International Journal of Software Engineering and Its Applications
Vol. 7, No. 3, May, 2013

The Study of AMGA RAP-based Web Application

Taesang Huh, Geunchul Park, Jae-Hyuck Kwak, Soonwook Hwang and Sunil Ahn*

National Institute of Supercomputing and Networking, KISTI, 245 Daehak-ro, Yuseong-gu, Daejeon, South Korea
{tshuh, gcpark, jhkwak, hwang, siahn}@kisti.re.kr

Abstract

The ARDA Metadata Catalog Grid Application (AMGA) web application has been widely used; however, it has drawbacks such as easy-to-use interface, no direct building of the Virtual Organization Membership Service (VOMS) proxy and no maintenance after AMGA server version 1.3. In response, we adapted a new development procedure and toolkit from Graphic User Interface (GUI) client, a Client/Server (C/S) program, to a web application to manage the both Eclipse Rich Client Platform (RCP) and Rich Ajax Platform (RAP) at the same time. The AMGA web application provides many interesting features for manipulation of collections, metadata schema, entries, access control, user/group information, federation and others. Additionally, this web application includes a powerful SQL query editor that enables users to make complicated sentences under specific query conditions. In this paper, we describe the implementation of the AMGA web application focusing on the transformation of AMGA Manager using Eclipse RCP to a RAP-based web application.

Keywords: *AMGA, Metadata catalogue, AMGA Manager, Grid computing, Eclipse, RCP, RAP, Web application*

1. Introduction

This paper focuses on the development of a general-purpose AMGA web application that provides a powerful Eclipse framework allowing engineers and programmers to quickly and easily embed data and metadata features inside their own applications, using a standard design-pattern-based approach. AMGA has powerful functionalities to ensure good performance and scalability, along with a multi-threaded multi-process based on Database (DB) connection pooling, a hierarchical collection structure, replication, and a federation mechanism for an efficient distributed environment. Information retrieval is one of the most important technologies in the internet or distributed computing system [1]. AMGA is used in the information retrieval system in the fields of scientific projects. There are many AMGA user communities worldwide: Belle II, INDICATE, DECIDE, DKRZ, and EUMEDGRID [2-3].

AMGA is universally recognized as the most effective metadata catalog middleware component in the Grid environment field for locating files using descriptive information about data and Grid authentication. Unfortunately, interacting with AMGA services is not always user-friendly, especially for non-expert users, because the only clients provided are Unix Command Line Interface (CLI) or APIs [4-6]. An AMGA web version was developed by the Istituto Nazionale di Fisica Nucleare (INFN). However, this was prototype application designed for a specific environment; there were no upgrades offered after AMGA server version 1.3, the interface is clumsy, and has

* Corresponding author: Sunil Ahn (Ph.D), Email:siahn@kisti.re.kr

limited login authentication for uploading of the proxy file generated by a pre-procedure of another User Interface (UI) machine. Moreover, there has been no professional Client/Server (C/S) software offered for efficient management of all AMGA contents [7].

In response to these problems, we determined the development and implementation procedure for a general-purpose and intuitive AMGA GUI toolkit from a C/S program based on transformation of Eclipse Rich Client Platform (RCP), which normally runs as a desktop application on a personal computer, to a Rich Ajax Platform (RAP)-based web application that can be deployed in other runtime environments with limited manpower. The result was AMGA GUI client (AMGA Manager), an easy-to-use and general-purpose GUI toolkit for AMGA [3]. On top of this framework, we built a web application that allows users to manage the metadata catalog and administrators to control AMGA services (setting the configurations in amgad file, and start/stop/restart service). We have also developed an AMGA web application for access and management of the metadata catalog from any platform, which requires a comprehensive approach that embraces technical, organizational and legal/philosophical dimensions [14]. Users need only a web browser and their valid authentications: Virtual Organization Membership Service (VOMS), Grid Security Infrastructure (GSI), Identifier/Password (ID/PW) and certificates [8]. After a successful login, users are able to browse the hierarchy of AMGA collections, to inquire about their schema, permissions and entry list. Users also have the ability to manipulate collections, their metadata schema, entries, access control, user/group information, federation, plain table, export/import metadata files, service configuration, service behavior and list of AMGA sites; all of this is accomplished via a user-friendly web interface that removes complexities to enable easier accessing Grid services, encapsulates AMGA syntaxes, and provides SQL editor for automatic query composition, various wizards, and specialized viewers.

