The many faces of the integration of instruments and the grid

F. Lelli*, E. Frizziero, M. Gulmini, G. Maron, S. Orlando, A. Petrucci and S. Squizzato

Italian National Institute of Nuclear Physics (INFN)
Viale dell'Universita, 2 35020 Legnaro, Padova, Italy

Dipartimento di Informatica
Universita Ca Foscari di Venezia
Via Torino, 155 30172 Venezia Mestre, Italy
E-mail: francesco.lelli@lnl.infn.it
E-mail: frizziero@lnl.infn.it
E-mail: gulmini@lnl.infn.it
E-mail: maron@lnl.infn.it
E-mail: orlando@dsi.unive.it
E-mail: petrucci@lnl.infn.it
E-mail: squizzato@lnl.infn.it
*Corresponding author

Abstract: 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 organisations can operate together in order to achieve common goals. However, concrete use cases demand a closer interaction between various types of instruments accessible from the grid on the one hand and the classical grid infrastructure, typically composed of Computing and Storage Elements, on the other. We cope with this open problem by proposing and realising the first release of the Instrument Element (IE), a new grid component that provides the computational/data grid with an abstraction of real instruments, and grid users with a more interactive interface to control them. In this paper we discuss in detail the implemented software architecture for this new component and we present concrete use cases where the IE has been successfully integrated.

Keywords: grid; instrument; Instrument Element; IE.

Reference to this paper should be made as follows: Lelli, F., Frizziero, E., Gulmini, M., Maron, G., Orlando, S., Petrucci, A. and Squizzato, S. (2007) 'The many faces of the integration of instruments and the grid', *Int. J. Web and Grid Services*, Vol. 3, No. 3, pp.239–266.

Biographical notes: Francesco Lelli is a taking postdoctorate studies, at the INFN. In the past three years he has been taking PhD studies in Computer Science at the University of Venice in parallel with an Associate Research position at INFN. He has published over 20 papers in international journals and conferences on subjects such as high performance computing, grid, web service, peer to peer, and artificial intelligence. He received a Master's degree in Computer Engineering from the University of Pisa in 2003 and is qualified to practise the profession of engineer. At that time he was Instructor of the

Cisco Networking Academy. He is currently involved in high-energy and nuclear physics experiments such as CMS at CERN and Advanced GAmma Tracking Array (AGaTA) and in the European project GridCC.

Eric Frizziero is a Software Engineer at the Italian National Institute of Nuclear Physics (INFN) – National Laboratories of Legnaro, Italy. He obtained his Master's in Computing Engineering in 2000 from the University of Padua and is qualified to practise the profession of engineer. After some years of experience in the IT industry, he is currently involved in the software designing and development of the Grid Enabled Remote Instrumentation with Distributed Control and Computation (GRIDCC) Project. He is also collaborating with the CERN CMS team to develop the software to control and monitor the CMS data acquisition system.

Michele Gulmini is a Technologist at the National Laboratories of Legnaro – INFN, Italy. He obtained his Master's in Computer Science from the University of Bologna, Italy, in 1996. He joined the Information and Electronics Service in Legnaro in 1996. Since 2002 he has been a full-time Project Associate in the CMS Data Acquisition team at CERN. He has extensive experience in the development of data acquisition, systems for nuclear and high-energy physics experiments. He is currently working in the design and development of the Run Control and Monitor System of the CMS experiment, and in the EU-funded GRIDCC project.

Gaetano Maron is a Senior Technologist at the National Laboratories of Legnaro – INFN. He graduated in Physics from the Padova University in 1980. He joined INFN in 1984 after four years' experience as a Software Developer at a leading Italian industrial automation company. He has been a Visiting Scientist at CERN since 1986. He is currently a team leader of the 'Information and Electronics Technologies Service' at the National Laboratory of Legnaro and Project Manager of the GRIDCC project, an FP6-funded EU project. He has extensive experience in designing and implementing data acquisition and online systems in physics experiments.

Salvatore Orlando is an Associate Professor at the Department of Computer Science, Ca' Foscari University of Venice. In 1985 he received a Laurea degree in Computer Science, cum laude, from the University of Pisa, and a PhD in Computer Science from the same university in 1991. Then he became a Postdoctorate Fellow of the HP Laboratories and the University of Pisa. In 1994 he joined the Ca' Foscari University of Venice as an Assistant Professor, and became an Associate Professor in 2000. He has published over 70 papers in international journals and conferences on several subjects particularly on parallel processing and data and web mining, and he has been on the program committees of many international conferences.

Andrea Petrucci is a Software Design Engineer at the National Laboratories of Legnaro – INFN, Italy. He received the Italian Laurea degree (MS) in Computer Science from the University of Bologna, Italy, in 2001. From 2002 to 2004 he undertook a research fellowship at the Department of Computer Science, University of Bologna. He joined the INFN in 2004 for the GRIDCC, a research project supported by EU funds. Since 2004 he has been an unpaid associate at CERN, and collaborates with the CMS Data Acquisition team. His main research interests are distributed and real-time systems applied to the control and monitoring of high-energy physics experiments.

Silvano Squizzato is a Software Engineer at the National Laboratories of Legnaro – INFN, Italy. He obtained his Master of Science in Electronic Engineering in 1998 and is also qualified to practise the profession of engineer.

Currently, he is involved in the software design and development for the GRIDCC, a research project supported by EU funds. He is also collaborating with the people realising the CMS Data Acquisition System at CERN. Previously, he worked as Solution Architect in the web designing and consulting area and in the telecommunications sector as well.

1 Introduction

Grid computing refers to the coordinated and secured sharing of computing resources among dynamic collections of individuals, institutions and resources (Cittolin *et al.*, 2002; Irving *et al.*, 2004).[1] It involves the distribution of computing resources among geographically separated sites (creating a 'grid' of resources), all of which are configured with specialised software for routing jobs, authenticating users, monitoring resources, and so on.

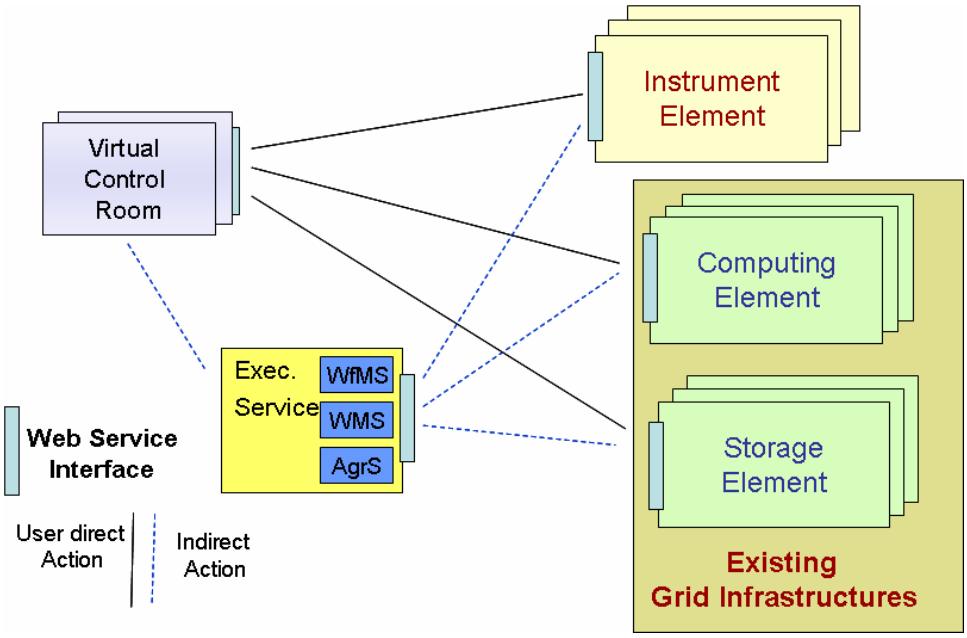
The operative core of the standard computational grid is mainly composed of two important elements: the Computing Element (CE) and the Storage Element (SE). The first one provides the final user with an abstraction of the back-end of the computational system. In other words it is where the execution of an application is performed. This is a very general element that allows different computational nodes, like a single processor or a complex computing cluster, to be seen as a homogeneous set of interfaces. The second one, the SE, provides a storage facility for the input and output of the applications that are executed on a CE. The storage can be as simple as a standard file-system, or a set of databases organised into a more complex structure. While remote control and data collection are part of the initial grid concept, most recent grid developments have been concentrated on the sharing of distributed computational and storage resources.

In this scenario applications that need computational power only have to use these grid elements in order to access an unlimited amount of computational power and disk storage. Unfortunately, as explained later in this section, concrete use cases require a strong interaction between the instrumentation and the computational grid; in addition, they need to be accessed directly from a remote site in the world. For instance, in the Compact Muon Solenoid (CMS) Data Acquisition (DAQ) (Cittolin *et al.*, 2002) system, where the data-taking phase of an experiment occurs, physicists need a single entry point to operate the experiment and to monitor detector status and data quality. In Electrical Utility Networks (or power grids (Irving *et al.*, 2004)), the introduction of very large numbers of 'embedded' power generators, often using renewable energy sources, creates a severe challenge for utility companies. In Geo-Hazards Systems a set of heterogeneous, geographically distributed sensors needs to be remotely accessed and monitored, while the combined instruments' outputs should be automatically analysed using the computational grid.

The Grid Enabled Remote Instrumentation with Distributed Control and Computation (GridCC) project (Maron *et al.*, 2005),[2–3] launched in September 2004 by the European Union, addresses these issues. The goal of GridCC is to exploit grid opportunities for the secure and collaborative work of distributed teams, in order to remotely operate and monitor scientific equipment using the grid's massive memory and computing resources for storing and processing data generated by this kind of equipment.

Our idea is to implement a software component that can satisfy all the above-mentioned requirements as a new grid component: the Instrument Element (IE). It consists of a coherent collection of services that provides all the functionalities to configure and control the physical instrument behind the IE and the interactions with the rest of the grid. Figure 1 gives an idea of the relationship between the IE and its users and between the IE and other grid components.

Figure 1 Interaction between the IE and other grid components



In order to achieve a fast and high level of interaction with the IE component, users can directly access the controlled instrument using a remote control room or Virtual Control Room (VCR) (Pugliese *et al.*, 2005). Or, as a second possibility, an instrument operation can be part of a complex work flow managed by a grid execution service (Alamri *et al.*, 2006) that allows the IE to access and converse with the computational grid. These ways to control the instruments are not mutually exclusive and can be performed in parallel where needed.

