A platform for linking ubiquitous devices and grid services

MOROHOSHI, Hiroyuki / 諸星, 宏行

(発行年 / Year)
2007-03-24

(学位授与年月日 / Date of Granted)
2007-03-24

(学位名 / Degree Name)
修士(理学)

(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

A Platform for Linking Ubiquitous Devices and Grid Services

Hiroyuki Morohoshi

Faculty of Computer and Information Sciences,

Hosei University, Tokyo, Japan

hiroyuki.morohoshi.f4@gs-cis.hosei.ac.jp

Abstract

With rapid advances in network technology and distributed computing, Grid computing has been widely used in not only the science but also the business. Grid computing plays an important role in our daily activities. People can access and receive services from the resource constrained ubiquitous devices such as PDA and mobile phone. To allow the ubiquitous device to use the Grid resource, it is necessary for the developer to make use of a platform or middleware for linking the devices to Grid services. This paper presents a platform that provides developers with a friendly environment of developing Grid services and accessing Grid services over Globus Toolkit 4 (GT4). Furthermore, the functions are extended for developing the mobile application to access the Grid resources from the ubiquitous devices. The developer can develop the Grid applications for the ubiquitous devices more easily by using this platform, and require less special expert knowledge. This paper describes the platform architecture, the design idea, system architecture, a sample application of skin checking, accessing to a skin-expert service from a mobile phone via the proposed platform, and evaluation and comparisons with other related platforms are given in the paper.

Supervisor: Runhe Huang, Professor

1. Introduction

Grid computing has applied in any cases in recent years with rapid advances in network technology and distributed computing. It has distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and, in some cases, high-performance orientation. One of the visions of Grid computing is to access computational resources automatically on demand in order to deliver the services required with the appropriate quality [1]. As a service-oriented approach to Grid computing is increasing adopted, many systems which can discover Grid resources and access them automatically are developed. In the past few years, small mobile computing devices such as cell phones, Personal Digital Assistance (PDAs), and other embedded devices have gradually become ubiquitous [1]. With improvements in both hardware and software, new opportunities have been provided for these devices. However, it remains a challenging objective to create complicated applications executing on small, light, and mobile devices because the small size and low weight requirement of mobile devices imposes a considerable restriction on their processing power, memory capability and battery capacity. These limitations make mobile devices a platform that would benefit from the Grid computing [2]. The main advantages of mobile Grid computing include mobile-to-mobile and mobile-to-desktop collaboration for resource sharing, improving user experience, convenience, convenience and contextual relevance and novel application scenarios. A grid-based mobile environment would allow mobile devices to become more efficient by off-loading resource-demanding work to more powerful devices or computers. By using the Grid resource from the ubiquitous device, only a cellular phone enables an impossible thing to realize. For example, the system is realized that analyzes user's skin image by Grid computing. The user takes the picture by the cellular phone with a camera, and it displays recommended cosmetics. The system is also realized that displays the route of the bicycle to consume the calorie using PDA with GPS. In addition, Grid computing can be used also in the scene such as control of the robot by voice recognition or image recognition, and advanced processing calculation, such as data mining and an expert system.

However, there are some problems in using the Grid resource from the mobile device. It is difficult to construct the Grid computing environment and impossible to use the Grid resource directly from the mobile device [2]. To access the grid resource, usually, it is necessary that the resource is expressed as service called grid service. Provisioning Grid services usually is done on a Grid infrastructure such as Globus Toolkit [3], which provides diverse fundamental functions of Grid. However, it is heavy and tedious and difficult for users to do installation and configuration. Previous research [4] was focused on developing a user-friendly platform for generating/deploying and accessing Grid services. This platform provides a class of functions for processing parameters input from a Grid service developer via GUI, a class of functions for generating necessary files of a Grid service, and a class of functions of creating the client program and facilitating access of deployed Grid services. As a natural succeeded research, this research is shifting its focus to provisioning Grid services to a user or an object with resource constrained ubiquitous devices rather than with ordinary note PCs or desktop PCs. Of course, the generation and deployment of Grid services could be similar without much revision but provisioning and accessing deployed Grid services is in need of a platform that links users to Grid services. The Grid client must be run on a platform, which is too heavy in terms of computation and processing power, memory, API and applications to a ubiquitous device like mobile phone. This paper realized the platform by adding the function which accesses a Grid resource from a cellular phone to it as plug-in. This paper describes the platform, including its architecture and implementation via an example Grid service and presents on example application, skin grading and caring service.

This paper is organized as follows. The platform architecture is to be described in Section 2. Section 3 is to discuss the Development of Mobile Grid Framework. The test bed application is to be described in Section 4. In Section 5, I will present make comparisons with some related work. Finally, conclusion is to be drawn and future work is to be addressed in the last section.

2. The Platform Architecture

A User-friendly Platform for Developing and Accessing Grid Services [4] provides the developer with the development environment which supports to construct the Grid computing environment. The fundamental concept of this platform is based on how the developer creates grid service easily and how the user accesses the Grid resource easily.

2.1 Outline of the Platform

This platform is focused on developing a user-friendly platform for generating/deploying and accessing Grid services. The platform includes a class of functions for processing parameters input from a developer via GUI, a class of functions for generating files required for defining Grid services specified, and a class of functions for creating client program and facilitating the access of deployed services. The purpose of research is to reduce the burden of development by requiring less special expert knowledge of a developer. Developing a Grid service and writing a client program for accessing a Grid service over Globus Toolkit original platform are not easy to both Grid service developers and the client side users. In the process of developing a Grid service, a developer must define a lot of files and execute commands manually, which require the developer specific expert knowledge and techniques. As for the client side users, they have to use special APIs provided by Globus Toolkit in order to access the deployed Grid services. To enhance Globus Toolkit practical usability and ease developers the development of Grid applications, the platform provides functions for both writing Grid services and accessing the deployed service from client side. In particular, the platform is emphasized on providing an easy environment that requires developers and users less special expert knowledge. Concretely speaking, it automates some processes like defining configuration files, writing programs, generating stubs, deploying services, writing client program, starting the Grid service container, and so on. It also uses wizards to guide developers and the client side users to input the minimal required parameters and data. Though many functions are added in the platform, they are easy to understand and use since it provides a user-friendly interface for both developers and users. A user-friendly interface is designed for developers with hints, samples, templates, and wizards to input necessary minimal parameters and values. The rest of file generations, command executions, and deployment of services are automatically conducted by the system itself.

2.2 Components of the Platform

This platform is mainly divided into four components, *interface*, *process*, *data*, and *plug-in*. Figure 1 shows an overview of the component of the platform.