The remainder of this paper is organized as follows. Section 2 discusses the background for the study. Section 3 focuses on the main topic: the design and implementation of the AMGA web application. Finally, Section 4 concludes the paper.

2. Background

In this section, we briefly describe web application technique, security in AMGA service and RCP-based AMGA Manager.

2.1. Rich Ajax Platform (RAP)

The RAP is an extension of the RCP for creating plug-in-based, interactive web applications. The most important advantage of using RAP is the reuse of existing Eclipse RCP and technologies such as the Standard Widget Toolkit (SWT) and JFace, as well as plug-ins, extensions and extension points, Open Service Gateway initiative (OSGi) services, and others. The HTML and JavaScript requirements of web applications are provided for RAP, and developers do not have to be familiar with those technologies [9]. In [10], the main benefit of the proposed solution asserts emphatically an increase of code reusability as well as platform independency, system scalability and modularity. The primary aim of the RAP project is to enable developers to build Rich Internet Applications (RIA) using Java, without having to learn a big stack of new technologies by means of the Eclipse development model. As far as I know, RAP allows developers to build rich, Ajax-enabled web applications using full Java™ libraries and Java-only APIs, by providing Web-enabled implementation of SWT,

JFace, and the Eclipse Workbench. Two technology projects (RAP2 and eRCP3) under the Eclipse.org umbrella provide an alternative runtime environment for RCP applications. In simple, practical terms, this means that applications normally running as desktop applications on a PC can be deployed to other runtime environments. Once a user has a ready-to-launch RCP-based application, transforming it from RCP to RAP makes it possible to launch it on a server through which clients can use it with a Web 2.0-centric interface on a browser. Significantly, there is no need for the user to install any further add-ons or plug-ins. In order to achieve full compatibility between the platforms, many concepts implemented in SWT need to be adapted to other runtimes. These are hidden behind the public API, which remains synchronous across all runtime projects [11]. The most important consideration for transformation of web applications is that RAP applications run multiple user sessions within a single application instance. Also, so as to change all dependencies according to RAP, RAP UI libraries and extension points must be considered.

2.2. Security in AMGA

To design the secure connection of AMGA web application, we needed to know the security of AMGA. AMGA's security mechanisms are contained entirely from the mechanisms provided by gLite. The basis for the Grid's security infrastructure is GSI and on top of GSI, a service to manage the security policy of a VO. On top of GSI, is the VOMS which was developed by the European DataGrid (EDG) project, a predecessor of the Enabling Grids for E-sciencE (EGEE) project [12]. Grid certificate, either GSI or VOMS, are the preferred way of authenticating users on a Grid setting, as depicted on Figure 1. For authorization, it would be possible to use the information included in the certificate directly, such as the distinguished name (DN), or the role and capabilities information encoded in a VOMS certificate. In fact, this is done by some catalogs, such as the Globus replica location service [13].

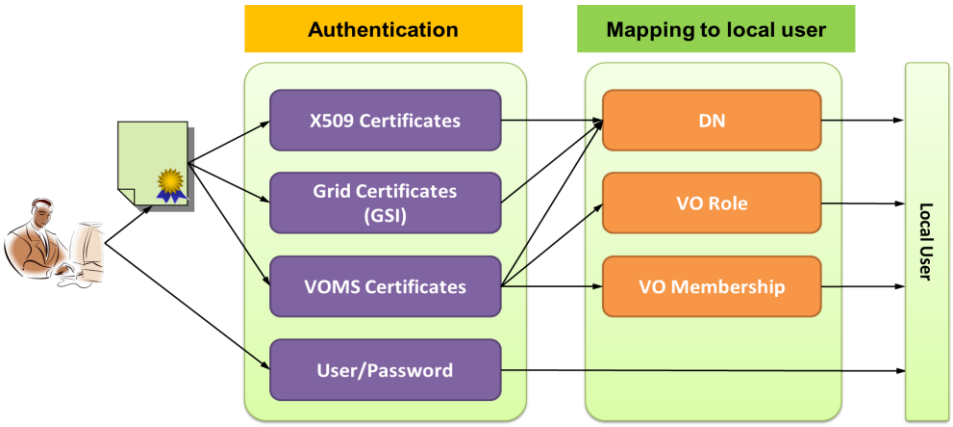


Figure 1. Security in AMGA

In addition, AMGA supports a fully independent local policy with its own users and groups in dealing with the GSI/VOMS policy. Authorization is done solely using this local policy. This level of indirection between the GSI/VOMS and the local security policy requires a system to map between the two of them. This mapping is done after authentication in accordance with a policy defined by the AMGA administrator, based on one of mapping rules (the DN, the VO role, and the VO membership) defined in the user certificate [14]. In a nutshell, there are 4 types of authentication which consists of x.509 certificates, Grid

certificates, VOMS Certificates, and User ID/Password authentication to access the AMGA service, which each login procedure needs.

