International Journal of Geo-Information

MDPI

Article

Linkage of OGC WPS 2.0 to the e-Government Standard Framework in Korea: An Implementation Case for Geo-Spatial Image Processing

Gooseon Yoon [1], Kwangseob Kim [2] and Kiwon Lee [3,*]

[1] Department of Information Systems Engineering, Hansung University, Seoul 136-792, Korea; gxexe3@naver.com
[2] Department of Information and Computer Engineering, Hansung University, Seoul 136-792, Korea; lovekph@nate.com
[3] Department of Electronics and Information Engineering, Hansung University, Seoul 136-792, Korea
* Correspondence: kilee@hansung.ac.kr; Tel.: +82-2-760-4254

Academic Editors: Ozgun Akcay and Wolfgang Kainz
Received: 15 October 2016; Accepted: 16 January 2017; Published: 20 January 2017

Abstract: There are many cases wherein services offered in geospatial sectors are integrated with other fields. In addition, services utilizing satellite data play important roles in daily life and in sectors such as environment and science. Therefore, a management structure appropriate to the scale of the system should be clearly defined. The motivation of this study is to resolve issues, apply standards related to a target system, and provide practical strategies with a technical basis. South Korea uses the e-Government Standard Framework, using the Java-based Spring framework, to provide guidelines and environments with common configurations and functions for developing web-based information systems for public services. This web framework offers common sources and resources for data processing and interface connection to help developers focus on business logic in designing a web system. In this study, a geospatial image processing system—linked with the Open Geospatial Consortium (OGC) Web Processing Service (WPS) 2.0 standard for real geospatial information processing, and based on this standard framework—was designed and built utilizing fully open sources. This is the first case of implementation based on WPS 2.0 running on the e-Government Standard Framework. Establishing a standard for its use will be important, and the system built in this study can serve as a reference for the foundational architecture in building geospatial web service systems with geodata-processing functionalities in government agencies.

Keywords: OGC WPS 2.0; the e-Government Web Framework; geospatial image processing; open source; ZOO-project

1. Introduction

With advances in computer technology and growth in demand for services utilizing it, diverse information has become available. For example, geospatial information and geo-based satellite images are used as base maps in services such as route guidance navigation, portal map service, facility management, and site suitability analysis, and are used in environmental application of nearly inaccessible areas and regional analysis through geo-based image processing. In addition, remotely sensed data and information may be used by stakeholders to make effective decisions in managing disasters [1]. It is expected that services will emerge in a much improved form as better means and methods are applied because of information technology development [2]. We should consider designs that are capable of managing complicated services systemically and expanding them easily. For engineering issues in geospatial applications, integration, customization, or optimization

with multiple close or loosely coupled technological components on matured or maturing stages are also important. Accordingly, necessity of international standard interfaces and standardized frameworks has been increased. This work presents an integrated application based on an open source strategy with heterogeneous standard sources, such as international standards for geospatial area and a standard framework for so-called electronic government.

The International Organization for Standardization (ISO), which develops and distributes internationally accepted standards, and the Open Geospatial Consortium, Inc. (OGC), which leads geospatial industry standards, are developing a standard for geospatial information. ISO is in the process of standardizing content related to collecting, processing, analyzing, and presenting geospatial information via the technical committee (ISO/TC) 211 applicable to Geographic Information System (GIS) standards [3]. OGC—an organization oriented toward open standards—researches and establishes technical standards for data compatibility and interoperability technical standards. The standards include Web Map Service (WMS), Web Feature Service (WFS), Web Coverage Service (WCS), and Web Processing Service (WPS) [4]. A system developed in compliance with international standards shows many advantages, such as information integration, quality improvement, consistency, easy maintenance, and cost reduction in utilization of geospatial information [5,6]. Additionally, it allows existing and new systems to share and distribute information, thereby improving compatibility and interoperability. Therefore, applying international standards should be considered in system design stages to facilitate efficient operation and geospatial service management. Among OGC standards, WPS—an interface and communication method in which geospatial processing can be defined and accessed from the Web—has compatibility with other OCG standard web services [7].

An electronic government, e-Government, in the Web environment is currently an important technical theme in most countries [8–10]. In South Korea, the e-Government Standard Framework is developed and distributed for free to improve web system quality and help standardize and ease development, and it is based on the Spring Framework. The Spring Framework provides a comprehensive programming and configuration model for Java-based application and supports infrastructure at the application level [11]. The e-Government Standard Framework—which offers application architecture, basic functions and common components necessary for web system development—provides features such as Development Environment, Runtime Environment, Administration Environment, Operation Environment, and Common Components. Through these functions, the framework resolves dependency on a specific vendor solution and provides standards that can be linked with commercial solutions, ensuring their interoperability. In addition, it provides a function for hybrid app development. A hybrid app utilizes elements of both native applications for a specific platform on a mobile device and web applications for multiple platforms available over the Internet by any browser. Because common components can receive common modules premade at the time of building the Web system, rapid development and quality gap reduction relative to different systems can be predicted. There is also the advantage of increased reusability of developed modules.

As recent web services deal with multiple types of data and a large volume of content, distributed systems are inclined to be dominant over centralized ones. Furthermore, interoperable geoprocessing functionalities via WPS interface standards offers user operation to the client by requesting algorithms or functions in remote servers without installation of external ones that the user wants. Meanwhile, the e-Government framework is a basic requirement to develop a public web services system. If geo-based service systems with geodata-processing functions should be operated in the distributed environments in public sectors, both WPS and the e-Government framework are crucial factors from the viewpoint of software engineering. This is the case for Korea, but this situation may occur in other countries that already have the e-Government framework, or plan to establish it. That is the main motivation of this study.