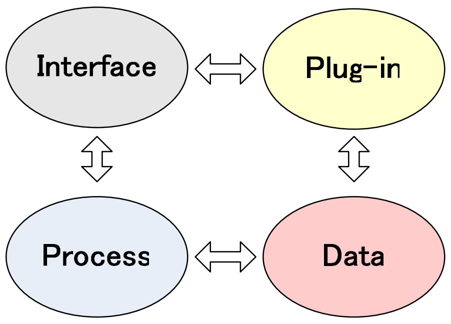
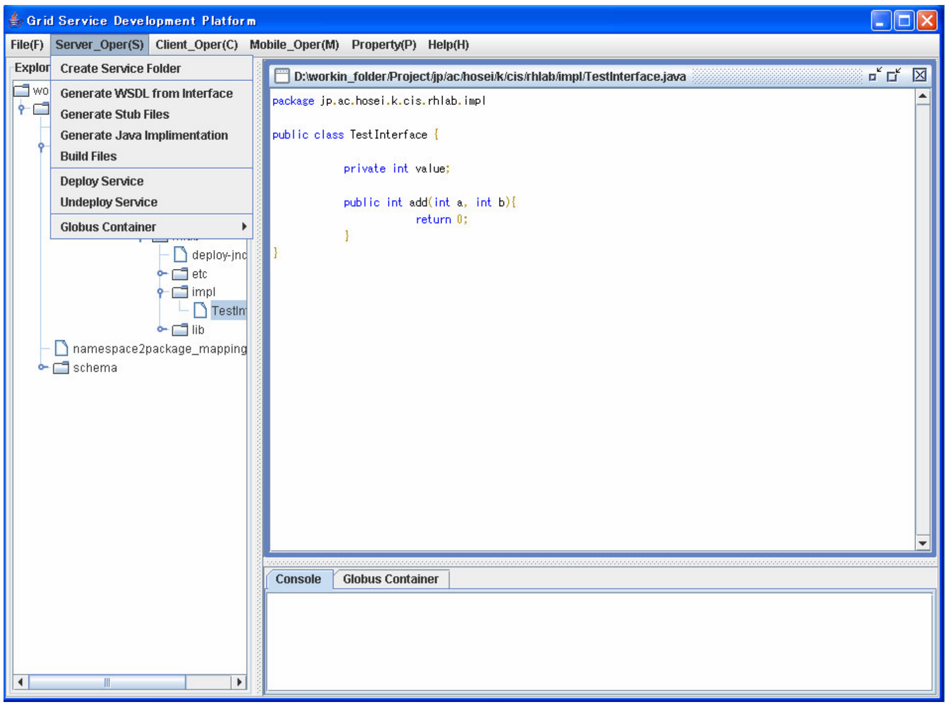


Figure 1. The components of the platform

Interface is a component that deals with the screen and the event. This component manages the design of the main screen. The main screen consists of some panels such as the menu, the directory tree, the file editors, and the process consoles. Also, this component has the subcomponents called tree, event, and editor. The tree subcomponent manages the directory tree, the event and the editor. The user interface is designed friendly for developer. The developer can develop Grid services and access the deployed services easily by selecting the menu. The main user-friendly interface is presented in Figure 2.



D:\workin_folder/Project/jp/ac/hosei/k/cis/rhlab/impl/TestInterface.java

```
package jp.ac.hosei.k.cis.rhlab.impl

public class TestInterface {

          private int value;

          public int add(int a, int b){
                    return 0;
          }
}
```

Figure 2. The interface of platform

Process is a component for developing the Grid service and generating the client access file. It is composed of the subcomponent such as file, service, client, and property, and executes the requested processing from the interface component. A file component is a component for opening and saving a file and so on. A service component is a component which is including the processing in order to develop the Grid service, and generates the project folder and any files (WSDL, implementation file, and stub files), builds the file, deploys the service, undeploys the service, and starts the globus container. The client component is a component for generating the Grid client to access the developed service. It is including the function for generating the stub files, generating the client access file, and running the grid client. Even if the

Grid service and the Grid client are in the different environment, the client can access the Grid service because the stubs are generable from a WSDL. A property component is a component for fundamental setup of the root directory of tree, and the location of the Globus Toolkit. *Data* is a component which manages the data/information of service. The kinds of information are the name of service, the namespace, the package, the portType, the location of the directory, etc. These are necessary information in order that the process component performs an operation. The developer develops the service and generates the client access file by using this information, and develops the plug-in including various function. *Plug-in* is a component to define new function that the developer wants to add it to the platform. The plug-in developer can add the function only by describing the interface and the event, and putting it into the component.

2.3 Upgrade to the Globus Toolkit 4

This platform is developed on top of Globus Toolkit 4 (GT4) and facilitates a part of OGSA services by providing various functions. The GT4 includes a complete implementation of the WSRF (Web Services Resource Framework) specification [5]. WSRF is a specification developed by OASIS. WSRF provides the stateful services though Web service generally doesn't have the state. Instead of putting the state in the Web service we will keep it in a separate entity called a resource, which will store all the state information. Besides the WSRF implementation, the toolkit includes a lot of components which we can use to program Grid applications. As for GT4, the following points are more excellent than GT3.

- The specification of WSRF is simpler than OGSI.
- WSRF is compatible with existing Web service.
- GT4 is steadier than GT3.

2.4 Template Function

This platform also has two enhancing functions. They are template and plug-in. Template is a function that the developer can generate the additional file easily on the development of Grid service and Grid client. For instance, it is possible to use for the generation of the Javadoc file of Grid service, implemented client file, and the ant file for processing any operation. This function uses the Java based template engine that is called Velocity [6]. Velocity's capabilities reach well beyond the realm of the web, though Velocity is often used for web development. For example, it can be used to generate SQL, PostScript and XML from templates. It can be used either as a standalone utility for generating source code and reports, or as an integrated component of other systems [6]. Also, the developer can use Grid service information provided by the data component from the template engine. The developer can put information (service name and namespace) of the Grid service into the file. Figure 3 shows the basic concept of Velocity. The file is generated with merging the vm file written with VTL (Velocity Template Language) and the object called VelocityContext to manage the value for setting in the file template.

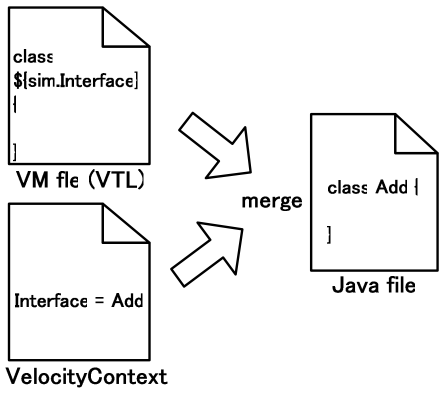
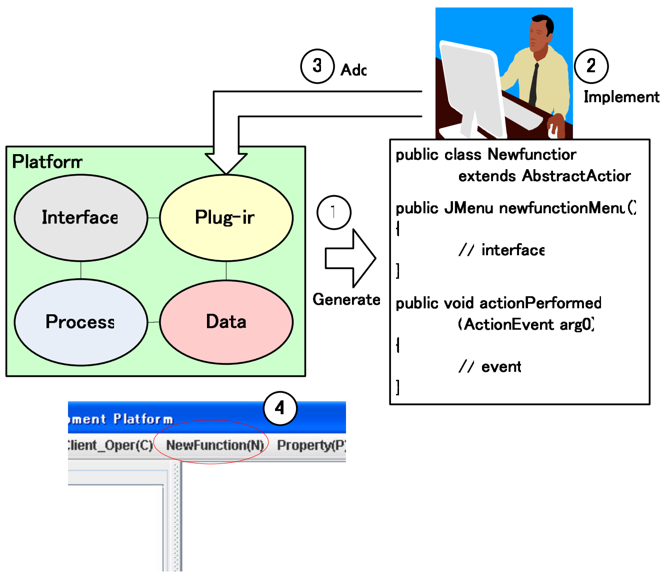