2.3. AMGA Manager

AMGA Manager is a general-purpose GUI toolkit for AMGA service. Its many easy-to-use features for improving usability and convenience include automatic query composition, importing and exporting of metadata into a spread sheet, and filtered metadata searching. Portability having been one of the most important toolkit-design considerations, the Eclipse Integrated Development Environment (IDE) helps AMGA Manager work on various heterogeneous platforms: Linux, Windows, and Mac. Such diverse features help users build metadata searching environments more easily and faster and boost productivity in managing large-size metadata on Grid environment. Figure 2 illustrates the connection with Grid proxy to access AMGA service, collection menu (management), attributes viewer to show attributes with data type and length, and schema browser including data, attributes, Access Control List (ACL) and index management.

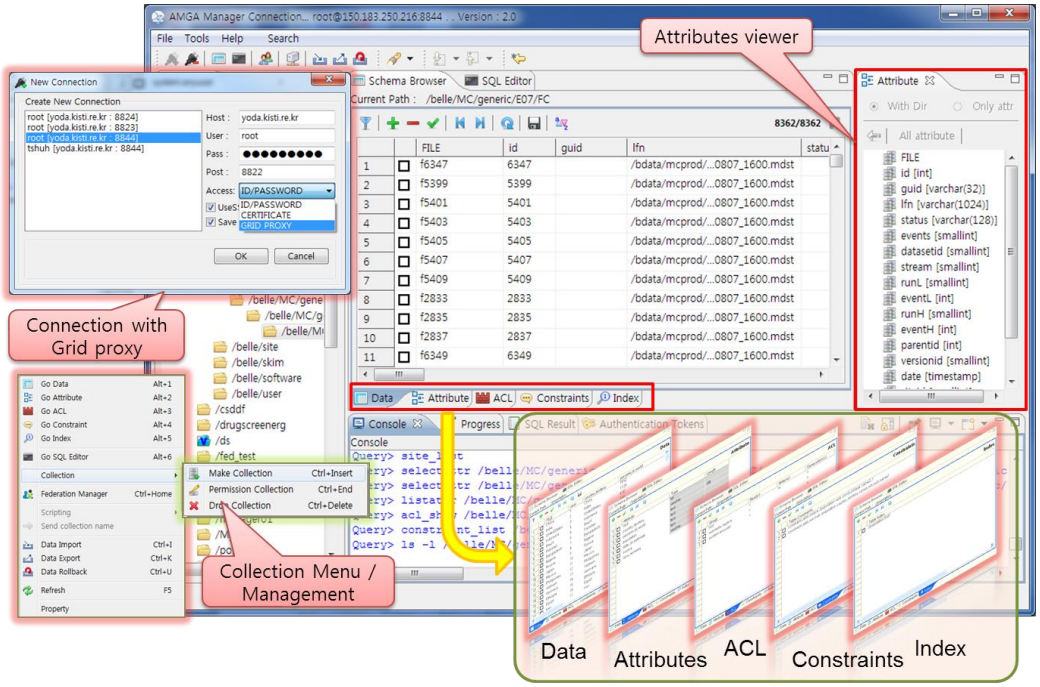


Figure 2. AMGA Manager: Schema Browser

3. Design and Implementation

3..1 User Requirements

We analyzed the requirements of the Belle II community, one of the largest user groups using AMGA [6]. As shown in Table 1, Belle II users wanted AMGA web application to have easy access, simple functions, management of server, low OS dependency, using various types of web browsers, and a large-scale data in/out processing.

