A New Grid Portal for Managing and Monitoring Application Execution on Global Grids through Multiple Devices

by

Kwai Wah, Ho

BSc (Hons) in Computer Science (De Montfort University)

Under the Supervision of:

Dr. Rajkumar Buyya

A minor project thesis submitted in partial fulfillment

of the requirement for the degree of

Master of Information Technology

Grid Computing and Distributed Systems Laboratory

Department Computer Science and Software Engineering

The University of Melbourne, Australia

June 22, 2004

Abstract

Previous G-Monitor was developed in response to the need for Web-based portals that hide low-level details of accessing Grid services for deployment and execution management of applications which enabled the user to upload their experiment files, start experiments, collect results and authenticate the grid proxy. However, the previous framework doesn't give access to grid services through handheld devices. In this paper, we purpose a new extensible framework and design methodology for grid access from any type of devices that based on Data Centric Model and Globalisation architecture where interaction was based on a combination of multiple modalities that are induced by different media and different navigation paradigms. This new portal was tested with Web-Based client, PDA client, and WAP phone client. In this paper, we have demonstrated that huge mode combinations in hypermedia can accommodate large user needs and tasks. Amongst such needs, that of accessibility holds pride of place. Accessibility isn't just about serving groups of disabled users [10], the same standards also enable web access by phone and Personal Digital Assistants (PDA). Also, we will demonstrate how it impacts on accessing the Grid environment through Grid Broker. In this experiment, our new G-Monitor interface will interact with the Gridbus Resource Broker [2], a part of Gridbus project [23].

Acknowledgement

I wish to express my deep gratitude to my supervisor and mentor Dr. Rajkumar Buyya. I thank him for his continuous encouragement, confidence and support, and for sharing with me his knowledge and experience. Moreover, I would like to thank my colleagues Martin Placek and Srikumar from **Gri**d Computing and **D**istributed **S**ystems (GRIDS) Laboratory. Martin Placek was the developer of the previous version of G-Monitor. He told me a lot of his past experiences on the previous G-Monitor, his valuable idea and skills he imparted through the collaboration. Besides that, I would like to thank Srikumar since he helped me to set up the testbed in his Grid Resource Broker that running in Sydney, Canberra, Adelaide and Melbourne University.

Contents:

Chapter 1: Motivation

1.1 Introduction

Grids provide the infrastructure to harness a heterogeneous environment comprising of geographically distributed computer domains, to form a massive computing environment through which large scale problems can be solved. For this to be achieved, Grids need to accommodate various tools and technologies that can support: security, uniform access, resource management, scheduling, application composition, computational economy and accounting [3][4]. In addition, Grids need to provide a ubiquitous user-interface that hides complexities associated with the deployment and management of execution applicable to experiments using distributed resources under *Globalisation* architecture. Globalisation is not a feature – it is an architecture [25]. In the context of G-Monitor, Globalisation is a suitable design, services and procedures so that G-Monitor can provide multilingual information with culturally correctness (for example, date, time, currency and number formats) from the intended broker such as Gridbus Broker, Storage Resource Broker and etc. However, the current version of G-monitor [1] was created to provide the users with a pervasive interface that allows them to monitor and control the execution of their applications. Besides that, it also provides a multi-user environment where the users are able to deploy and collect experiment files, authenticate their grid proxy and analyze graphs illustrating experiment progress. Nevertheless, we belief today "Networked World", all devices will require the ability to communicate with environment. This makes the terms of "Mobility" and "Roaming" become very important and necessity for everyone in their daily life. Hence, people wish to get information easily in any location and any time.

Mobile application is considered a type of hypermedia application since it combines the flexibility of navigation based-access to information with typical use of hypertext and allows communication with multiple media. By the nature of hypermedia applications, it is supported with multimode interaction. In this thesis, we use the 1st version of G-Monitor as our case study in this thesis and provide a better framework and

methodology, which give access through different type of devices while hiding all the implementation details of the services, allowing it to be used independently of the hardware or software platform by having a framework that loosely coupled, component-oriented, cross-technology implementations. It illustrated in Figure 1.

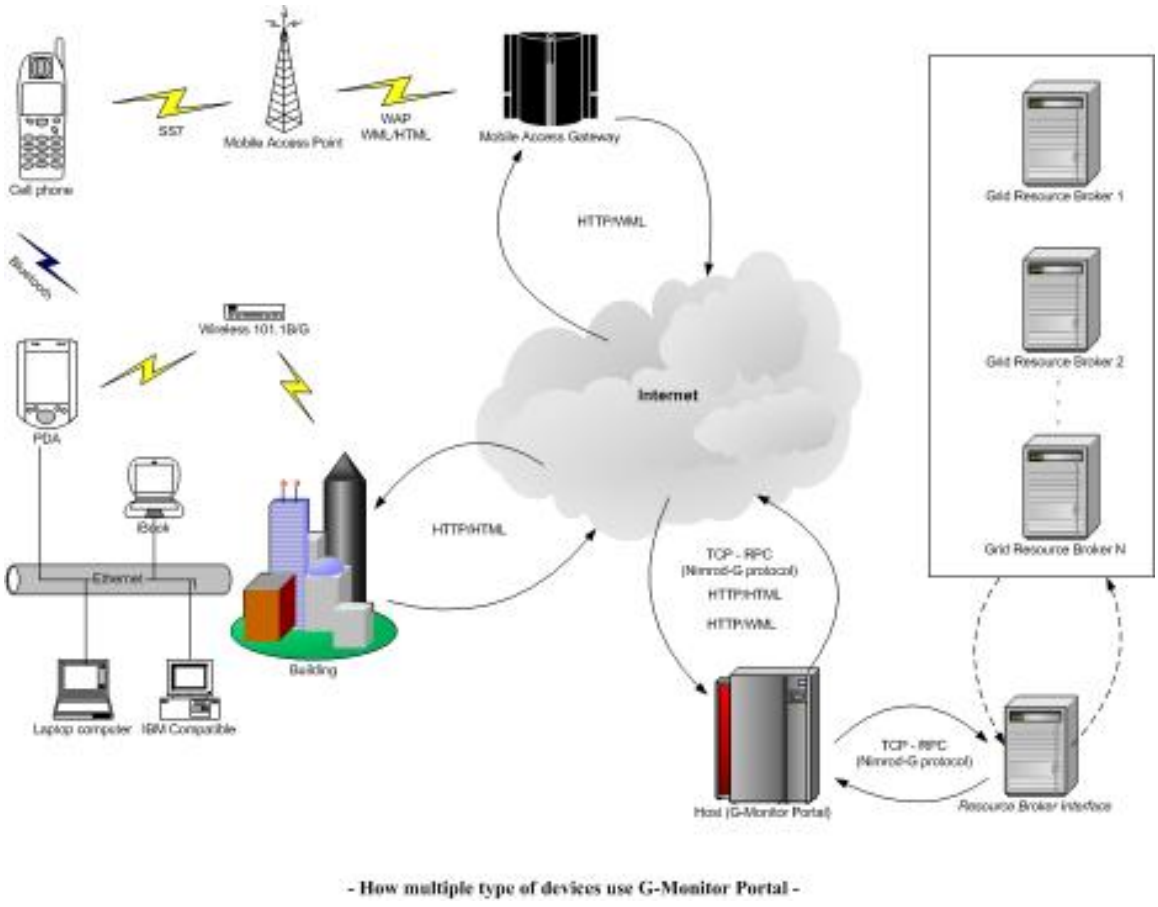

Figure 1: How multiple type of devices use G-Monitor Portal.

A sample wide-area Grid computing environment is shown in Figure 2. In this environment, the Grid consumers interact with GRBs (Grid Resource Brokers) such as Nimrod-G [7] responsible for scheduling applications on the distributed resources, based on their availability, cost, capability, and user-specified QoS (Quality of Service) requirements.

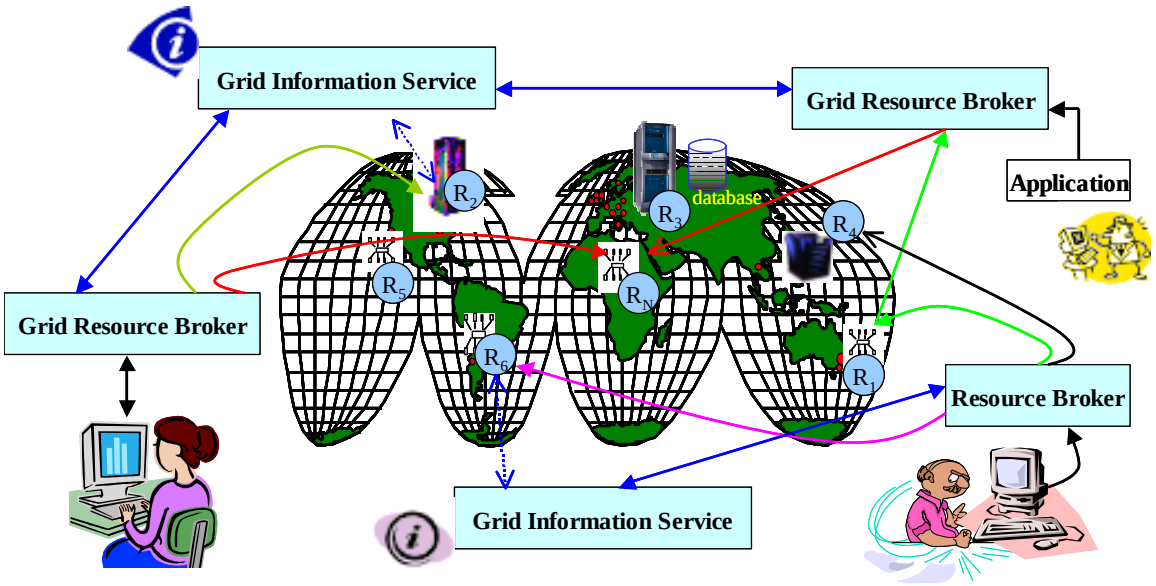

Figure 2: A Sample Grid Computing Environment and key components

1.2 Related works

In the context of design methodology for grid access from handheld devices, it is advisable to mention the Condor Grid Computing from Mobile Handheld Devices [12], which provides the technical issue, based on the hierarchical design methodology. However, in that paper they just provide the system overview design and its prototypes. In addition to that, basically they are only focusing on how their framework can be access from handheld devices. Besides that, they are a number of projects have explored development of toolkits for development of Grid portals and construction of application specific portals. Some representative efforts include GridPort [5] and HotPage [6] from an Diego Supercomputing Centre, GPDK (Grid Portal Development Kit) [7] from Lawrence Berkeley National Laboratory, Legion portal [8] from the University of Virginia, GRB portal [9] from University of Lecce, SGE (Sun Grid Engine) Technical Portal [10] from Sun Microsystems, and PBSWeb [11] from the University of Alberta. Unlike these systems, the unique core objective of G-Monitor is that it has been designed to provide access to high-level Grid services (e.g., the Nimrod-G broker and Gridbus scheduler). As high-level Grid services (e.g., Nimrod-G) are in turn implemented using low-level Grid services (e.g., Globus), they hide issues related to the identification of resources that are suitable for running user applications and their aggregation. Conversely, in our new G-Monitor methodology, we combine the Computer Science discipline and Human Computer Interaction discipline to provide a feasible, flexible and extendable model to enhance our previous G-Monitor architecture for our future development.