Inside the IE a set of web service interfaces called Virtual Instrument Grid Service (VIGS) allows the user to remotely access the real instrument, thus plugging the system itself into the grid. The IE can provide facilities for interactive cooperation between computing grid resources and applications that have real-time requirements or need fast interaction with CEs and SEs. Finally, the IE can also be linked to existing instrumentation in order to provide grid interaction and remote control to stand-alone resources.

The term 'instrument' describes a very heterogeneous category of devices. A set of use cases (Cittolin *et al.*, 2002; Irving *et al.*, 2004; Siaterlis *et al.*, 2005; Tham and Buyya, 2005; McMullen *et al.*, 2005)[1, 3–9] have been extensively analysed in order to collect the functional and nonfunctional requirements of this new grid component. From an intuitive point of view the IE should:

- provide uniform access to the physical devices
- allow standard grid access to the instruments
- allow cooperation between different instruments that belong to different Virtual Organisations(VOs).

We need to point out that IE users do not need to be human beings. Other software components must be able to control the instruments.

Figure 2 shows a use case diagram of the system. A user of the IE can have one of three roles:

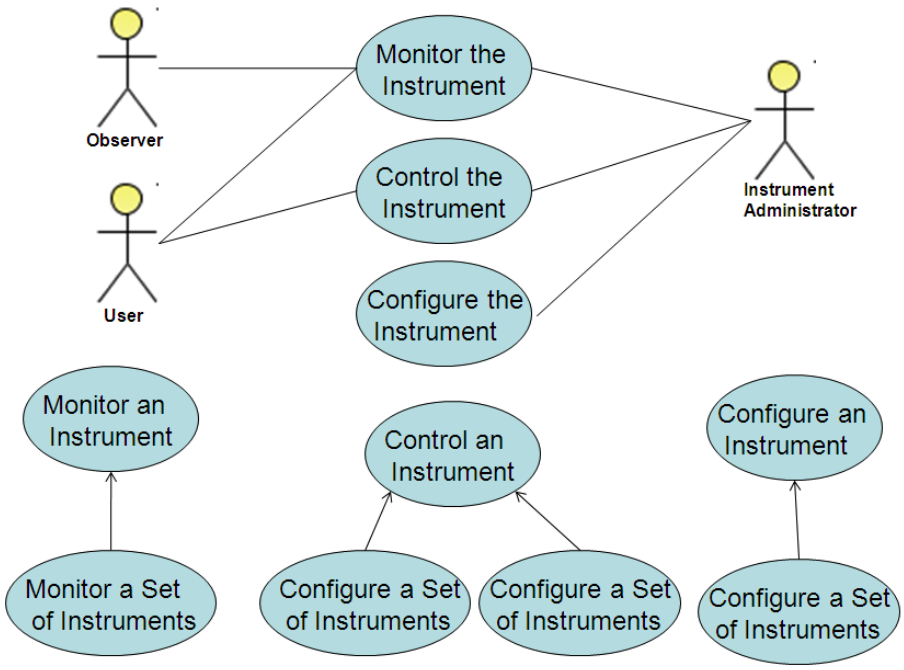
1 Observer – one that has the right to monitor the operation of the instrument.

2 Operator – one that can instantiate an instrument configuration, and control and monitor the instrument.

3 Administrator – one that can create an instrument configuration that can be accessed by the observer user and/or the operator user.

The *monitoring operation* is intended as the possibility to retrieve all information that can be used to determine the operational status and to track the operation of an Instrument System.

Figure 2 IE use cases



The *control operation* is intended as the possibility to act on one or more instruments and move acquired data to and from the computational grid.

The *configure operation* is intended as the possibility to create instruments orchestration (*i.e.*, configuration or partition) that can be used by other actors.

The *instantiation operation* (which is considered a control extension) is intended as the possibility to create instruments, if the physical instruments provide this functionality, or to link to them otherwise.

At any moment there can be multiple observers, multiple administrators, but at most one operator that utilises a particular instrument. A second operator user that tries to control an instrument already controlled by another operator should be treated by the system as an observer user. If the real instrument can be partitioned into subsystems, multiple operators should be able to access different instrument partitions via the same IE.

Close to these functional requirements, a grid of instruments introduces a set of nonfunctional requirements, as summarised in Table 1. Take, for example, the possibility of controlling about 10^4 instruments simultaneously in an interactive way. This introduces the need to provide a scalable system (marked as *Scalability* requirement) equipped with a certain *Quality of Service* (QoS) guarantee, as will be explained in further sections.

Table 1 Nonfunctional requirements

Nonfunctional requirements	*Type*
O(10^4)nodes/instruments must be controlled and monitored	Scalability
The nodes/instrument should be accessed through the web.	Remote access
The nodes/instrument should be accessed in a homogeneous way.	Standardisation
Round-trip time to reach all the nodes must be in the order of human reaction time.	Quality of Service
Online diagnostics and possible error recovery	Autonomic

In a grid of instruments, like the computational one, interoperability (marked as *Standardisation* in Table 1) is a mandatory requirement. Therefore the only possible communication between Grid subcomponents is via Web Services (WS).

For complex systems, in addition, online diagnostics, error recovery and device organisation robustness (marked as *Autonomic*) should be provided. Finally, we need to consider that different instruments use different technologies and protocols in order to be accessed.

The above-mentioned functional and nonfunctional requirements represent the basic building block of this new grid component. In particular we can observe that the control of instruments demands both deep interaction between users and devices, and the adoption of highly interoperable solutions that only SOA-based web/grid services can offer.

The rest of this paper is organised as follows. In Section 2 we present a classification of instruments and a uniform model for the control of each type of instrument. In Section 3 we present the IE system as the way to integrate instruments and the grid. In Section 4 the technological choices for the implementation of the first release of the proposed IE model are discussed. In addition, Section 5 shows some performance tests on the current implementation, while Section 6 we give an overview of the applications that already use the current implementation.

2 Instrument classification

In grid terminology the words 'instrument', 'sensor' and 'device' are used to identify a piece of equipment that needs to be initialised, configured, operated (start, stop, standby, resume, application-specific commands), monitored or reset. Remotely accessed resources play a key role in the grid paradigm. Our classification includes only those devices that can be plugged into a network. In addition, the instruments must be controlled and monitored remotely in order to enable cooperation with other Grid components, such as the CE, SE and other instruments. Normally, such functionalities are not part of simple devices, simply because they request too many resources in terms of computational and/or electrical power.

To summarise, from a grid perspective, a device can belong to one of the following categories:

- dummy instrument
- smart instrument
- smart instrument in an *ad hoc* network.

The first type comprises very simple hardware. The instrument in this category uses Data Acquisition (DAQ) for remote operation; the data is collected with a set of devices that are physically linked together. The devices enable remote network organisation and provide higher-level functionalities. These instruments are typically deployed in remote sites far from the base station. In addition, only the remote (local from a sensor point of view) DAQ collector can be accessed in a remote way, thus acting as a proxy for the sensors that are physically connected to it. Nonpolarisable Petiau electrodes[8] used in the IMAA network are examples of dummy sensors.

'Smart instruments' make up a large category comprising devices that provide all the functionalities needed to be remotely accessed and controlled. High-energy physics and particle physics experiments use these kinds of sensors (Cittolin *et al.*, 2002).[4, 9] For instance, a smart instrument can be an electronic card that acquires data from the concrete detector and dispatches it to one or more machines that perform data aggregation. Typically, these devices are close together and physically connected via a fast communication channel, like a 1 GB Ethernet or an optic fibre. Performances and scalability are an issue and key requirements of these instruments as part of a distributed system. An additional requirement imposed on these devices is autonomicity. The electronic front-end and the information collector (event builder or EVB) of a high-energy experiment is composed of thousands of nodes (usually powerful PCs), thus, basic fault tolerance and dynamic instrument reconfiguration must be part of the device's functionalities.

The smart instrument in an *ad hoc* network can be seen as a specialisation of the previous category. Such devices need not be in close physical proximity, but rather they can remotely communicate through specialised wireless connections. In general, batteries fuel these devices and mobile sensors are part of this category. The challenge is to minimise the energy consumption of the communication channel. Then, we should keep battery consumption uniform between different devices to minimise human interaction with them.

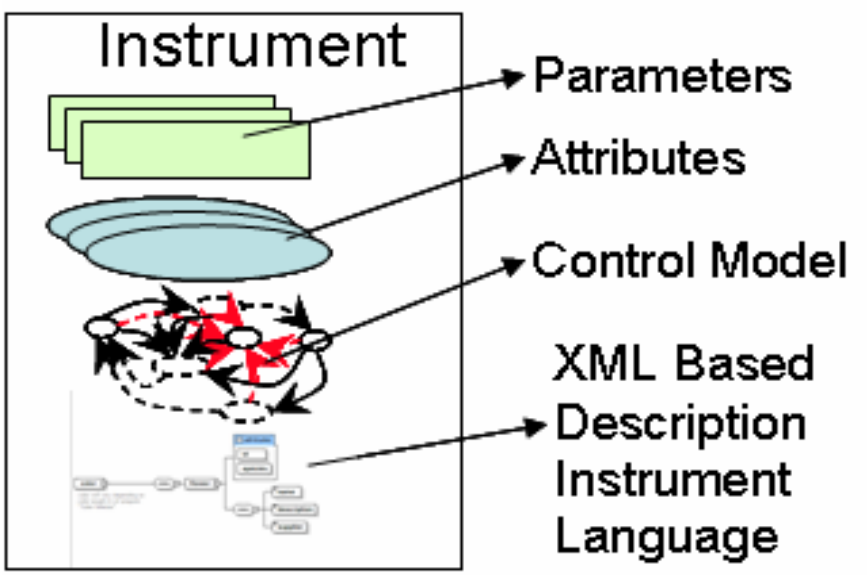
A uniform instrument model

Since the instruments are heterogeneous in nature, one current shortcoming is that the applications that use them (*e.g.*, DAQ codes) must have a complete operational model of the instruments and sensors they work with. This makes maintaining investments in these codes difficult and expensive when the underlying instrument hardware is changed and/or improved. A primary design goal for this section is to externalise the instrument description so that applications can build an operational model 'on the fly'. This approach makes it possible to preserve investments in DAQ codes as instrument hardware evolves and to allow the same code to be used with several similar types of instruments.

