WPS-BASED TECHNOLOGY FOR CLIENT-SIDE REMOTE SENSING DATA PROCESSING

E. Kazakov, A. Terekhov, E. Kapralov, E. Panidi *

Saint-Petersburg State University, Institute of Earth Sciences, Department of Cartography and Geoinformatics,
33-35 10-th line V.O., 199178 Saint-Petersburg, Russia – panidi@yandex.ru

KEY WORDS: Remote Sensing Data Web Processing, Web Processing Services, OGC WPS, Client-Side Web Geoprocessing, Hybrid Geoprocessing Web Services

ABSTRACT:

Server-side processing is principal for most of the current Web-based geospatial data processing tools. However, in some cases the client-side geoprocessing may be more convenient and acceptable. This study is dedicated to the development of methodology and techniques of Web services elaboration, which allow the client-side geoprocessing also. The practical objectives of the research are focused on the remote sensing data processing, which are one of the most resource-intensive data types.
The idea underlying the study is to propose such geoprocessing Web service schema that will be compatible with the current server-oriented Open Geospatial Consortium standard (OGC WPS standard), and additionally will allow to run the processing on the client, transmitting processing tool (executable code) over the network instead of the data. At the same time, the unity of executable code must be preserved, and the transmitted code should be the same to that is used for server-side processing. This unity should provide unconditional identity of the processing results that performed using of any schema. The appropriate services are pointed by the authors as a Hybrid Geoprocessing Web Services (HGWSs).
The common approaches to architecture and structure of the HGWSs are proposed at the current stage as like as a number of service prototypes. For the testing of selected approaches, the geoportal prototype was implemented, which provides access to created HGWS. Further works are conducted on the formalization of platform independent HGWSs implementation techniques, and on the approaches to conceptualization of theirs safe use and chaining possibilities.
The proposed schema of HGWSs implementation could become one of the possible solutions for the distributed systems, assuming that the processing servers could play the role of the clients connecting to the service supply server.
The study was partially supported by Russian Foundation for Basic Research (RFBR), research project No. 13-05-12079 ofi_m.

1. PROJECT MOTIVATION

Web-based geospatial data representation and data manipulation technologies got extensive development and rapid growth in recent years. The improvement of the network infrastructure and hardware facilities entails enhancement of transfer and store abilities when operating with large geospatial datasets, which are especially important when using remote sensing data. Additionally, the improvements of Web software technologies become the basis for the implementation of Web-based systems, which able not only to transfer and display the geospatial data through the Web, but also to provide tools for geoprocessing and spatial analysis using Web interface, directly in the Web browser. The Web 2.0 paradigm (Governor, 2009), the concept of SaaS, i.e. Software as a Service, (Software & Information Industry Association, 2001) and the HTML5[1] standard should be mentioned as examples of such improvements. The development of interactive data manipulation techniques is one of the current trends in geospatial Web.
An important phase in the development of technologies for processing geospatial data in the Web was the publication of the Open Geospatial Consortium Web Processing Service (OGC WPS) standard in 2007 (Schut ed., 2007). This international standard formalized the Web implementation rules for the services of geospatial data processing. The standard describes the order of request-response interaction between server and client computers, when accessing geospatial data processing services via Web.
However, a significant feature of the open source WPS standard and of the proprietary ecosystems (e.g. ESRI ArcGIS Online[2]), which allow Web publishing of the geoprocessing tools, is their focusing on the server-side data processing. The software tools that provide decentralized data processing, on the client side or using grid computations, are developed only within the sporadic research projects (Coene at al., 2007) or referred by some authors as a promising concept (Keens ed., 2007).
Nevertheless, in some situations, the use of server-side computations for geospatial data processing may be unjustified or inconvenient. For example, when the user handles the data, which have distribution restrictions, and cannot be transmitted to a third-party server. In this and in some other situations, a possible solution is to transmit the executable code from the server to the client instead of transmitting the raw data to the server for processing (Panidi, 2013). In this case, the problem is reduced to providing of executable code portability, i.e. the software components suitability for use on the client side without installation. Additionally, the facilities for execution and running of the code on the client should be provided.
Moreover, to provide maximum flexibility of the geoprocessing schema and maximum comfort for user, it is reasonable to provide the choice possibility for running processing on the server or on the client side. At the same time, of course, the unity of the executable code should be ensured. It means the possibility

* Corresponding author.
[1] http://www.w3.org/TR/html5/
[2] http://www.arcgis.com

of use the same computational core on the server and on the client, to provide identical processing results, either of the client-side or server-side computations.

