an Instrument Manager

Instrument manager is the actual protocol adapter used to access remotely an instrument. The first step in creating a new manager is writing the XML file that fully describes the instrument. This XML file contains the list of possible instrument states that together define the state machine, together with the details about the commands, attributes, and parameters. Next, for each command, attribute, or parameter, the developer must write the implementing class that extends the ones provided by the framework. Most of the time it suffices to implement a single method for each such class.

The instrument's XML descriptor must conform to the document type definition (DTD) file. Advanced IDE tools like Eclipse provide substantial aid in building and validating the file against DTDs.

Setup of four attributes is required to configure an IM: *name*, *implementation*, *id*, and the *initialStatus*. *Implementation* field contains the fully qualified name of the main class of the IM. *InitialStatus* indicates the state in which the instrument manager should be when first created. Note that the state of the instrument manager is not the same as the state of the instrument itself. An instrument may be already switched on and running while the IM is in the *off* state. (In such case, the meaning of the IM's *off* state might be disconnected from the instrument, while *on* state would mean successfully connected to the instrument.) The IE internally identifies each IM by a unique *id*. Such an *id* is also used to assign an IM to a context and to represent the IM on the client side. *Name* attribute has similar function and is used only internally in the IE, thus it is redundant and likely shall be removed in the future. An example IM and its state machine configuration may be seen in XML Descriptor 2.1.

Algorithm 2.1 IM and State Machine definition

```
<instrumentManager      name="Thermostat"
                        implementation="it.trieste.elettra.uos.test.impl.
                             Thermostat"
                        id="InstrumentManagerID" initialStatus="off">
<stateMachine>
        <status statusName="on"/>
        <status statusName="off"/>
        <status statusName="error"/>
        <status statusName="any"/>
</stateMachine>
...
</instrumentManager>
```

The IM – implementing class must extend the framework's *InstrumentManager* class. It is the class that actually connects to the physical instrument. All other classes that implement commands, attributes, and parameters use this one as the common link.

4.2 Attributes

Attributes are instrument (sensor) variables. Attribute values are normally set through commands, but since in some control systems (e.g., Tango) they may be set directly, we allow for that option as well. The outcome of setting an attribute is identical to running a command that alters that attribute value.

To configure an instrument attribute, the user must specify three mandatory fields: *name*, *description*, and the *enableInStatus*. All remaining fields are optional. The *enableInStatus* defines the state in which the attribute is active (and thus may be monitored) and the unique attribute name. The *name* identifies the attribute in the framework and in the user interface. Attribute's *description* is intended as a help for the final users. The optional *implementation* field contains the fully qualified name of the implementing class. If omitted, the framework will attempt to construct the implementing class name from the instrument manager implementation directory and the attribute name. Another optional field is *subscribable*: if set to *true* each variation of the attribute value registered by the IM will trigger a JMS message. *Lockable* indicates whether the access to the attribute may be locked. *Accessibility* field indicates the type of the attribute, read-only, write-only, or read and write. Read-only attributes cannot be locked. The last optional field is the *unit* of measurement. An example of attribute configuration may be seen in XML Descriptor 2.2.

Nested element *policy* defines how the instrument manager reads the attribute value. *Direct* means that each read is triggered by the user request, while *polling* periodically (*time* value in milliseconds) reads the value. Also, there is a possibility for the IM developer to define a custom policy extending a framework class. Nested element *value* defines the value type of the attribute. Handled types are *string*, *short*, *integer*, *long*, *float*, *double*, *calendar*, *vector*, and *enumeration*.

The implementing class must extend the framework's *Attribute* and implement the getter and setter methods for the reading and writing of the attribute's value on the physical instrument (in most cases, the setter will be a no-operation method, since physical instruments often do not allow for setting of the attributes).