This study reflects some benefits of standards and technology that are deemed important as systems become more complicated, and constructs a number of integrated trial system examples in connection with these merits. Among the reflected standards and technologies are WPS,

a geospatial standard, and the e-Government Standard Framework. The trial system had the ability to perform satellite image-processing functionalities and used a request interface based on the WPS 2.0 standard. The ZOO-Project [12], an open source framework to create and chain WPS-compliant web services, was used as a WPS platform to conform to WPS 2.0. The ZOO-Project provides various components to utilize WPS and comprises Server, Services, API (application programming interface), and Client. A geospatial information-processing interface can be used, and its function is provided via Server and Services. While the WPS interface offers ways to implement geospatial processing on the Web, appropriate functions should be developed separately for real processing. For this purpose, the Geospatial Data Abstraction Library (GDAL), a geospatial input/output library, and the Orfeo ToolBox (OTB), which focuses on satellite image-processing functions, were used. Both are open sources. The trial system was developed based on the e-Government Standard Framework. In South Korea, the e-Government Standard Framework is a set of specifications to guide the implementation of all types of web-based information systems for public services supported by governments and public enterprises. It uses basic requirements to develop web services. When government institutions or agencies develop a web-based system to serve geo-based data services and their derived contents, they should adopt this standard guide. When a development project is conducted based on this framework, complicated tasks—such as specific data processing in a certain application field—are partially supported, as is the case with Spring Framework features. Because common component functions are also available, the burden of development may be eased once the utilization method is understood. It is possible to deliver various information to the client—such as function lists, input variables, and status values—through developed modules. This trial system is a prototype for public services for interoperable processing of geo-based images among two or more remote servers managed by governments or public sectors. Therefore, it needs the WPS platform to receive satellite image-processing requests and execute them accordingly. The module in the trial system follows the WPS 2.0 communication method and is able to implement communication with other WPS 2.0-compliant servers.

2. OGC WPS 2.0 and Open Source

The OGC, an international standardization body, has established several standards to provide geospatial services on the Web. These OGC standards, widely used in industry and the academy, have also had great influence internationally [13–15]. Among them, WPS is a standard referred to when developing a web system to support geospatial processing. If developed in compliance with the standard, web systems can interact with other servers and improve system reusability. In addition, geospatial-processing applications can be improved with enhancement of interoperability and accessibility of geospatial information [16]. WPS 2.0 is currently the latest version, and it has been modified with added functions to meet the requirements of enhanced web technology. One of the major changes is that WPS 2.0 supports synchronous and asynchronous processing, while WPS 1.0 supports synchronous processing only. In conducting geospatial processing on the Web with the support of asynchronous processing, it is possible to build a system in which a new or the next geospatial-processing service can be initiated without waiting for completion of a previously executed geospatial process. WPS 2.0 has six interface definitions on flows, usages, requests, and responses: GetCapabilities, DescribeProcess, Execute, GetStatus, GetResult, and Dismiss [17].

GetCapabilities is an interface that returns metadata on WPS servers with a form of XML documents. Metadata include geospatial-processing function lists and WPS interface communication methods. The DescribeProcess interface returns detailed information on geospatial-processing functions in XML document form. When requesting DescribeProcess, identifiers should be sent together for geospatial-processing functions for which detail is desired. The returned detailed information includes descriptions of the function, input value, and result values. Execute is an interface that requests the execution of geospatial function processing. This interface does not wait for completion of the execution request, but it immediately returns a JobID (a job identifier) to XML

documents. The processing progress status and result value can be identified through the JobID. Several JobIDs can be created for one function, and each JobID performs geospatial processing individually. This new feature was added with the structure change, which supports asynchronous processing. GetStatus, an interface that shows progress status, returns one of four statuses to XML documents: Running, Succeeded, Failed, and Accepted. GetResult is an interface that returns geospatial-processing function result values corresponding to a JobID. If the progress status obtained through GetStatus is "Succeeded", XML documents containing the result value can be created on request. Dismiss is an interface that terminates a process corresponding to a JobID. Examples of WPS standard-compliant open sources include 52° North, Deegree, GeoServer, and PyWPS [18].

The ZOO-Project, an open source platform that supports both WPS 1.0 and WPS 2.0, was applied to the trial system for WPS 2.0 application. The ZOO-Project is developed in C, Python, and JavaScript and provides a developer-friendly framework for building WPS servers. For this purpose, it offers components to build and utilize WPS: ZOO-Kernel, ZOO-Services, ZOO-API, and ZOO-Client. ZOO-Kernel, a CGI (Common Gateway Interface) program, has interfaces and communication methods following WPS standard, with requests and responses conforming to the standard. ZOO-Services is a component compatible with ZOO-Kernel, and its main role is to manage and provide geospatial-processing function as a service. Services offered by ZOO-Services are composed of configuration files and source codes to be executed. These services can be developed and configured independently, or they can support interworking with geospatial open sources such as GDAL, OTB, the Computational Geometry Algorithms Library (CGAL) [19], Geographic Resources Analysis Support System (GRASS) GIS [20], and System for Automated Geoscientific Analyses (SAGA) GIS [21]. In the case of separate developments, interoperability is ensured across diverse environments because many types of development languages (including Python, PHP, Java, and JavaScript) are supported. When the developed service is registered on ZOO-Services, ZOO-Kernel provides the corresponding service via WPS request. A WPS server can be built using these two services as essential components of the ZOO-Project. ZOO-API, a library composed in JavaScript, provides an API (Application Programming Interface) that creates or executes services to be registered on ZOO-Services on servers. ZOO-Kernel and the JavaScript engine, SpiderMonkey, are required on the server-side to use ZOO-API. Lastly, ZOO-Client is a JavaScript API that can be used on the client side. It offers ways to interact with other WPS servers, including the ZOO-Project. ZOO-API and ZOO-Client, which are optional, provide web system development methods following WPS. With all the components provided by ZOO-Project and additional client developments, it is possible to build a geospatial-processing web system. However, the main focus of this study is building a system following and utilizing WPS 2.0 linked to the e-Government Standard Framework. Furthermore, the system was built considering linkages to open sources that will support WPS 2.0 in the future. Therefore, only essential components of the ZOO-Project were used, excluding ZOO-API and ZOO-Client, which were optional components.

Using interfaces and communication methods defined in WPS 2.0 makes it possible to show process information and build process-handling user interfaces. Process progress status can be checked in real time after a process execution request, and functions such as killing processes during processing can be implemented. This functionality is possible because the WPS 2.0 standard supports asynchronous processing when handling geospatial information, and a system structure capable of multiprocessing can be built using this feature.