In general, the industrial scale technologies for client-side computations are standardized and used widely. However, in the case of geospatial data processing, the main obstacle to the development of the software tools, which can be requested via the Web and can be executed optionally on the server or client side, is the absence of common approaches and standards. This finding became the starting point of our project.

2. PROJECT SCOPE AND TOOLS

Currently the OGC WPS is the only international standard that regulates the implementation rules for geoprocessing Web services. Due to this fact, we decided to use it as a basis of our elaborations. Thus, the main objective of the project was the development of the WPS standard extensions, which would provide not only geoprocessing on the server side, but also transmitting executable code and running it on the client. We suppose that this novation will not violate current conventional approaches to the Web geoservices implementation. Moreover, it will provide a synergistic effect by adding more flexible implementation schema.

Current WPS standard assumes three types of client requests to be executed one by one:

1. getCapabilities request
2. describeProcess request
3. execute request

The first one allows the client to request and receive an XML file containing the metadata of the WPS compatible service that published on the server. Metadata file contains the names and general descriptions of the separate processes available as part of this service.

The describeProcess request allows to get another XML file with a detailed description of one or more processes. This file contains number and types of the inputs and outputs for each process.

Finally, the execute request allows to transmit an XML file to the server, with the input values required for the process execution. If necessary, it includes links to the local files to be uploaded.

As a result of the execute request the server returns another XML file that contains calculation results or links to output files.

Our idea is to introduce a fourth type of request named getProcess request (Panidi, 2014). This request must be generated using the results of getCapabilities and describeProcess requests. As a response on this request, the server should return to the client a detailed description and download links to the software components that are required for selected process or processes execution on the client side (Figure 1).

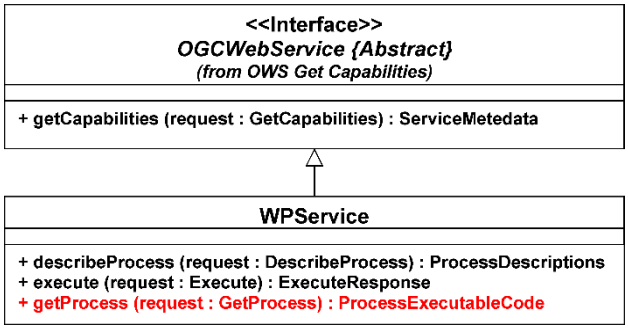

Figure 1. Modified WPS Interface Diagram with getProcess method added

The described approach will allow to implement geoprocessing Web services suitable for use with the existing WPS clients, as well as newly developed, which will be capable to support the loading of executable geoprocessing code and to run it on the client side. Services with a similar double sided architecture we offer to name as Hybrid Geoprocessing Web Services (Panidi, 2013).

3. FUNCTIONALITY SCHEMA

Due to the fact that the usual client-server paradigm, which the current WPS standard version assumes, will be added with a new method, there is a need to revise some data structure organization on the server side.

A basic description of each process must be supplemented with two entities:

1. A flag indicating availability of the client side version of the process for a particular software platform
2. List of modules and applications required to be downloaded on the client for local execution

In addition, to describe the software components that can be downloaded, we propose two new notions, which are functional modules (fModules) and extra applications (extraApps).

The term functional modules denotes the separate files that playing the role of containers for processing algorithms. The code presented in these files explicitly in the form of some functions, program classes or scripts. These can be Python[3] scripts, DLLs (Dynamic Link Libraries), Java[4] bitecode files etc., depending on the client operating system and runtime environment which are used for the process execution.