XML Descriptor 2.2 Attribute definition

```
<attribute      enableInStatus="on" name="CurrentTemp"
                description="Read_the_temperature_value_from_the_sensor."
                implementation="TempSensor" subscribable="TRUE" unit="Degree_C">
        <doubleValue value="0.0" />
        <policy>
                <polling time="10000" />
        </policy>
</attribute>
```

4.3 Parameters

Parameters regard instrument settings only. Common practice is to set parameters directly, but they may be set also through commands. The procedure required to configure a parameter is almost identical to the one seen for the attributes. Configuring a parameter is the same as configuring an attribute, except that the *subscribable* and *lockable* fields are omitted. Also the required nested elements are the same as for the attributes. An example of parameter configuration may be seen in XML Descriptor 2.3.

The implementing class must extend the framework's *Parameter* and implement the getter and setter methods for the reading and writing of the parameter's value on the actual device.

XML Descriptor 2.3 Parameter definition

```
<parameter      enableInStatus="off" name="StratingTemp"
                description="The target temperature"
                implementation="" unit="Degree C">
        <doubleValue value="19.5" />
        <policy>
               <polling time="10000" />
        </policy>
</parameter>
```

4.4 Commands

Commands include both the state transitions and the regular commands that do not trigger a state change in the instrument. From the implementation point of view, the two are just the same. As with attributes and parameters, the implementation of a command should be written in a class that extends the one provided by the middleware, while the configuration is performed in the instrument deployment descriptor.

There are four mandatory fields that must be specified when configuring a command. The first one is the command *name*. The remaining three deal with the instrument state. The field *initialStatus* defines the state in which the command may be triggered, *finalStatus* is the uniquely defined state that results from the command execution. The *errorStatus* is the state to which the instrument moves in case an error occurs during the execution. Optional fields are *description*, *implementation*, and *lockable*. Nested elements to commands are command input parameters. Each command may have an arbitrary number of *commandParameters*. Commands configuration may be seen in XML Descriptor 2.4.

Command parameters also have a couple of required fields (*name* and *description*) and a couple of optional ones like the *unit* of measurement and a field that states whether that parameter is a *mandatory* input to the command.

For the attributes, parameters or command parameters that are numeric or calendar types, it is possible to define the range of admissible values (*min*, *max*) used

both by the IE2 and the VCR. An attempt of setting a variable to a value out of range would cause an exception.

XML Descriptor 2.4 Commands definitions

```
<command finalStatus="on" name="TurnOn" errorStatus="error"
     initialStatus="off" />
<command finalStatus="off" name="TurnOff" errorStatus="error"
     initialStatus="on" />
<command finalStatus="on" name="ChangeTemperature" errorStatus="error"
     initialStatus="error">
        <commandParameter name="Temperature" description="" mandatory="TRUE"
            unit="Degree C">
        <doubleValue />
        </commandParameter>
</command>
```

The *Command* class must implement the *executeCommand* method. Mostly, what it does is a simple forward of the user's call to the physical instruments API.

4.5 Deployment

Once the XML descriptor is completed and all the command, attribute and parameter classes are implemented, everything should be packaged in a jar file that must have the following manifest attribute: *InstrumentManager: DescriptorFileName.xml.* The IE ignores jars lacking such attribute. Jars are then deployed to the proper location in the Tomcat's webapp folder. Sample Ant build file provided with the framework greatly simply building, packaging, and deployment operations.

4.6 Front-End (VCR)

The VCR is currently the only available front-end to the IE2. Once registered in the Virtual Organization's BDII registry, Instrument Elements appear in the VCR's resource browser. A click on the IE link starts a https secure connection with the chosen instrument element and expands the resource browser view of the IE's tree-like structure containing contexts and instrument managers. Selected managers appear in the instrument control portlets, an example of which can be seen in Fig. 2.3. The VCR also offers a user-friendly interface to the traditional grid resources of the VO. A user can access data stored in the LFC file catalog and the storage elements or submit jobs to the workload management system by simply filling a form. Security is handled transparently for the user. First time registration into the portal requires the upload of the user's certificate into a MyProxy [22] server. For the successive logins into the portal, the user is asked for his user name and password only, while the VOMS authorization will be performed automatically. Collective use of the resources required by most applications can be programmed using the scripting capability of the VCR or by integrating external workflow management