Figure 3. The concept of the Velocity

2.5 Plug-in Function

Plug-in is a function which the developer can insert the original function which is developed into a platform. To do that, it is necessary that the developer describes the interface and processing in the plug-in program. The developer can automatically generate the program with the platform. The interface part basically describes about the menu bar, and the processing is also called by the event of the menu. To add the plug-in, it is necessary to add the program to the plug-in component. Figure 4 shows the image for adding the plug-in to the platform. If incorporated, new function will be displayed on the platform. Moreover, various additional features can be developed by combining this plug-in function with the template function. This thesis shows the plug-in for developing the framework to use the Grid resource from the mobile device. Concretely speaking, it includes the function to generate a mobile access client, copy the stubs to the Grid client, and run. The following section shows the architecture for the developer to use the Grid resource from the cellular phone over the platform.



```
public class Newfunctior
        extends AbstractActior
public JMenu newfunctionMenu()
{
        // interface
}
public void actionPerformed
        (ActionEvent arg0)
{
        // event
}
```

Figure 4. Add the new function to the platform

3. Development of Mobile Grid Framework

In this section, the framework that uses the Grid resource is defined from a mobile device. First, this section explains the method and the problem to use the Grid resource from the mobile devices, and then, explains about the Mobile Grid Framework developed by the platform. Finally, this section shows the example of using the Add service that does a simple calculation from mobile device based on the framework.

Generally, there are at least three possible ways to allow a ubiquitous device to use Grid services [2]. One way is to make a device end as a Grid service client and execute a Grid service directly. The second one is to execute a Grid service directly from a device end in the OGSI.NET [7] framework. Both approaches require the device end powerful enough to be able to install a Grid platform and run on it and having Java based OGSI implementation or OGSI.NET implementation, which is not practical since those devices have severe limitations in system resources such as processing power, memory, and battery power and are stripped down runtime environments, API, and applications. The third way is to develop and place a gateway between a device and Grid services, which has an interface to Grid services and an interface to the device, and executes Grid services instead of the device and returns results to the device. This paper takes the third approach. To well assist the explanation, a mobile phone is used as a representative of so-called ubiquitous devices in the following text.

The proposed system needs the following processes for developing a mobile application to use Grid services via the gateway.

- Developing the Grid services.
- Creating the Grid service client.
- Deploying the Grid service client on the gateway.
- Creating the Mobile client for accessing the gateway.

It is very hard for the developer to do all these processes from the beginning to constructing the system. Though the first two processes can be executable in previous research, the rest processes cannot. This thesis adds the function for executing the rest processes to the platform as a plug-in, and enhances it to execute all steps (server, client, and mobile) instead of the developer. The developer can realize the framework for accessing the Grid resource from mobile device easily by using the platform. Also, it supports the knowledge of the Grid computing, the web programming, and the mobile programming. In addition, the other advantage of this framework, mobile client device don't have to have the excess resource because the stubs and Globus API are included on the Gateway. Figure 5 shows the platform's role and position in a system that allows mobile phone to access Grid services.

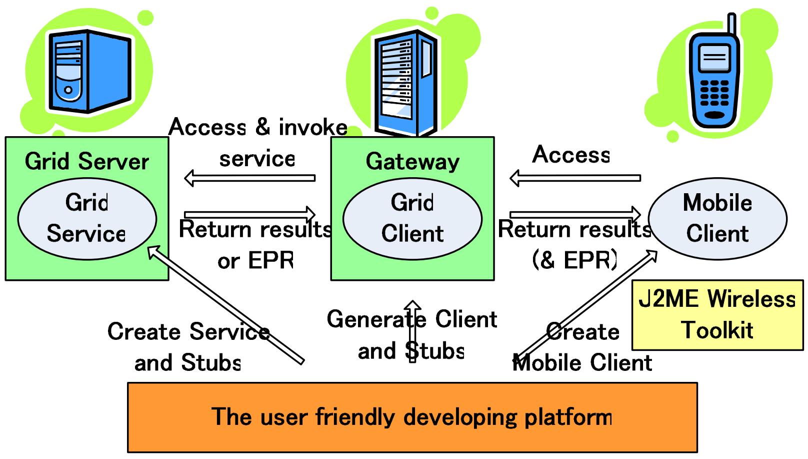


Figure 5. The role and position of the platform

This framework adopts the Http connection for the communication procedure from mobile to the Gateway. Because it has the following merits.

- Any MIDP terminal the HTTP communication is supported though it doesn't necessarily support the socket communication and the Datagram communication.
- It is portability because the Http communication doesn't depend on the network compared with the socket and the Datagram.
- The Http communication can easily solve the problem like the network security and the firewall, etc.

3.1 Grid Server Side Architecture

To use the Grid resource from a ubiquitous device, the developer of the system has to develop three parts as the Grid service, the client on the gateway and the application on the ubiquitous device. To develop the Grid service, the platform provides the developer with various functions. Their functions are briefly described list below.

- ***Create Service Folder*** generates a service package including files and folders required for writing a grid service. The minimal required information for writing a grid service such as service name, base package, and so on, is input from a wizard window.
- ***Generate WSDL and Impl*** generates a WSDL and Java implementation file from the interface by using Java reflection API.
- ***Generate Stub Files*** generates the stub files by using ant tools. The build folder is generated and put in the service package, and then the stubs folder including the stub files is generated and put in the build folder. The stubs are used for communications between the server and the client, and they are imported into implementation files.
- ***Build Files*** generates the Gar file from all files including the service stub files, WSDD file, JNDI file, the code added by the developer, and so on. The generated Gar file as a developed grid service is to be deployed into the grid service container.
- ***Deploy Service*** unpacks the Gar file and copies the files (WSDL, compiled stubs, compiled implementation, WSDD, JNDI) into key locations in the GT4 by using ant tools.
- ***Undeploy Service*** removes services and deletes the files from GT4.

- ***Globus Container*** includes two functions, “Start” and “Stop”. If “Start” is selected, the container is started, and if “Stop” is selected, the container is stopped. When the container starts up, the developer sees a list with the URIs of all the deployed services.

The developer can easily create the Grid service on Globus Toolkits via a user-friendly interface by this server function of the platform, and the Grid service is automatically deployed into the Globus container. Associated with each deployed Grid service, there are four important components in the Globus container, a Factory Service, an Instance Service, a Resource Home, and a Resource [8]. Their relation is shown in Figure 6. The factory service is service for creating new resource composed of zero or more resource properties. The factory service provides a ***createResource*** operation that returns an endpoint reference (EPR) to the Grid client. This operation in fact includes the following three steps:

① Retrieve the resource home.

② Use the resource home to create a new resource.

③ Create the endpoint reference we will return to the Grid client. This EPR must contain the instance service’s URI and the new resource's key.

The instance service provides the Grid client with the operations associated with the specified Grid service. The invoking an operation is shown as follows:

① Read the EPR and finding the resource by using the Resource Home.

② Operates on the resource properties.

③ Send the result to the Grid client.