Extra applications are active software components with a particular functionality that can be executed directly. Any compiled programs for geospatial data processing, that are able to run in the client operating system after downloading, can be used as extra application, including programs that require runtime engine (e.g. Python and Java programs). Good examples of such programs are open source SAGA[5] GIS, GRASS[6] GIS, and others.

Transmitting of extraApps to the client computer should be conducted in a minimum configuration (core only) and in portable-assembly. Software components, which are not included in the program core, should be transmitted as fModules.

Therefore, every process designed for execution on the client side, could be composed of an unlimited number of functional modules, which provide particular processing algorithms. Each module must have a unique name and should be organized into a file of specific type. The module should be supplemented with description, table of compatibility with different operating systems, and list of additional program resources, which are used in this module (for example, it can refer to io_gdal module that executed through the SAGA command line, which is used as extraApp to provide this execution).

As the software components are obtained from the server, the mirror program structure is formed on the client. However, this structure is filled out with the components of the loaded processes and their dependencies only (Figure 2).

To organize software components downloading and geoprocesses running on the client computer, the special software client should be deployed in the client operating system. This software client plays the roles of the user interface and runtime environment for client side geoprocessing. As in the case of current WPS standard, the software client can be implemented in any form of Web browser interface, and in the form of a desktop graphical user interface (standalone or as a part of the desktop

[3] http://www.python.org
[4] http://www.java.com
[5] http://www.saga-gis.org
[6] http://grass.osgeo.org

geoinformation system, i.e. GIS). This software client is direct provider of the HGWSs in the client environment. However, the implementation of this runtime environment in the form of browser-based interface can be difficult, due to the built-in browser security, which can block execution of the desktop applications from a browser.

In some cases the HGWS runtime environment must be complemented by external runtime environment for the code execution, for example it may be bytecode interpreter.

The objectives of the HGWS runtime environment include the generation of a graphical user interface and execution of the process in the case of absence of the process extraApps, which are suitable for immediate launch.