The presented instrumentation model is used in order to meet the functional requirements. It can be used for each device independent of the category to which it belongs. In the following sections, we will better understand why the implementation of this model is really dependent on the instrument types.

Figure 3 shows our instrument abstraction model.

Figure 3 Abstraction model of a generic instrument



We can consider a generic device as a collection of parameters, attributes and a control model, plus an optional description language. The more detailed parameters are variables on which the instrument depends, like range or precision values of a measure, while attributes refer to the effective object that the instrument is measuring.

The main difference between parameters and attributes is that typically the first are accessed in a polling mode, while for certain types of attributes, like a cam streaming, a publish/subscribe or a stream access method is more appropriate. Therefore, both push and pull access ways must be supported for attributes.

The control model represents the list of commands that the instrument can support. This list can be expressed using a state machine model, a Petri Net, *etc.* We have to note that in this abstraction, the term 'command' refers to both the actions that change the

instrument status and those that do not. Parameters, attributes and the control model can give the possibility of controlling every possible instrument or sensor, but what is still lacking is the possibility to allow the device to describe itself, giving the user the possibility to understand what exactly it can do with this instrument. In order to achieve this goal, an XML-based description language is also part of our instrument abstraction. Languages like the SensorML[1] or OWL-DL (Smith *et al.*, 2003) can be use to describe the semantics of the particular instrument.

The presented abstraction provides a uniform layer across different devices and can be used as a building block for the control of complex systems such as the CMS experiment (Cittolin *et al.*, 2002), which is composed of about $2^* 10^7$ hardware components and about 10^4 machines dedicated to online event processing. In order to simplify the control of these systems, instruments can be logically (or physically) grouped into hierarchies, from which data can be aggregated or for which control commands affect multiple sensors or actuators. What is needed in this case is an instrument aggregation model, like the one that we will present in the next section, which is capable of controlling all these devices in a congruent way.

3 Instrument aggregation model

The integration of a single instrument into the grid is a relatively simple task. Problems arise when millions of different instruments need to interoperate with each other in order to achieve common goals, giving the external users a well-defined entry point. In order to simplify this task, some services could be created around the instrument to allow a uniform interaction, by giving the illusion of controlling a single device.

We define the term 'Instrument Element' (IE) as a set of services that provides the needed interface and implementation that enables the remote control and monitoring of physical instruments. The IE needs to be really flexible; in the simplest scenario this abstraction can represent a simple geospatial sensor or an FPGA card that performs a specific function, while in a more complex network of sensors it can be used as a bridge between the sensors and the computational grid. Finally, the IE can be part of the device instrumentation, permitting the organisation of the instrument into a network that allows grid interaction.

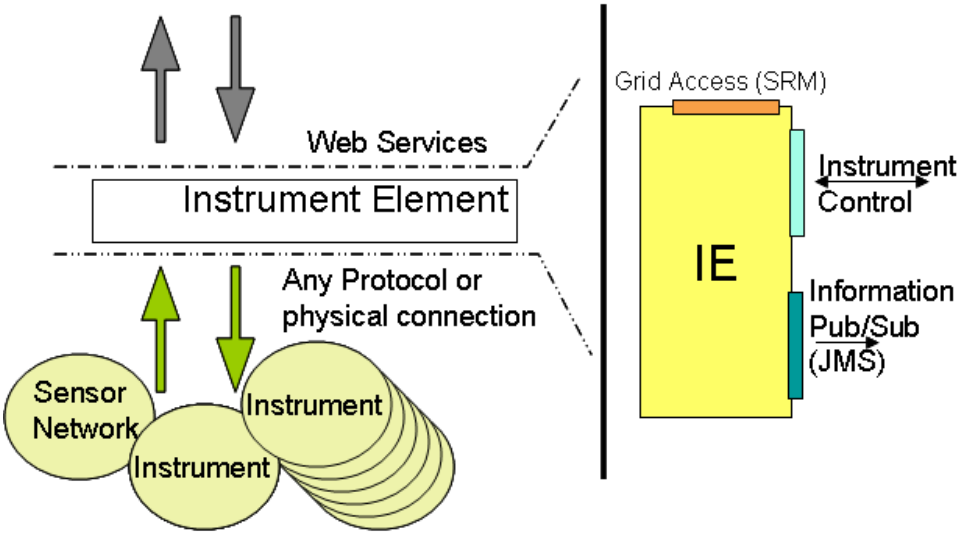
Unlike the CE and the SE, this grid component is accessed using interactive computational job execution and requires a close interaction with the users sitting in the VCR (Pugliese *et al.*, 2005).

If we see the IE as a black box that allows the interaction of instruments in a uniform way, we can identify three different communication channels: firstly, a uniform interface that allows the control of the different system devices in a uniform and coherent way; secondly, an output channel that allows a fast instrument cooperation that permits the reception of asynchronous data and monitors information that came from the instrument attributes; and finally, a set of services that allows the interaction between the instrument and the standard grid system.

Many similarities can be seen in the black box model presented in Figure 4 and in the one discussed in Section 2.1. The main difference is that the IE represents a collection of instruments that can work together or simply belong to the same organisation, so an index service that addresses them is mandatory. Finally, we can note that the IE itself is consistent with the model presented in Section 2.1; therefore an IE can be part of other

IEs. In addition, the IE also acts as a protocol adapter, providing a uniform way to control heterogeneous devices. We believe that web services are an excellent choice when there is a need to provide a common language to the cross-domain collaboration and to hide the internal implementation details of accessing specific instruments. Standards like SWE,[1] JMX (McManus and Sun Microsystems, Inc., 2003), or IVI[6] and P2P Systems like JINI[10] and Freenet (Clarke *et al.*, 2002),[11] have been analysed in order to ensure that the front-end (the Virtual Instrument Grid Services or VIGS) final methods are really capable of providing a generic instrument virtualisation.

Figure 4 The IE abstraction



A final remark: as already introduced at the beginning of this paper, the control of instruments demands deep interaction between users and devices. Consider that when the access is performed via internet using WS, the remote invocation time becomes critical in order to understand if a service can be controlled properly or the delays introduced by the wire are unacceptable.

To summarise the mentioned requirements, we can define two different types of access with different QoS:

1 Strict (hard) guarantees – the response to a requested service is reliable; in this case the availability of an Agreement Service (Maron *et al.*, 2005) that performs advanced reservation and can negotiate 'interaction times' with a component is necessary.

2 Loose (soft) guarantees – the response to a requested service is unreliable. Therefore QoS is provided on top in a best-effort infrastructure. In this case a prediction method based on a statistical approach must be provided.

Techniques that allow interactive web service improvement and prediction of a remote method execution time that can be used in this particular context have been presented in Lelli *et al.* (2006a) and Lelli *et al.* (2006b).

What follows is a description of the main building blocks and a detailed description of the most important ones.

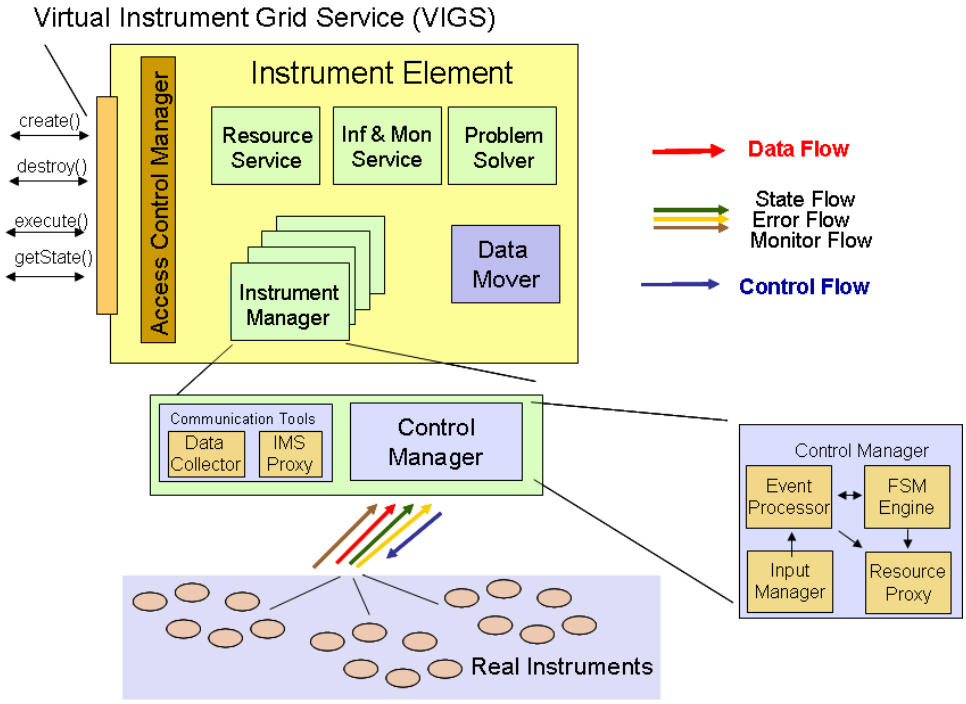
3.1 Instrument aggregation building blocks

This section describes the main IE building blocks, presenting the set of additional services that can simplify the access and the control of instruments. We present the mentioned services as centralised components and in Section 3.5 we discuss where these services should be implemented in a centralised or in a distributed way.

Figure 5 represents the detailed IE architecture. What follows is a short description of each subsystem component.