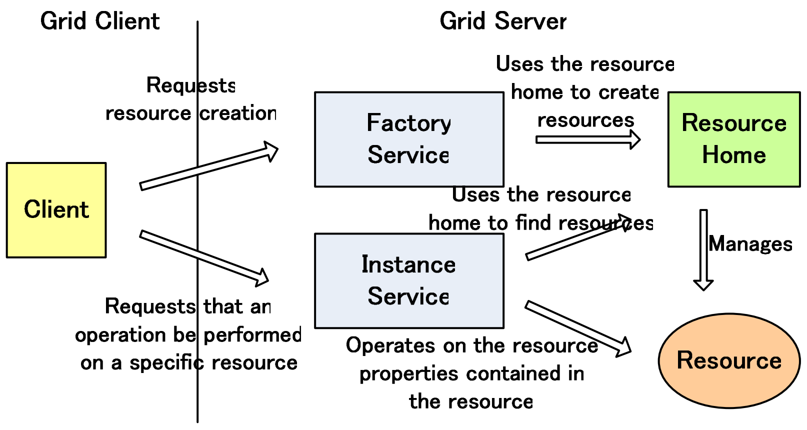

Grid Client
Grid Server
Requests resource creation
Client
Requests that an operation be performed on a specific resource
Factory Service
Uses the resource home to create resources
Resource Home
Uses the resource home to find resources
Instance Service
Manages
Operates on the resource properties contained in the resource
Resource

Figure 6. The relation of the four components in the Grid server

The resource of the grid service exists one for each mobile device. Other devices cannot access the same resource. It is identified with the resource key. This has the merit that the gateway does not need to have the useless resource. To manage information of all mobile phone puts a big load for the gateway. This problem will be solved by distributing the resource and managing information by using the Grid computing.

3.2 Grid Client Side Architecture

Then, this section explains the Grid client to access the Grid service. The platform also provides the developer with the function to develop the Grid client. Their functions are briefly described list below.

- ***Generate Client Stubs*** imports WSDL file from the server and generates stubs.
- ***Generate Client File*** generates a client file for accessing a deployed grid service.
- ***Run*** runs and accesses the service with some arguments from the client.

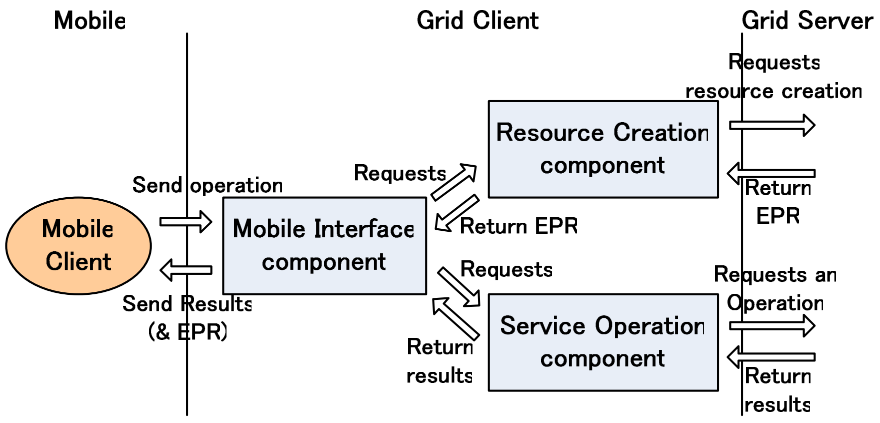


Figure 7. The relation of the components in the Grid client

The Grid client program is created by client function of the platform and placed in a gateway. It is including three components, Mobile Interface, Resource Creation, and Operation Service. The mobile interface acts as a communication window between a mobile phone and the Grid client. It receives a request of performing a particular operation and forwards the request to the service operation, and receives the operation results from the service operation and forwards the results to the mobile phone. The service operation receives a request of performing an operation and invokes an instance service in the Grid server. The resource creation provides a function that makes a request of creating the resource to the Grid server and gets the EPR returned and passes it to the mobile phone via the mobile interface. The relation of the three components is shown in Figure 7.

3.3 Mobile Device Side Architecture

Finally, it explains architecture on the mobile device side accessed the Grid client. A mobile access client is written based on the JavaME platform and the MIDP library [9]. In this section, first it explains the JavaME platform, and then it explains the J2ME Wireless Toolkit that is the simulation tool to execute the application, finally it explains about the function of the platform on the mobile device side.

3.3.1 Java 2 Platform, Micro Edition (JavaME)

JavaME technology was originally created in order to deal with the constraints associated with building applications for small devices [9]. For the purpose Sun defined the basics for JavaME technology to fit such a limited environment and make it possible to create Java applications running on small devices with limited memory, display and power capacity. JavaME platform is a collection of technologies and specifications that can be combined to construct a complete Java runtime environment specifically to fit the requirements of a particular device or market. This offers a flexibility and co-existence for all the players in the eco-system to seamlessly cooperate to offer the most appealing experience for the end-user. The JavaME technology is based on three elements;

- A configuration provides the most basic set of libraries and virtual machine capabilities for a broad range of devices,
- A profile is a set of APIs that support a narrower range of devices, and
- An optional package is a set of technology-specific APIs.

Over time the JavaME platform has been divided into two base configurations, one to fit small mobile devices and one to be targeted towards more capable mobile devices like smart-phones and set top boxes. The configuration for small devices is called the Connected Limited Device Profile (CLDC) and the more capable configuration is called the Connected Device Profile (CDC).

There are two advantages of JavaME. One is to be adopted by two or more communication company, and to be able to distribute the application program to more users. The second is to provide a general developer with the environment to develop and distribute the application program. From these advantages, there is a possibility that any developer can create a mobile application using more attractive grid service, and a mobile application using the grid computing will spread. However, it is necessary to understand any difficult steps and complex specification to develop the mobile application that uses the grid resource. However, the user will be able to develop the mobile application more easily by using the platform.

3.3.2 Sun Java Wireless Toolkit

The Sun Java Wireless Toolkit (J2ME Wireless Toolkit) is a state-of-the-art toolbox for developing wireless applications that are based on JavaME's CLDC and MIDP, and designed to run on cell phones, mainstream personal digital assistants, and other small mobile devices [10]. The toolkit includes the emulation environments, performance optimization and tuning features, documentation, and examples that developers need to bring efficient and successful wireless applications to market quickly.

3.3.3 Mobile Side Functions

The platform provides the user with the environment to develop the framework for accessing the Grid client on the gateway from the JavaME access client on the mobile phone. Their functions are briefly described list below.

- ***Deploy files on server*** copies the grid client, the stub files, and the Globus Toolkit API onto the key location on the gateway.
- ***Server*** includes two functions, "Start" and "Stop". If "Start" is selected, the server is started, and if "Stop" is selected, the gateway server is stopped.
- ***Generate Midlet Suite*** generates the files and the packages to develop a mobile application.
- ***Deploy files on Wireless Toolkit*** copies the file, the package, and the Java file which is described onto the key location of J2ME Wireless Toolkit.
- ***Run*** executes the developed application by using J2ME Wireless Toolkit.