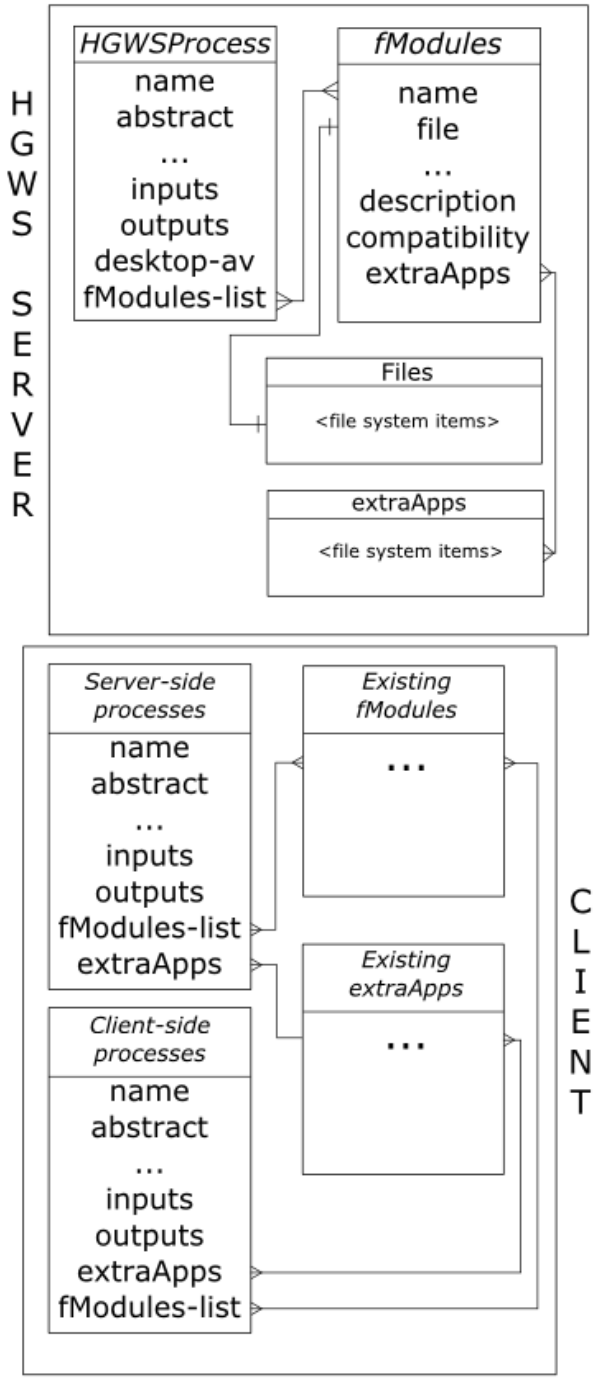

Figure 2. ER-map of process components located on server and client

4. PROGRAM INTERFACE IMPLEMENTATION

The general schema of the interaction between the client runtime environment (RE) and the WPS/HGWS server is a cascade type and includes four basic steps.

Step one (Figure 3), assumes that the user choose a server and the client RE establishes connection to it. Client generates standard request getCapabilities, and receives the response, which is extended with the desktop-available attribute. For each of the processes this attribute indicates downloading availability of the software components for client-side geoprocessing. Client RE analyzes the response and gets a list of processes that are available for client-side execution.

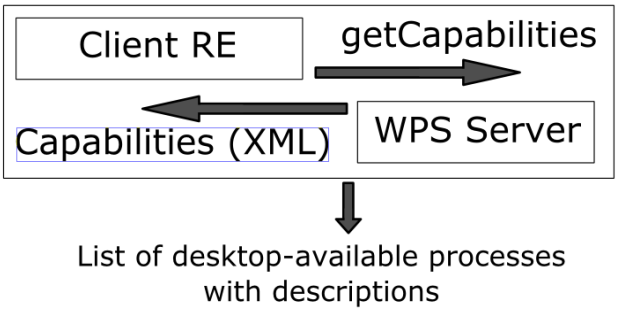

Figure 3. First step of data exchange – requesting capabilities and generating list of available processes

At the step two, the user choose any of the available processes to execute. After that, the client RE automatically executes a describeProcess request, and generates a graphical user interface (UI) for selected process, in the case of desktop execution. The parameters for UI generation are extracted from describeProcess response, using its list of process inputs and outputs.

Additionally, the base interface for the process is generated, which is the program function that enables linking of the graphical UI and the functional module(s) or extra application(s) of selected process. At the time of the process execution, the base interface redirects all the process inputs to the corresponding functional module or extra application (Figure 4).

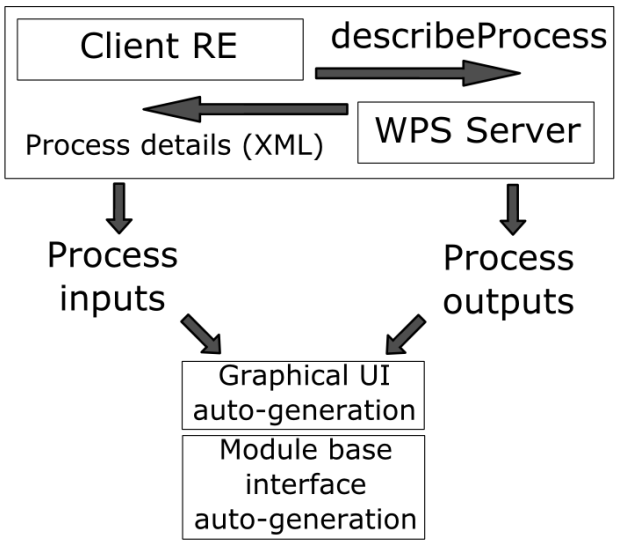

Figure 4. Second step of data exchange. Requesting process details and generating UI and base interface of functional module

At the step three, the application executes a new request to the server, which is getProcess. The request involves getting a list of all required to be downloaded and executed software components of the process, namely the functional modules and extra applications (Figure 5). The structure of the request includes a number of compatibility settings that characterize the client computer (Table 1).