1.3 Use Case Study

Research carried out at The University of Melbourne, by the Experimental Particle Physics (EPP) group [13] and the GRIDS Lab, involves investigation into the development of Grid technologies for high energy physics. The EPP group is a member of the Belle collaboration, which consists of 400 physicists from 50 institutions around the world [14]. The Belle experiment, situated at the KEK B-factory in Japan, is operational and has been collecting data and producing results for a number of years. G-Monitor has been used in the Belle application analysis on global Grids demonstration at the 4th PRAGMA workshop [15] [16] held in June 2003, Monash University, in Melbourne, Australia. The demonstration involved the construction of a data grid test bed. The grid test bed was constructed within 9 days prior to the PRAGMA workshop and was made possible by the loan of high performance servers from IBM Asia Pacific. The demonstration setup depicting application deployment and access locations of various Grid entities is shown in Figure 3. Servers from around Australia were used to analyze data collected from the Belle simulations. During the demonstration G-Monitor was used to manage and monitor experiment progress at both application and job levels. An application jobs execution statistics plot from the experiment are shown in Figure 4 and this is on of those functionalities from the current G-Monitor that will be bring over to our new framework.

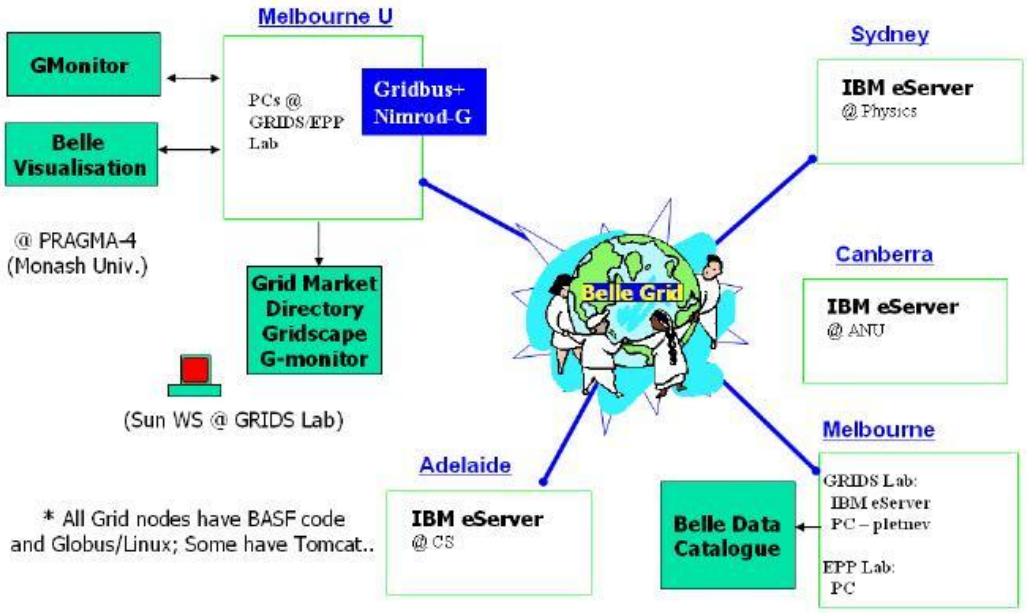

Figure 3: G-Monitor Usage during the HPC Challenge Demo @ SC 2002

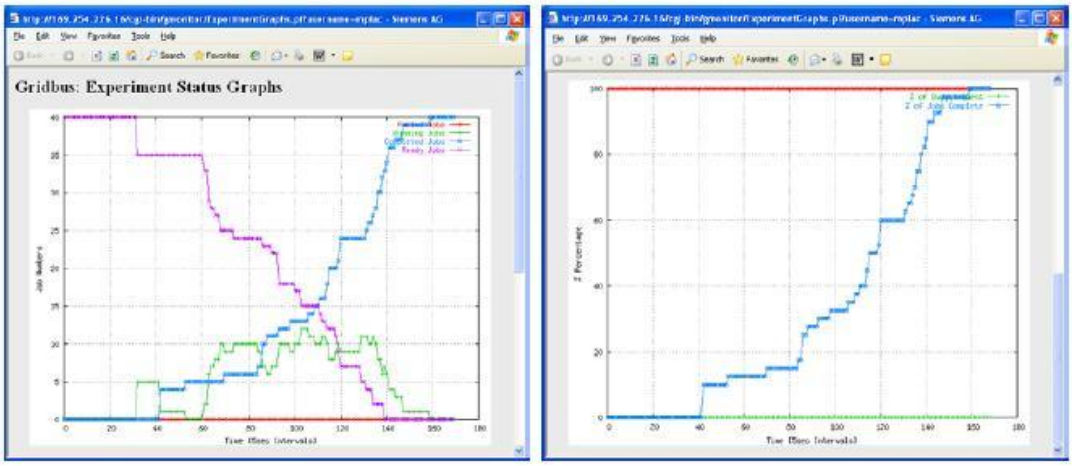

Figure 4: Sample of Graphs results generated by G-Monitor

In our new framework, we will map and bring over all the functionality from the previous G-Monitor and provide some more other features to extend the core functionality of the G-Monitor itself to coup with the complexity of Grid Architecture and it's usage. Moreover this new framework will serve the request not only from web browser only but even more, such as PDA, J2ME enabled device, WAP phone, Web Services and off-line browsing capabilities. These additional features are:

- Customization of PDA screen output using parameters setting based on user or device preference.

- Ability to handle multiple languages and provide multilingual data, based on Globalisation Architecture with the use of Locale Model.
- Support browsing from other handheld devices not only for web browsing.
- Generate dynamic jobs information table based on screen size and user preference.
- Sharing “Grid Proxy Authentication” Permission Security.
- Graph generator using SVG (New XML standard) – Testing Phase.

Below are those functionalities from the current G-Monitor that will be bring over to our new framework:

1. Retrieve and set QoS (quality of service) parameters, such as Deadline, Budget and Optimisation preferences.
2. Monitor/Control Jobs Information, such as Start, Stop, Grid node status and Execution time.
3. Monitor Resource status, such as Server name, Host name, Service cost and Status.
4. Monitor Experiment status, such as Deadline, Budget, Job status and Resource status.
5. Multi-User Interface User login and User preferences relating to available brokers and interface.
6. Real time graphs, Refer to Figure 3
    - Graphs of the experiment progress are provided throughout experiment execution.
7. Start Experiment
    - Ability to upload experiment files using the G-Monitor interface and start experiments.
8. Grid Authentication
    - Ability to update Grid Authentication, without the need to access the Nimrod Broker.
9. Collect Experiment Files
    - Upon completion, experiment files with results can be collected.

Chapter 2: Mobile Appliance on HCI

2.1 Our Methodology for designing Mobile Appliance Application.

Grids enable the sharing, selection, and aggregation of a wide variety of resources including supercomputers, storage systems, data sources, and specialized devices that are geographically distributed and owned by different organizations for solving large-scale computational and data intensive problems in science, engineering, and commerce [24]. Hence, we come out with a new methodology and framework to overcome our previous G-Monitor. Moreover we apply the Human Computer Interaction principal in our context to enhance usability in our new G-Monitor Web-based portals that meant to hide low-level details of accessing Grid services for deployment and execution management of applications. The reason behind of applying principal of HCI in our G-Monitor context is because we want to provide a uniform interface that can support not only expert user or technical person but also novice person. This uniform interface must map all our supported devices that can access to our G-Monitor portal. Today information retrieval is needed for 24 hours a day, 7 days a week, regardless of their location or activity at the moment. Therefore, we come out with a new framework that supports access through handheld devices such as PDA and cellular phones. Now scientists can access to our new G-Monitor portal anytime and anywhere. We studied and come out with a new methodology that uses HCI principal in our design discipline.

Human-Computer Interaction (HCI) for mobile appliance is a discipline concerned with design of technological guidelines that are effective, efficient and satisfying to use in mobile context approaches. Mobile devices present HCI designers with five main challenges that were posted in the CHI conference in the year 2001. Today, they still remain the top challenges in Mobile HCI with reference to the CHI conference that was last held in September 2003.

Limited input/output facilities, Since the early eighties, much work was done on improving the readability and comprehension of information displayed on small screens such as those of cash machines (ATM). Duchnicky, Kolers [17] and Dillon et al [18] discovered that users are able to read information that is displayed on very small screens comprising only a few lines of text without adversely affecting the ability to understand and compile information. Nowadays, however, most mobile users claim that mini screens are totally inadequate for information presentation and it is mainly because of their long-term preferences (Desktop PC) quality rather than readability issues**!** Most handheld browsing often feels like browsing on a PC with a shrunken desktop. Over-reliance on scrolling is a big problem in current handheld browsers. Users are only able to view a small and limited portion of each page, which causes them to lose comprehension of the overall context. Also, in terms of usability users may feel lost in a large page and be forced to remember the locations of items as those items scroll out of view. According to Nielson "*Experience from many other user interface platforms indicates that a bigger screen leads to better usability than a small screen and that a graphical user interface adds even more usability*"[19] Currently, most researchers are focused on how to best use the limited area to present the increasing functionality of the devices as well as presenting complex information on these limited screens by using early research finding in the field of information visualisation. Certainly on a larger display surface, improvements in layout and font are desirable for the sake of readability and quality, but on a smaller display, the improvements must be evaluated to determine whether better readability is worth any sacrifice in the amount of text displayed. In general, portability tended to outweigh these problems when the texts were shorter and the reading shallower, more for the purpose of familiarization with the text. Research found that we could characterize on-screen reading practices on the handheld as converging on quick reading, skimming, and scanning to meet the needs of high time-constraints [20]. In this paper, we also addressed the fact that it was useful to utilise multiple modalities to access and interact with information services via any mobile devices. This can ensure that at least one mechanism is available for every person and situation. In particular, the combination of voice input together with graphical selections, as well as the combination of graphical and audio output, is very likely to become the predominant interaction paradigm for users

of tiny mobile devices. By using this multi-interface, the user can flexibly switch between several information display modes, including the display of text and graphics, audio display and effective combinations of multiple modes. During the designing stage, we can decide what display modes are the most functional based on available HCI guidelines detailing which media combinations are more appropriate considering a mobile user's task and situation.