- *Instrument Managers (IM)* – the instrument managers are the parts of the IE that perform the actual communication with the instruments. They act as protocol adapters that implement the instrument-specific protocols for accessing the IE's functions. One IM can control more than one single device and it is coherent with the model presented in Section 2.1. In other words, it can be seen by other IMs as an instrument, allowing a hierarchic partition of the controlled devices where the complexity of the system requires such a control structure.
- *Resource Service (RS)* – this service organises in groups all the resources that belong to the system, in order to facilitate the access. In this context a resource can be any hardware or software component that can be managed directly or indirectly through the network.
- *Information and Monitor Service (IMS)* – this service disseminates monitoring data to the interested partners, giving a single access point to all the information produced by the instruments.
- *Problem Solver (PS)* – it has the main task of collecting alarms, errors, warnings and messages, which in our models are instrument attributes and parameters, in order to identify error conditions (Kalganova *et al.*, 2006). We have to note that an error recovery can be part of the IM control logic, but while this component is mainly involved in the online recovery, the PS can act offline to try to discover unknown rules.
- *Access Control Manager (ACM)* – it is responsible for checking user credentials and deciding whether an external request should be processed by the IE (Maron *et al.*, 2005).
- *Data Mover (DM)* – since we cannot assume that instruments are complex devices, they need an external service in order to deal with the classical computational grid. This component provides this functionality in a centralised way. In a fully distributed scenario like Sensor Network, a decentralised approach could be more appropriate; therefore part of this functionality could be part of the IM component. This service provides the SRM[12–13] interface to any external storage (SE) or processing elements (CE). It finds the 'best' mechanism, such as GridFTP (Allcock *et al.*, 2005; Silva, 2003) or other transport protocols, to move a file from one storage resource to another. For more demanding applications, grid standards could be inappropriate due to the high bandwidth requested, and a streaming output and/or an MPI interface that allows push subscription capability of the instrument attributes might be needed.

Figure 5 The IE architecture



3.2 *The Instrument Manager (IM) abstraction*

As already mentioned, the IM is the component that deals with the instrument or the instrumentation of the physical device. Each IM is responsible for performing the actual communication with the controlled set of instruments. It acts as a protocol adapter that implements the instrument-specific protocols for accessing their functions. Considering that instruments are heterogeneous in nature, this component is also equipped with a plug-in/driver-based system, which allows an easy reuse of code by changing just the communication library that implements the device-dependent communication protocols. For simple devices that require a minimal control structure, no additional code must be written; at the same time, in multiple and complex control devices this component provides all the basic building blocks that allow the customisation of the control system. Finally, the IM conforms to the model presented in Section 2.1; therefore it can be seen by other devices as an instrument itself, allowing instrument aggregation and hierarchic organisation, thus achieving the goal of breaking down the complexity of the system or organising the instrument in groups.

In addition, the IM can allow the cooperation with other devices; this is a key feature when working in a grid of instruments. Instruments in the ‘smart sensor’ category need to exchange data without loss, as fast as they are being generated. The data collector devices, or some other dedicated set of instruments, process the data (filter and aggregate it) and move it to a final location. Afterwards the data, via the data mover, is stored in a repository or in a SE for future offline analysis.

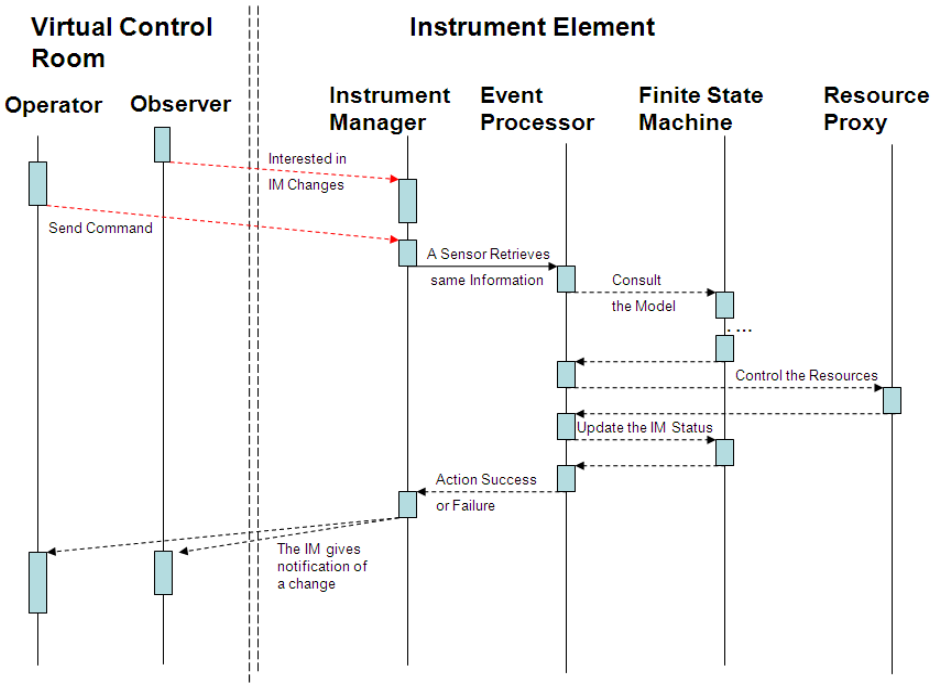
The main components of each IM are as follows:

- The *Communication Tools* can be used by every single subcomponent in order to publish/receive information such as logs, errors, states, configuration, *etc.*, and to receive messages coming from other components. This service also acts as a proxy for the higher-level Data Mover service. If data produced by the set of controlled instruments arrives at a large rate, this low-level service makes the movement of such data more manageable. Finally, these tools represent the instrument front-end of the Information and Monitoring System.
- The *Control Manager* is the component that actually controls the instrument and/or the instrumentation. The typical use case of this subsystem is to receive inputs from the users that are controlling the devices. It can also receive inputs like states or errors that come from the physical controlled instrument. So it can react in an autonomous way to unexpected behaviour of the controlled resources by allowing automatic recovering procedures to be started. This autonomic action becomes critical if the IM controls a large set of instruments, while for simpler devices such functionality becomes less important compared to the need to have a plug-in system that integrates the device drivers.
- The *Input Manager* waits for an external input and presents it to the Event Processor. Inputs can come from users, other instrument managers, or from the devices themselves in the case of smart instruments. In this last case, considering that each instrument has a different way of communicating, a driver component must be provided and loaded inside the particular IM.
- The *Resource Proxy* represents the IM instrument front-end. The control library used in order to exchange messages with the physical device must be plugged inside; this is the minimal customisation action that must be performed in order to plug a generic device into the grid.
- In the *Event Processor* component, the command will be elaborated in order to control the instrument properly. By default, events just trigger the proper action in the proper resource proxy. At the same time, using the plug-in system, this component can be used in order to provide an aggregate and more complex control that allows the possibility to plug inside every possible algorithm: expert systems, fuzzy logic, custom if-then-else, Neural Networks, *etc.* As a final remark, this component represents the basic infrastructure of the Problem Solver, allowing recovery action and/or fault-tolerant procedure, in case of subsystem failure.
- The *Finite State Machine Engine* is specifically designed to simplify the Event Processor algorithms, providing simple call-back mechanisms when writing the action that must be performed according to a particular triggered transition. It also provides the possibility to perform introspection by external users that want to control the particular IM.

Figure 6 shows in detail the interactions occurring in the IM subcomponent. The dotted lines are optional actions that can be performed or not on the basis of the particular received input, while the other lines are mandatory. In addition, the first two lines are actions triggered by the users in the VCR. From the temporal point of view, events coming from user commands or instrument messages are received from the input manager and the information is processed in the Event Processor submodule. Such

a submodule, on the basis of the received input, can decide to perform a state transition according to its Finite State Machine (FSM), and/or control the real instruments via the Resource Proxy module. Once the action is successfully or unsuccessfully performed, a notification is sent back to the users.

Figure 6 VCR-IE interactions



3.3 *Resource Service (RS)*

The complexity of the information managed by the RS is really instrument dependent and ranges from a practically fixed configuration to a configuration and orchestration of thousands of nodes (Cittolin *et al.*, 2002; Irving *et al.*, 2004).[1, 9] In any case, it provides a uniform way and a single point of access to the information related to a particular instrument from external users.

If a system, like a sensor network, allows the possible use of a subset of devices, it also manages this partitioning. In addition, this service can act as a super peer in dynamic instrument networks, where simple devices can appear and disappear. Finally, this service can permit a reservation of the system, allowing authorised users the possibility to bookmark resources.

Figure 7 classifies the information that has to be retrieved from the Resource Service for every instrument.

From a semantic point of view, we can divide the information into three different categories:

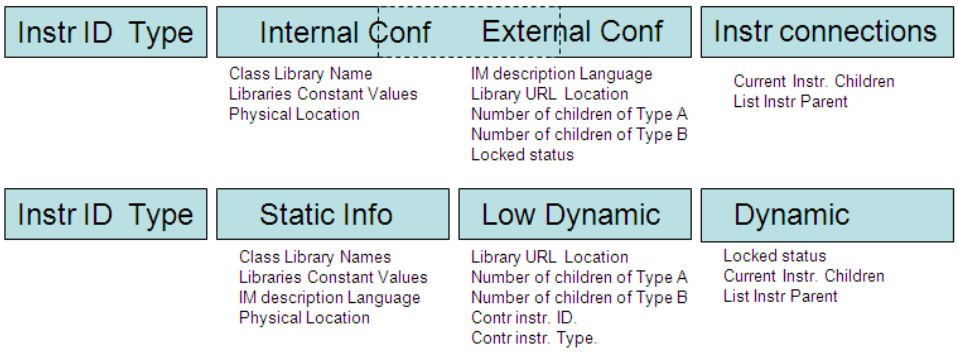
1 Information such as physical locations of configuration file or driver type, *etc.*, which is only internal to the IE and is needed in order to ensure the correct instrument instantiation.

2 Information that can be modified at runtime by the users and that could change the global behaviour. The numbers and types of instruments that should be used to perform particular aggregate actions are typical of the information that belongs to this category.

3 Information that identifies the instrument topology, *i.e.*, both potential and actually performed intra-instrument connections.

The same information could be categorised from the dynamicity point of view: (a) Static information refers to data that will be defined at deployment time and will never change in the future. As its opposite, (b) Dynamic information consists of data that can change in an automatic way, without user intervention. In the middle, we have (c) Low Dynamic information, which corresponds, for example, to adjustments performed by the users at runtime.

Figure 7 Classification of the information contained in the resource service



We can note that most of the information is close to the instrument, and belongs to a particular instance. In addition, considering a set of instruments as one single device, static information introduces rigidity to the system while low dynamic information introduces complexity in the global system usage. As we will point out later in Section 3.5, complex static systems configuration, which typically are simpler to implement, could be the solution in use cases, where all instruments need to be in a consistent state in order to produce a coherent output (Cittolin *et al.*, 2002).[9] Unfortunately, this solution remarkably increases the configuration problem, providing a fixed structure that, in the case of a subsystems fault, must be manually reconfigured by the users. In highly dynamic systems like the one described in Irving *et al.* (2004) and Note 1, this solution is simply unusable because the introduction of a new node in the system triggers a total reconfiguration.

