Interactive Visualization of Grid Monitoring Data on Multiple Client Platforms

Lea Skorin-Kapov[1], Igor Pandžić[2], Maja Matijašević[2], Hrvoje Komerički[2], and Miran Mošmondor[1]

[1] Research and Development Center, Ericsson Nikola Tesla, Krapinska 45, HR-10000 Zagreb, Croatia
{lea.skorin-kapov, miran.mosmondor}@ericsson.com
[2] FER, University of Zagreb, Unska 3, HR-10000 Zagreb, Croatia
{igor.pandzic, maja.matijasevic, hrvoje.komericki}@fer.hr

Abstract. Most current Grid monitoring systems provide a visual user interface. With recent advances in multimedia capabilities in user terminals, there is a strong trend towards interactive, multi-modal and multi-platform visualization. In this paper we describe a multi-platform visualization architecture and a Web based service built upon it, which provides a view of the monitored Grid hierarchy, and the values of selected monitoring parameters for different Grid sites. We demonstrate the application on four platforms: a desktop Personal Computer (PC), a handheld PC, a Java-enabled new-generation mobile phone, and a Wireless Application Protocol (WAP) enabled mobile phone.

1 Introduction

Grid technologies have been described as supporting the sharing and coordinated use of diverse resources in distributed "virtual organizations" [1]. As Grid structure is inherently difficult to monitor and manage due to its many geographically and organizationally distributed components, tools and services to assist humans in such tasks have become a necessity. To date, a number of Grid monitoring systems has been developed to enable monitoring and displaying Grid topology, its components, and their selected parameters, for example, resource availability, load distribution and task performance [2][3][4][5]. In addition to data collection, most current systems also include a means for Web-based visualization of the monitoring information gathered by the system.

With recent advances in multimedia capabilities in user terminals, there is a strong trend towards improved interactivity as well as rich media spaces for information visualization [6]. Visualizations are becoming *multi-modal* and *multi-platform*, i.e. they may combine *various media* such as text, hypertext, pictures, multi-dimensional graphics, audio, and video, on a *wide range of client (end-user) platforms*, from PCs to new-generation mobile phones.

In this paper we describe a multi-platform visualization architecture and a Web based service built upon it, which provides a user with a view of the monitored Grid hierarchy, and the values of selected monitoring parameters for different Grid sites.

The data source used by our prototype was a central data repository provided by the MonALISA system [4], however, the described architecture is independent of data source, and the particular implementation described in this paper could be tailored to work with a different database or data source (e.g., a publish/subscribe system) by use of Web services. We also believe it to be suitable for integrated monitoring systems [7], where data could come from different sources but could be presented through a common visual interface. The prototype service we implemented provides an interface for users to view Grid configuration and monitoring data, such as load, data rates, and memory on different platforms. The capability for multiplatform visualization and flexibility in introducing new visualization techniques make our approach a viable alternative to custom-made graphical user interfaces.

The paper is organized as follows. First, we briefly analyze the properties of typical client platforms with their communication and presentation capabilities. Next, we describe the proposed architecture and its implementation. Finally, we demonstrate the application of the proposed architecture on Grid monitoring data visualization on four platforms: a desktop Personal Computer (PC), a handheld PC, a Java-enabled new-generation mobile phone, and a Wireless Application Protocol (WAP) enabled mobile phone.

2 Client Platform Capabilities

With the growing heterogeneity arising from differences in client devices and network connections there is a need to adapt services to device networking and presentation capabilities. Various parameters dictate client platform capabilities (e.g. available memory, processor speed, graphics card, display) and connection types (e.g. LAN, Wireless LAN, General Packet Radio Service (GPRS)). Table 1 gives an overview of some client devices that were used in this work as test platforms, with their respective characteristics, processing capabilities, and communication capabilities.