Varying and incomplete context information, One of the most challenging issues is relatively on how to adhering Human Mental Model concurrent to design around the apparently, which conflicting goal of adapting to changes in a context. This problem needs to be dealt with carefully whereby the system should behave as the Human Mental Model and adhere to the principle of least astonishment. Human mental models are generally simpler than reality since people build models of how the world works and that comes from the past experience, instruction and training. We use previous G-Monitor as a vehicle to explore this issue.

Controls vary within context, target users and environment. The concept of control is another fuzzy decision that should be analysed in HCI studies since it needs to be considered whether or not it is best to provide the user with a high level of control with complex or arduous interaction or to provide high level control with limited choices. There are pros and cons in these issues. The advantage of a complex or arduous interaction is that users are enabled to dynamically change the context based on his/her preference, but novice users might find it is too daunting and would prefer a simpler approach such as a "single button for action or shortcut" which falls under limited choice control. The natural problem of information pulling is that users would forget the action required, in order to retrieve information using their device. This problem seemed to be a challenge for HCI studies because it wasn't a simple interface design issue, but rather a problem to do with the human short-term memory issue whereby humans can hardly memorise all stuff at once. It wasn't surprising that the information pushing mechanism may help in this issue since actual presentation of context-aware information is triggered

by contextual events (e.g. Alert user when their job finished). In this matter, HCI studies will help us to figure out the interesting variations on the 'control' issue concerns on how the system is perceived by the user within the environment; either system acts as a companion (Information Pull) or guardian (Information Push).

Aiming on common users, Normally users will not have any idea or probable training on the application or even device, so the application interface should be designed as "*one suits all"*. Time is valuable for users, they will not want to read help manuals and have to learn how to use the application. Their expectations are to be able to spend time on the system accomplishing their goals with little or no frustration. In practice, task and user analysis are the most difficult jobs in HCI, acquired because it requires a deep understanding of the users, rather than the design of the functionality itself as the system interface must match target users' skills, expectation and needs. In the real world, users are extremely diverse, "The design is satisfactory to one, doesn't mean it will be satisfactory to others". In this case, we should study the entire targeted users and come out with a superset of interface that nearly satisfies all users based on HCI guidelines. Exploiting the user's prior knowledge by making interface objects seem like objects that the user is familiar with can reduce user feeling of complexity to the system. There are a few distinct gold guidelines that were raised by the well-known graphic designer, Clement Mok and the Usability engineer, Jakob Neilson in the year 2000 CHI conference.

Clement Mok claims " *good is about clarity of communication, or good is about usability or good is about performance, when in fact a good product should nuances of grey in between* " [21]

Jakob Neilson claims, " *Feel always dominate more than Look!* " [21]

Mobility and Multitasking, In practice, people often buy a specific mobile device to encourage a specific interaction and mobility. It may be a pioneering conversation piece with a style they consider attractive, or it may be a device, which supports an active role-playing where the user interacts with others people around him or her. With the advent of GPRS, aimed at increasing the data rate to 115 kbps as well as other emerging high-bandwidth bearers such as Wi-Fi, 3G and later 4G, the reality of access speeds equivalent or higher to that of a fixed-line scenario become evermore believable. This makes the terms of "*Mobility*" and "*Roaming*" become very important and necessary for everyone in their daily lives. Now people wish to obtain information easily in any location and at any time - this is the reason the term Mobility shows a tendency to become an important issue in our context of G-Monitor. In the future "*Networked World*", all devices will require the ability to communicate in a pervasive environment. In terms of usability studies, users will be transparent to the network protocol and systems will be designed in such a way that it is enabled to carry out network migration on its own without user intervention. However, in HCI there weren't any appropriate guidelines to deal with network transparency for users. In terms of Multitasking, mobile devices should be designed to enable them to accept input events at all times, even while executing commands. The successful desktop design, which enables multitasking capabilities, can be used as a guideline on designing mobile systems. A good example of multitasking that is currently supported in mobile devices is its "call waiting" capabilities.

2.2 Implication for Accessibility Guidelines for Navigation on Mobile Devices.

There is growing amount of legislation and guidelines relating to accessibility on the Web, which include the World Wide Web Consortium (W3C) and its Web Accessibility Initiative (WAI). However, most of the guidelines are difficult to apply in a mobile context since it is either too general or too specific (HTML) so that a wide range of products is not supported. In an ideal world, a mobile application or site needs to be designed to be informative, attractive and accessible from anywhere and anytime. To be able to deal with those issues, we need to involve an in-depth knowledge of how people interact with information by navigating through sites.

When designing mobile-enabled application such as the G-Monitor portal we should minimize the adaptation of the users existing mental model to reduce volatility when using the application. In practice, most people design applications in such way that users need to navigate by scrolling or paging since there is a huge amount of data being displayed. However, there are still a number of people designing mobile applications in a way that summarises text as well as selects keywords and reduces the amount of images. With this method, users can generally navigate for more information if they request. In general navigation will affect user experience and it is based on context as well as user needs. The issue of navigation will be discussed now**…**

Reflections on Small Screen Devices Since the display size of mobile devices such as the PDA is much smaller than that of desktop computers, the amount of information that is visible at one time is dramatically reduced. As a result, this will increase the number of pages, therefore making it necessary that content be reorganized or modified in order to suit small screen devices. Jones et al (1999) found that path lengths were shorter for small screen users than users who used the normal sized computer screens. Besides that, we found that small screens with short lines slow down the speed of reading by disrupting the normal pattern of eye movements. Moreover, since the amount of information that can be displayed on the screen at one time is limited, much time is spent manually

scrolling or paging the text. Also, we found that the organisation of the content and the purpose of the service are key issues in determining the usability of mobile services such as in G-Monitor. From the previous implementation of G-Monitor, we argued that we should reduce the amount of information and to make the content more focused based on the "Value-Focused" approach, whereby we should provide main core navigation based on user wishes, concerns, problems, and values in mobile applications. When we redesign our G-Monitor layout, we found that constantly visible navigation bars require too much display space and reduce the amount of space for context. Inside the article "End of Web Design", Jakob Neilson also claims that we shouldn't spend screen space on navigation except when absolutely necessary. We do agree with his approach, but from our point of view, although context is the main priority for mobile devices we still need to spend time investigating how to make a reusable, shortcut and meaning navigation for users in mobile devices without spending extra screen space. We found many research groups trying to use transparent widgets, separate navigation layers, and rapid serial visual presentation RSVP of text techniques to implement navigation for browsing on small displays. This is a good sign in the Mobile Device arena.

Accessibility Guidelines on Navigation Refer to Web Accessibility Guidelines for Navigation section, we summarized the important issues into 3 sub categories, ❶ Be sure that all links indicate that they are links. ❷ Be sure that all links clearly indicate their destinations.❸ Provide a search facility or index for direct access to content. In the real world, users prefer an easy way rather than a hard way for navigation, since humans are mostly lazy and they prefer all devices to work as in their mental model without having to relearn. Navigation in the context of mobile devices is the activity by which users move or navigate around the facilities that are provided on the mobile devices. Ease of navigation is the second most important usability issue of websites (2^{nd} priority) [1], after performance. However, for mobile device guidelines, I argued that it should be categorised as the 1^{st} priority since a well-designed application or site should allow users to navigate quickly and efficiently to the content. This makes it easy for users to figure out where to find a specific piece of information without getting lost or sidetracked. Although Context is important for users in mobile environment, but without a proper

form of navigation, context will be hidden in user perspective. Having good navigation can improve user experience on the product. A good interface of navigation will help users build a mental map of the screen structure of the application or site. This will provide ease-of-use paradigm on the application itself and reduce the amount of time users spend on navigation when using the application.

Chapter 3: A new G-Monitor Architecture

3.1 Previous Architecture.

The advent of the previous version of G-Monitor was to overcome problems and limitations associated in using interfaces with heavy bandwidth requirement. It was host by Apache web server and programmed using Perl that called by Apache's CGI module. Therefore, it only supports access through web browser only. In addition to that, the current implementation is based on Design Centric Model. It was fully tested with the Nimrod-G server and Gridbus Broker whereby the transport modules were implemented using IO-Socket module, Net-FTP-Common module, and Net-Telnet module. Besides that, previous version stores information locally on the G-Monitor server as a normal plain text format to remembers user preferences and we always encounter data integrity problem. Hence, in the our new framework, we stores every single user information and their preferences in MYSQL databases since it store information in more secure manner and provides SQL facilities. The previous implementation, take the advantage of Perl scripting since it provide a free GD graphics library to generate graphs in Portable Network Graphics, PNG format and open source Telnet and FTP Module. In our new framework, we provide access through different type of devices while maintaining the capability of lightweight interface and provide multilingual capabilities. The previous G-Monitor framework is shown in figure 5.