3. The e-Government Standard Framework in South Korea

In South Korea, the e-Government Standard Framework (eGovframework) has been developed and applied so as to establish a basic environment standard required for development of web service systems applied to public projects. The eGovframework aims to standardize software and improve qualities and reusability of web services. It also intends to reduce product quality gaps between businesses and improve investment efficiency. The major features of the eGovframework include open standard conformance via open source utilization, submission of standards linkable to commercial

solutions, implementation of nationwide standardization of web system development, flexibility and ease of replacement through modularization of each service, support for mobile web and hybrid apps, and environment provisions for web system development. Use of the eGovframework for web system development offers many advantages, such as cost reduction through reuse of common components, resolution of dependency on specific vendor solutions, improved interoperability with commercial solutions, and easy maintenance. Further, the eGovframework has been distributed for free to promote its usage in private sectors as well as public projects, and as of June 2016, it has been applied to 649 public and private information system projects in areas including administration, housing, disaster prevention, and statistics. As proof of its practicality, the eGovframework has been rapidly spreading across South Korea and has been applied to government information systems in various countries: the e-learning system in Saudi Arabia, the urban administration system of Da Nang city in Vietnam, the medical information platform in Mexico, the electronic customs system in Ecuador, the e-bidding system in Tunisia, and so forth [22].

The eGovframework is composed of application architecture, Runtime Environment, Development Environment, Operation Environment, Management Environment, Mobile Device API, and Common Components, which are required for building web systems. The Runtime Environment, based on the Spring Framework, is an application environment that provides common modules necessary for execution. The Spring Framework, an open source web framework based on Java, offers various services for dynamic web system development. The Runtime Environment is comprised of 7 service groups—including common foundation, display processing, mobile display processing and data processing—and provides 38 services in total. The Development Environment is a component offering an environment required for web system development. A host of environments, including Data Development Tool, Test Automize Tool, Code Inspection Tool, Template Project Generation Tool, and Common Component Tool, can facilitate building of automated and optimized development environment. The Operation Environment provides a monitoring tool, a communication tool, and a batch operation tool in the Runtime Environment. The Management Environment manages the version and status of the eGovframework. The Mobile Device API offers various APIs capable of directly accessing and using mobile device resources in mobile hybrid apps. In addition, it provides Runtime Environment APIs that support implementation and execution of device applications based on web resources, and Development Environment APIs that can facilitate device application development in the Android-based environment. Lastly, Common Components are a collection of developed components focusing on common reusable functions in building web systems. The Common Component is designed and developed in accordance with Model, View, and Controller (MVC) Architecture, based on the eGovframework.

Table 1 shows the compositions and types of Common Components that are composed of Common Technological Service, Elementary Technological Service, and New Mobile Common Component.

This is a classification of results considering frequency of redundant developments, reusability, and standardization application. This classification also elicits functions with high development productivity and efficiency, required for building web systems. Explanations for each function are as follows: Common Technological Service, a common component that runs on the eGovframework, comprises user directory/authentication, security, statistics/reporting, collaboration, user support, system management, system/service integration, and digital asset management, providing 136 components in total. Elementary Technological Service is a common component that works in a normal Java environment, regardless of the eGovframework. The Elementary Technological Service provides 104 components including utilities such as calendar and format/calculation/conversion. The New Mobile Common Component offers functions optimized for mobile devices, utilizing the User Experience (UX) support function. Additionally, the New Mobile Common Component, which includes general common components, provides 11 components, including mobile common technology, support service, and mobile device support components.

Table 1. South Korean e-Government Standard Framework (eGovframework) component list.

Component Type		Component
Common Technological Service	User Directory/Authentication	General Login, login with authentication token, login policy, etc.
		Converted from Mobile Common Component: General Login
	Security	Services including authentication, permission administration, encryption/decryption, etc.
	Statistics/Reporting	Services including Statistics on posting, access, report, etc.
	Collaboration	Services including Bulletin board, community, directory, etc.
		Converted from Mobile Common Component: services including Bulletin board, community, directory, etc.
	User support	Services including user administration, inquiry administration, questionnaire administration, FAQ, Q&A, etc.
		Converted from Mobile Common Component: services including user administration, inquiry administration, questionnaire administration, FAQ, Q&A, etc.
	System Management	Services including common code management, menu/log administration, institution code, batch management, etc.
	System/Service Integration	Services including Institution/Interface administration, etc.
	Digital Asset Management	Services including Knowledge, Knowledge Map, Knowledge Evaluation, etc.
Elementary Technological Services (Common Utilities)		Services including calendar, format/calculation/conversion, validity check for format/calculation/conversion, etc.
New Mobile Common Component	Mobile Common Service	Real-time Notification Service, Mobile Chart/Graph, Mobile Photo Album, Synchronize service, Off-line service
	Support	Offline web service, MMS service connection, OPEN-API Connection Service
	Device support	Location information connection, Multimedia control, Mobile Device Identification

Reference: http://www.egovframe.go.kr.

4. Linkage of WPS and the eGovframework

Various model studies based on WPS were reviewed for reference to design an integrated trial system linking WPS 2.0 to the eGovframework. The system design and WPS utilization case studies include development of geospatial-processing workflow design tools. First, Open Modeling Interface (MI), WPS, and Sensor Web Enablement (SWE) were implemented; Second, various geospatial information analysis model encapsulation methods were developed following the WPS standard; Third, combining Open MI and WPS, a web service model was formed; Finally, a web service design for automatic quality evaluation was applied [23–26]. These study cases used WPS-related information for reference in stages for designing and building systems. Research cases on developing open sources, such as 52° North [27] and PyWPS [28], were also consulted. Subjects of other open source case studies included models capable of creating thematic maps on the Web by linking WMS, WFS, and WPS; geospatial information distributed processing implementation utilizing WPS; and design and development of geospatial automatic interpolation web services [29–31]. Reference cases also included use of WPS 1.0 and WPS 2.0 together, linking the Spring Framework to WPS, and visualization of public data and geospatial data based on the eGovframework [32–34].