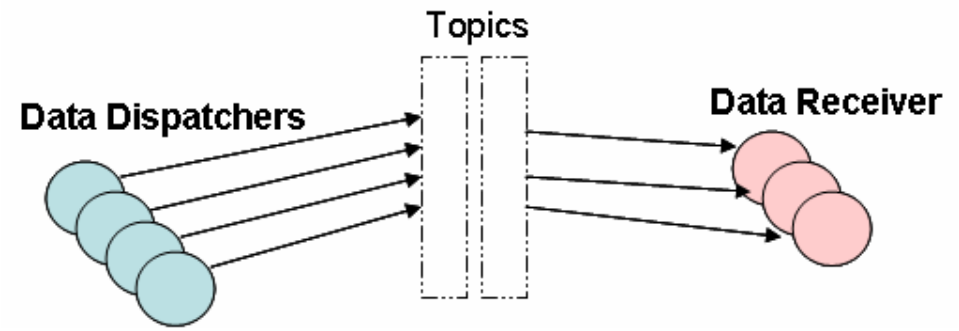
3.4 Information and Monitor Service (IMS)

Instruments and Instrument Managers dispatch data and information that can be monitored via the IMS. We have to point out that more demanding applications (Cittolin *et al.*, 2002) request that this service handle about 10^5 messages per second. Therefore this system cannot be implemented in a centralised way. In addition, standard message formats like SOAP cannot be used in these components due to the overhead

that this serialisation introduces (Lelli *et al.*, 2006b). Finally, taking into account that instruments are independent and weakly coupled with each other, an information and monitor system should preserve the mentioned properties.

As a result, architectures such as the one presented in Note 14, Byro *et al.* (2004) and Brookshier *et al.* (2002) appear to be the most appropriate for this task. In these systems, peers publish information on a given topic for subscribers that have previously indicated their interest (see Figure 8).

Figure 8 Publish/Subscribe architecture



Using this particular approach, we allow peers in the network to appear and disappear dynamically, preserving a certain robustness with respect to system faults (Eugster *et al.*, 2003).

Once the connection to the particular information channel has been established, publishers can start sending data. Considering that high throughput is a must in these cases, a bridged/relay (Brookshier *et al.*, 2002)[15] solution appears to be the only way to preserve this characteristic.

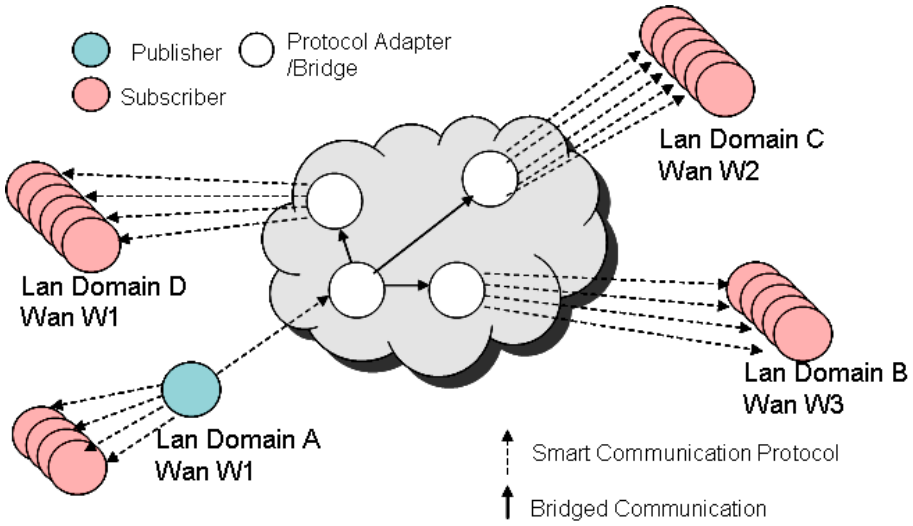
Figure 9 tries to explain what was mentioned before in a simplified scenario, where one publisher, which belongs to a particular private network, tries to send information to peers disseminated throughout the world. Using a smart protocol, the publisher can reach peers located in the same LAN and/or communicate to the bridge component that is connected with other bridges via standard communication protocols, like TCP and HTTP. Bridges, once they have received the messages, firstly convert them into a more suitable format, and secondly send them to peers that are in the same LAN domain. Multicast protocols can be used to reduce the number of messages that publishers need in order to continuously improve the performance of the system (Carmeli *et al.*, 2004).

3.5 Static versus dynamic aggregation models

The term ‘Instrument’ describes a very heterogeneous category of devices. Refer to Cittolin *et al.* (2002), Irving *et al.* (2004), Siaterlis *et al.* (2005), Tham and Buyya (2005), McMullen *et al.* (2005) and Notes 1, 3–9 as a large set of examples. Even if we believe that a uniform and coherent set of services can facilitate their aggregation and the interoperability across different organisations, the approach to the implementation could be really different. In Cittolin *et al.* (2002), for instance, the CMS data acquisition phase can start only if all the instruments of the system are of the same status, while in Irving *et al.* (2004), McMullen *et al.* (2005) and Note 1, instruments can dynamically join

the system. In the previously mentioned cases we cannot assume that the index of all instruments, which is the base abstraction of the Resource Service, is static. The IM behaviour needs to dynamically adapt itself to the dynamic, existing instrument structure. This particular functionality is typical of P2P (Taylor, 2004) systems, wherein the network can dynamically adapt to peer changes. Incidentally, a single and relayable entry point of this information is mandatory if we want to provide a set of instruments as a service for the computational grid.

Figure 9 IMS architecture



Considering the categorisation of the information that we defined in Section 3.3, we can note that static information and some of the low dynamics belong to a particular instrument instance, while instrument topology information must be a sheered attribute between devices in order to avoid collisions, thus organising the instruments in the proper way. In a typical discovery based on P2P, a peer announces itself to the network, giving other peers the possibility to perform query and exchange of data.

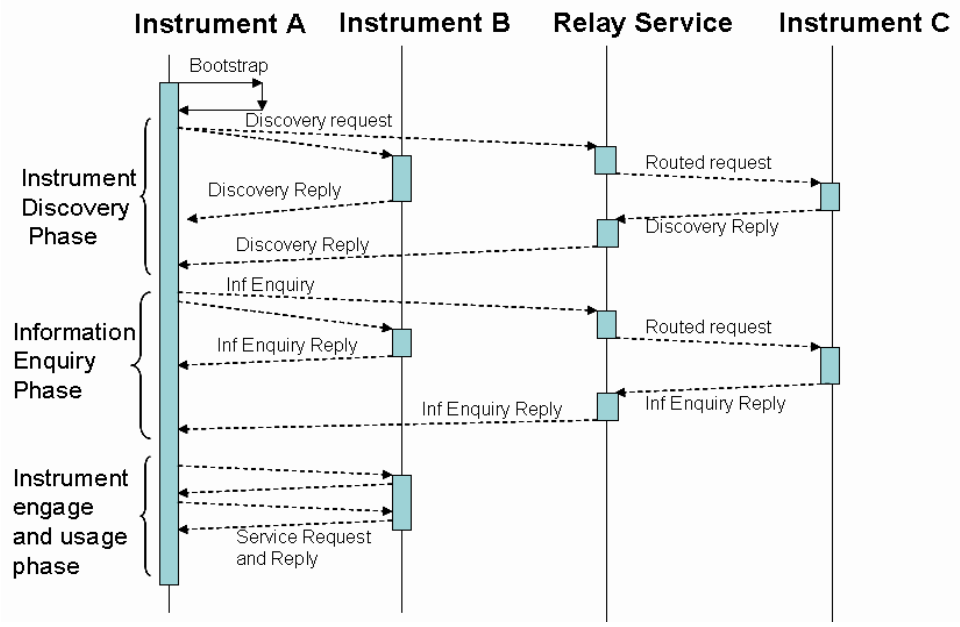
In this scenario, instruments can dynamically engage other existing instruments, performing a system lookup and allowing the dynamic determination of the peers' topology.

This approach distributes the information to the instruments, thus breaking the global configuration into several parts that dynamically change during the system usage.

Figure 10 explains the dynamic joining of an instrument into the system. After a bootstrap, the instrument sends a discovery request to other peers and Relays forward this request to unreachable devices. Instruments reply to this request by announcing their presence in the network and then the new instrument enquires of the others in order to discover what type of device they are. Once the instrument finds the needed resources, it engages and uses them.

If an instrument disappears from the system, other devices can repeat the discovery/information enquiry phases in order to try and find the needed resources. In addition, this operation can also be repeated in case of failure in order to detect the recovery of the needed subcomponent, allowing an autonomic behaviour.

Figure 10 Instrument discovery interaction diagram



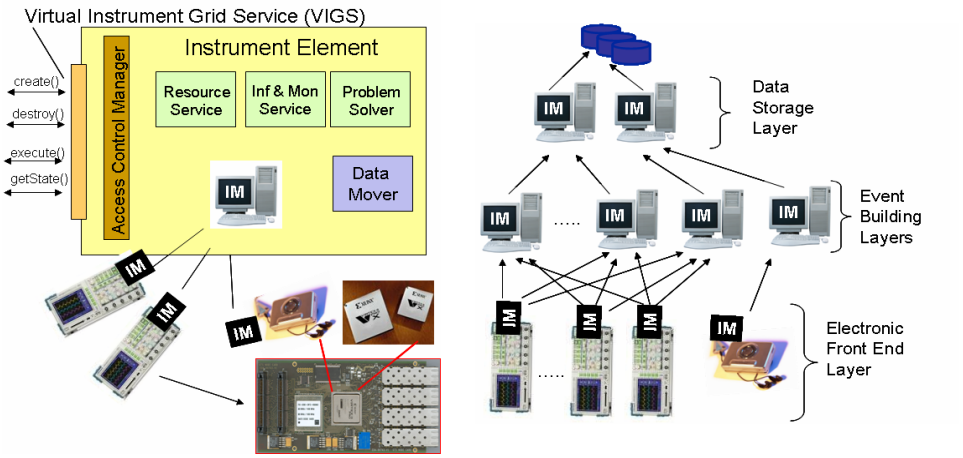
In this P2P scenario, the information is no longer centralised but is distributed in the system. Therefore this approach complicates monitor functionalities that also need a discovery system in order to detect the actual instrument topology. In other words, the Resource Service end point must periodically repeat the instrument discovery and information enquiry phases, as with all the system devices, in order to detect the status of the entire system. Alternately (or in parallel), instruments can periodically send an advertise message in order to inform interested peers of their status.