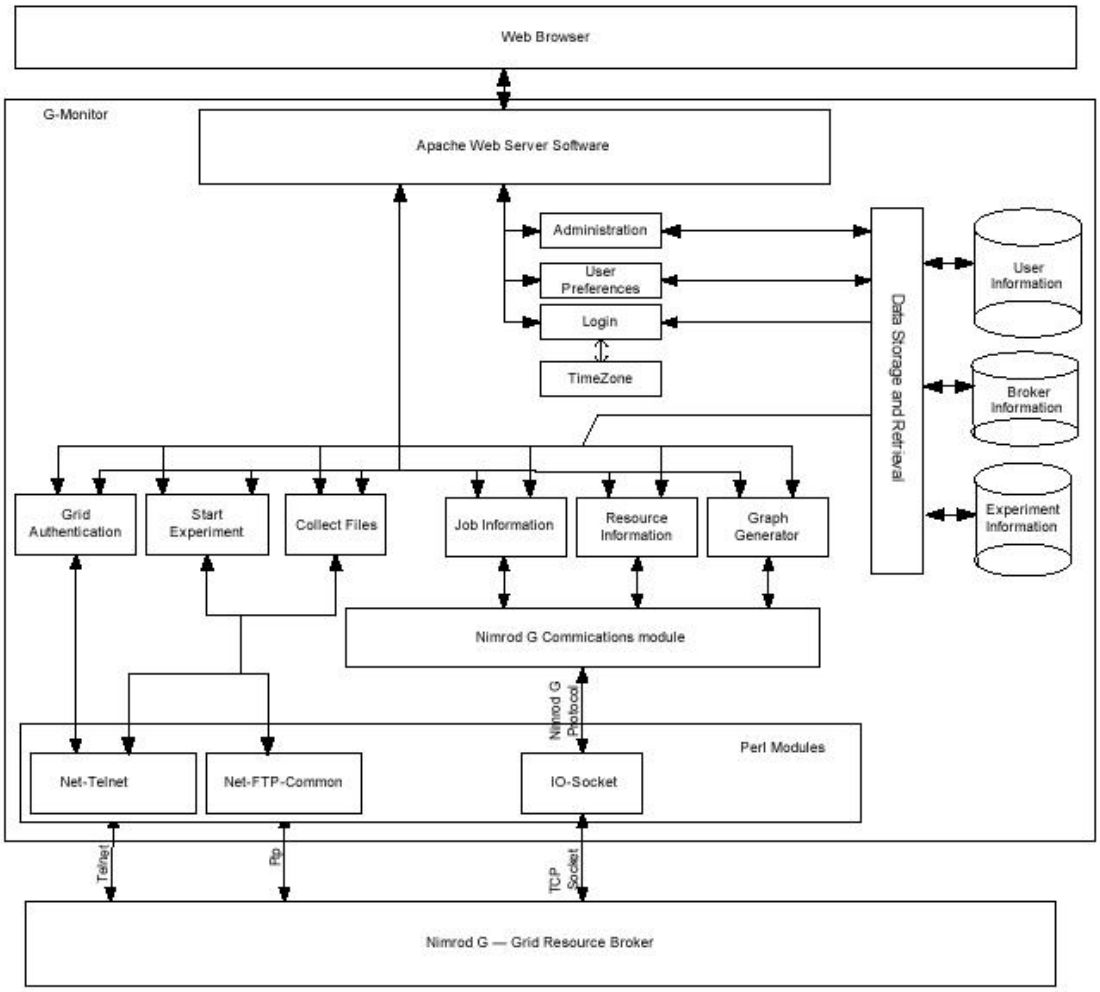

Figure 5: Previous G-Monitor's Architecture

3.2 New Framework with Globalisation capabilities.

In our new framework of G-Monitor, we decided to keep the existing Apache web server since it provide the Apache SOAP module for the Web Services functionality. Since, our new framework switch from pure Perl language to Java and X-Family technologies such as XML, XSL and XPath, we have to use Tomcat server resides behind Apache Web server to deploy JSP pages as the web module in server side. In testing stage, we are using Tomcat version 3.2.3 rather than Tomcat 5 because it easy to check error from the server console. Once we finalise our framework and everything, we will deploy our whole portal under Tomcat 5, known as production server. Figure 6 shown how this new framework being divided into component level and how these components interact among themselves. The first level is front-tier level since it consists all type of devices that are able to use our portal services. In this stage, we currently only targeted on PDA, Web Browser, WAP phone and J2ME enabled devices. Our next target client is .NET Web Services enabled application since we wish to provide services that enable any applications to use our API through Web Services call. Our Web Services will accepts all requests from others systems across the Internet or an Intranet or mediated by lightweight application. This will allows any network-enabled system interact with our backend Web Services module for our structured XML data generated by MiddleNet Module which extracting information from Gridbus Broker. Eventually system integration through Web Services will happen dynamically at runtime but now Just-in-time integration will herald a new era of B2B integration over Internet. This is why we need to deploy Web Services component inside our current framework. Besides that we have a Multilingual Content Transformation module to transform our information into the targeted devices that defined in front-tier. In session 3.2.2, we will explain all the Business & Communication Logic Component and Server-Side Components.

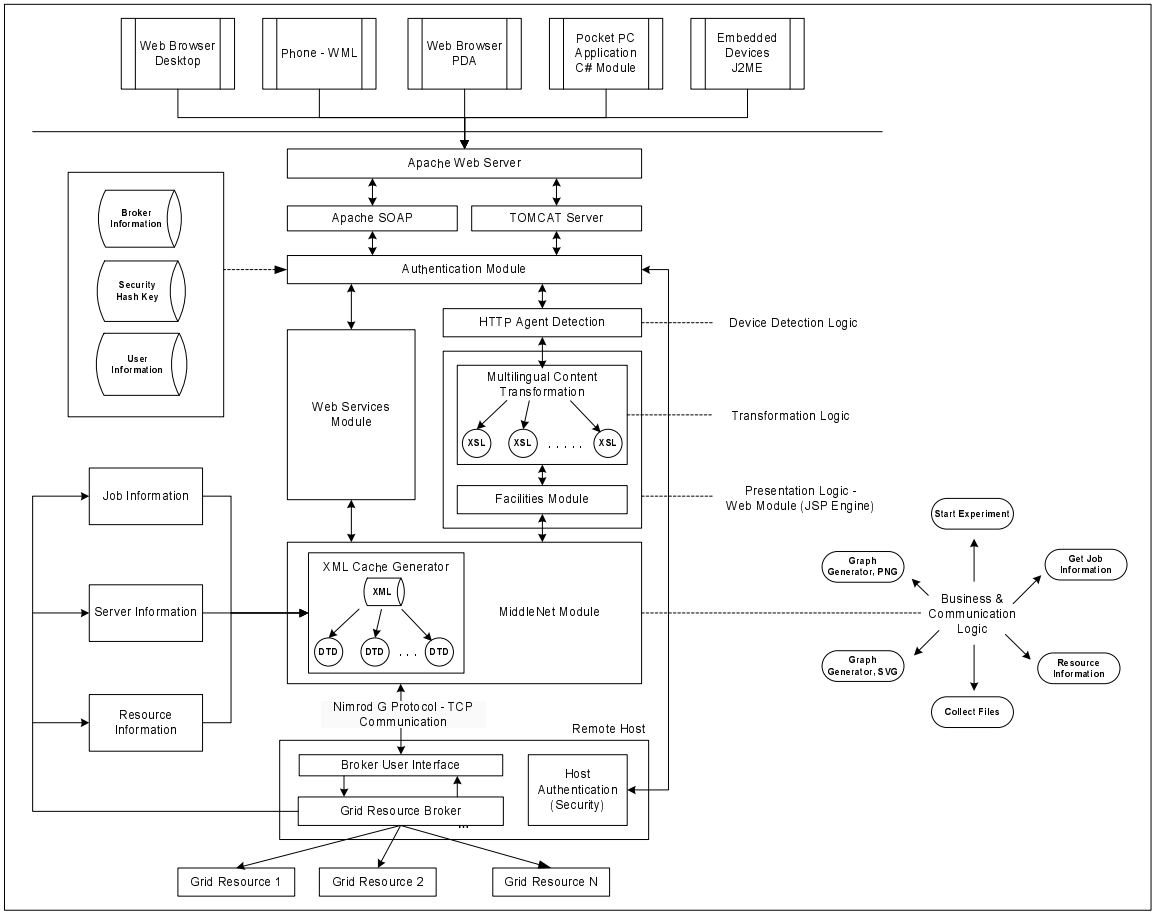

Figure 6: New G-Monitor Architecture

3.2.1 Globalisation Architecture.

How can globalization features enhance the users experience customer when using G-Monitor portal? The intention of G-Monitor was to provide a ubiquitous user-interface that hide low-level details of accessing Grid services for deployment and execution management of applications. In general, scientists or researcher as a group in all around the world with different language background are much more likely to visit and use this portal if they can read its contents easily. Hence, this portal should serve multiple languages and providing users with globally enabled portal, as it is not use by English like people only. In order for us to implement multilingual for G-Monitor portal, we decided to use the Unicode technology to encode XML data to enable Java's capabilities for multilingual support, since it is the universal character-encoding scheme for written characters and text in a consistent way for handling multilingual text interchange internationally. At the very least, this portal must be able to:

- Provide jobs, resources, brokers information and registration form in the user's language and cultural setting (for example, date format or currency symbol), based on the browser's setting or the user's selection.
- Accept input in the user's language and cultural setting.
- Users can choose language and cultural preferences.
- Information should be store and display on the user's own language and intended format (for example, name and address).

In globalisation the word "locale" was borrowed by software engineering from geography to indicate that the distribution of human cultural expectations of computer behavior fall into clusters that can be grouped together, most commonly by language and country or region [25]. We used the concept of "locale" to classify different cluster to different language specification and cultural specification. Moreover, we have to study how to handle numbers in different cultural or locale even if the decimal system (base 10) is used in almost every country of the world but we found number format vary considerably and in some important context, traditional (non-decimal) numbering systems still in used. In our context, currency format becoming an important issue for us

to handle and different countries or regions have different formats and rules for currency, for example in Europe is called Egyptian Pound, US is called US Dollar, China is called Yuan Renminbi, Malaysia is called Ringgit Malaysia. Besides the currency format, we found that there are currency separators rules that we need to follows, for example Pound used apostrophe for thousand separator while US, China and Malaysia used comma for their thousand separator. In globalization point of view, input and output of multilingual data is an important issue but in our context we only emphasize on output since we have no interest on handle complex input, especially Chinese. The reason behind is we do not want to spend time on researching on language aspect and come out with some grammar checking capabilities in our portal to verify user input. Currently we are in the progress of researching how to fully utilize the ICU libraries that maintained by IBM for the benefit of IBM and its customers. This library makes us easy to add robust Unicode and internationalization support to various aspects in our context. We are planning to use:

- Calendar support
- Character set conversions
- Number and currency formatting
- Date and time formatting
- Time zones

For more information on ICU4J, please refer to http://oss.software.ibm.com/icu4j/ and for ICU4C, please refer to http://oss.software.ibm.com/icu/

3.2.2 Web Server Tier.

Below describes all components inside our framework and its details description. Table 1 describes all Business & Communication Logic Component. While, Table 2 describes all Server-Side Components reside in our framework.

Table 1: Business & Communication Logic Component

Business & Communication Logic Component	Description
Authentication	Responsible for Grid Proxy Authentication, telnet remote broker and issue grid-proxy-init commands based on user specification
Start Experiment	Provides the user with the functionality to start experiments on the broker using existing files or upload on real time from those supported G-Monitor interfaces.
Collect Files	Upon completion of experiment the user is able to download their resultant experiment files from the broker.
Get Job Information	List all the job status and description to the user.
Resource Information	Lists all the available resources to the user.
Graph Generator, PNG (Existing) - Currently act as the default graph generator.	Runs constantly on the actively monitored experiment and any experiments the user wishes to monitor in the background. The scripts responsible poll the Broker for the current experiment status at user specified interval and log it in a file. This file is then used to generate experiment graphs.
Graph Generator, SVG	Generate graph based on the job status and resource status at requested time in Scalable Vector Graphics (SVG) format. This is another application of XML that designed to create two-dimensional graphics using XML. SVG 1.1 received the W3C recommendation status on Jan 2003 and SVG 1.2 is currently in the Working Draft status. Since SVG 1.1 is so new in W3C, only mozilla browser supported with the plug-in. In order to view SVG graphs on unsupported browser, you have to download SVG Viewer from Adobe Web Site [22]. Currently, we still testing on the capabilities and usefulness on the SVG Graph Generator.