The integrated trial system, designed by linking WPS 2.0 with the eGovframework and comprised of the server and client, was built using a number of open sources. Table 2 shows the environments and open sources used to build the trial system. The Web environment was constructed using Ubuntu, Apache, and Tomcat, on which the eGovframework-based web system was built. WPS standard application and satellite processing were implemented using open sources, without its direct implementation. The ZOO-Project was utilized for WPS standard application, and GDAL and OTB were used for satellite image processing. GeoServer [35], a geospatial data server, was employed to manage satellite images and processing results; the client can call geospatial data easily and visualize it. On the client side, JQuery [36], a JavaScript library, and OpenLayers 3 [37], a web-mapping library, were used to compose the user interface (UI) and visualize processing results based on data returned to the WPS interface.

Figure 1 shows the design diagram of the integrated trial system. The client is composed of WPS 2.0 request modules, modules composing the UI based on returned data, and modules visualizing satellite images and processing results. The request modules conduct requests through

XML binding in accordance with the request method of the WPS interface. For this purpose, a request schema appropriate to each interface needs to be built for each module. For building schema and request, GetStatus and GetResult request modules can be used after an Execute request because they require JobID values. The UI composition module comprises satellite image-processing lists, processing function, and progress status on the client screen. The visualization module visualizes background maps, satellite images registered on GeoServer, and processing results. Processing results are visualized using GeoServer layer names returned from GetResult requests. All client-side modules were implemented using jQuery, and, in the case of the visualization module, OpenLayers 3 was utilized to visualize required geospatial information.

Table 2. Web Processing Service (WPS) 2.0 processing system based on the eGovframework using open source.

System Environment			Name/Version
Server	WPS 2.0 Processing	Operating System	Ubuntu/14.04.4 LTS
		Web Server	Apache/2.4.7
		Web Container	Tomcat/8.0.36
		SDK (Software Development Kit)	JDK (Java Development Kit)/7
		Standard Web Framework	The eGovframework in Korea/3.5.1
		WPS Platform	ZOO Project/1.5
		Geospatial Data Server	GeoServer/2.8.3
	Satellite Image Processing Software based on Open Source	Processing	Orfeo ToolBox/5.2.1
		I/O Libraries	GDAL/1.11.2
	Client	JavaScript Library	jQuery/3.1.0
		Web Mapping Library	OpenLayers/3.17.1

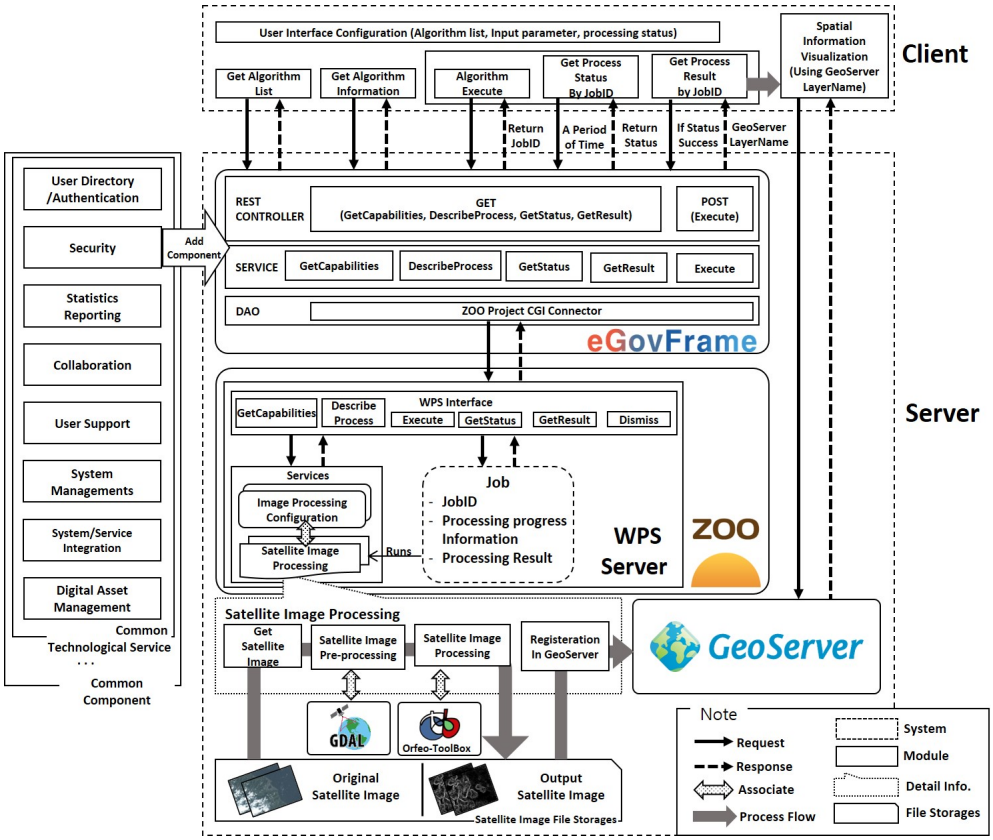

Figure 1. System design based on the eGovframework using open sources for geoprocessing and manipulation of geo-based images, including the ZOO-project.

The server is composed of a web system based on the eGovframework, the ZOO-Project, and GeoServer. The Rest Controller receives WPS 2.0 requests from the client on the Web system; GetCapabilities, DescribeProcess, GetStatus and GetResult requests are made with the GET method and Execute requests with the POST method. Rest Controller runs services corresponding to the received requests, connects to the ZOO-Project via the Data Access Object (DAO), and retrieves XML documents matching the request. The service extracts only necessary information from the XML documents, sends it to Rest Controller, and returns it to the client. Information extracted per request includes the satellite image-processing function list, satellite image-processing function information details, processing result, and JobIDs, which are contained in metadata. The ZOO-Project is designed to return information requested from the DAO or carry out an Execute request. This function exists because the DAO conducts connection requests to the ZOO-Kernel through the ZOO-Project CGI Connector. The ZOO-Kernel plays a server role through interaction with ZOO-Services, and ZOO-Services enables responses corresponding to WPS requests through registered services. While services included in ZOO-Services can be developed either independently or with open sources capable of interworking, this study provides services linking GDAL, OTB, and GeoServer. Because only open source linkage was considered in service development, Python was used among the development languages supported by the ZOO-Project.