For the purposes of this paper we divide client platforms with regards to visualization capabilities into three groups (full, midi, mini) as follows:

Full clients: platforms with enough processing power and visualization capabilities to locally process the raw data received from a server and visualize it while simultaneously supporting all visualization modes (2D and 3D graphics, tables, text, etc). In addition, the full client may offer better interaction capabilities with the data. Although other types of full clients may emerge, the representative of a full client is a PC (Windows, Linux, etc.) running a standard Web browser with Java support. Example interfaces and visualization tools may be implemented as Java applets (e.g. Shout3D) and standard Hypertext Markup Language (HTML) elements, so there is no need to download/install any software on the client machine. Additional software, however, may be installed if needed depending on the interface implementation (e.g. Virtual Reality Modeling Language (VRML) [8] plug-in for 3D graphics). Client hardware and software may vary from lower-end configurations to higher-end configurations.

Midi clients: platforms with medium processing power and visualization capabilities, powerful enough to present a user interface, simple 2D/3D graphics, as well as text and tables. The representative of a midi client would be a Personal Digital Assistant (PDA) (e.g. Compaq iPAQ or Palm) or a PDA-integrated mobile phone (e.g. Sony Ericsson P800).

Mini clients: platforms that have insufficient processing power for local processing of raw data, and insufficient presentation capabilities for showing either 2D or 3D graphics. Such a terminal would receive pre-formatted Wireless Markup Language (WML) pages ready for presentation instead of raw data.

Table 1. Client platforms – processing and communication capabilities

		Full Client PC (low-end)	Full Client PC (high-end)	Midi Client PDA	Midi Client Smart Phone	Mini Client Mobile phone
General Characteristics	Terminal example	Desktop PC low-end	Desktop PC high-end	Compaq iPAQ 3870	Sony Ericsson P800	Ericsson R520s
	Operating system	Windows 2000/XP	Windows 2000/XP	Windows Pocket PC (CE 3.0)	Symbian 7.0	proprietary
	Browser	IE 5.5 / Netscape 6.0	IE 6.0 / Netscape 7.1	IE 3.02	Opera 6.0 / SE R101	EricssonR520 R201 WML browser
		HTML, VRML, Shout3D	HTML, VRML, Shout3D	HTML, VRML, Shout3D	HTML WML	WML
Processing Capabilities	Processor	PIII 800 MHz	P4 2.66 GHz	Intel Strong ARM SA 1110 206 MHz	ARM 9 156 MHz	N/A
	Speed	1066 MIPS	4904 MIPS	235 MIPS	N/A	N/A
	Memory size	128 MB	512 MB	64 MB	16+16 MB	N/A
	Display size	1024x768	1280x1024	240x320	208x320	101x65
	Color depth	32-bit	32-bit	12-bit	12-bit	1-bit
	Sound	16-bit	16-bit	8-bit	16-bit	none
		Stereo	Stereo	Mono/speaker, Stereo/headph.	Mono/speaker, Stereo/headph.	none
Communication Capabilities	Network connec-tion	LAN 100 Mbps	LAN 100 Mbps	GPRS CS-2 53.6 kbps / WLAN 11 Mbps	GPRS CS-2 53.6 kbps	GSM 9.6 kbps / GPRS 48 kbps
	Latency	<10 ms	<10 ms	1 s (GPRS)	1 s	0.5 / 1 s
	Jitter	<1 ms	<1 ms	N/A	N/A	N/A
	Packet loss	<1%	<1%	N/A	N/A	N/A
	BER	$<10^{-8}$	$<10^{-8}$	10^{-3} / 10^{-5}	10^{-5}	10^{-3} / 10^{-5}

In addition to service customization based on client processing and communication capabilities, customization may be introduced through user preferences. For example, a user may wish to filter some media types (e.g. sound, video, etc.) in order to increase response time from the system and thus increase the functionality of a specific application interface at the expense of richness of presentation detail.

3 Multiplatform Universal Visualization Architecture

The architecture we apply to for visualization of Grid monitoring data is the Multiplatform universal visualization architecture (MUVA) [9]. MUVA provides universal visual access to data independent of the client platform, while automatically adapting delivery modes to the particular platform. Rather than developing data visualization applications designed to run on a target (group of) client platforms, we separate the platform adaptation procedure on the output side from the implemented data visualization technique, and so facilitate flexible client access. In addition, by separating the data source on the input side from the visualization technique, the result is reusability of such techniques across a wide range of application domains and data sources. For example, the implementation of a technique such as a 3D tree representation of a hierarchical structure may be reused when developing any application which makes use of hierarchical data visualization.