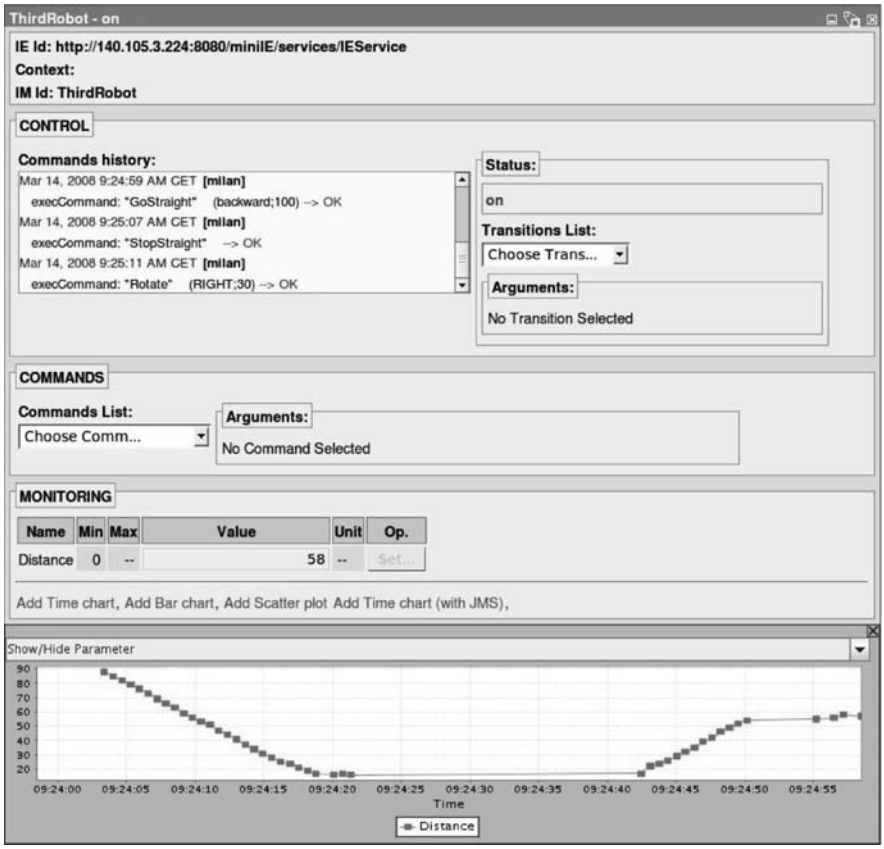

Fig. 2.3 Instrument control portlet: Lego mindstorm robot control

systems. A set of tools provided by the VCR (e-logbook, chat, integrated third party tools like Skype and EVO [9]) supports collaborative efforts.

5 Conclusions

A lesson we learned from GRIDCC is that the instrument element middleware should offer simple, straightforward procedures to the application developers in order to favor a wider adoption of the grid technologies. Otherwise, the risk is that the sheer complexity of the deployment process might mask the true benefits that this technology offers and discourage its widespread use. With the IE2 framework, we tried to make the developer's task much simpler than before.

Although the IE2 is yet to undergo extensive testing in the DORII pilot applications, we can already state that the IE2 provides a flexible solution for connecting a variety of devices to the grid. In particular, IE2 is suitable to control and monitor

both large and complex facilities such as ELETTRA accelerator, as well as simple sensors or even remotely controlled robots like the Lego Mindstorm [19].

An advanced GUI like the VCR makes a perfect front-end to the IE2 and together the two tools offer the scientific community an excellent entry point into the grid world.