As an open source utilized in linkage services, GDAL carries out tasks converting server satellite images into applicable satellite images, and OTB conducts satellite image-processing using the converted images. Finally, the processing result is registered on the GeoServer, and the client visualizes the processing result. The DAO of the Web system can carry out information and Execute requests on the satellite image-processing function by linking to ZOO-Project. On an Execute request, the ZOO-Project creates a JOB and executes registered services. The JOB has a JobID, progress status of service, and result values and can respond to service status or result requests.

5. A Trial System with Geospatial Image-Processing Function

A trial satellite image-processing system was built based on the diagram designed in Figure 1.

The eGovframework-based web system, which can be provided with a common component function, offers a development environment based on Eclipse [38]. In the development environment, it is possible to build an eGovframework-based web project. The project can add a common component function, which is shown in Figure 2. When common components are added, required functions can be selected and added; in the figure, the mobile common function and user authentication functions were added. When the common component function is added, the eGovframework package is created in the Java package, which includes packages related to mobile common functions and the eGovframework. By utilizing added packages, an ability to construct a web environment capable of running in a mobile environment and an authentication function, such as login, can be applied to the Web system. As the result of the application, Figure 3 shows a login combo box of the Web system and the entry panel accessed from a mobile device. Because of the user authentication common component, the login page is loaded when the Web system is accessed. Likewise, adding the mobile common component function causes the loading of a login page dedicated to mobile devices when accessed from the mobile web. Therefore, the common component function enables developers to add necessary functions immediately without the need for developing necessary functions on a web system.

Figure 4 shows the client UI screen constructed through WPS 2.0 requests. Figure 4a presents the satellite image-processing function list. When the client requests a function list, the Web system conducts the GetCapabilities service and receives XML documents containing metadata. The Web system refines the documents, extracts lists of satellite-processing functions and returns them to the client. Based on this, the satellite image-processing function list is visualized. Figure 4b is a modal view to implement the satellite image-processing function. When the data requested are necessary for modal view composition, the Web system runs the DescribeProcess service. After running the service, XML documents are received that contain detailed information of the

satellite image-processing function. The Web system extracts only necessary information to conduct the satellite image-processing function, and transfers it to the client. The client creates a modal view, based on the information received.

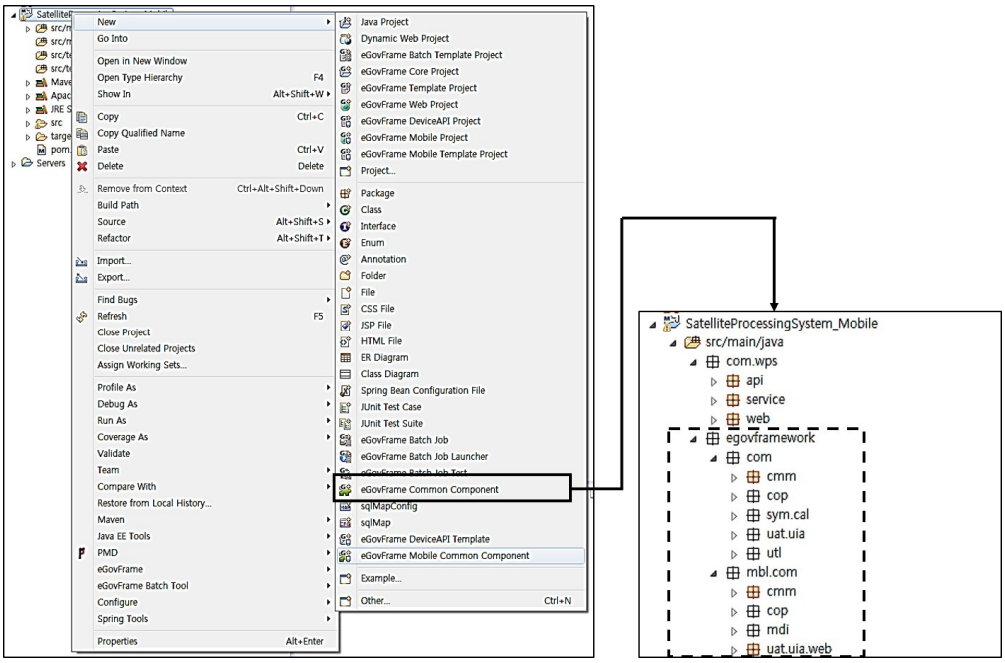

Figure 2. Application of common components for the eGovframework in South Korea.

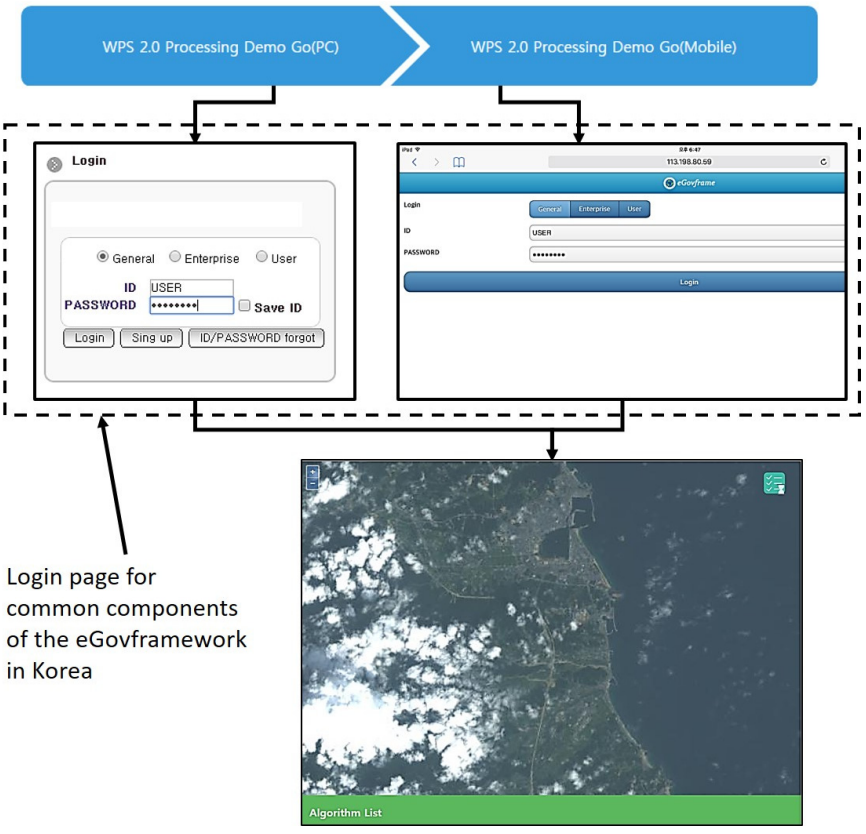

Figure 3. Automatic loading of additional common components by user authentication.