Table 2: Server-Side Components

Server-Side Components	Description
Device Detection Logic - HTTP Agent Detection Tier	Detect the supported MIME type and forward it to Transformation Logic module. Please refer to details explanation at "Identifying the client's capabilities to serve the proper content" in Web Server Tier section.
Transformation Logic – Multilingual Content Transformation Tier	Content all the design of presentation tier for supported client in XSL format. Moreover, it is a bean that due with the issue of Globalisation and usage of locale model.
Presentation Logic - Facilities Tier	This tier provide interface to Grid Resource Broker by given Facilities for users to invoke!
Presentation Tier or Web Module	Is a presentation tier for client! This tier is sub divided into 3 sub tier that contains HTTP Agent Detection Tier, Multilingual Content Transformation Tier and Facilities Tier
XML Cache Generator	This tier only generate XML content that response from Grid Resource Broker when user invoke functionality in facilities tier. Please refer to details explanation at "XML Cache Generator or Generating XML content" in Web Server Tier section.
Web Services Module	Provide Web Services interface to Grid Resource Broker via MiddleNet bean. This services use by C# .NET Mobile Web Services Application or any type of Web Services Client.

Chapter 4: Design

4.1 Data Centric Model VS Design Centric Model.

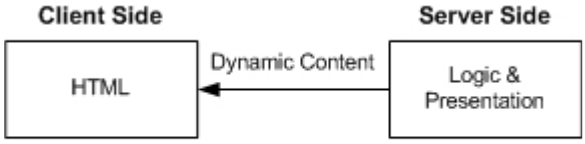

Figure 7: Previous G-Monitor Implementation

Problem:

Combination of design code (for example, HTML, WML and etc) + Script provides no separation of logic and presentation. Previous G-Monitor Implementation was design based on Design Centric Model, which shown in figure 7.

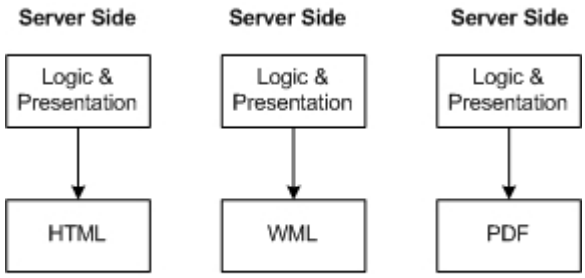

Figure 8: Design Centric Model

Most of the web site written in these technologies (Perl, PHP & etc) will have problem in developing website that serve one content for multiple different type of client or design. Since these technologies need to be code in such way that business logic, content (data) and design (style) of the document need to be mixing together (Design Centric Model). The previous implementation of G-Monitor is the example of problem that we encountered since it developed using Perl scripting language whereby it mixes the presentation tags with content. Based on previous implementation, if we going to develop the portal to be able to access from different type of devices which illustrated in Figure 8 or even multilingual portal which followed the Design Centric Model, we have to write 3 pieces of code that combine logic, data and design.

Moreover, it needs more times to do separation between data and design when doing maintenance or re-design the portal. Besides that, encapsulate the data from the site is very hard and can be costly if the complexity increase. Hence, this will duplicate our work and effort when develop such a portal that support accessing from different type of devices, that illustrated in figure 9.

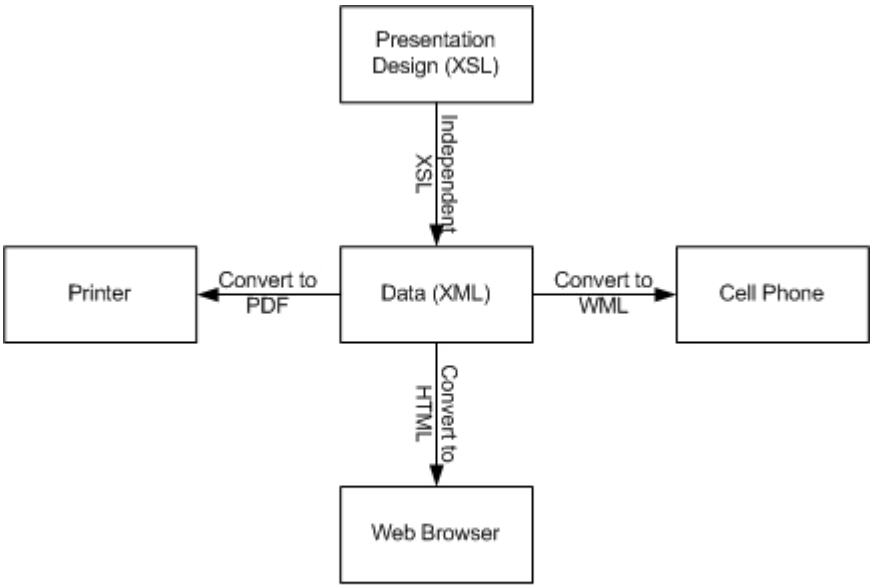

Figure 9: XML is independent from medium

Solution: With the use of XML, XSLT & JSP technology is capable of overcome the problem on designing site that can support different type of devices that follows Data Centric Model paradigm rather than Design Centric Model. Java technology, Java Beans enables the creation of platform-independent business logic modules in backend and JSP enables to act as a controller in MVC model, while XML complements java by enabling the creation of platform-independent business data that can be encoding in Unicode, UTF-8, Big5 and etc. Since Java technology and XML are built on Unicode, programs and data can be fully internationalised. Besides that, both technology enable software design based on loose coupling where by reduces restrictions and eliminates similarity requirement between cooperating system and avoids problems that occurs as a result of foreseeable changes to the software. In addition to that, XML is the best technology to fit with Globalisation architecture.

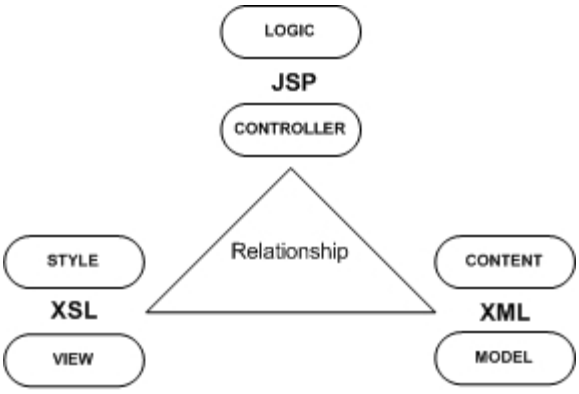

Figure 10: "XML, XSL & JSP" Model

Our design methodology, "XML, XSL & JSP" model shown in figure 10 follows the Model-View-Controller model which allows web sites to be highly structured and well-designed, reducing duplication efforts and site management costs by allowing different presentations of the same data depending on the requesting client (HTML clients, PDF clients, WML clients) and separating out different contexts with different requirements, skills and capacities. Our framework is based on separation of:

1. Content
    - Information of the real content
2. Style
    - Presentation, look & feel
3. Logic
    - Business Logic

This Model is similar to Model-View-Controller (MVC) model, whereby XML document acts as the Model, and the JSP in the Web Server Tier as the Controller, and the XSL file is the View. Then, by changing only the XSL file, different views can be generated. Besides that it follows pure Data Centric Model rather than Design Centric Model. Benefit of our "XML, XSL & JSP" model that shown in figure 10 is easily extend to support Globalisation capabilities by providing same set of content or model in different language and different set of style or view based on cultural needs on supported locales, as shown in figure 11. The difference from previous implementation of G-Monitor is that cultural and language-independent program code calls cultural and language-dependent information at runtime, thereby greatly reducing the expenditure of cost and effort otherwise invested throughout the product life cycle.

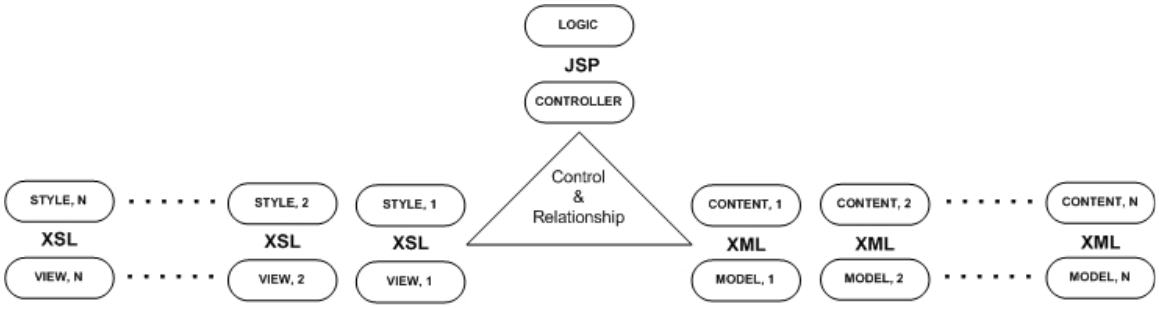

Figure 11: "Single Controller control multilingual transformation & multilingual data"

4.2 Identifying the Client's Capabilities to serve the proper content.

The content dynamically generated by the JSP in facilities module or in Presentation Logic must be in a format supported by the requesting client. In our case, our G-Monitor portal is able to access from different type of clients (for example: Desktop Web Browser, PDA Browser, WAP Phone and J2ME). Hence, our "`HTTP Agent Detection`" is responsible to identify the capabilities of a requesting client in order to respond in a proper format. The selection of the proper content type is made according to a configuration table (`MIMETypeMap.properties`) that maps a given category of clients to a particular MIME type. This MIME type is used as the response content type. The two steps are:

1. **Identify the client and its capabilities**.

   The identification of the capabilities of a requesting client is based on the values of both fields of the HTTP request header: `User-Agent` and `Accept`. According to the specifications of HTTP 1.1, the User-Agent field can contain product tokens (i.e., product names and versions) and comments identifying the requesting client. While Accept field in the header contains the lists of MIME types accepted by the client. All of the below results are taken in the testing stage of our new G-Monitor portal. For example, when issuing a request from Internet Explorer 6.1, Mozilla 1.5 and Nokia 5100, the User-Agent field is respectively:

```
UserAgent (Internet Explorer 6.1):
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)

UserAgent (Mozilla 1.5):
Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:1.5) Gecko/20030925

UserAgent (Nokia 5100):
Nokia5100/2.0 Profile/MIDP-1.0 Configuration/CLDC-1.0
```

For example, when issuing a request from Internet Explorer 6.1, Mozilla 1.5 and Nokia 5100, the Accept field in the header is respectively:

**** The token */* stands for any MIME type***

Accept (Internet Explorer 6.1**):**
/

Accept (Mozilla 1.5**):**
text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1

Accept (Nokia 5100**):**
text/vnd.wap.wml, application/vnd.wap.wmlscriptc, application/vnd.wap.xhtml+xml, application/xhtml+xml, application/java

2. **Map the client to a MIME type**.

   We design the framework in order to for the configuration file `MIMETypeMap.properties` able to add more supported MIME types in the development time without recompile the our "`HTTP Agent Detection`" class or even recoding. This configurable file contains user agent categories with its associated MIME types, which located externally from the "`HTTP Agent Detection`" bean. This technique allows the definition of generic and specific mapping rules, starting with the most specific and ending with the most generic. The process of mapping the client to a MIME type as follows and shown as UML Sequence Diagram in figure 12 and 13:

   1. "`HTTP Agent Detection`" retrieves the value of the User-Agent header field from the request.
   2. For each product token, "`HTTP Agent Detection`" extracts the fully qualified product name (i.e., the product name and its version) and the product name by itself.
   3. "`HTTP Agent Detection`" looks up the corresponding MIME type in the configuration file `MIMETypeMap.properties`, starting with the fully qualified name. If this fails, it proceeds with the product name; if this fails it iterates again on the next product tokens

4. If none of the product tokens can be mapped to a MIME type, it gets the value of the Accept header field from the request. Then, for each advertised accepted MIME type, it looks up the corresponding MIME type in the map. Ultimately, it returns the default MIME type: `text/html`
5. The return of the response MIME type will be forward to "Multilingual Content Transformation" bean to further process on the transformation.