The platform provides the user with the environment to develop the framework for accessing the Grid client on the gateway from the mobile phone. The platform automatically puts the Grid access client, the stubs, the API of Globus Toolkit on the gateway, and sets the environment variables. Also, in regard to the development of the mobile application, BtoG platform provides not only the functions for executing the JavaME application, for example, methods to start and destroy the application, but also functions for communicating with the Grid client on the gateway by HttpConnection. The Midlet suite including the mobile access client is generated by the platform, and it is copied onto the key location of the Sun Java Wireless Toolkit (J2ME Wireless Toolkit) [10]. To develop the mobile application, the platform integrates the J2ME Wireless Toolkit, and the developer can simulate the mobile application. When the JavaME client accesses to the gateway, the mobile phone can refer to the EPR by storing the resource key and the service URI at the storage area called RMS (Record Management System). RMS is a data base system of a flat file format. Each table of the data base system is called "Record store", and each entry in the record store is called "Record". The API to create the record store and access to the record is described in the javax.microedition.rms package [11]. Because the manifest file and the jad file are also provided by the platform, a developer is able to execute the JavaME client over the J2ME wireless toolkit. Figure 8 shows the Midlet suite generated by the platform. The J2ME wireless toolkit can compile these files and generate a jar file. The user uses the application by downloading the jar file and jad file into the mobile device.

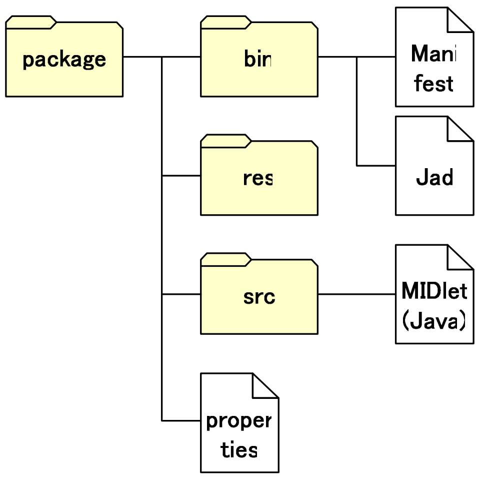


Figure 8. The hierarchy of the Midlet suite

3.4 Add service

Below gives an example to show how a user accesses a Grid service, named add() service. The add() service provides a simple calculation service that can be accessed from a mobile device. When JavaME client accesses the add() service for the first time, the Grid Client makes a request of creating the resource to the Grid server. The Grid server creates the resource and returns the EPR, and Grid client receives and saves it. The Grid client invokes the operation, add() , with the received EPR and receives the results from the Grid server and sends the results to the JavaME client via the mobile interface component.

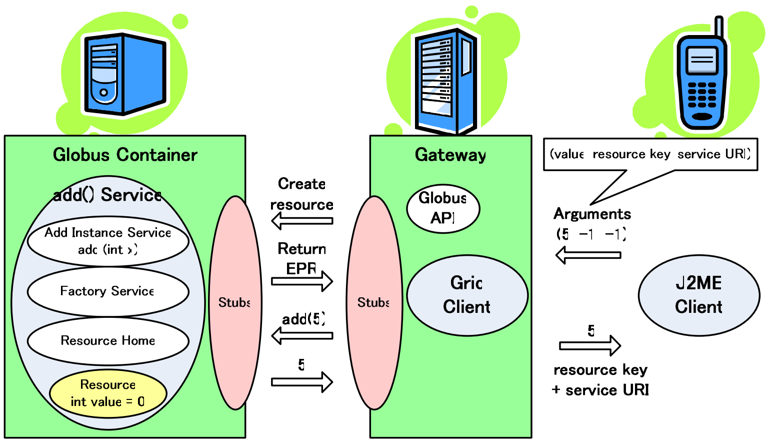


Figure 9. Accessing the Grid service, add(), from a mobile phone (for the first time)

The first time the JavaME client accesses the add() service, the resource is created and the EPR (including URI and resource key) is sent to the JavaME client. From the second time and onwards, there is no more need of the resource creation, instead the Grid client receives the EPR with which the Grid service, add(), is accessed and operated. The rest of the processes are same.

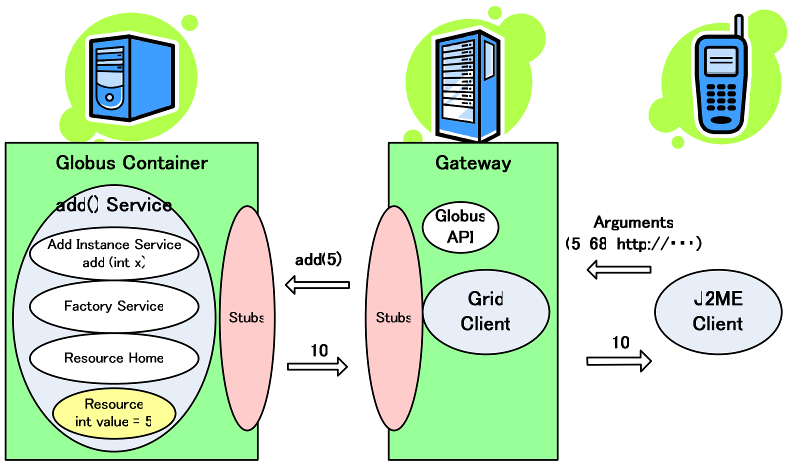


Figure 10. Accessing the Grid service, add(), from a mobile phone (from the second time and onwards)

4. Sample Application

This paper shows the skin diagnosis system as the sample application which the developer can develop on top of the platform. Skin diagnosis and care has been a popular topic with improving life standard, in particular to women. People were used to have to go to a beauty salon or clinic for getting skin diagnosed since it requires experts' knowledge or uses some software for diagnosis, which requires a great amount of processing and computation resources. But nowadays it becomes possible to diagnose skin in a cosmetic shop or counter or even recently just using a mobile phone due to advanced grid computing, ubiquitous computing, and Web technologies. The mobile phone based skin-grading system [12] is such a representative example application developed by Mizokuchi laboratory at Tokyo University of Science. A skin-diagnosis system in fact requires skin image processing for extracting skin related parameters, data mining process for extracting skin-diagnosis related rules, and skin-expert rulebase system for making recommendations. All processes are time consuming and in need of a great amount of processing power and computation resources, which are not be able to be performed in a mobile phone or not easy to be done in a personal PC at home. Therefore, skin-diagnosis is a good example application of grid service systems. In this section, first it introduces skin-grading system that Mizokuchi team developed, then it explains the problem of the system, and finally it proposes the Skin-diagnosis system based on the proposed framework.

4.1 Skin-Expert System

Mizokuchi's team has developed a system for sales of cosmetics based on a system called Skin-Expert, a skin-image grading service that includes analysis and diagnosis. Skin-Expert analyzes a customer's current skin quality from a picture of the skin. Several parameters are extracted by image processing, and the skin grading is done by rules generated by data mining from a baseline of grades given by human skin-care experts. Communication with the Skin-Expert is through a cellular telephone with a camera, using e-mail software and a Web browser. The User photographs the own skin using the camera in a standard cellular telephone and then send an e-mail message that includes the picture as an attachment to their analysis system. Other parameters associated with the user (for example, age and gender) are included in the body of the message. The picture is analyzed by their skin-grading system, and the results are made available as a page in HTML format on a user-accessible Web site. An e-mail is sent when the results are available, usually within minutes. The User checks the results by using a Web browser on his/her cellular telephones. The output not only provides a grading result but also gives recommendations for the care and cosmetics that are most suitable for the user. The system integrates cellular communication, Web technology, computer analysis, data mining, and an expert system. Though user uses only a cellular telephone with very little computing power as the front end, his/her can take advantage of intelligent services such as computer grading and data mining. The user does not need to think about what is running in the background, and there is no requirement that end users have any special hardware.