3.6 Embedded devices

In this context we refer to an embedded device as one with limited computational and network capacity. In order to reduce the system requirements, IM with very limited functionality can be directly installed into the electronic front-end. This enables other IMs to remotely contact this device using a low-level communication channel, allowing a bridged communication that can enable a more elaborate interaction.

For complex systems such as in Cittolin *et al.* (2002),[9] embedded applications that run in *ad hoc* electronic cards can also use the communication tools to dispatch the data just acquired to the thousands of nodes that constitute the event builder layer. Each Event Builder (EVB) machine, which can be seen as an instrument, operates as a subscriber to the messages (data) sent by the publishers – the devices on the cards. The selection of the required devices is enabled by associating a topic with a device. The incoming data messages are then sampled using the publish/subscribe selector capabilities (Brookshier *et al.*, 2002).[14] The data is further processed and an event is generated and sent to a subset of machines that perform an additional intermediate step of ‘collection and aggregation’. Finally, this data is sent to the last layer of machines, which save the data via instrument managers directly connected to the data mover.

Figure 11 Embedded devices



4 Current implementation

Several technologies have been used in order to implement the first Java-based IE release, such as tomcat + axis as a web service engine, which provides a WS-I compliant[16] software end point. The usage of a quite mature open source software has driven our choices in this field.

The Resource Service uses JDBC[17] connections to MySQL and Oracle DB in order to retrieve the instruments configuration information, while the IE-embedded GUI is based on JSP[18] and AJAX (Mahemoff, 2006) technologies.

Finally, the standard Apache Logging System,[19] such as log4j, has been adopted in order to acquire runtime information of each single software subcomponent.

To ensure the high throughput needed by the IMS component presented in Section 3.4, we have also used a JMS[14] implementation on top of a high-performance Reliable Multicast Messaging (RMM) layer (Carmeli *et al.*, 2004). RMM allows hosts to reliably exchange data messages over the standard IP multicast network. It exploits the IP multicast infrastructure to ensure scalable resource conservation and timely information distribution, with reliability and traffic control added on top of the standard multicast networking. The RMM-JMS extension is a very efficient Java implementation of the JMS standard using RMM services.

RMM-JMS supports:

- multicast transport for pub/sub messaging – supporting the JMS topic-based messaging and API, with matching done at the IP multicast level. The transport is a Nack-based reliable multicast protocol.
- Direct (brokerless) unicast for point-to-point messaging – supporting the JMS Queue-based messaging and API. The transport is the TCP protocol.
- Bridged/Brokered unicast transport for pub/sub messaging.

Section 5 shows some performance tests on the current implementation while Section 6 provides an overview of the current applications that are using the proposed architecture and the related implementation.

5 A benchmark on the current IE implementation

In this section we present some IE benchmark tests. The goal is to figure out the scalability and flexibility of the first release of the IE.

5.1 Command reception performances

The following section describes tests related to the receiving capability of the IE's Control Manager component.

Figure 12 shows which components of the Control Manager are involved in these tests. Two tests have been performed:

- Test 1 stresses the capability of the control manager to answer the clients' requests. Typical clients in this case are the VCR and Execution Services, and also generic clients. Figure 13 reports the results. Around 50 invocations per second have been achieved, each invocation triggering a state transition in the FSM engine.
- Test 2 stresses the capability of the control manager to react to asynchronous messages, as happens when an instrument has to send error and information messages, and even state changes. Figure 14 reports the results obtained by varying the number of instruments sending messages to the Input Manager and then to the Event Processor.

Figure 12 IE invocation benchmark test description

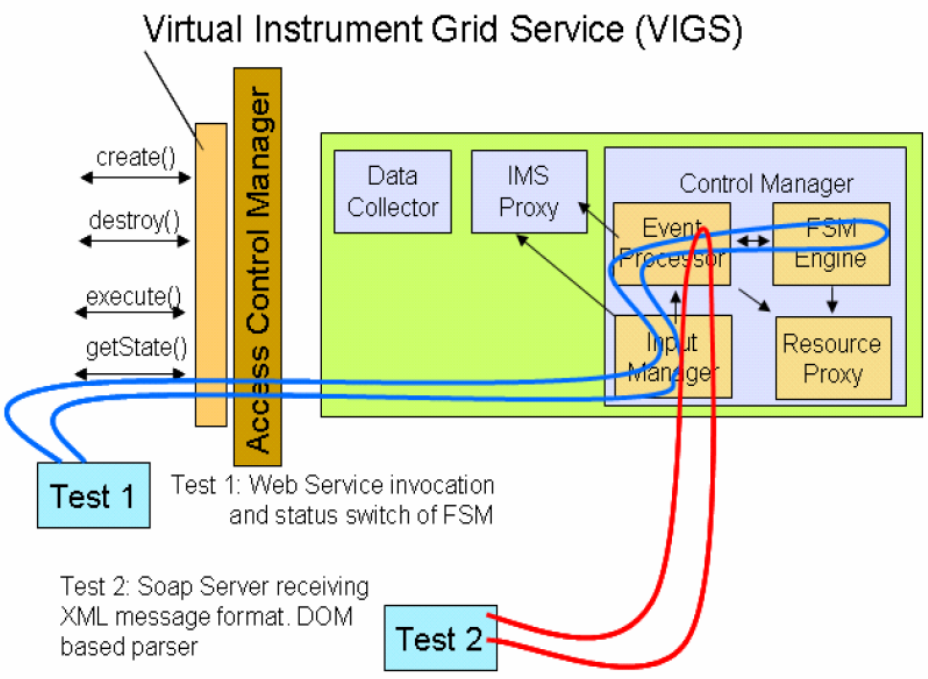


Figure 13 Test 1 results: VCR invocation capability

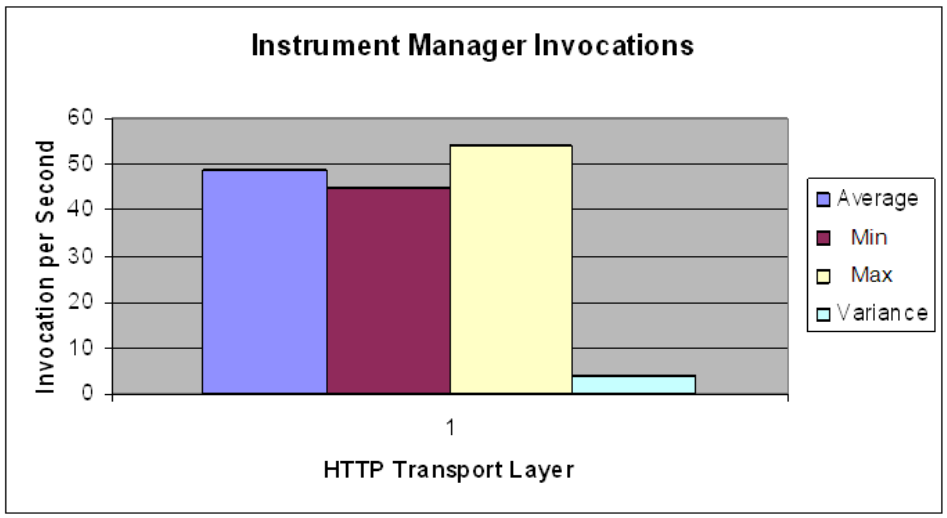
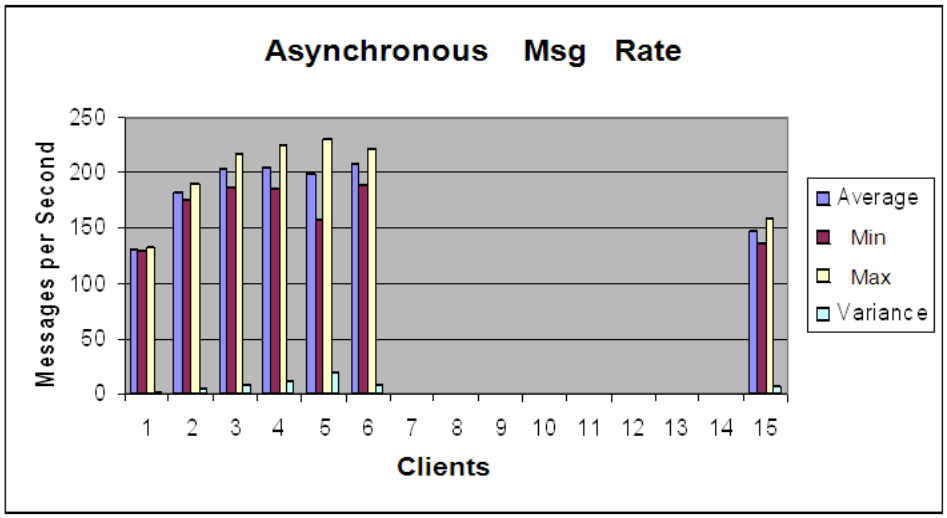


Figure 14 Test 2 results: IE message-handling capability



5.2 Command distribution performances

These tests aim to measure the distribution capability inside an IE when a tree Instrument Manager structure is used to reach the controlled instruments. Figures 15–17 depict the hierarchy that was used.

In Figure 15 a single Instrument Manager was controlling from 10 to 120 instruments exchanging SOAP messages over HTTP. Figure 16 shows a similar case, but with three Instrument Managers running in the same machine and controlling the same number of resources. Finally, in Figure 17, three Instrument Managers are instantiated in three different machines.