References

1. ALBA – The Light Source for Spain. http://www.cells.es/
2. Apache Axis. http://ws.apache.org/axis/
3. Apache Tomcat. http://tomcat.apache.org/
4. BDII – Berkeley Database Information Index. https://twiki.cern.ch/twiki//bin/view/EGEE/BDII
5. CMS – The Compact Muon Solenoid experiment. http://cms.cern.ch/
6. DORII – Deployment of Remote Instrumentation Infrastructures. http://www.dorii.eu/
7. ELETTRA Sincrotrone Trieste. http://www.elettra.trieste.it/
8. ESRF – European Synchrotron Radiation Facility. http://www.esrf.eu/
9. EVO – The Collaboration Network. http://evo.caltech.edu/
10. E. Frizziero, M. Gulmini, F. Lelli, G. Maron, A. Oh, S. Orlando, A. Petrucci, S. Squizzato, and S. Traldi. Instrument element: A new grid component that enables the control of remote instrumentation. In *International Conference on Cluster Computing and Grid (CCGrid)*, Singapore, May 2006.
11. g-Eclipse Integrated Workbench Framework to Access the Power of Existing Grid Infrastructures. http://www.geclipse.eu/
12. GLite: Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite/
13. GRIDCC – Grid Enabled Remote Instrumentation with Distributed Control and Computation. http://www.gridcc.org/
14. INFN Grid – The Italian Grid Infrastructure. http://grid.infn.it/
15. Instrument Element VIGS WSDL. http://gladgw.lnl.infn.it:2002/rcms/services/IEService?wsdl
16. int.eu.grid – Interactive European Grid Project. http://www.interactive-grid.eu/
17. JMS – Java Message Service. http://java.sun.com/products/jms/
18. LCG – LHC Computing Grid Project. http://lcg.web.cern.ch/LCG/
19. LEGO MINDSTORMS NXT. http://mindstorms.lego.com/
20. F. Lelli, E. Frizziero, M. Gulmini, G. Maron, S. Orlando, A. Petrucci, and S. Squizzato. The many faces of the integration of instruments and the grid. *International Journal of Web and Grid Services*, 3(3):239–266, 2007.
21. F. Lelli and G. Maron. Integrating instruments into the grid. *Electronic Journal*, November 2006. GRID Today, Supercomputing '06 Special Coverage, http://www.gridtoday.com/grid/1086062.html
22. MyProxy, a Globus Project that Develops Software for Managing X.509 Credentials. http://dev.globus.org/wiki/MyProxy
23. Narada Brokering Project. http://www.naradabrokering.org/
24. OGC – The Open Geospatial Consortium. http://www.opengeospatial.org/
25. M. Prica, R. Pugliese, C. Scafuri, L. Del Cano, F. Asnicar, and A. Curri. Remote operations of an accelerator using the grid. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Grid Enabled Remote Instrumentation*, Signals and Communication Technology, pp. 527–536.

Springer US, 2008. ISSN 1860-4862, ISBN 978-0-387-09662-9 (Print) 978-0-387-09663-6 (Online).
26. R. Ranon, L. De Marco, A. Senerchia, S. Gabrielli, L. Chittaro, R. Pugliese, L. Del Cano, F. Asnicar, and M. Prica. A web-based tool for collaborative access to scientific instruments in cyberinfrastructures. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Grid Enabled Remote Instrumentation*, Signals and Communication Technology, pp. 237–251. Springer US, 2008. ISSN 1860-4862, ISBN 978-0-387-09662-9 (Print) 978-0-387-09663-6 (Online).
27. RINGrid – Remote Instrumentation in Next-Generation Grids. `http://www.ringrid.eu/`
28. RMM – Reliable Multicast Messaging by IBM Research. `http://www.haifa.ibm.com/projects/software/rmsdk/index.html`
29. Synchrotron SOLEIL. `http://www.synchrotron-soleil.fr/`
30. Tango Control System. `http://www.tango-controls.org/`

http://www.springer.com/978-1-4419-5595-1

Remote Instrumentation and Virtual Laboratories
Service Architecture and Networking
Davoli, F.; Meyer, N.; Pugliese, R.; Zappatore, S. (Eds.)
2010, XXVI, 519 p., Hardcover
ISBN: 978-1-4419-5595-1