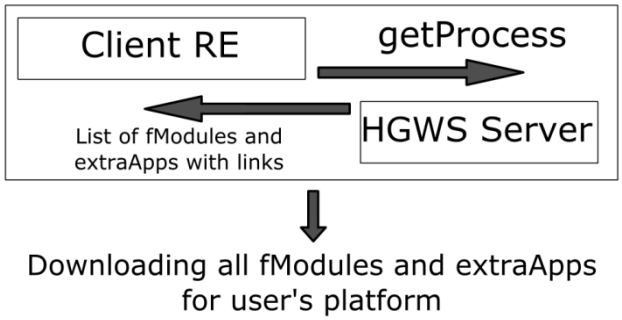

Figure 5. Third step of data exchange. Requesting process executable code and downloading program components with dependencies

Name	Presence	Description
Request	Required	Identifies service request. Must be "getProcess"
Service	Required	Identifies service type. Must be "HGWS"
Version	Required	Identifies service version
Identifier	Mandatory	Process identifier as listed in the Capabilities document
Platform	Required	Identifies user's desktop platform. Could be "Win32", "Win64", "Mac", "Linux Debian" etc.

Table 1. getProcess request parameters

The getProcess request schema example is as follows:

http://some.host/server?
Request=getProcess&
Service=HGWS&
Version=x.0.0&
Platform=Win32
Identifier=index-NDVI

As a result, the HGWS server returns the XML-document containing the description of all the software components that are needed to use the process on the client side. This document includes unlimited number of descriptions of the functional modules and extra applications, and all of the components may refer to each other. Each software component is supplied with the description and download link. This link can be static or temporarily generated for downloading at the time.
The schema of this document should be as simple as it is possible. The example is as follows:

```
<schema xmlns="..." xmlns:ows="..." xmlns:wps="..." targetNamespace="..." version="..." xml:lang="...">

<annotation>...</annotation>

<import namespace="..."/>

<element name="ProcessExecutableModules">...</element>

<!—Platform parameters-->
<complexType name="Platform">...</complexType>

<!—Modules parameters-->
<complexType name="fModules">
 <complexContent>
  <name>...</name>
  <annotation>...</annotation>
  <compability>...</compability>
  <type>...</type>
  <usedExtraApps>
   <complexContent>
    <name>...</name>
    <annotation>...</annotation>
    <compability>...</compability>
    <type>...</type>
    <downloadLink>...</downloadLink>
   </complexContent>
  </usedExtraApps>
  <downloadLink>...</downloadLink>
 </complexContent>
</complexType>
</schema>
```

Finally, at the step four, the client RE downloads the functional modules and extra applications that were not previously downloaded. The downloaded files are stored on the client so that the software components can access each other if necessary.
As a result, the client forms the software infrastructure that is ready to perform computations and data processing (Figure 6). This infrastructure is accessible through the client RE.

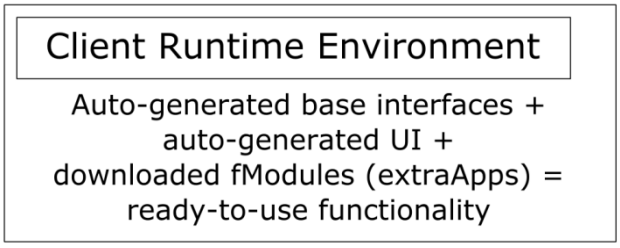

Figure 6. Processes on client side, in the case of desktop client

To implement geoprocessing tools in accordance with the HGWS concept and to test described principles, the server application (HGWS server) was developed. This software server works in conjunction with the Web server (the Microsoft IIS[7] was used) and the WPS server (the PyWPS[8] server was used). The HGWS server is responsible for handling the request getProcess and generation of links to functional modules and extra applications.

5. USER INTERFACE IMPLEMENTATION

As the described schema of the client-side services execution is based on the OGC WPS standard, it allows to access the server-side HGWS services directly using any existing client applications that implement WPS protocols. Also, some

[7] http://www.iis.net

[8] http://pywps.wald.intevation.org

functionality updates and implementation of getProcess request is needed for accessing the client side services.