MUVA has been designed as a flexible and modular architecture comprised of a collection of software modules. Fig. 1 presents a conceptual view of the MUVA architecture. Crucial parts of the architecture are the *visualization tools*, which represent various modes and concepts of visualizing the data (e.g. text, table; graph; chart; tree). Visualization tools are separated from actual client devices by *platform drivers*, designed to adapt the data delivery mode to specific platforms. On the input side, actual data collection is separated from the abstract visualization tools. This allows for any data source to be connected simply by developing thin *application interfaces*. The result is quick adaptability to various specific application domains.

The *service logic* provides the necessary intelligence for connecting application interfaces, visualization tools, and platform drivers depending on the particular client platform capabilities and user request. Each component of the architecture contains several modules, where not all of them have to be used in each application. The modular design and separation of MUVA components allows for easy addition, modification, and maintenance of software modules. A more detailed description of MUVA components is given below.

Visualization tools are responsible for one particular mode of visualization (tool), e.g., a 3D structure of an input hierarchy, a simple table, bar-chart, pie-chart, etc. Each tool is *standardized*, in terms of (1) input data parameters that can be fed to it through its API; (2) requests it can receive through its request interface; and 3) visual output it produces in reply to a given request.

Platform drivers are implemented for each supported platform. They render (visualize) formatted data received from visualization tools on the screen, and enable user interaction. The communication with the visualization tools may be through the network or local, depending on the location of the visualization tool in relation to the platform driver. Any or both may in certain cases be located on the server side, and in other cases on the client.

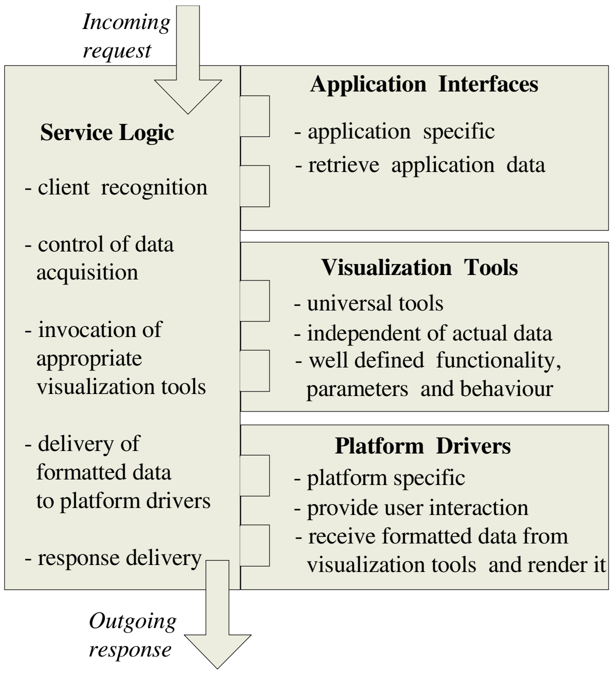

Incoming request
Service Logic
- client recognition
- control of data acquisition
- invocation of appropriate visualization tools
- delivery of formatted data to platform drivers
- response delivery
Application Interfaces
- application specific
- retrieve application data
Visualization Tools
- universal tools
- independent of actual data
- well defined functionality, parameters and behaviour
Platform Drivers
- platform specific
- provide user interaction
- receive formatted data from visualization tools and render it
Outgoing response

Fig. 1. MUVA concept

Application interfaces are responsible for retrieving data from a data source via a standard application specific API. Retrieved data is converted to Extensible Markup Language (XML) format based on the specified input interface for visualization tools.

Service logic encompasses modules that provide the intelligence needed to bring together the components of the architecture in order to enable universal visual access and delivery mode adaptation.