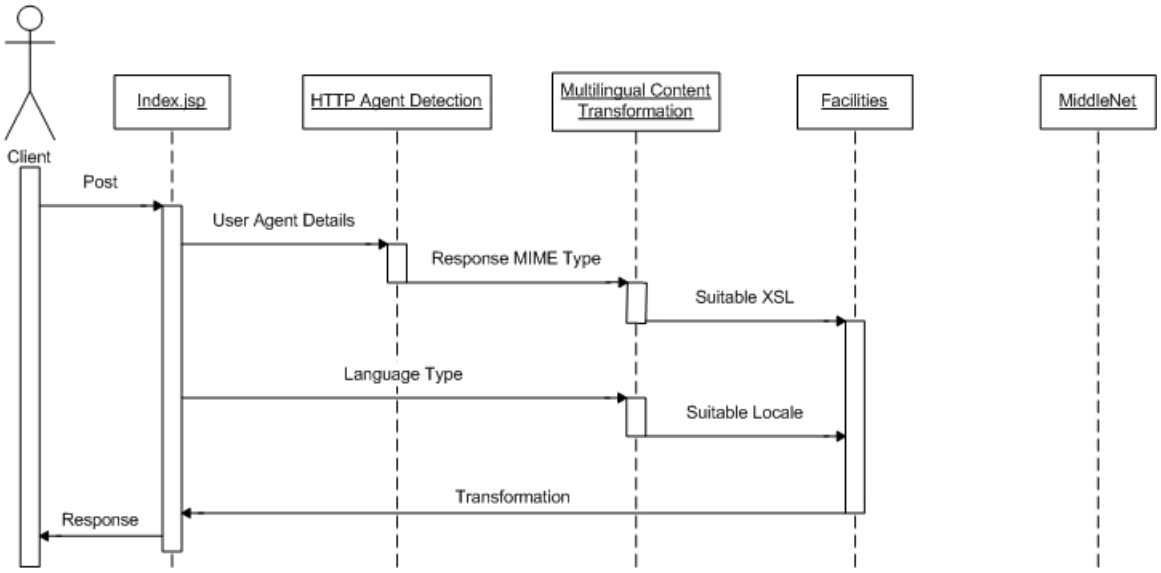

Figure 12: Detect type of client that makes request

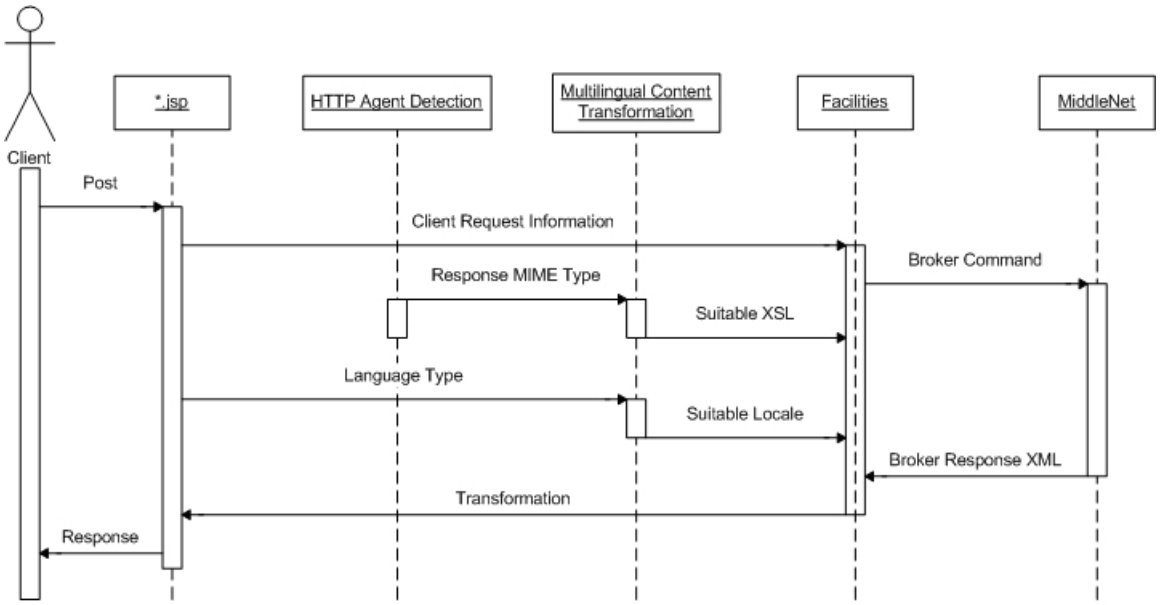

Figure 13: Normal Client Request

4.3 XML Cache Generator or Generating XML Content.

In our application, XML Cache Generator that resides in MiddleNet bean is used only to generate the XML response content from Grid Resource Broker. The XML content for each response is generated:

1. By constructing the corresponding DOM (Document Object Model) tree according to the corresponding DTD (Document Type Definitions).
2. By serializing it to generate the XML stream or output as XML file type.

This approach ensures a clean generated document without having problem of unclosed tags when writing a java program to generate XML stream or file. Furthermore, having the XML documents represented internally as a DOM tree allows for more effective post-processing, in our case is applying style sheet. MiddleNet bean will generate the DOM trees when the results return from the Grid Resource Broker using the Nimrod-G protocol by calling its `data2XML (String inputStream, String type)` method. Once a DOM tree is created, it is serialized on the response output stream to generate the actual response that apply with appropriate style sheet. The whole scenario is illustrated in figure 13.

Chapter 5: Implementation

5.1 Cross-Technology Implementation.

5.1.1 JDBC-MYSQL.

In this framework, we using MYSQL database to store sensitive information such as broker details and user information in backend system rather than just flat file. Hence, we using JDBC native driver to communicate with MYSQL database since the backend framework was programmed in Java. The following are the functionalities that we chose to used to optimise the communication between JDBC native driver with MYSQL database:

1. Prepared Statement
    - In Most Cases, SQL Statement will be sent to DBMS right away, where it will be compiled. As a result, the PreparedStatement object contains not just a SQL Statement, but also an SQL statement that has been precompiled. This means that when the PreparedStatement is executed, the DBMS can just run the PreparedStatement's SQL statement without having to compile first.
    - It reduce execution time even if my backend java code execute Statement object many times
2. Using Transactions
    - It allows SQL statements to be group together for execution as a unit and helps to preserve the integrity of the data in a table.
    - We use transactions when we need to add entry simultaneously for "brokers" and "usersbrokers" table.

5.1.2 Java Server Pages.

Java Server Pages, JSP technology is an open, freely available specification developed by Sun Microsystems. JSP pages use JavaBeans technology as the component architecture to enable the separation between presentation tier and business tier. Moreover, it provides a number of capabilities that ideally suited for working with XML since it can use the full power of the Java platform to access programming language objects to parse and transform XML messages and documents. Below shown how we use JSP to import the XML parser and XSLT transformation engine inside my code:

```
<%@ page import="javax.xml.transform.*"%>
<%@ page import="javax.xml.transform.stream.*"%>
```

Furthermore, JSP technology provides an abstraction mechanism to encapsulate functionality for ease of use within a JSP page. **JavaBeans** is a portable, platform-independent java component model that lets us write components and reuse them everywhere in our framework. In this framework, we using JavaBeans as server side component to augment JSP pages because:

- Since this project is work for GRIDS Laboratory and Resource Broker [2] developed by Srikumar was programmed in JAVA. It makes system easy to integrate and able to reuse code.
- It allows separation of business logic and presentation logic. Thus, you can alter the way data is displayed without affecting business logic.
- As reusable components since it allows us to reuse the components in other application in within the framework.
- Protecting the framework intellectual property by keeping source code secure.

5.1.3 XML

XML stands for the e**X**tensible **M**arkup **L**anguage and developed by W3C (World Wide Web Consortium) as a subset of SGML, primarily to overcome limitations in HTML. HTML has more than 100 tags and supported by thousands of application including email, editor, browser and more. Even though HTML was so important and successful in Internet arena but it has some shortcomings and it has turned into a maintenance nightmare for W3C. HTML tags serve primarily purpose of describing how to display a data item, conversely XML enabled heterogeneous computing environment to share information over World Wide Web since XML tags describe the data itself or self-describing and it can be semantically independent or dependent. Hence it possible for program to interpret data in many ways, can filter the document based upon its content, can restructure it to suit the application's needs and so forth. The other advantage is XML concentrates on the structure of the document and this makes it independent of the delivery medium. In this way XML brings the same cross-platform benefits to information exchange as the Java programming language has for processing. Moreover, User-defined tags is another key advantage on developing new language based on XML standard, the good examples are WML and SOAP.

Below is example of how XML represents structure and semantic of a login page:

Structure for " `Login.xml` "

```
<?xml version="1.0" encoding="utf-8"?>
<Login>
 <form validate="mainMemberProcess.jsp" method="post">
   <input-element>
    <content>Member ID</content>
    <input type="text" name="frmMemberID"/>
   </input-element>
   <input-element>
    <content>Member Password</content>
    <input type="password" name="frmMemberPassword"/>
   </input-element>
 </form>
</Login>
```

Besides that, XML is used to represent the information or message given by Resource Broker via TCP communication channel or it is a data format that represents data in serialized form that can be transported over network from one endpoint to another:

Example of " <session_id>-getjobinfo.xml "

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE NimrodG_Command SYSTEM "Common-SingleInfo.dtd">
<NimrodG_Command>
  <getjobinfo>
    <jobID>j45</jobID>
    <status>ready</status>
    <host>none</host>
    <server>s3</server>
    <time>none</time>
  </getjobinfo>
</NimrodG_Command>
```

5.1.4 DTD

DTD stands for the **D**ocument **T**ype **D**escriptor (or an equivalent to XML Schema) and it is a schema for a set of XML Documents and used to describe the structure of XML documents. An XML document is valid if a DTD or XML schema is associated with it and if the document complies with that DTD or X-Schema. Without DTDs or their equivalent, XML will never reach its full potential because a tagged document is not very useful without some agreement among inter-operating applications as to what the tags mean. We use DTDs in our framework to check the validity of the XML data generated by MiddleNet module when ever it communicate with Grid Resource Broker using Nimrod-G (Broker User Interface or API).

For example when MiddleNet send a Nimrod-G command "`getdonejobs`" to Grid Resource Broker via TCP communication channel:

Example of issuing Nimrod-G Command "getdonejobs"