Table 1. Belle II User Requirements Analysis

Classification	User Requirements
Accessibility	• Minimize firewall and security issues • Run stand-alone against Grid UI • Use easily with Grid authentications (Grid/VOMS proxy) • Use directly with X.509 certification and ID/PW authorization
Functions Simplification	• Simplify a number of functions for Belle II general users compared with AMGA Manager
Server set-up	• Modify environment variables in configuration files • Manage a list of AMGA Sites • Manage AMGA service running (start, stop, restart)
Client OS	• Reduce the OS dependency
Interface	• Support various web browsers (IE, Firefox, Chrome, etc.)
Data In/out	• Support a large-scale data in/out

3.2. Design

The design AMGA web application shares the AMGA Manager design, because this development is to be transformed mainly in consideration of multi-users and RAP plug-ins, according to the concept of reuse packages developed by Eclipse RCP (according to the RAP project, 70% - 90% reuse is possible) [8]. But all features of AMGA Manager are imported into the web application along with these added administrator functionalities: controlling the AMGA configuration and the behavior of AMGA services. In Figure 3, this administrative module directly accesses the AMGA services without using any AMGA APIs or run Linux commands on the OS platform. Also shown is a list of other, related AMGA services written according to the amgad configuration, which allow a site administrator to restart the AMGA service after changing certain variables, in order to improve service performance or change service environments.

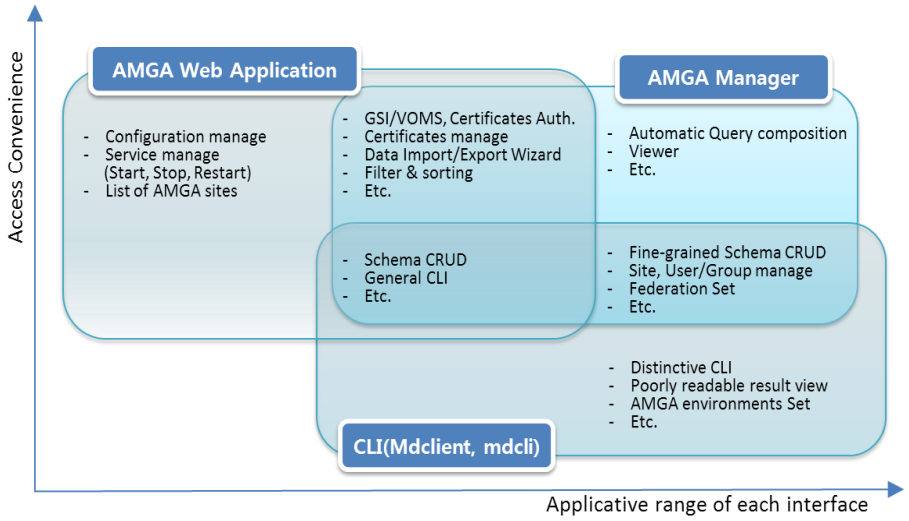


Figure 3. Design Range of AMGA Web Service

The other functionalities of the AMGA web application are the same as or similar to AMGA Manger in the aspect of basic GUIs. Although AMGA web application does not have delicate and complicated functionalities, it provides more convenient accessibility compared to other AMGA interfaces: AMGA Manager, AMGA APIs and mdclient/mdcli.

3.3. Architecture

The architecture of AMGA web application is shown in Figure 4. AMGA web application interacts with AMGA server through a set of AMGA Java API and AMGA configuration and service control API after authentication/authorization in the connection module. The AMGA configuration and service control API allows to control the service and to change the service parameters at AMGA server. The RCP source files of collection Management, metadata management, properties management and SQL editor modules are reused and service management is made of new RAP coding. Then all RCP components are changed to a web service, based-on the concept of service-oriented computing, at transformation layer: RAP-based AMGA web application [15].

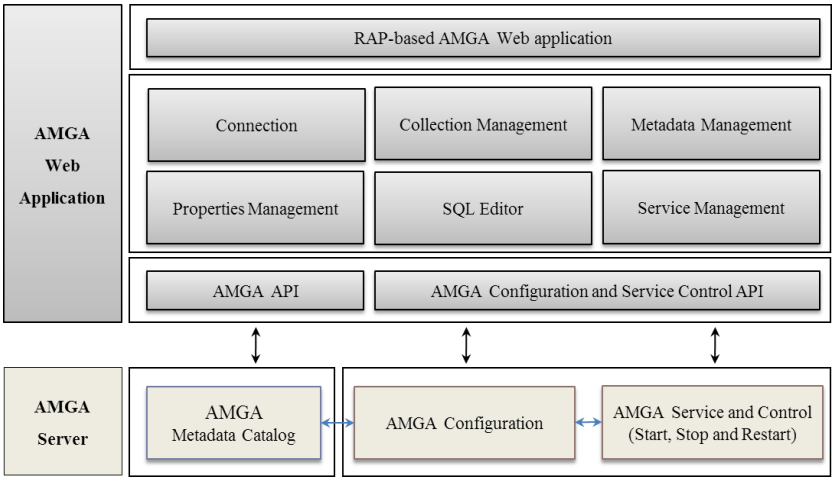


Figure 4. AMGA Web Application Architecture

3.4. Development environment

The development environment of AMGA web application consists of the following three categories:

(1) Server environment:

Server environment has the main role to provide web service relating to AMGA service.

(2) Client development environment:

Client development environment allows developers to transform RCP, AMGA Manager to RAP, and develop the components of a web service and test the function level in the AMGA web application.

(3) Client testing environment:

Client testing environment is used only for the purpose to test the web service while running.

The development environment, including the test and debug environments, contains an SLC5 OS-based tomcat servlet container interacting with AMGA (ODBC: psqlodbc-08.03.0200, Backend DB: Postgresql). We used Eclipse for RCP and RAP developers (a development toolkit known as Integrated Development Environment (IDE)) in the coding, debugging and testing phases in order to accelerate the overall development process. As a prerequisite, JDK 1.7 was installed on the server side as well as client side. On the client side, the web application was tested in several web browsers (Opera, Chrome, Internet Explorer, and Firefox) working on various OS platforms (Mac, Debian, and Windows) as in Table 2.

Table 2. Development Environment of AMGA Web Application

Server environment	OS	SLC5
	Web server	Apache Tomcat 6.0.35
	Java	JDK 1.7
	Backend DB	Postgresql 9.0.9
	ODBC	psqlodbc-08.03.0200
	AMGA	AMGA 2.3(server/client)
Client development environment	OS	Windows 7
	Development toolkit	Eclipse for RCP and RAP Developers
	Java	JDK 1.7 for Windows 64bits
Client Testing environment	OS	Mac/Debian/Windows XP/7
	Java	JDK 1.7 for Mac/Linux(32/64bits)/Windows(bit/64bits)
	Web browser	Opera, Google Chrome, Internet Explorer, Firefox

3.5. Eclipse for RAP

Table 3. Plug-ins for AMGA Web Application

AMGA Manager	kr.re.kisti.amga.editor, kr.re.kisti.amga.voms, kr.re.kisti.amga.editor.feature
gEclipse & VOMS plug-ins	eu.geclipse, eu.geclipse.core, eu.geclipse.core.filesystem, eu.geclipse.core.jobs, eu.geclipse.core.reporting, eu.geclipse.efs.gridftp, eu.geclipse.glite.info, eu.geclipse.globus, eu.geclipse.globus.ui, eu.geclipse.info, eu.geclipse.jsdl, eu.geclipse.jsdl.model, eu.geclipse.keystore.ui, eu.geclipse.ui, eu.geclipse.voms, eu.geclipse.voms.ui, eu.geclipse.workflow.model, org.bouncycastle, org.globus
UI plug-ins	org.eclipse.rap.ui, org.eclipse.nebula.widgets.nattable.core, org.eclipse.nebula.widgets.nattable.dataset, org.eclipse.nebula.widgets.nattable.extension.glazedlists
Entry Point	org.eclipse.rap.ui.entrypoint

The Eclipse environment setup for the implementation of AMGA web application proceeded with the procedure below.

- Install an Eclipse and install RAP tools into the Eclipse IDE

- Run Eclipse.exe

- Download AMGA Manager plug-ins from SVN[1] and relating plug-ins in Table 3.

- Setup and check AMGA web application running

- Check web browser: http://127.0.0.1:9090/amga

The RAP tools provide a dedicated launcher to start RAP application directly from the IDE. The AMGA web application is running at a default port chosen by the launcher if the RAP application is selected the without port number and the running information is registered in Eclipse Run menu.

Figure 5 shows our AMGA configuration in the **Run/Run Configuration** menu to create a new launch configuration and to start RAP applications directly from the RAP launcher that was used for applications based on OSGi. During this time, we checked internal browser, wrote '9090' in the port configuration field and 'amga' for Servlet Path and, selected fms in order to open AMGA web application in a browser.