To illustrate the typical usage scenario, we start from the incoming client request. Upon receiving a client service request, the client's preferences and platform capabilities are identified. One method of identification is based W3C Composite Capabilities / Preference Profile (CC/PP) Recommendation for device independence [10], a proposed industry standard for describing delivery context. The client profile data format is based on the Resource Description Framework (RDF). A client profile may either be sent directly as an extension to a HyperText Transfer Protocol (HTTP) request, or referenced from a remote location using a Uniform Resource Locator (URL). The implementation of client identification based on a set of generic profile parameters allows for on-the-fly identification of the capabilities of an increasing number of end-user devices.

The service logic retrieves raw data independently of the platform capabilities through invocation of application interface modules. The raw data is then sent to appropriate visualization tools. Formatted data received as the output from visualization tools is then delivered to necessary platform drivers. The service logic layer provides the logic necessary to select adequate visualization tools and platform drivers to produce the final content, adapted to the given client platform.

In order to demonstrate the proposed approach in a real world scenario, a prototype Web based service was implemented. In the following section, we describe a service providing multiplatform visual access to Grid monitoring data. Service implementation helps to demonstrate the separation between application interfaces, visualization tools, platform drivers, and service logic components, as well as the main communication channels involved.

4 Implementation

The service implemented in this work provides an interface for users to view network configuration and monitoring data, such as load, data rates, and memory on different platforms. As mentioned earlier, the data source used was a central data repository provided by the MonALISA system, which provides a distributed monitoring service and was in this case used to monitor hundreds of AliEn Grid sites (http://alien.cern.ch/).

Users access the service by entering a unique URL, which is independent of the client device being used. Requested data, which is then retrieved from a central repository and described using XML, is dynamically converted to a format suitable for displaying on the client device. The different formats that were used include VRML, HTML, and WML. The service implementation was tested on four platforms: a desktop PC, a handheld PC, a Java-enabled PDA-type mobile phone, and a standard WAP-enabled mobile phone.

Where possible, the monitored network configuration (or a particular sub-configuration) was visualized using the 3D Cone Tree technique [11]. Cone Tree is an interactive visualization technique suitable for hierarchical structures. The root of the network hierarchy is located at the tip of a transparent cone. When a level in the hierarchy is expanded (on user click), its children nodes are distributed at equal distances around the base of a cone. The user interface is enhanced by enabling interactive viewing, zooming, expanding and collapsing of parts of the structure. A formal user study using a cone-tree-based file system visualization showed that although Cone Trees are not suitable for all tasks, users "were enthusiastic about the cone tree visualization and felt it provided a better 'feel' for the structure of the information space" [12].

We now describe the implementation of the MUVA system architecture components.

Service logic – Client recognition functionality was implemented using Apache server 2.0.47. Due to fact that the developed prototype service was intended to be made available to users outside of a laboratory environment, and the current lack of widespread availability of CC/PP compliant terminal browsers, implemented client recognition is in this case based simply on identification of the browser type specified in the User-Agent header field of the HTTP request.

Data is retrieved in standard XML format through invocation of application interface components and formatted depending on the platform capabilities and requests of the client. Requests are directed towards a Java servlet that is run using

Apache Tomcat Server 4.1.27-LE-jdk14. The servlet requests data, and invokes the necessary visualization tool. In cases when 3D content is generated, the Cone Tree tool is called. Once the VRML result is received, the servlet calls on a platform driver to further adapt the VRML file for rendering in a Shout3D (Eyematic Interfaces Inc., www.shout3d.com) applet, after which the HTTP response is sent to the client.

In cases where the content generation is based on HTML or WML format, the servlet passes retrieved XML data to Apache Cocoon 2.0.4. Data is formatted into HTML tables or histograms using Extensible Stylesheet Language Transformations (XSLT) technology. Where necessary, additional platform drivers are invoked to further adapt the format, prior to sending the response to the client.