```
Processing Command: getdonejobs
Return results: j115 j139 j73 j155 j135 j31 j58 j81 j18 j145 j187
Nimrod-G Protocol: <job_0>whitespace<job_1>whitespace...<job_N>
```

Generated XML result must conform the Nimrod-G protocol using DTD as below:

"Common-SingleInfo.dtd"

```
<!ELEMENT NimrodG_Command (getjobinfo|getserverinfo)>
<!ELEMENT getjobinfo (jobID,status,host,server,time)>
<!ELEMENT getserverinfo (serverID,host,user,status)>
<!ELEMENT jobID (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT host (#PCDATA)>
<!ELEMENT server (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT serverID (#PCDATA)>
<!ELEMENT user (#PCDATA)>
```

“Common-Jobs.dtd”

```
<!ELEMENT NimrodG_Command
(getdonejobs|getexecjobs|getfailjobs|getreadyjobs|getjobs)>
<!ELEMENT getdonejobs (jobID*)>
<!ELEMENT getexecjobs (jobID*)>
<!ELEMENT getfailjobs (jobID*)>
<!ELEMENT getreadyjobs (jobID*)>
<!ELEMENT getjobs (jobID*)>
<!ELEMENT jobID (#PCDATA)>
```

5.1.5 XSLT

XSLT stands for the e**X**tensible **S**tylesheet **L**anguage **T**ransformation and it is a mechanism to convert an XML document from one schema to another. A stylesheet specifies a number of template-matching rules and applies them in a recursive tree-traversal similar to the Document Object Model (DOM) paradigm. An XSLT engine can use this stylesheet to transform XML documents to another format (such as WML, SVG, PDF and etc) depending on what kind of client made the request. For example, if a request coming from a web browser, it might be returned in XHTML or HTML format, while the same request coming from a wireless device, it might be returned in WML. An XSLT stylesheet's syntax is very expressive and contains a full repertoire of loops, conditionals and mathematical expressions, along with function-like constructs and the concepts of scope and recursion. In our implementation, we used the following nodes of an XSLT stylesheet:

- `<xsl:stylesheet>`
- `<xsl:template match="" name="">`
- `<xsl:element name="">`
- `<xsl:attribute name="">`
- `<xsl:apply-template select="">`
- `<xsl:text>`
- `<xsl:value-of select="">`
- `<xsl:for-each select="">`
- `<xsl:call-template name="">`
- `<xsl:if test="">`
- `<xsl:choose>`
- `<xsl:when test="">`
- `<xsl:otherwise>`
- `<xsl:sort>`

In this framework, we used Java API for XML Processing (JAXP), which enables applications to parse and transform XML documents using an API that is independent of particular XML processor implementation. Below is part of the code that we extracted

from the framework. The first line shown how we create an instance of the `TransformerFactory` class and used it to create a Transformer Object that reads the XSLT stylesheet and convert the templates within it into a Templates object:

```
// XML Transformation inside JSP

TransformerFactory tFactory = TransformerFactory.newInstance();

Transformer transformer = tFactory.newTransformer(new
StreamSource(xslFile));

transformer.transform(new StreamSource(xmlFile), new
StreamResult(out));
```

Below is one of the XSL that apply in this framework that works as Parameter Builder and Form Handler:

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

      <!-- ******* Parameter Builder ******** -->
      <xsl:template name="paramBuilder">
        <xsl:param name="name" />
        <xsl:param name="value" />
        <xsl:if test="string-length($value)>0">
            <xsl:value-of select="$name" />=<xsl:value-of
             select="$value" />,
        </xsl:if>
      </xsl:template>
      <!-- ******* Form Handler ********* -->
      <xsl:template name="formText">
        <xsl:param name="size" select="'20'" />
        <xsl:param name="name" />
        <xsl:param name="value" />
          <input type="text" name="{$name}" size="{$size}">
             <xsl:attribute name="value">
                  <xsl:value-of select="$value" />
             </xsl:attribute>
          </input>
      </xsl:template>
      <xsl:template name="formTextArea">
            <xsl:param name="rows" select="'5'" />
            <xsl:param name="cols" select="'40'" />
            <xsl:param name="name" />
            <xsl:param name="value" />
            <textarea name="{$name}" rows="{$rows}" cols="{$cols}">
                  <xsl:value-of select="$value" />
            </textarea>
      </xsl:template>
       ..... More
</xsl:stylesheet>
```

5.1.6 Session State

We implement session state component to keeps track the session between users and the portal since it consists of caches and session information associated with a particular session initiated by user when the login to the portal. This technique enables the portal to identify a user across more than one page request or visit to the portal. The current implementation of session between HTTP client and HTTP server contains information about environment cache, a credentials cache, session ID, member information and device type which can be added using interface that provided by JSP or Servlet container. This information is important when setup user environment with portal environment since access to any G-Monitor functionality, including the credentials file is controlled by a session ID generated by JSP runtime-code. Moreover, bind objects to sessions allowing user information to persist across multiple user connections. The session will valid as long as users keep using the same terminal or browsers when using the services after they logged in. The session ID is large numeric sequence of random numbers that is extremely hard to falsify. The current implementation of session ID is constructed by function that provided inside `java.lang.String` package that called `getId()`. The identifier is assigned by the servlet container and returns a string containing the unique identifier assigned to this session. If any time, the session ID from the browser doesn't match the saved session ID, the session is terminated. Thus, a user's session can be compromised only if he/she communicates his session ID to an intruder and keeps the session valid by interacting with the portal.

5.1.7 Post-Processing for Output Customisation and Filtering

This portal was designed based on selective data dissemination where specific XML data is selectively generate directly from the response of Grid Resource Broker or relayed to a large number of distributed clients. This XML data will share by heterogeneous medium thus transformation between medium-specific XML formats are necessary. Therefore, we used XSLT stylesheet to transform entire XML stream or document from one medium format into another and uses XPath Query to control the part of the XML tree to be transform. Moreover, we also use XPath Query to customise the XML and filtering multiple XML data for later post-processing (such as medium design transformation, data for business logic processing and data from Web Services Querying) since XPath was proposed by W3C as a standard for addressing parts of XML and as a filter specification language. XPath enables us to refer to specific section of XML documents since it get its name from its use of path notation as in URLs for navigating through the hierarchical structure of an XML document. In XPath language, paths can be specified as absolute paths from the root of the document tree or as relative paths from a known location. A query path expression consists of sequence of one or more location steps. The hierarchical relationship between the nodes are specified in the query using parent-child "/" operators and ancestor-descendant "//" operators. For example the below XPath query is taken out from one of the XSL transformation files that addresses all `jobID` element descendants of all `getfailjobs` elements that are direct children of `NimrodG_Command` (root) element in the document (`$documentName1`):

```
select='document($documentName1)/NimrodG_Command/getfailjobs//jobID
```

This powerful language also allows the use of a wildcard operator "*" that matches any element name at a location step in a query. Moreover, it allows us to include one or more filters to further improve the selected set of nodes. Each filter is a predicate that is applied to the element(s) addressed at that location step and all filters at a location step have to be evaluated as TRUE in order for the evaluation to be continue to the descendant location steps. For example, consider the query that extract from below figure:

```
test="$pageNumber &lt;= $PCounter and $pageNumber &gt; 0"
```

This query will return TRUE if the variable pageNumber is less or equal to variable PCounter and variable pageNumber must greater than 0. For another example, consider the query of:

```
//NimrodG_Command/getfailjobs/jobID > 20
```

This query will selects all jobID elements where jobID greater than 20.

The advantage of XPath in XSLT implementation allows us to easily implement customisation and filtering capabilities on top of our Grid-Monitor framework. This customisation and filtering allows us to post information to multiple devices based on the devices information retrieval capabilities. Besides that we can easily transform the unstructured data from Grid Resource Broker into devices based oriented structured data. The meaning of devices based oriented structured data in our context is structure-based information that based on the constraints of the devices itself. For example, our portal will filter all unnecessary information before post the customise information to mobile devices in order for them to render it. The below code “XPath Query – Dynamic Job Table Generator” is one of this portal’s code that exemplified the power of the XPath language and XSLT language in our context.

“XPath Query – Dynamic Job Table Generator”

```
<xsl:template name="tableGenerating">
 <xsl:param name="TCounter" /><xsl:param name="PCounter" />
 <xsl:param name="JobType" /><xsl:param name="size" />
 <xsl:param name="NoColumn" />

  <xsl:if test="$pageNumber &lt;= $PCounter and $pageNumber &gt; 0" >
  ...
    <xsl:for-each select="jobID">
     <xsl:sort order="ascending" data-type="number" />
      <xsl:if test="position() &lt;= ($TCounter * $pageNumber) and position()
       &gt; (($pageNumber - 1) * $TCounter)" >
      <xsl:choose>
        <xsl:when test="position() mod $NoColumn = 0" >
         ...
          <xsl:value-of select="substring(substring-
           before(substring($JobType,4,string-length($JobType)),'Jobs'),-
           ((string-length(substring-before(substring($JobType,4,string-
           length($JobType)),'Jobs'))-2)),string-length(substring-
           before(substring($JobType,4,string-length($JobType)),'Jobs')))" />
         ...
         <xsl:if test="position() != ($TCounter * $pageNumber)" >
          ...
         </xsl:if>
        </xsl:when>
        <xsl:otherwise>
         ...
         <xsl:value-of select="substring(substring-
           before(substring($JobType,4,string-length($JobType)),'Jobs'),-
          ((string-length(substring-before(substring($JobType,4,string-
           length($JobType)),'Jobs'))-2)),string-length(substring-
           before(substring($JobType,4,string-length($JobType)),'Jobs')))" />
         ...
         <xsl:if test="position() = $size" >
            <xsl:call-template name="TDGenerator">
             <xsl:with-param name="PCounter" select="$NoColumn - ($size mod
              $NoColumn)" />
              <xsl:with-param name="temp" select="1"/>
            </xsl:call-template>
         </xsl:if>
        </xsl:otherwise>
      </xsl:choose>
      </xsl:if>
    </xsl:for-each>
   ...
  </xsl:if>
</xsl:template>

<xsl:template name="TDGenerator">
 <xsl:param name="PCounter"/>
 <xsl:param name="temp"/>
   <xsl:if test="$temp != ($PCounter + 1)">
       ...
            x  
       ...
       <xsl:call-template name="TDGenerator">
           <xsl:with-param name="PCounter" select="$PCounter" />
            <xsl:with-param name="temp" select="$temp + 1"/>
       </xsl:call-template>
   </xsl:if>
</xsl:template>
```

Chapter 6: Experimental Results

6.1 Client Side.

Our portal can be used on any type devices that supporting mark-up language such as HTML, XHTML, WML, XML and SVG. Plus in future, it also supports Web Services client (C# application client) and J2ME client. After we study on HCI approach, we came out with a design that suits all type of user including expert and novice user. Moreover, since our framework is based on Globalisation architecture, our client will receive the appropriate type output based on cultural, location, time and language. Rather than that, the user can change their user preferences and customise the interface portal. Figure 14 is shown the sample of Web Browser portal user interface. Besides that, our portal also supports WAP Client and tested with Nokia 5210 SDK, which developed by Nokia. The limitation of mobile screen size makes PDA widely used. Therefore, our portal can be access through PDA browser or PDA client application. In our current framework, PDA client application meant to be the Client version of Web Services application and it is still under research. Figure 16 shown the screen shot of the PDA user interface when accessing portal services.