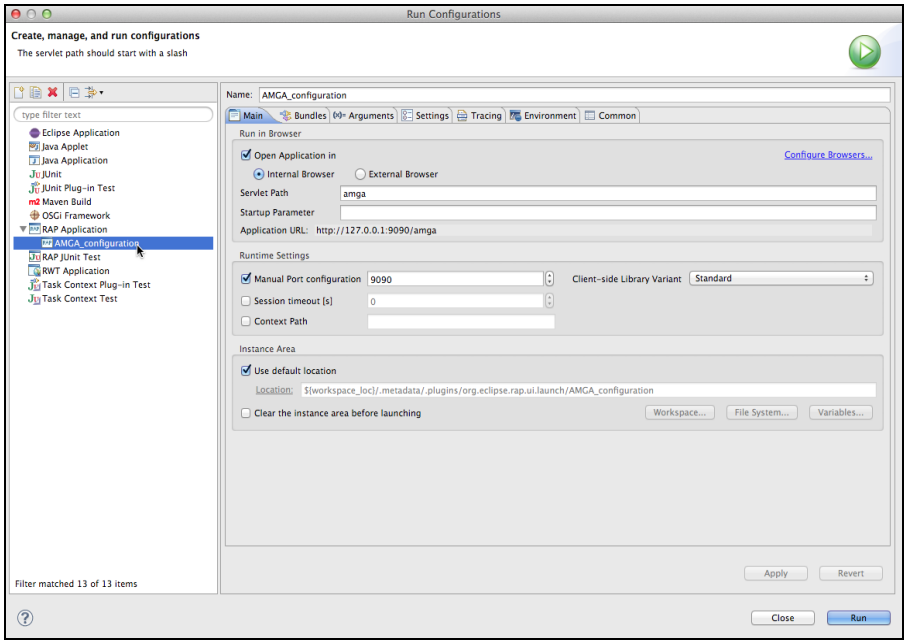


Figure 5. AMGA Web Application

3.6. Implementation

We first changed all of the RCP-based AMGA Manager dependencies to RAP versions and performed debugging so as to convert the RCP source files to RAP ones. Since all AMGA Manager APIs were not available in RAP, and some AMGA Manager APIs (such as Graphic Context (GC), StyledText, FileDialog and MouseMove Event) caused up to approximately 1,200 errors. RAP-unsupported APIs, as described above, were needed to change to alternative APIs: Text for StyledText, Canvas for GC, Upload for FileDialog, and Canvas for MouseMove. FileDialog isn't particularly supported by RAP environment due to browser security restriction, so we had to substitute FileDialog with Upload plug-in supporting HTTP streaming.

[1] https://amga-qui.googlecode.com/svn/

```
if (Activator.getCtx().getAuthMode()==MDServerConnectionContext.AUTH_GRID_PROXY) {
    AuthenticationTokenManagermanager=AuthenticationTokenManager.getManager();
    if   (manager.getTokenCount() > 0) {
            java.util.List<IAuthenticationToken>tokens=manager.getTokens();
            for (IAuthenticationTokentk:tokens){
                if   (tk.getID().equals(Activator.getTokenID())){
                    try {
                        break;
                    }   catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
    }
}
```

Figure 6. Example of Implementation of VOMS API

Figure 6 is an example of implementation of VOMS API. Because the plug-ins relating to gEclipse VOMS are running on the Eclipse RCP, it is impossible to be supported from Eclipse RAP. So we needed to implement VOMS API directly for Java using the library. For that reason, we changed org.eclipse.ui to org.eclipse.rap.ui, the plug-in for RAP, in consideration of the UI libraries because RCP and RAP needed different platforms. Session-based singleton was used for a web application to provide multi-users simultaneous access. We also developed add-on interfaces to handle the AMGA configurations and services, and to show the list of AMGA sites.