Application interfaces – Data is always requested by application interfaces and returned in standard XML format. The interface towards the actual data repository storing monitoring data collected by the MonALISA system is based on Web Service technology. Connectivity to the Web Service was provided using Apache Axis 1.1 open source solution, the follow up on the Apache SOAP project. The stub code for the Web Service was generated by Axis' WSDL2Java utility and modified according to our needs. The Web Service returns values in the form of Java beans, that are then transformed to XML format.

Visualization tools – Once data is retrieved, visualization tools are needed to generate the actual data representation. Various visualization techniques were used, including text, 2D graphics, and 3D graphics.

The creation of a VRML Cone Tree display based on an input hierarchy was implemented using Java. In general, any form of hierarchical data structure may be given as a valid input. Optional additional parameters may be applied to create a simpler Cone Tree display (e.g. when the client device is a PDA), where certain elements of the tree hierarchy are filtered (e.g. display only nodes or clusters belonging to a particular farm). If a particular terminal is not capable of displaying or rendering complex 3D graphics, data is transformed to simple HTML tables or histograms by using XSLT.

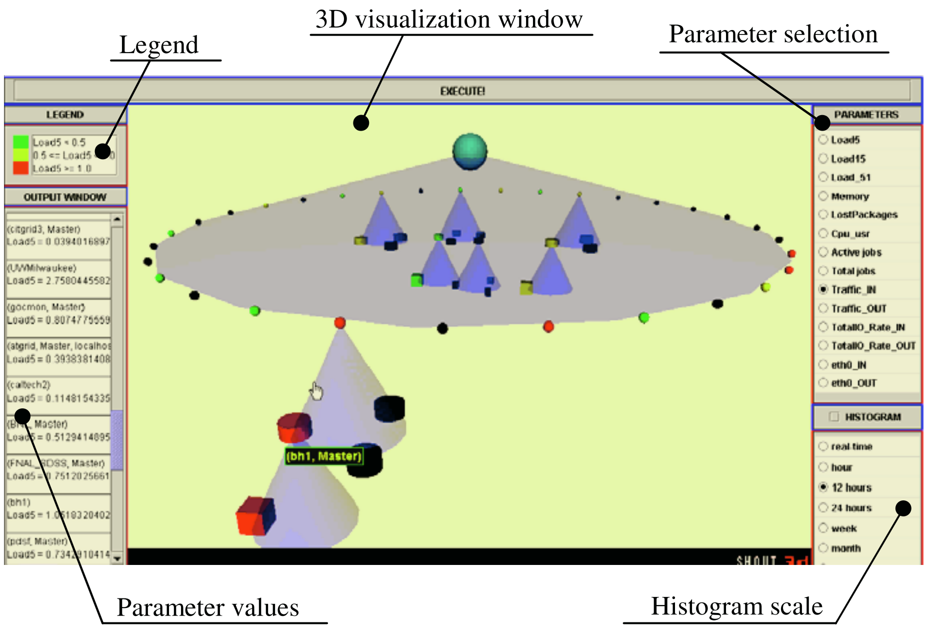
Platform drivers – The interface displaying the 3D scene, designed to be viewed on a client with a standard Web browser and Java support, was implemented as a Java applet and based on the Shout3D engine. Shout3D is a library of Java classes for rendering 3D scenes over the Internet, thus offering the user the ability to view and interact with 3D scenes without the need for any additional plug-ins. In a different set-up, Cortona VRML plug-in and Pocket Cortona (for rendering on the iPAQ PDA) were used to render the 3D scene. Within the scope of MUVA, the Shout3D applet classes and Cortona plug-in are all considered platform drivers.

In addition to displaying 3D content on the PC and iPAQ clients, we implemented a C++ application to dynamically generate a 3D scene (in our case a 3D Cone tree display) on the Sony Ericsson P800 mobile device. The DieselEngine SDK 1.3 was used for software support. It is a collection of C++ libraries for creating 3D applications on mobile devices. Additional software used included Symbian UIQ v7.0 SDK and Metroworks CodeWarrior for Symbian OS. The application that was built reads a VRML file dynamically generated by the Cone tree visualization tool, parses the file, converts it to Diesel3D scene format and displays the content to the user. Additional interaction modules for navigation, camera manipulation, and object selection within the scene were also implemented. Also, XSLT files were implemented to further adapt content for display on a particular device. This includes creation of WML format for display on a WAP-enabled mobile phone.