Mizokuchi's team tests his skin-grading system in the Grid computing environment [12], constructed a 100 megabyte per second optic fiber network It uses a mail server and a Web browser to communicate a mobile phone with the Grid server. The Grid computer could process sixteen requests per second. Ten to fifteen accesses were available per second for the Web server. The mail server could receive only two request messages per second, that is to say, only two requests could be processed per second since the mail server is the bottleneck in the environment.

4.2 Skin-Grading System on top of the proposed framework

The proposed system on top of platform strips down the mail server so as to overcome the bottleneck problem occurred in the mail server. Instead, the Grid client is automatically deployed in a gateway and a mobile phone communicates with the gateway rather than directly with the Grid server. Besides, the platform includes the Grid service, Grid client and mobile client developing platform, which would make development of similar skin-diagnosis and care system easier. Figure 11 presents the working flow of the skin-diagnosis system on top of the platform.

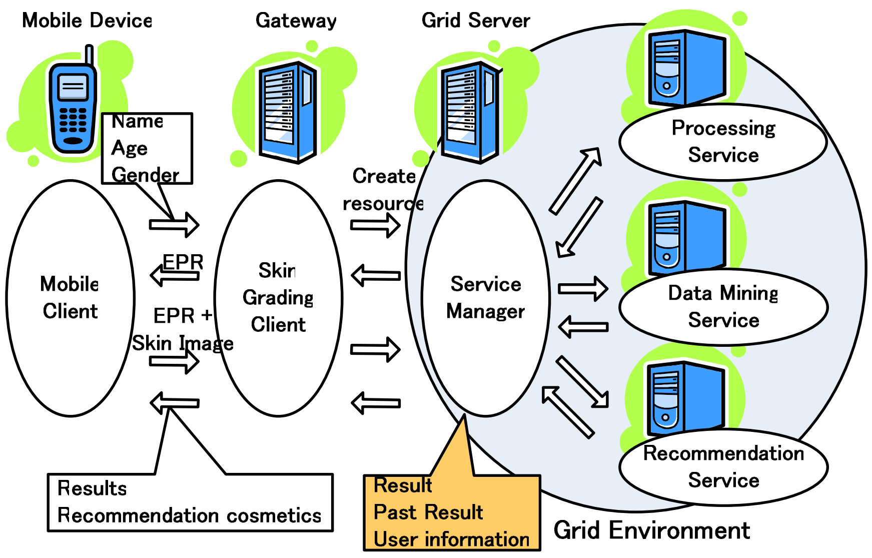


Figure 11. Working flow of the Skin-diagnosis system

First, a user inputs and sends personal information (name, age, gender and so on). The gateway requests the service manager to create the EPR and returns back to the mobile phone. These processes are not need next time. Then, a user takes a photograph of her skin with a mobile phone with camera function and the skin image with resource key and service URI is sent to the gateway. The skin grading client requests to operate the service manager which directs the request to all related services such as the processing service, the data mining service, and the recommendation service. The service manager has the resource properties (result, past result and user information). If the service manager doesn't have the resource properties, the skin grading client creates them by the factory service in the service manager. The processing service receives the skin image from the service manager, analyzes it and sends the results back to the service manager. The data mining service receives the user information and the analysis results and extracts from the received data, information, and results to format a set of rules, and the location of the rulebase is sent back to the service manager. The recommendation service includes an inference engine to make recommendations for the use with the rulebase service and sends the recommendation cosmetics back to the service manager. Finally, the skin grading client receives the results and the recommendation cosmetics, and sends the results to the mobile phone end. The UML diagram of skin-diagnosis system is shown in Figure 12.

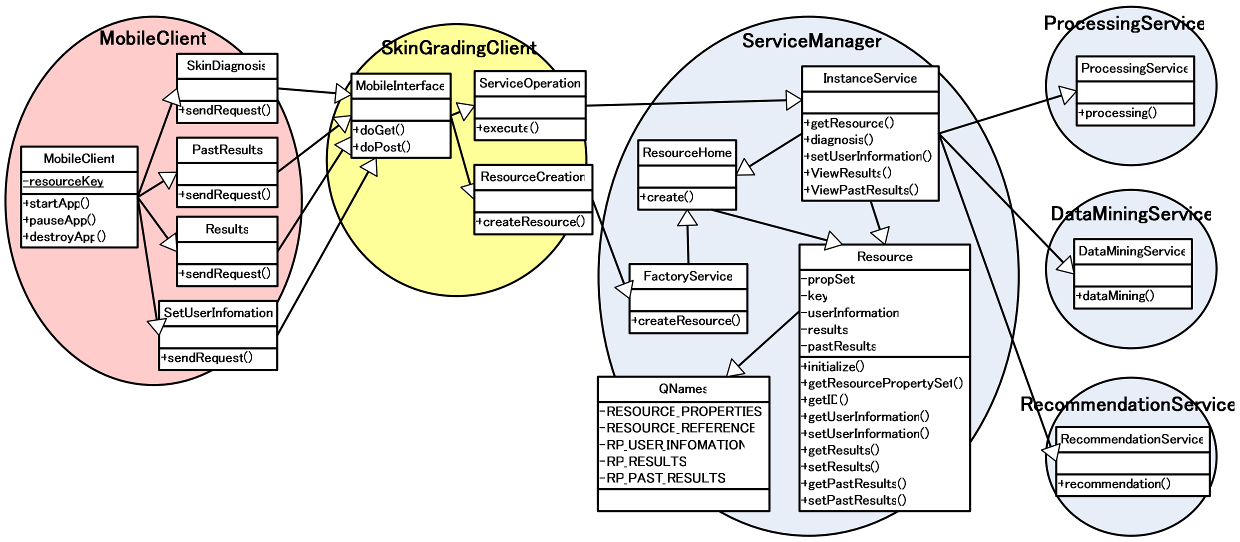


Figure 12. UML diagram of the skin-diagnosis system

A *MobileClient* is the main class in the mobile client. The method *SkinDiagnosis* can send a request of skin diagnosis to the Grid client. A user can refer to the results by using the method *Results*. Also a user can refer to the past results by using the method, *PastResults*. *SetUserInformation* is the class for inputting user information (For example, name, age, gender, and so on). *MobileInterface* is the main class in the skin grading client. This class receives the requests from the mobile client, and returns the diagnosis results. *ResourceCreation* is the class to get the EPR. This class requests the creation of a resource, and sends the EPR of the resource to the *MobileInterface*. *OperationService* is the class for requesting the operation. *ResourceHome* is the class to manage the resources. *FactoryService* uses this class to create new resources. Also, *InstanceService* uses this class to operate the request on the resource properties. *InstanceService* requests *ProcessingService* for skin analysis, requests *DataMiningService* for extracting data to make rules, and requests *ResommendationService* for finding the recommendation cosmetics. The time sequence of a scenario in skin diagnosis system is given in Figure 13. This is an example when the user diagnosis the skin for the first time.