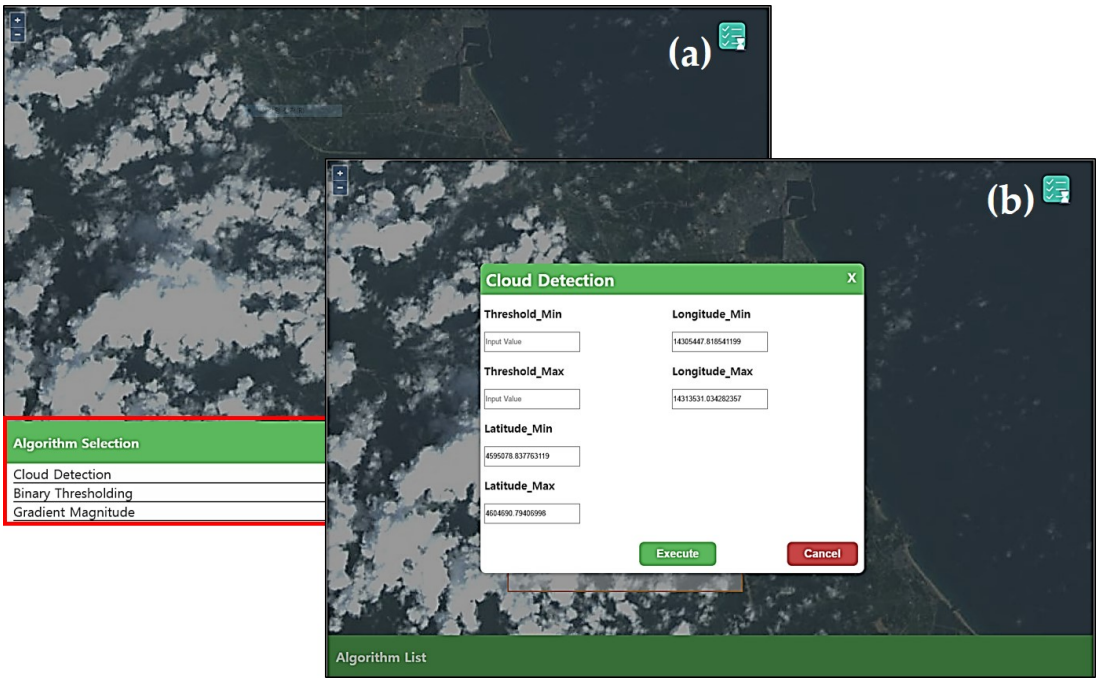

Figure 4. Configuration of the user interface on the client side: (**a**) select algorithm—GetCapabilities request; (**b**) algorithm modal view—DescribeProcess request.

Figure 5 shows the progress status and results of the satellite image-processing function. Click event of the Execute button in the satellite image-processing modal view initiates an Execute request. Then, the WPS server receives JobID values and implements satellite image-processing based on input values in the modal view. The received JobID values are used to receive progress status and results of the satellite image processing function. Figure 5a shows the current progress status of the satellite image-processing function. When the progress status information on the Web system is requested, it is retrieved and returned through the GetStatus service. To show processing status continuously, progress status information is requested periodically until the satellite image-processing function is completed. Upon completion of processing, it is indicated as complete, and the processing result information is requested. In the Web system, the GetResult service is carried out to fetch a processing result. The processing result is a layer name registered on GeoServer. Using the GeoServer layer name, the client brings the processing result, visualizes it on the client, and shows it as in Figure 5b. Figure 5c indicates the satellite image multiprocessing stages and processing result. The screen image on the left, which displays the satellite image-processing progress status, shows that two processing functions have been activated. The satellite image-processing functions under activated states are Cloud Detection [39] and Gradient Magnitude functions [40]. In the UI showing processing progress status, the progress status on top indicates the processing function executed first. The UI shows that the Cloud Detection function is still running, but the Gradient Magnitude function is completed, displaying the processing result on the client. The image on the right indicates that the Cloud Detection function is completed and shows the processing result. This proves that support for asynchronous processing enables implementation of such a multiprocessing function. This result also shows that another function can be implemented without having to wait for the completion of the previously executed processing function.