5 Results

The result is a multiplatform Grid data visualization service that is developed in a modular fashion in order to facilitate adaptation across different application domains. The implemented service provides visualization of monitoring parameters for a large number of Grid nodes. The monitored nodes are arranged in a hierarchical manner into farms and clusters, referring to the geographical and/or logical grouping of nodes into virtual computing systems. The MonALISA system collects monitoring data from all distributed sites and stores the data in a central repository. Data is collected by our service from the central repository in CERN via a Web Service interface.

A view of the *full client* display interface is shown in Fig. 2. Upon initial loading, the *3D view window* renders the 3D scene displaying the dynamic node configuration. The *Parameters window* enables a user to choose a monitoring parameter. The *Histogram window* enables a user to choose between displaying real-time data and history data. Once the user has chosen a parameter and histogram button, clicking on the "Execute!" button will initiate the coloring of tree nodes and writing text to the output window. Parameter values are retrieved for each node, and coloring is based on the range that the value fits into. These ranges are displayed in the *Legend window*.



- 🟩 Load5 < 0.5
- 🟨 0.5 <= Load5 < 1.0
- 🟥 Load5 >= 1.0

```
(citgrid3, Master)
Load5 = 0.0394016897
(UWMilwaukee)
Load5 = 2.7580445582
(gocmon, Master)
Load5 = 0.8074775559
(atgrid, Master, localhos
Load5 = 0.3938381408
(caltech2)
Load5 = 0.1148154335
(BNL, Master)
Load5 = 0.5129414895
(FNAL_SDSS, Master)
Load5 = 0.7512025661
(bh1)
Load5 = 1.0510320402
(pdsf, Master)
Load5 = 0.7342910414
```

PARAMETERS: ( ) Load5 ( ) Load15 ( ) Load_51 ( ) Memory ( ) LostPackages ( ) Cpu_usr ( ) Active jobs ( ) Total jobs [✓ Traffic_IN] ( ) Traffic_OUT ( ) TotalIO_Rate_IN ( ) TotalIO_Rate_OUT ( ) eth0_IN ( ) eth0_OUT

HISTOGRAM: ( ) real-time ( ) hour [✓ 12 hours] ( ) 24 hours ( ) week ( ) month

Fig. 2. Visualization using a *full client*: 3D visualization and text windows

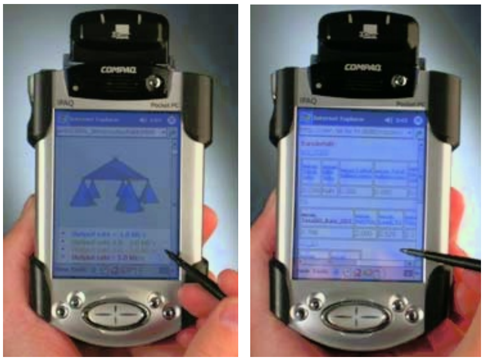


Fig. 3. Visualization using a *midi client* (iPAQ)

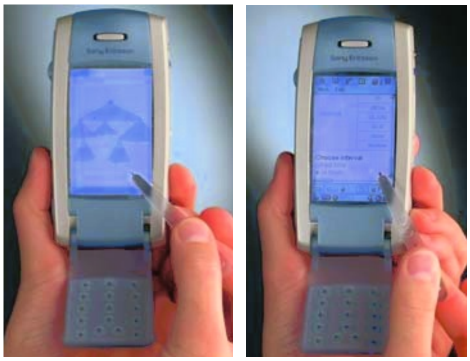


Fig. 4. Visualization using a *mini client* (Sony Ericsson P800)

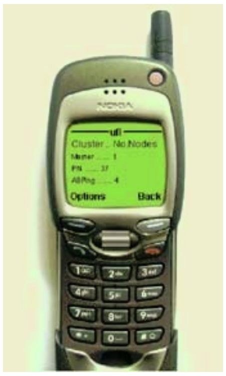


Fig. 5. Visualization using a *mini client* (WAP enabled mobile phone)

When a user accesses the same service using an iPAQ (Fig. 3), the same data is represented using a combination of HTML pages (to specify the network configuration) and simpler 3D Cone trees with only subsets of nodes shown (e.g. a user chooses to view monitoring parameter values for a specific cluster). In cases where a large hierarchy needs to be presented, small display size and lower processing capabilities make it more efficient to present the data using a simpler method, such as a table that a user can scroll through.