Figure 15 Instrument manager hierarchy performance tests: one IM

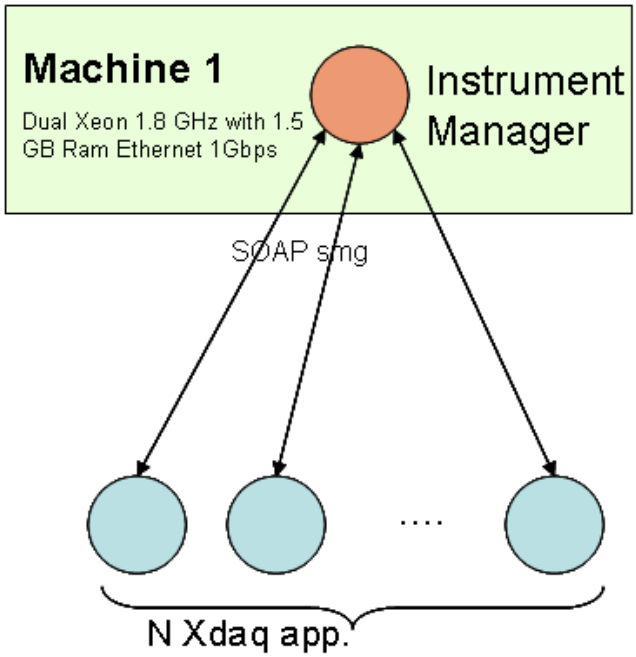


Figure 16 Instrument manager hierarchy performance tests: three IMs

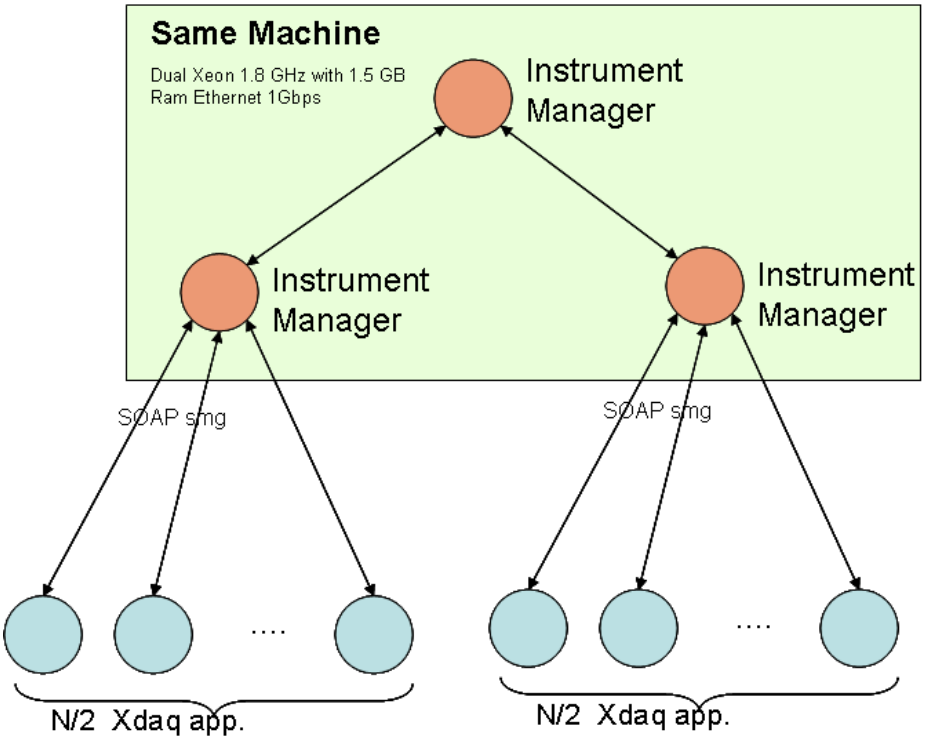
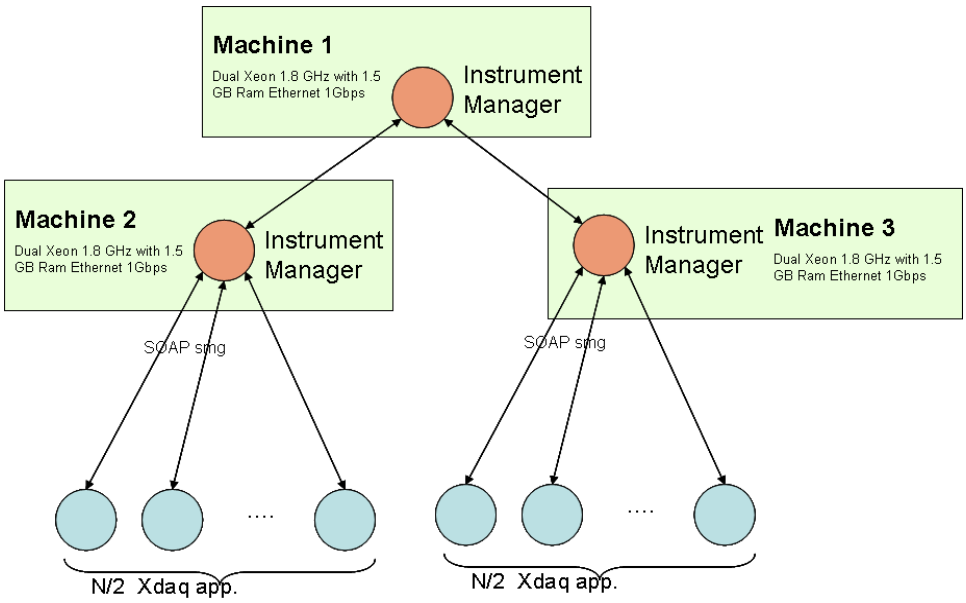


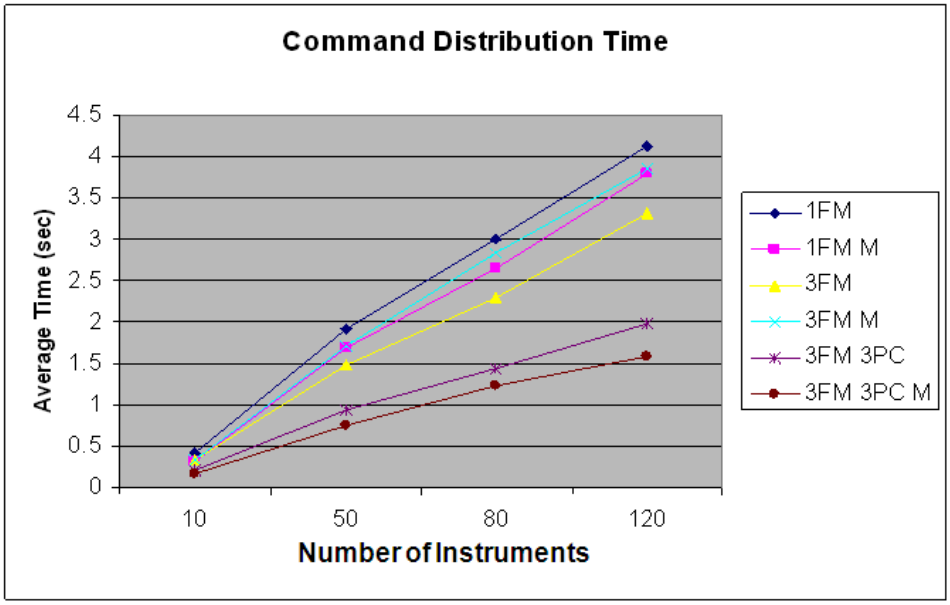
Figure 17 Instrument manager hierarchy performance tests: three IMs in three machines



The graph in Figure 18 shows the time spent to distribute a command by varying the number of controlled instruments for all the three above-mentioned cases using a parallel and a sequential command distribution.

This shows that a distributed configuration of the Instrument Managers can ensure better performance where it is needed.

Figure 18 IE command distribution benchmark test results



6 Applications that are using the current release of the IE

The first IE released is currently used to manage the following use cases.

6.1 The Compact Muon Solenoid experiment

In the Compact Muon Solenoid (CMS) (Cittolin *et al.*, 2002) Data Acquisition (DAQ), the IE is the master controller when the experiment is acquiring data. Approximately $2 * 10^7$ electronic devices need to be accessed and controlled by about 10^4 different machines scattered over a high-bandwidth network. The IE instructs the subpart of this experiment to act according to the specific needs of a data collection session. The main functions of the IE in this case are:

- to control and monitor the entire instrument, ensuring the correct and proper operation of the CMS experiment
- to control and monitor the DAQ system
- to provide user interfaces and allow users to access the system from anywhere in the world.

6.2 The Intrusion Detection System

One of the main challenges in the security management of large high-speed networks is the detection of suspicious anomalies in network traffic patterns due to Distributed Denial of Service (DDoS) attacks or worm propagation (Siaterlis *et al.*, 2005). In the Intrusion Detection System (Siaterlis *et al.*, 2005), anomaly sensors measure various network elements linked to grid-controlled Instrument Managers within a real-time domain. The IE problem solver provides algorithms aimed at fusing the collected knowledge to analyse individual domain-state reports, originated from heterogeneous sensors, to deduce a global view of security incidents and so detect Distributed Denial of Service.

6.3 The power grid

In electrical utility networks (or power grids; Irving *et al.*, 2004), the introduction of a very large number of 'embedded' power generators, often using renewable energy sources, creates a severe challenge for utility companies. In addition, power systems involve many geographically distributed participants: generator owners, transmission network operators, load managers, energy-market makers, supply companies, and so on. GridCC technology allows the generators to participate in a virtual organisation, and consequently to be monitored and scheduled in a cost-effective manner.

6.4 The Synchrotron Radiation Storage Ring

In the Elettra[4] Synchrotron Radiation Storage Ring, the IEs control and monitor many instruments (mainly sensors). The rate of incoming control and monitoring data is different from the previous use cases; it is smaller than the data rate in the CMS scenario, yet, since it resides at a critical point of alerting against major catastrophes, it imposes high requirements on response times for alerts at the human-machine level interface.

6.5 The IMAA sensor networks

Operating since 2003, the Institute of Methodologies for Environmental Analysis (IMAA) has installed a network of sensors devoted to providing real-time information on several meteorology and physical-chemical parameters of soil and subsoil, useful for describing landslide dynamics, and possibly to detect warning conditions in a timely fashion (Perrone *et al.*, 2004; Perrone *et al.*, 2005). Monitoring is done with a multichannel data-logger[7] on the monitored side to collect/aggregate information coming from different sensors, both passive and active.[8] Different sensors are wired to a data-logger while this device is remotely controlled via a wireless network. The IE virtualises the access to the sensors controlling multiple DAQ devices, allowing grid access to the instruments.

6.6 The Advanced Gamma Tracking Array experiment

AGATA,[9] the Advanced Gamma Tracking Array, is a 4π array of segmented coaxial detectors. The design consists of a geodesic tiling of a sphere with 12 regular pentagons and 180 hexagons. Sensors, which are front-end electronic devices, are linked to the detector to dispatch the just-read information to the event-building farm that will merge it into physical events. The IE controls every single subcomponent such as PSA farms, event builders, tracking farms and data servers of this complex distributed application.