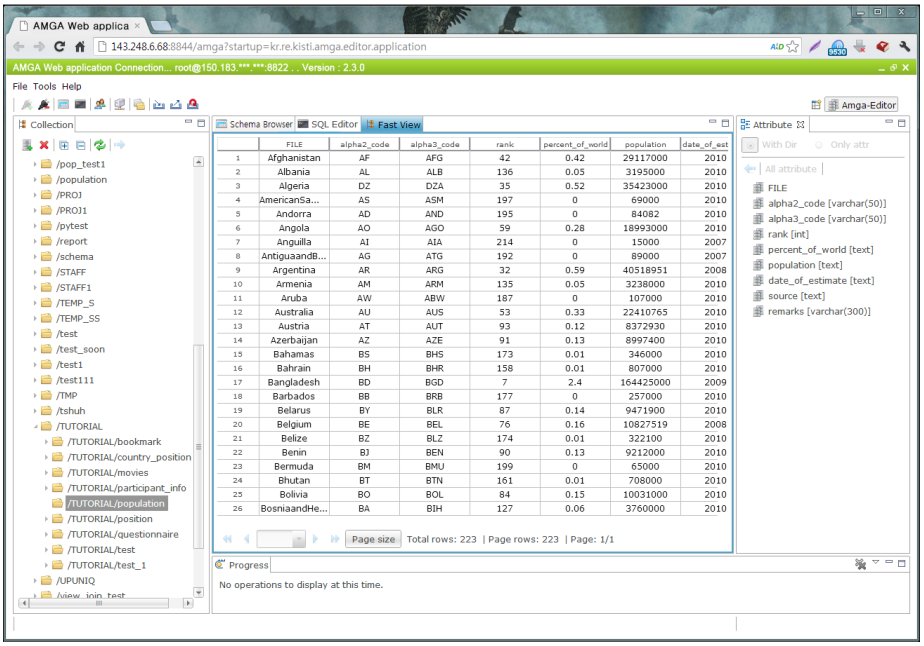


Figure 7. Main Web Page

As illustrated in Figure 7, after a successful login with proper authentication, users will be able to browse AMGA collections hierarchically according to the specific

metadata schema with its entries and attributes. At this time, the most of the code is being executed on the server side, whereas the thin-client side, users' PC running the web browser, is only updated when needed.

4. Conclusions

In this paper, we have discussed implementation of a general-purpose AMGA web application through transformation of the RCP-based AMGA Manager to a RAP-based web service through Belle II user requirements analysis. A powerful RAP enables to quickly and easily embed data and metadata features inside their own web service. The AMGA web application provides the basic manipulation functionalities of AMGA Manager as well as administrator functionality for manipulation of AMGA configurations and handling AMGA services. After a successful login, users are able to browse the hierarchy of AMGA collections, to inquire about their schema, permissions and entry list. Users also have the ability to manipulate collections, their metadata schema, entries, access control, user/group information, federation, plain table, export/import metadata files, service configuration, service behavior and list of AMGA sites; all of this is accomplished via a user-friendly web interface that removes complexities to enable easier accessing Grid services, encapsulates AMGA syntaxes, and provides SQL editor for automatic query composition, various wizards, and specialized viewers.

In the future, we plan to conduct a further analysis on the requirements of the Belle II community, which conduct high-energy physics in Japan by porting the AMGA web service to it; on the basis of that analysis, we will introduce new features that better reflect users' requirements. Ultimately, we aim to provide not only precisely customized AMGA web services for Belle II but also generic AMGA web services for existing AMGA users. Analogously, we will be able to provide this for other R&D communities such as biomed infrastructure and medical data management in France, DKRZ climate research in Germany, the digital library in Italy, and so on.

Acknowledgements

This work has been partially funded by the European Commission as part of the EMI (Grant Agreement INFSO-RI-261611) project.

References

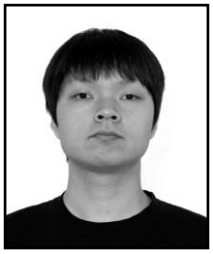
[1] J. Lee, "Design of Document Profile Database for Browsing in Information Retrieval Systems", International Journal of Software Engineering and Its Applications, vol. 1, no. 1, **(2007)**, pp. 78-88.
[2] AMGA, "ARDA Metadata Grid Application", Available at: http://amga.web.cern.ch/amga, **(2008)** August 24.
[3] T. Huh, S. Hwang and G. Park, "Implementation of AMGA GUI Client Toolkit : AMGA Manager", The Journal of the Korea Contents Association, vol. 12, no. 3, **(2012)**, pp. 421-433.
[4] N. Santos and B. Koblitz, "Metadata services on the grid", Proceedings of the Advanced Computing and Analysis Techniques, DESY, Zeuthen, Germany, **(2005)** 22–27 May.
[5] B. Koblitz, N. Santos and V. Pose, "The AMGA Metadata Service", Journal of Grid Computing, vol. 6, **(2008)**, pp. 61-76.
[6] S. Ahn, J. H. Kim, T. huh and S. Hwang, "The Embedment of a Metadata System at Grid Farms at the Belle II Experiment", Journal of Korean Physical Society, vol. 59 , no. 4, **(2011)**, pp. 2695-2701.