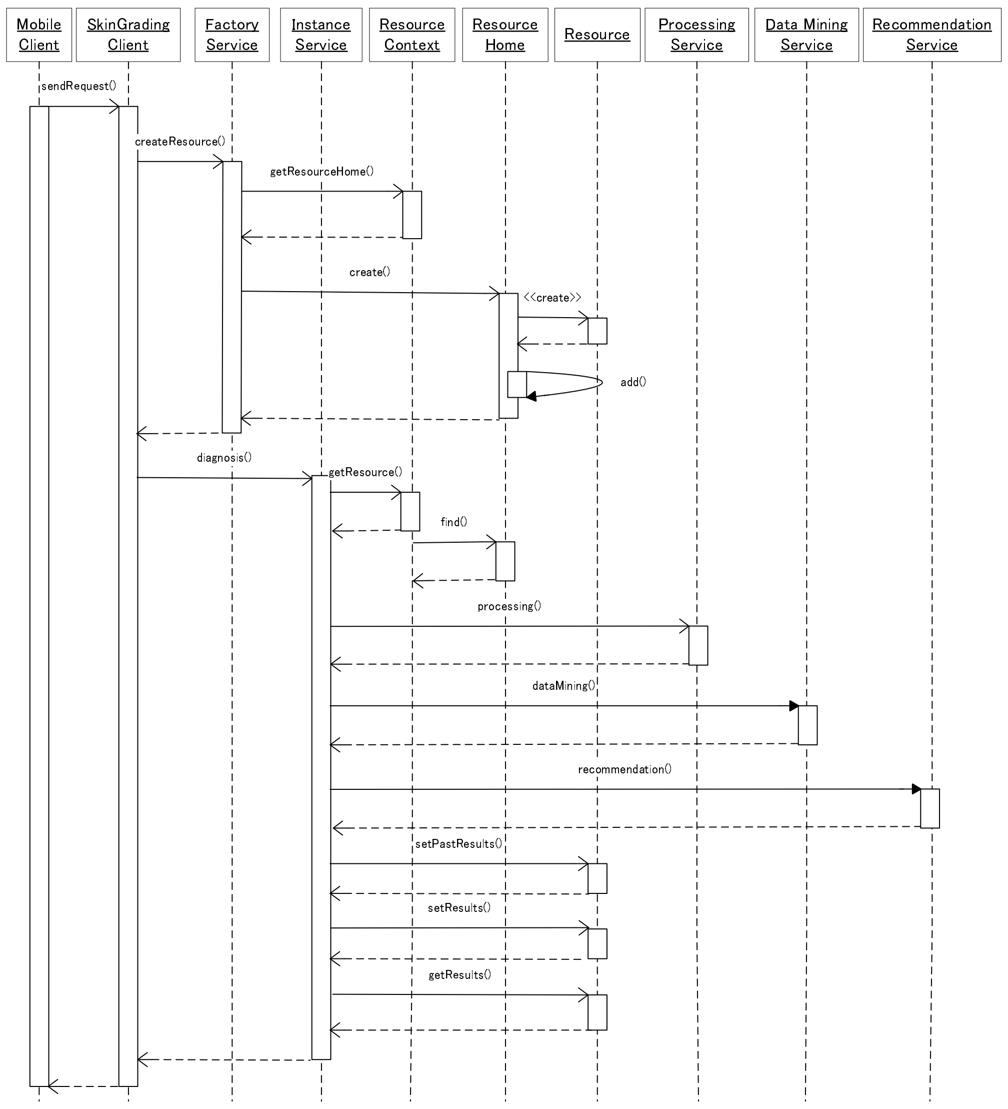


Figure 13 Time sequence of a case in skin-diagnosis system

5. Related Work and Comparisons

The proposed platform has the feature that it supports all steps from the development of the Grid service to the mobile client. However, there are a number of development environments for developing the Grid service and accessing the Grid resources. This section introduces those tools and compares them with the proposed platform. Also, to make the Grid resources available and accessible anytime anywhere from ubiquitous devices, a number of researches for accessing the Grid resources have investigated the possibilities. This section also shows those researches, compares these researches with the proposed framework, and compares the skin-expert system developed by Mizokuchi team with the skin-grading system based on the proposed framework.

5.1 Related work for developing grid services

To allow Globus programmers to develop Grid services easily, Globus Service Build Tools [13] developed by the GSBT project team provides an environment. It integrates all the steps from code to deployment. Below lists are functions of the Globus Service Build Tools.

- Generation of Namespace-to-package mappings file
- Generation of WSDD file
- Generation of JNDI deployment file
- Initial WSDL generation
- Initial Java generation
- WSDL↔Java synchronization
- GAR generation from Eclipse

The advantage of this tool is that Eclipse is a Java environment and Plug-in for Eclipse makes implementations of defined Grid services more effective in Eclipse environment. Moreover, it provides the function to convert the implementation file to WSDL automatically. However, it has some disadvantages that it provides only functions that mainly facilitate server side capabilities and a limited number of functions such that developers are restricted to few Grid services.

The Java Commodity Grid (CoG) Kits [14] allow Grid users, Grid application developers, and Grid administrators to use, program, and administer Grids from a higher-level framework. This tool kits allow for easy and rapid Grid application development. They encourage collaborative code reuse and avoid the duplication of effort among problem solving environments, science portals, Grid middleware, and collaboratory pilots. It offers components that provide mainly client and limited server side capabilities. The advantage is that it hides the complex infrastructure and provides convenient access to Grid functionality through pure Java client-side classes and components. However, a user has to have Java programming skill and has to familiar with CoG Kit classes and components, and CoG Kit itself is a quiet complicated commodity framework and it will take users lot of time before utilizing its functions.

The proposed platform has two features that should be emphasized. One is that it is aimed at facilitating capabilities of both creating the Grid services and accessing the deployed services. While the Globus Service Build Tools is mainly focused on providing easier environment only in Eclipse environment for developing the Grid services with less consideration of client program that runs and accesses the deployed services. In reverse, the Java CoG Kit is mainly for providing client side functions and components with little consideration of writing the grid services. Another feature of the proposed platform is that the platform is designed for naïve developers who have less special expert knowledge about Java programming, web service and Grid service programming. Without knowing of the detail about infrastructure, writing Java code, and writing (Grid) web service descriptions, developers can write the Grid services, run the deployed services and access the services with assistant of the user-friendly interface and wizard guidance.

5.2 Related researches for accessing grid resources

Millard, etc.[2] describe their attempts of writing Grid clients for mobile devices, such as a PDA which have restrictive computational and storage facilities, and implementing a Web-based proxy that uses Globus Toolkit 3 to talk to the Grid services. Their experiences are based on an implementation of a mobile grid client for an existing Web-based e-learning system. In developing Mobile Grid clients for the grid services, they have investigated three main approaches. There are Java-based Grid clients, .Net-based Grid clients and Proxy-based Grid clients. They tried several mobile JVMs on their IPAQ PPC looking for one that could support a minimum set of APIs to invoke GT3 Grid services. However, they have concluded that there is no Java-based OGSI implementation for mobile devices.

Chu, etc. [7] designs, implements and evaluates Mobile OGSI.NET, which extends an implementation of Grid computing, OGSI.NET, to mobile devices. Mobile OGSI.NET emerged as the only real attempt to provide an OGSI implementation for mobile devices. However, OGSI.NET is inappropriate because it is not available on mobile device, has questionable interoperability with existing Grid systems such as GT3, and has been deprecated in favor of WSRF.