The visualization is also displayed on the Sony Ericsson P800 mobile device, where data is again represented using a combination of HTML pages and 3D content (Fig. 4).

If a user using a WAP enabled mobile phone accesses the service, the result is data presented in WML format. Simple WML pages allow the user to list through the network configuration, and request monitoring parameter values (Fig. 5).

6 Conclusion

In this paper, we have presented the concept of a multiplatform universal visualization architecture and its application to visualization of Grid monitoring data on multiple platforms. The key features of the proposed approach are independence of data acquisition and the thin adaptation “layer” for the platform on which the data is visualized. The presented case study demonstrates the visualization of Grid monitoring data on multiple platforms: a desktop PC, a handheld PC, and various mobile phones. Our future work is directed towards extending the system with the ability to interact not only with the display, but also with the Grid itself, as well as towards introducing new visualization tools, suitable for mobile devices.

References

1. Foster, I., C. Kesselman. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers (1998)
2. Andreozzi S., N. De Bortoli, S. Fantinel, A. Ghiselli, G. Tortone, C. Vistoli. GridICE: a monitoring service for the Grid, Proceedings of the Third Cracow Grid Workshop, Cracow, Poland (2003)
3. Balaton, Z., P. Kacsuk, N. Podhorszki, F. Vajda, Comparison of Representative Grid Monitoring Tools, Laboratory of Parallel and Distributed Systems, Computer and Automation Research Institute of the Hungarian Academy of Sciences, Technical Report LPDS-2/2000 (2000)
   [Available: http://www.lpds.sztaki.hu/publications/reports/lpds-2-2000.pdf]
4. Newman, H. B., I. C. Legrand, P. Galvez, R. Voicu, C. Cirstoiu. MonALISA: A Distributed Monitoring Services Architecture. Proceedings of 2003 Conference for Computing in High Energy Nuclear Physics, La Jolla, California, USA (2003)
5. R-GMA: Relational Grid Monitoring Architecture, http://www.r-gma.org
6. Card, S. K., J. D. Mackinlay, B. Shneiderman. Readings in information visualization: using vision to think. Morgan Kaufmann Publishers (1999)
7. Mambelli, M., R. Gardner. Integration of Monitoring Systems for Grid Environments, Proceedings of the 13th IEEE International Workshop on Enabling Technologies WET-ICE: Infrastructure for Collaborative Enterprises (2004) 266-267

8. Information Technology – Computer graphics and image processing – The Virtual Reality Modeling Language (VRML) – Part 1: Functional specification and UTF-8 encoding. ISO/IEC 14772-1:1997 (1997)
9. Skorin-Kapov, L., D. Mikic, H. Komericki, M. Matijasevic, Pandzic, I. Multiplatform Universal Visualization Architecture, Proceedings of the Second International Conference on Advances in Mobile Multimedia 2004, Bali, Indonesia (2004)
10. Butler, M., F. Giannetti, R. Gimson, T. Wiley. Device Independence and the Web. IEEE Intenet Computing 6, 5 (2002) 81–86
11. Robertson, G. G., J. D. Mackinlay, S. K. Card. Cone trees: Animated 3D visualization of hierarchical information. Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology, ACM Press, New York, USA. (1991) 189–184.
12. Cockburn A., B. McKenzie. An Evaluation of Cone Trees, In People and Computers XIV: Proceedings of the HCI 2000, 14th Annual Conference of the British Human Computer Interaction Group (2000) 425–436