[7] S. Scifo and V. Milazzo, "Amga wi - amga web interface", Conference on Hypermedia And Grid Systems, IEEE, **(2007)**.
[8] N. Santos and B. Koblitz, "Security in distributed metadata catalogues", Concurrency and Computation: Practice and Experience, vol. 20, no. 17, **(2008)**, pp. 1995-2007.
[9] Eclipse Foundation, Rich Ajax Platform (RAP), Available at: http://www.eclipse.org/rap/.
[10] C. Pastrone, M. Spirito, R. Tomasi and F. Riz, "A Jabber-Based Management Framework for Heterogeneous Sensor Network," International Journal of Software Engineering and Its Applications, vol. 2, no. 3, **(2008)**, pp. 9-24.
[11] B. Muskalla, "Patterns for Single-Sourcing RCP and RAP applications", EclipseSource version 1.1.4"
[12] R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, K. Lorentey and F. Spataro, "From gridmap-file to voms: Managing authorization in a grid environment", Future Generation Computer Systems, vol. 21, no. 4, **(2005)**, pp. 549-558.
[13] A. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman and R. Schwartzkopf, "Performance and scalability of a replica location service", Proc. of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC-13), IEEE Computer Society, **(2004)**, pp. 182-191.
[14] S. Al-Fedaghi, "Developing Web Applications", International Journal of Software Engineering and Its Applications, vol. 5, no. 2, **(2011)**, pp. 57-68.
[15] N. Santos and B. Kobliz, "Security in distributed metadata catalogues", Concurrency And Computation-Practice & Experience, **(2008)**, vol. 20, pp. 1995-2007.

Authors

Taesang Huh received his B.S. degree in electric, electronic and computer engineering from Sungkyunkwan University (SKKU), Korea, in 2000 and the MS degree in information and communications engineering from Gwangju Institute of Science and Technology (GIST), Korea in 2002, respectively. He joined Korea Institute of Science and Technology Information (KISTI) in 2002 and he is a senior researcher of National Institute of Supercomputing and Networking (NISN) at KISTI. His research interests include Metadata Catalog, Distributed Computing, Cloud Storage, e-science and information system.

Geunchul Park received his B.S. and the M.S. degree in computer engineering from Chungang University (CAU), Korea, in 1998 and in 2000, respectively. He joined KISTI in 2006 and he has been a senior researcher of NISN at KISTI. His research interests are in the areas of Metadata Catalog, Grid Computing, Cloud Computing, Data Management, e-science and database.



Jae-Hyuck Kwak received the B.S. degree in Information and Computer Engineering from Ajou University in 2001 and the M.S. degree in Electrical and Computer Engineering from Seoul National University in 2003. He joined KISTI in 2003. Currently, he is a senior researcher of NISN at KISTI. His research interests lie primarily in the issues concerning High Performance Computing, Distributed Computing Technology, and Data-intensive Computing.

Soonwook Hwang received the B.S. degree in mathematics and the MS degree in computer science from Seoul National University (Korea), in 1990 and in 1995, respectively. He also received the Ph. D. degree in computer science from University of Southern California in 2003 under the supervision of Dr. Carl Kesselman, one of pioneers in Grid computing. He was a visiting scholar at Information Sciences Institute (ISI) in US in 2003. He worked for Japanese National Grid Initiative (NAREGI) as a researcher, which is a Japanese National Grid project started in 2003 for five years, aiming at developing grid middleware for next-generation Cyber Science infrastructure. In 2006, he has joined Korea Institute of Science and Technology Information (KISTI) and has been a principal researcher of Supercomputing center. His research interests are in the areas of Grid computing, high throughput computing, Cloud storage, e-science and information system. Dr Hwang has been Editor-in-Chief of the Journal of Convergence Information Technology since 2009 and AMGA supervisor in European Middleware Initiative (EMI) since 2010.

Sunil Ahn is a research staff in the supercomputing center at KISTI in Korea. He obtained his Ph.D degree in parallel computing from Seoul National University (Korea). He has several published journals and conference articles largely in the grid and its application field.