Guan, etc. [1] presents a system infrastructure that allows local mobile devices to interact with the Grid. Central to the infrastructure is a proxy with the ability of dual connectivity to transfer the request from the mobile device to the Grid. In their system, there are mobile devices, proxy devices, and Grid services. Mobile devices are usually portable but resource-limited devices with various sorts of networking capability. Proxy devices might be a desktop computer or a small server available for nearby mobile devices via the local wireless LAN. It is also connected to the Grid via a high-speed network. All kinds of Grid services are registered at the proxy, providing a convenient mechanism for mobile devices to find the needed Grid service. The requests from mobile devices are first processed on the proxy, and if necessary suitable Grid services requests are invoked. Once the proxy receives the result from the Grid, it generates and serves the appropriate display form to the mobile devices. Although it enables mobile devices to make effective use of Grid service, they can be used only in the local area.

GridLite [15] is an extensible framework that provides services to users on ubiquitous, resource-limited devices within a Grid infrastructure. It uses a server infrastructure for provisioning of persistent services, and smart helper services running on the "lite" devices which tap into this infrastructure. One of the goals of GridLite research is to define a Grid architecture which manages these devices such that their resource constraints are minimized by the intelligent Grid infrastructure. This is done by defining new services for managing various resources. Another goal of GridLite is to provide mobile users ubiquitous access to Grid resources, higher productivity and entertainment. However, currently GridLite framework can not work with Web Service Resource Framework (WSRF).

Grabowski, etc. [16] also examine the problem of giving the Grid users a possibility to access their applications and resources from any place using mobile devices. Because of limitations of mobile devices, they assumes adopting gateway between the client and the Grid and develops a set of applications in the client-server model with the J2ME CLDC/MIDP-client, and portlet server working with GridSphere[17]. The GridSphere Portal is a Java Servlets/JSP based framework that builds upon the Java CoG Kit [14], Globus, and MyProxy [18] to support such features as single sign-on, job submission, and data movement. As a production environment, the GridSphere Portal provides online tools for administering users of the portal, as well as the Grid resources and services user access. However, the GridSphere only provides the basic infrastructure required for developing and administering Web portals. Though it is a framework to make use of the grid services, it can not build the Grid services.

5.3 The advantages of BtoG

The proposed platform has some advantages to achieve ubiquitous Grid computing compared with these related researches. They are as follows:

- Accessing Grid resources anytime anywhere from ubiquitous devices.
- Working with Web Service Resource Framework (WSRF) that is common standard.
- Reducing data of ubiquitous device by using JavaME and gateway.
- Providing all the steps from developing Grid services to accessing from ubiquitous devices.

The proposed platform supports all processes from the development of the Grid service to the access by the mobile devices. This is more advantageous than the tool that develops only the Grid service like Globus Service Build Tools, and the tool that develops only the part of accessing the Grid resources such as Java Cog Kit. The developer only has to use only the proposed platform without installing various tools in the computer, and using them.

5.4 Comparison with the Skin-Expert system

Compared to the skin-diagnosis system developed by Mizokuchi's team, the proposed system on top of the platform strips down the mail server so as to overcome the bottle neck problem occurred in the mail server. Mizokuchi's team has set up a mail server on each machine to avoid the bottle neck. They enable to process sixteen requests per second. Even if one hundred thousand requests arrive at the same time, all requests can be finished in about one and one-half hours. However, the more the mail server is set up, the more the administrator has to engage in maintenance and security. The proposed system will enable to process almost equal requests without setting up the mail server.

6. Conclusions

This paper presents a user-friendly platform not only for developing Grid services and the Grid client program but also creating JavaME client to enable ubiquitous devices like mobile phone, PDA, etc. to access the Grid resources. It has several advantages over other related work, such as working with the standard framework (WSRF) and emphasizing the feature of accessibility of Grid resources anytime anywhere any ubiquitous devices. An example application of skin grading and caring on top of the platform is compared with the similar system developed by Mizokuchi. It is addressed that the proposed system strips down the e-mail servers, which are required in Mizokuchi's system, to overcome the bottleneck problem of processing requests.

This platform can provide the developer with an environment to develop a framework for using the Grid resources from the mobile device. However, there are still remaining work and improvements for future. The specification and the functions of mobile device vary with different models. This is the platform for developing the applications based on MIDP. However, it is not suitable for developing the application based on Doja, the unique profile developed by the company, NTT docomo in Japan. It is necessary to improve the platform that can correspond with different profile formats and communication protocols. Since the platform is open source, it is possible to build further necessary functions for corresponding with different profiles and add them to the platform as plug-in.

References

[1] T.Guan, E.Zaluska, D.Roure, "A Grid Service Infrastructure for Mobile Devices", *Proceedings of 1st Semantic Knowledge and Grid conference,* Beijing, China

[2] David E. Millard, Arouna Woukeu, Feng(Barry) Tap, Hugh C. Davis, "Experiences with Writing Grid Clients for Mobile Devices", *Proc' of 1st International ELeGI Conference*, Vico Equense - Napoli(Italy), 2005.

[3] The Globus Alliance, http://www.globus.org/toolkit/

[4] Hiroyuki Morohoshi, Runhe Huang, and Jianhua Ma, "A user-friendly platform for developing and accessing grid services", *Information Systems Frontiers*, Springer Netherlands, 2006.

[5] Oasis, "Web Services Resource 1.2 (WS-Resource)", 1 April 2006, http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

[6] The Apache Jakarta Project, http://jakarta.apache.org/velocity/

[7] D.Chu, and M.Humphrey, "Mobile OGSI.NET: Grid Computing on Mobile Devices", 5th IEEE/ACM International Workshop on Grid Computing - Grid2004 (at Supercomputing 2004). Nov 8 2004, Pittsburgh, PA

[8] Borja Sotomayor, “The Globus Toolkit 4 Programmer's Tutorial”, http://gdp.globus.org/gt4-tutorial/

[9] Sun Microsystems, “Java ME Technology”, http://java.sun.com/javame/technology/index.jsp

[10] Sun Microsystems, “Sun Java Wireless Toolkit for CLDC”, http://java.sun.com/products/sjwtoolkit/index.html

[11] Yu Feng, Jun Zhu. *Wireless Java Programming with J2ME*, ASCII, 2002

[12] Hironori Hiraishi, Fumio Mizoguchi, “A cellular telephone-based application for skin-grading to support cosmetic sales”, *AI Magazine*, American Association for Artificial Intelligence, 2004

[13] Java Cog Kit, http://wiki.cogkit.org/index.php/Main_Page

[14] Globus Service Build Tools, http://gsbt.sourceforge.net/

[15] Raj Kumar, Xiang Song, “GridLite: A Framework for Managing and Provisioning Services on Grid-Enabled Resource Limited Devices”, 2005

[16] Piotr Grabowski, Bartosz Lewandowski, “ACCESS FROM J2ME-ENABLED MOBILE DEVICES TO GRID SERVICES”, 2004

[17] GridSphere, http://www.gridsphere.org/

[18] MyProxy, http://grid.ncsa.uiuc.edu/myproxy/