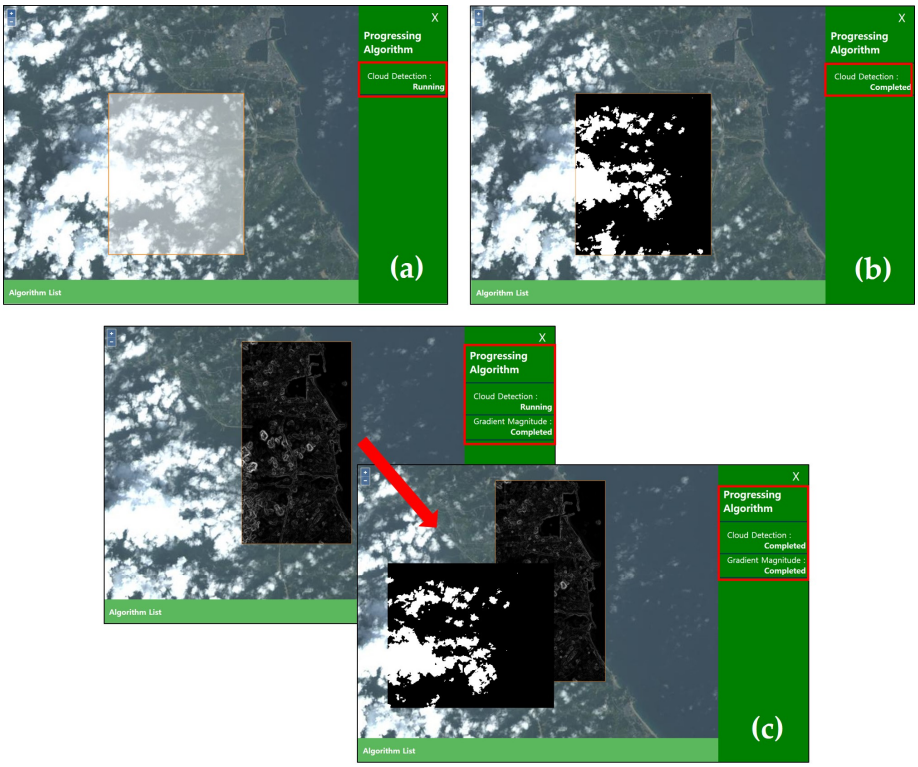

Figure 5. Application of the progressing algorithm: (**a**) running process—GetStatus request; (**b**) completed process and result—GetResult request; (**c**) multiprocessing and results.

6. Discussion

In comparison with WPS 1.0, improved features in the standard interface WPS 2.0 were observed when the processing system using WPS 2.0 was implemented. The ZOO platform offers WPS communication through CGI. When a service is provided using CGI, the server may become vulnerable due to excessive loads caused by multiple connections. Because this can cause real service trouble, other communication methods should be presented. Existing open sources, excluding the ZOO platform, comply with WPS 1.0, while technical implementation for WPS 2.0 is underway or is not yet planned. In addition, WPS 2.0 research is trailing far behind those on WPS 1.0; performance studies on WPS 2.0 are required. Information and communication sectors provide a variety of technologies and platforms. Understanding current trends and measuring performance in various infrastructures can maximize utilization.

The link to the user authentication function with WPS 2.0 is one of the common component functions provided by the eGovframework-based system. Therefore, functions required for a web system can be provided without the need for independent development, enabling a highly scalable structure to be built. Also, this structure enables system development, testing, and management through various components offered by the eGovframework. Because the eGovframework is provided through a virtual machine, which is a feature of Java, basic hardware specifications should be good enough to implement services. It is convenient that the standard framework common components can be added immediately according to the specific functions required. However, some unwanted components can be added when common components are added due to their interdependency, which affects system quality and maintenance. The organization behind the eGovframework provides a lightweight framework; therefore, a light version could be considered to solve this issue. When the eGovframework is utilized, areas of improvement need to be verified through performance and code quality tests.

7. Conclusions

Information and communication technologies will continue to progress, promising users more convenient services in various sectors. However, more time needs to be spent on management as systems grow more complicated. The same applies to systems in the geospatial sectors, and solutions should be sought to provide and manage geospatial services effectively. For this purpose, in this study, a trial system was built linking WPS 2.0, an international standard related to geospatial processing, to the eGovframework developed in South Korea. The objective was to suggest plans to provide geospatial services effectively. The trial system, a web system capable of online satellite image processing, used an open source method. Because WPS 2.0 was applied in building the trial system, the system has a structure capable of implementing consistent interfaces and sharing functions with other systems when providing geospatial processing functions on the Web. In addition, with asynchronous processing capability, flexible processing is possible in terms of getting functions. The ability to use and comply with international standards when providing geospatial processing on a Web system has been confirmed by using the ZOO-Project platform in order to comply with WPS 2.0. This system has a structure that can be modified to link open sources once WPS-related open sources support WPS 2.0 in the future. At present, many geospatial service systems are available on the Web, but this trial system is the first with WPS 2.0 and the eGovframework based on no-cost full open sources. The geo-based image operation in this system is an example demonstrating linkage of applied technologies. Other functionalities or algorithms for further geodata processing can be added to this design and architecture. This is an implementation case for Korea. However, this case can be a useful example for other countries that already have the e-Government framework or plan to establish it, because WPS and e-Government framework are crucial elements if geo-based service systems with geodata-processing functions are to be operated in the distributed environments and in the public sectors.

Acknowledgments: This research was supported by the National Land Space Information Research Program from the Ministry of Land, Infrastructure and Transport, Korea (No. 14NSIP-B080144-01).

Author Contributions: Kiwon Lee conceptualized the research objectives, drafted the manuscript and provided revisions. Gooseon Yoon and Kwangseob Kim, under Kiwon Lee's supervision, performed data processing.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Aye, Z.C.; Sprague, T.; Cortes, V.J.; Prenger-Berninghoff, K.; Jaboyedoff, M.; Derron, M.-H. A collaborative (web-GIS) framework based on empirical data collected from three case studies in Europe for risk management of hydro-meteorological hazards. *Int. J. Disaster Risk Reduct.* **2016**, *15*, 10–23. [CrossRef]
2. Shekhar, S.; Feiner, S.K.; Aref, W.G. Knowing where you are in space and time promises a deeper understanding of neighbor ecosystems and the environment. *Commun. ACM* **2016**, *59*, 72–81. [CrossRef]
3. ISO/TC 211 Geographic Information/Geomatics. Available online: http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees/iso_technical_committee.htm?commid=54904 (accessed on 30 September 2016).
4. Open Geospatial Consortium. Available online: http://www.opengeospatial.org/ (accessed on 30 September 2016).
5. Lee, K.; Kang, H-K. ISO and OGC Standards for Geo-Spatial Image Information and Suggestions for Their Applications. *Korean J. Remote Sens.* **2010**, *26*, 451–464.
6. Klug, H.; Kmoch, A. A SMART groundwater portal: An OGC web services orchestration framework for hydrology to improve data access and visualisation in New Zealand. *Comput. Geosci.* **2014**, *69*, 78–86. [CrossRef]
7. Schut, P. *OpenGIS Web Processing Service*; Open Geospatial Consortium Inc.: Wayland, MA, USA, 2007; p. 88.
8. E-Government, GIS, and Good Governance. Available online: http://transformgov.org/en/Article/105412/ (accessed on 22 November 2016).
9. Tsai, N.; Choi, B.; Perry, M. Improving the process of E-Government initiative: An in-depth case study of web-based GIS implementation. *Gov. Inf. Q.* **2009**, *26*, 368–376. [CrossRef]