6.7 The device farm

A set of telecommunications measurement instruments of various categories (signal generators, signal analysers, channel emulators), interconnected via programmable switching matrices, are accessed and remotely controlled as WS, and they are integrated in the IE architectural framework. The concept of 'virtual instrument' allows the remote control and visualisation of the results of the measurements carried out by an instrument (Vollono and Zinicola, 2005). In most cases, it also allows the execution of the functions of more instruments found in the laboratory.

6.8 Meteorology systems

The meteorological application of GridCC architecture consists of the Skiron/Eta weather forecasting system (Kallos, 1997). It aims to predict and analyse hazardous weather events. A HELLASGRID cluster at IASA is currently used as a test-bed for the model configuration and execution under the GridCC environment. Further developments of the model algorithms and operation scripts have been prepared in order to optimise the model utilisation through the IE components. The system operates daily in deterministic mode. The stochastic mode is under development, although a beta version is currently being tested.

The normal operation of the computational model runs in a deterministic fashion, covering the entire Mediterranean region. An emergency condition is assumed if a hazardous weather event is detected in the computational domain from the current operational cycle. In this case, the system can be switched to the stochastic mode instead of the normal deterministic execution.

7 Conclusion

The GridCC project integrates instrumentation into traditional computational/storage grids. We cope with this open problem by proposing and realising the first release of the IE, a new grid component that provides the computational/data grid with an abstraction of real instruments, and grid users with a more interactive interface to control them. The main benefits of the proposed solution are:

- A highly modular and flexible solution:
  - a The middleware deployment is organised into pluggable independent components.
  - b The IE architecture is not tied to specific technologies or third-party software.
  - c Multiple and independent I/O interfaces: Commands and Controls directed at Instruments, Fast Data Publishing, Data Movement to/from the grid.
- High scalability:
  - a IE is portable to different platforms.
  - b The target environment ranges from large farms to embedded devices.
- IE middleware aims to be used in production environments:

  Mature middleware has been used, whenever possible, to assure robustness and stability.
- Fine customisation of the IE:

  The IE is easily adaptable to diverse scenarios and needs.
- High level of abstraction in the IE interfaces:
  - a Service Oriented Architecture
  - b Adherence to standards: WS-I-compliant web services.
- EGEE (gLite) compatibility:

  Many grid facilities and tools can be reused, for instance GridFTP or SRM interface for data movement.

Acknowledgement

The GridCC project is supported under EU FP6 contract 511382.

References

Alamri, A., Eid, M. and El Saddik, A. (2006) 'Classification of the state-of-the-art dynamic web services composition techniques', *Int. J. Web and Grid Services.*

Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L. and Tuecke, S. (2005) 'GridFTP: protocol extensions to FTP for the grid', http://www-fp.globus.org/datagrid/gridftp.html (updated January 2002).

Brookshier, D., Govoni, D., Krishnan, N. and Soto, J.C. (2002) *Published JXTA: Java P2P Programming*, Sams Publishing.

Byro, R., Coghlan, B., Cooke, A., Cordenonsi, R., Cornwall, L., Craig, M., Djaoui, A., *et al.* (2004) 'R-GMA: production services for information and monitoring in the grid', *AHM2004*.

Carmeli, B., Gershinsky, G., Harpaz, A., Naaman, N., Nelken, H., Satran, J. and Vortman, P. (2004) 'High throughput reliable message dissemination', *SAC '04: Proceedings of the 2004 ACM Symposium on Applied Computing*, New York, NY: ACM Press, pp.322–327.

Cittolin, S., Varella, W.S.J., Racz, A., Della Negra, M. and Herve, A. (2002) *CMS TDR 6.2 The TriDAS Data Acquisition Project and High-level Trigger CERN/LHCC*, December.

Clarke, I., Miller, S., Hong, T., Sandberg, O. and Wiley, B. (2002) *Protecting Free Expression Online with Freenet*.

Eugster, P.Th., Felber, P.A., Guerraoui, R. and Kermarrec, A. (2003) 'The many faces of publish/subscribe', *ACM Comput. Surv*., Vol. 35, No. 2, pp.114–131.

Irving, M., Taylor, G. and Hobson, P. (2004) 'Plug into grid computing', *IEEE Power & Energy Magazine*, March–April, pp.40–44.

Kalganova, T., Suppharangsan, S., Taylor, R., Alsaif, M. and Lelli, F. (2006) 'Towards development of problem solver in the grid environment with monitoring and control of instrumentation', *International Conference on Intelligent Engineering Systems (INES)*, London, UK, June.

Kallos, G. (1997) 'The regional weather forecasting system SKIRON', *Proceedings of the Symposium on Regional Weather Prediction on Parallel Computer Environments*, October.

Lelli, F., Maron, G. and Orlando, S. (2006a) *Enabling the Web Service Quality of Service*, Istituto Nazionale di Fisica Nucleare Pre-Printing ID:INFN-LNL-212(2006), September.

Lelli, F., Maron, G. and Orlando, S. (2006b) 'Improving the performance of XML based technologies by caching and reusing information', *Proc. of International Conference of Web Services (ICWS06)*, IEEE Computer Society, Vol. 1, September, pp.689–700.

Mahemoff, M. (2006) *Ajax Design Patterns*, O'Reilly Media, 1st ed., 1 June.

Maron, G., Lenis, A., Moralis, S., Grammatikou, M., Karounos, T., Papavassiliou, S., Maglaris, V., *et al.* (2005) 'GridCC architecture design', GridCC Project deliverable www.gridcc.org, May.

McManus, E. and Sun Microsystems, Inc. (2003) *JSR 160: JavaTM Management Extensions (JMX) Remote API 1.0*, October.

McMullen, D.F., Devadithya, T. and Chiu, K. (2005) 'Integrating instruments and sensors into the grid with CIMA web services', *Proceedings of the Third APAC Conference on Advanced Computing, Grid Applications and e-Research (APAC05)*, September.

Perrone, A., Iannuzzi, A., Lapenna, V., Lorenzo, P., Piscitelli, S., Rizzo, E. and Sdao, F. (2004) 'High-resolution electrical imaging of the Varco d'Izzo earthflow (Southern Italy)', *Journal of Applied Geophysics*, Vol. 56, No. 1, pp.17–29.

Perrone, A., Zeni, G., Piscitelli, S., Pepe, A., Loperte, A., Lapenna, V. and Lanari, R. (2005) 'On the joint analysis of SAR interferometry and electrical resistivity tomography surveys for investigating ground deformations: the case-study of Satriano di Lucania (Potenza, Italy). Remote sensing of environment', *Journal of Applied Geophysics*, Vol. 1, pp.486–504.

Pugliese, R., Asnicar, F., Del Cano, L., Chittaro, L., Ranon, R., De Marco, L. and Senerchia, A. (2005) 'Collaborative environments for the GRID: the GRIDCC multipurpose collaborative environment', *Tyrrhenian Workshop*, Sorrento, July, Vol. 1, pp.252–260.

Siaterlis, C., Lenis, A., Moralis, A., Roris, P., Koutepas, G., Androulidakis, G., Chatzigiannakis, V., *et al.* (2005) 'Distributed network monitoring and anomaly detection as a grid application', *HP Openview University Association Plenary Workshop (HP-OVUA) Porto*, July.

Silva, V. (2003) 'Transferring files with GridFTP', April, http://www.128.ibm.com/developerworks/grid/library/gr-ftp/?ca=dgr-lnxw03GridFTP.

Smith, M.K., Welty, C. and McGuinness, D.L. (2003) 'OWL web ontology language guide, W3C', http://www.w3.org/TR/owl-guide/.

Taylor, J. (2004) *From P2P to Web Services and Grids, Peers in a Client/Server World*, Springer, October.

Tham, C.K. and Buyya, R. (2005) 'SensorGrid: integrating sensor networks and grid computing', *Invited Paper in CSI Communications, Special Issue on Grid Computing*, Computer Society of India, July.

Vollono, A. and Zinicola, A. (2005) 'A new prospective in instrumentation interfaces as web services', *Proc. of Tyrrhenian Workshop*, Sorrento, Italy, July.

Notes

1 OGC Sensor Web Enablement, www.opengeospatial.org/functional/page=swe, last visit 1 January 2007.

2 GridCC Project, http://www.gridcc.org/, last visit 1 January 2007.

3 Grid Enabled Remote Instrumentation with Distributed Control and Computation (GridCC) Project Annex I, http://www.gridcc.org/getfile.php?id=1436e, 2005.

4 Synchrotron Radiation Storage Ring Elettra, http://www.elettra.trieste.it/index.php, last visit 1 January 2007.

5 GridCC Use Cases, https://ulisse.elettra.trieste.it/tutos.

6 Interchangeable Virtual Instrument Foundation, http://www.ivifoundation.org/, last visit 1 January 2007.

7 Data acquisition system (Keithley instruments model 2701, with plug-in switching modules model 7702).

8 Non-polarisable petiau electrodes (SDEC electrodes, pb/pbcl2-nacl) soil probes made by http://www.campbellsci.com/sensors: Time Domain Reflectometer (TDR) and a thermometer; atmospheric probes made by http://www.campbellsci.com/sensors: Pluviometer and thermometer.

9 AGATA Advanced Gamma Tracking Array design specification, http://agata.pd.infn.it/Agata-proposal.pdf.

10 Jini Project, http://www.jini.org/, last visit 1 January 2007.

11 The Freenet Project, http://freenet.sourceforge.net/, last visit 1 January 2007.

12 EGEE Middleware Architecture and planning, EGEE Project Deliverable, EGEE-DJRA1.1 -594698-v1.0, Chapter 9. Also available at https://edms.cern.ch/document/594698/, July 2005.

13 SRM: Storage Management Working Group, http://sdm.lbl.gov/srm-wg/, last visit 1 January 2007.

14 JMS Standard API, http://java.sun.com/products/jms/, last visit 1 January 2007.

15 Mantaray Project, http://www.mantamq.org, last visit 1 January 2007.

16 Web Service Interoperability Organization, http://www.ws-i.org/, last visit 1 January 2007.

17 JDBC Standard API, http://java.sun.com/javase/technologies/database.jsp, last visit 1 January 2007.

18 JSP Standard API, http://java.sun.com/products/jsp/, last visit 1 January 2007.

19 Apache Logging Project, http://logging.apache.org/, last visit 1 January 2007.