6.1.1 Desktop Web Browser.

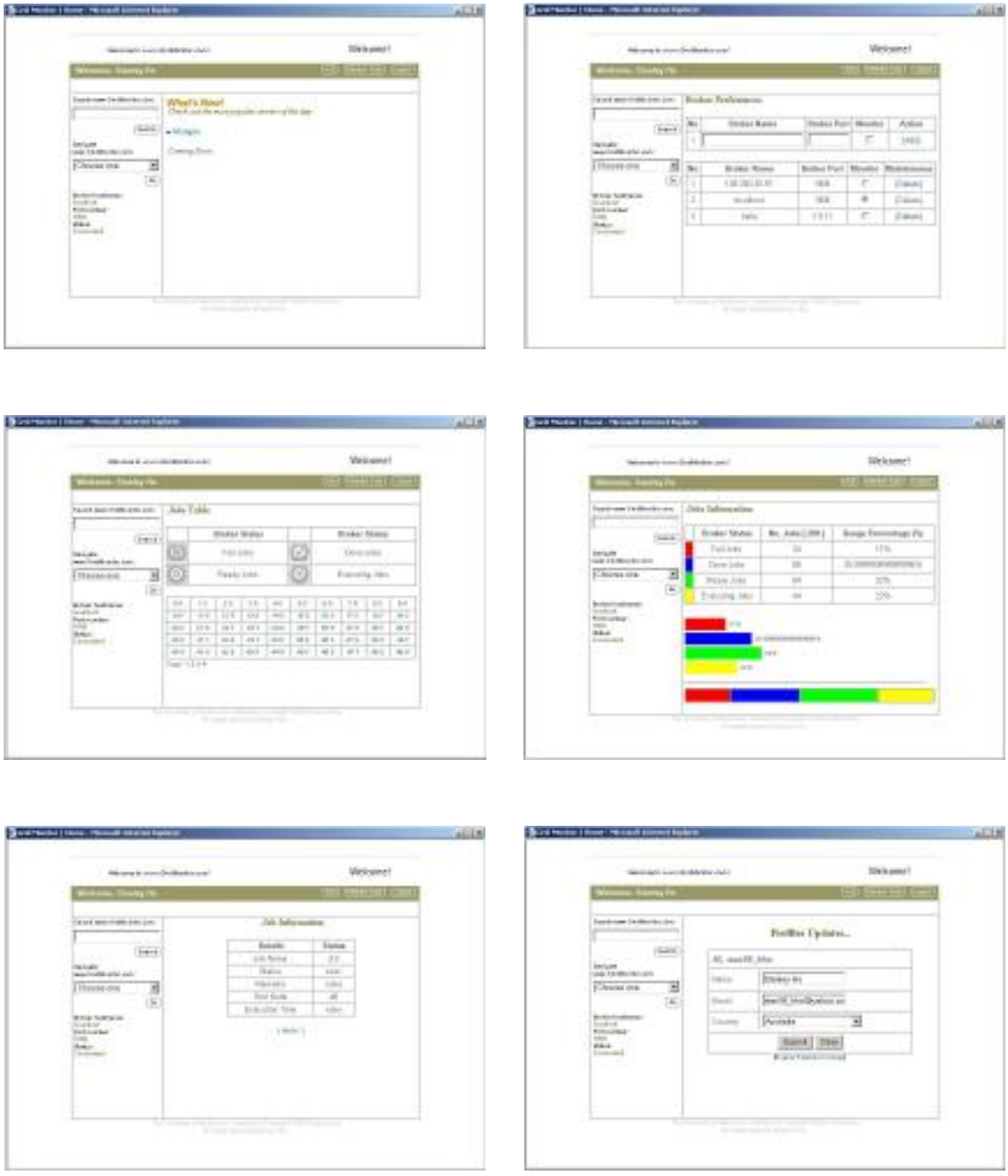

Figure 14: Screen shot from Desktop Web Browser

6.1.2 WAP Phone

We tested on Nokia 5100 SDK which running on Windows 2000. This SDK is provides full Nokia WAP phone API. This SDK is meant for developer like us to use in testing stage. More details specification of the Nokia Developer Toolkit, please refer to Nokia Forum: http://www.forum.nokia.com [29]

Figure 15: Screen shot from Nokia 5100 SDK

6.1.3 PDA Client

We tested on Compaq iPAQ H3900 Series which running on Windows CE 4.20.

Figure 16: Sample of PDA User Interface taken by digital camera

Chapter 7: Conclusion and Future works.

In this paper, we have proposed a Globalisation Framework (G-Monitor) for Managing and Monitoring Application Execution on Global Grids through “Multiple Devices”. Moreover, we also came out our methodology for Mobile Appliance that based on Human Computer Interaction discipline. All of these researches that we carried out was to improve the previous version of G-Monitor Architecture and came out with a more robust architecture that can coup the changes of the current technology. However, we are still in the earlier stage of having a full version of globalization portal due to the language barrier problem but our designed architecture was based on globalisation point of view. As the result of our effort, our portal can be access through most of the handheld devices and in the future we are targeting for more type of users and even Agent based application. The below are some of the future plan for our portal:

- Supporting Client based Web Services.
- 100% globalisation based portal.
- Support more language such as French and Japanese.
- Real-Time statistic report from the grid resource broker.
- Support Push message mechanism, send SMS or email to user when requested experiment is finished.
- Provide namespace for all the XML elements that we used. This will prevent duplication of same XML element name using by other people.

References:

[1] Martin Placek, and Rajkumar Buyya "G-Monitor Enhanced: A Web Portal for Managing and Monitoring Application Execution on Global Grids", Proceedings of the Internasional Workshop on Challenges of Large Applications in Distributed Environment (CLADE 2002), In conjunction with the 12th International Symposium on High Performance Distributed Computing (HPDC 2003), June 21-24, 2003 Seattle, USA

[2] Srikumar Venugopal, Rajkumar Buyya and Lyle Winton "A Grid Resource Broker for Scheduling Distributed Data-Oriented Application on Global Grids", IEEE Supercomputing Conference (SC 2004), NOV 2004, Pittsburgh USA.

[3] I. Foster and C. Kesselman (editors), "The Grid: Blueprint for a future Computing Infrastructure", published by Morgan Kaufmann, USA 1999"

[4] R. Buyya, D. Abramson, and J. Giddy, "An Economy Driven Resource Management Architecture for Global*" Computational Power Grid*s, Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA., June 2000.

[5] M. Thomas, S. Mock, J. Boisseau, M. Dahan, K. Mueller, D. Sutton, "The GridPort Toolkit Architecture for Building Grid Portals", Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing, Aug 2001.

[6] NPACI HotPage - https://hotpage.npaci.edu/

[7] J. Novotny, "The Grid Portal Development Kit", Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov-Dec 2002.

[8] A. Natrajan, A. Nguyen-Tuong, M. Humphrey, M. Herrick, B. Clarke, and A. Grimshaw, "The Legion Grid Portal", Journal of Concurrency and Computation: Practice and Experience (CCPE), Volume 14, Issue 13-15, Wiley Press, Nov.-Dec., 2002.

[9] G. Aloisio, M. Cafaro, P. Falabella, C. Kesselman, R. Williams, "Grid Computing on the Web using the Globus Toolkit", Proceedings of the 8th International Conference on. High Performance Computing and Networking Europe (HPCN Europe 2000), Amsterdam, Netherlands, May 2000.

[10] Sun, "Sun Grid Engine Portal", http://www.sun.com/solutions/hpc/pdfs/TCP-final.pdf

[11] Sun, "Sun Grid Engine Portal", NPACI HotPage - https://hotpage.npaci.edu/

[12] Francisco J. Gonzalez, Javier Vales, Miron Livny, Enrique Costa and Luis Anido, "Condor Grid Computing from Mobile Handheld Devices", Mobile Computing and Communication Review, Volume 6, Number 2.

[13] Experiment Particle Physics group, The University of Melbourne, http://epp.ph.unimelb.edu.au/epp/

[14] Grid Computing prototype demonstrated in Melbourne, The Age, June 5, 2003 http://www.theage.com.au/articles/2003/06/05/1054700318561.html

[15] PRAGMA Demonstration, http://epp.ph.unimelb.edu.au/epp/grid/pragma.html

[16] PRAGMA, Pacific Rim Applications and Grid Middleware Assembly, The 4th Workshop, June 5-6, 2003, http://www1.qpsf.edu.au/pragma/index.html

[17] Duchnicky, R L & Kolers, P A (1983) "Readability of text scrolled on visual display terminals as a function of window size," *Human Factor*s, 25:683–692

[18] Dillon, A, Richardson, J & McKnight, C (1990) "The Effect of Display Size and Text Splitting on Reading Lengthy Text from the Screen," *Behaviour and Information Technology*,

[19] Nielsen, J (1999). "Graceful degradation of scalable internet services, WAP: wrong approach to portability," Alertbox 31/10/1999 http://www.useit.com/alertbox/991031.html

[20] Catherine C. Marshall and Christine Ruotolo, "A Study of Reading on Small Form Factor Devices" ACM paper year 2002.

[21] Conversation with Clement Mok and Jakob Neilsen on CHI 99 to addressed " Web design limit to HCI"
[22] Download SVG Viewer from Adobe Web Site
http://www.adobe.com/svg/viewer/install/main.html
[23] **Gri**d Computing and **D**istributed **S**ystems (GRIDS) Laboratory Project,
http://www.gridbus.org
[24] Mark Baker, Rajkumar Buyya and Domenico Laforenza, "Grids and Grid technologies for
wide-area distributed computing" Software – Practice and Experience (SP&E), 2002 in press.
[25] Xiao hui Zhu, Ming Zhu Cui, Bei Shu, Yi Zhen Xu, Xia Li, Ming Li, Fei Qu "e-business Globalization Solution Design" ibm.com/redbooks
[26] Unicode Home Page
http://www.unicode.org
[27] Web Services Internationalization Usage Scenarios
http://www.w3c.org/TR/ws-i18n-scenarios/
[28] XPath 1.0 specification and 2.0 requirement.
http://www.w3c.org/TR/xpath/
http://www.w3c.org/TR/xpath20req/
[29] Nokia Developer Forum
http://www.forum.nokia.com/main.html/