10. Cordella, A.; Tempini, N. E-government and organizational change: Reappraising the role of ICT and bureaucracy in public service delivery. *Gov. Inf. Q.* **2015**, *32*, 279–286. [CrossRef]
11. Spring. Available online: https://projects.spring.io/spring-framework/ (accessed on 10 October 2016).
12. ZOO-Project. Available online: http://www.zoo-project.org (accessed on 2 March 2016).
13. Lopez-Pellicer, F.J.; Béjar, R.; Florczyk, A.J.; Muro-Medrano, P.R.; Zarazaga-Soria, F.J. A Review of the Implementation of OGC Web Services across Europe. *Int. J. Spat. Data Infrastruct. Res.* **2011**, *6*, 168–186.
14. Li, W.; Wang, S.; Bhatia, V. PolarHub: A large-scale web crawling engine for OGC service discovery in cyberinfrastructure. *Comput. Environ. Urban* **2016**, *59*, 195–207. [CrossRef]
15. Han, W.; Di, L.; Yu, G.; Shao, Y.; Kang, L. Investigating metrics of geospatial web services: The case of a CEOS federated catalog service for earth observation data. *Comput. Geosci.* **2016**, *92*, 1–8. [CrossRef]
16. Zhao, P.; Foerster, T.; Yue, P. The Geoprocessing Web. *Comput. Geosci.* **2012**, *47*, 3–12. [CrossRef]
17. Mueller, M.; Pross, B. *OGC WPS 2.0 Interface Standard*; Open Geospatial Consortium Inc.: Wayland, MA, USA, 2015.
18. Swain, N.R.; Latu, K.; Christensen, S.D.; Jones, N.L.; Nelson, E.J.; Ames, D.P.; Williams, G.P. A review of open source software solutions for developing water resources web applications. *Environ. Model. Softw.* **2015**, *67*, 108–117. [CrossRef]
19. The Computational Geometry Algorithms Library. Available online: https://www.cgal.org/ (accessed on 21 December 2016).
20. Geographic Resources Analysis Support System (GRASS) Geographic Information System (GIS)—Home. Available online: https://grass.osgeo.org/ (accessed on 21 December 2016).
21. System for Automated Geoscientific Analyses (SAGA) Geographic Information System (GIS). Available online: http://www.saga-gis.org/ (accessed on 21 December 2016).
22. eGovFrame Portal. Available online: http://www.egovframe.go.kr (accessed on 30 September 2016).
23. Yue, P.; Zhang, M.; Tan, Z. A geoprocessing workflow system for environmental monitoring and integrated modelling. *Environ. Model. Softw.* **2015**, *69*, 128–140. [CrossRef]
24. Yue, S.; Chen, M.; Wen, Y.; Lu, G. Service-oriented model-encapsulation strategy for sharing and integrating heterogeneous geo-analysis models in an open web environment. *ISPRS J. Photogramm.* **2016**, *114*, 258–273. [CrossRef]
25. Castronova, A.; Goodall, J.L.; Elag, M.M. Models as web services using the Open Geospatial Consortium (OGC) Web Processing Service(WPS) standard. *Environ. Model. Softw.* **2013**, *41*, 72–83. [CrossRef]
26. Xavier, E.M.A.; Ariza-Lopez, F.J.; Urena-Camara, M.A. Web service for positional quality assessment: The WPS TIER. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2015**. [CrossRef]
27. 52° North Initiative for Geospatial Open Source Software GmbH—Home. Available online: http://52north.org/ (accessed on 21 December 2016).
28. PyWPS Home. Available online: http://pywps.org/ (accessed on 21 December 2016).
29. Pebesma, E.; Cornford, D.; Dubois, G.; Heuvelink, G.B.M.; Hristopulos, D.; Pilz, J.; Stohlker, U.; Morin, G.; Skoien, J.O. INTAMAP: The design and implementation of an interoperable automated interpolation web service. *Comput. Geosci.* **2011**, *37*, 343–352. [CrossRef]
30. Rautenbach, V.; Coetzee, S.; Iwaniak, A. Orchestrating OGC web services to produce thematic maps in a spatial information infrastructure. *Comput. Environ. Urban* **2013**, *37*, 107–120. [CrossRef]
31. Giuliani, G.; Nativi, S.; Lehmann, A.; Ray, N. WPS mediation: An approach to process geospatial data on different computing backends. *Comput. Geosci.* **2012**, *47*, 20–33. [CrossRef]
32. Yoon, G.; Lee, K. WPS-based satellite image processing on web framework and cloud computing environment. *Korean J. Remote Sens.* **2015**, *31*, 561–570. [CrossRef]
33. Yoon, G.; Lee, K. Testing application of Web Processing Service (WPS) standard to satellite image processing. *Korean J. Remote Sens.* **2015**, *31*, 245–254. [CrossRef]
34. Kim, K.; Lee, K. Visualization of Geo-spatial data and public data using mobile operating environment in the eGovernment standard framework. *J. Korea Spat. Inf. Soc.* **2015**, *23*, 9–17. [CrossRef]
35. GeoServer. Available online: http://geoserver.org/ (accessed on 21 December 2016).
36. jQuery. Available online: http://jquery.com/ (accessed on 21 December 2016).
37. OpenLayers 3—Welcome. Available online: https://openlayers.org (accessed on 21 December 2016).
38. Eclipse—The Eclipse Foundation Open Source Community Website. Available online: https://eclipse.org/ (accessed on 21 December 2016).

39. Feature Extraction in OTB. Available online: https://www.orfeo-toolbox.org/SoftwareGuide/SoftwareGuidech14.html#x43-23800014.8 (accessed on 21 December 2016).
40. Basic Filtering in OTB. Available online: https://www.orfeo-toolbox.org/SoftwareGuide/SoftwareGuidech8.html (accessed on 21 December 2016).