As it was mentioned above, the client application could be a Web application, a desktop application, or a built-in desktop GIS module (i.e. QGIS[9] WPS client). For the proposed approaches validation and testing, we implemented our own Web client[10].

This elaboration is needed because of two main reasons, which are the unsuitability of existing software clients to full support of HGWS schema, and secondly, the need to have a full and clear access to the source code of the client.

Concerning the suitability for client-side geoprocessing, the client should be able to parse the desktop-available attribute in the getCapabilities response, and to provide some correct reactions when this attribute is presented. Firstly, the client must provide the processes execution in the corresponding client runtime environment, using getProcess response data. Additionally, the desktop client should provide graphical user interface generation, using getCapabilities response data.

Finally, since the server-side process execute response may include the download links to the processed files (raster, vector or text data) it leads to a number of potential inconveniences. When there is a need to have the availability of the processing results during some time, we should provide some tools for convenient storing of the links.

The main tasks of the elaborated HGWS Web client include parameterization and execution of selected processes on the server side, storage of the obtained geoprocessing results on the server and providing of the re-access to the stored data. The Web client also should allow to manage the data storage on the server. The storage could preserve as downloadable files of the processed data as well as the user raw data, which could be used as processing inputs. User access to the storage is also carried out via the Web client.

Thus, the user cannot lose the processing results links, when accidentally updating the page, for example. The processing results can be downloaded at any time and not only immediately after the processing. On the other hand, there is no need to download the result, and then upload it to the server as the inputs for another process. It is important when running a sequence of processes within a single technological chain.

Provision of the data store abilities requires the implementation of a registration and authorization system on the server. Furthermore, to ease personal data store, the user interface should provide some functionality for metadata representation (e.g. the number of channels of the raster, raster size, etc.).

Due to these needs, our HGWS Web client was supplemented with the user's personal account, which is responsible for the management of data, uploaded by authorized users (Figure 7).

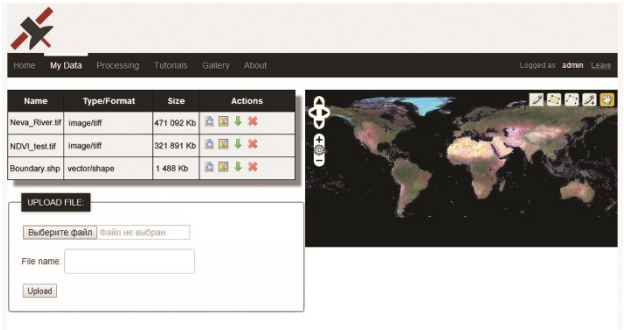

Figure 7. Web interface of the personal account

Web client includes the Web interface and the server components. The Web interface consists of several Web pages, which are responsible for the following tasks:

1. Registration data collecting when registering the new users
2. Entering of the user authorization data
3. Managing of the stored geospatial data on the server
4. Generating and visualizing of a list of available geoprocesses
5. Parameterizing of the process before execution
6. Sending raw data to the server (if necessary)
7. Getting the server responses and its interpretation

Web interface was implemented using the client-side JavaScript[11] Web scripting language and the jQuery[12] JavaScript library.

Server components of the Web client include a set of the server-side scripts developed using the PHP[13] Web scripting language. The server components performs:

1. Registration and authentication of the users
2. Validation and storage of the geospatial data, uploaded by the users
3. Generation of graphic files for each dataset, for preview in a browser
4. Putting the geoprocessing results into the user store

Currently, users can upload the raster datasets in the GeoTIFF format, vector datasets in archived Shape-files, and a simple text files. All the uploaded files are available for renaming, downloading and deleting.

For the raster datasets, the server generates two JPG images at uploading. The first one is the image in source data map projection for previewing directly in a browser, and the second is the image in personal account Web map projection for visualize as an overlay on the world map.

After the one of available geoprocesses at the data processing Web page was selected, the Web interface automatically generates a Web form for data input with the required number of input fields. The input type of each field corresponds to the process input parameter or input dataset, such as single-channel raster image for example.

Summarizing the description of the elaborated Web client, we may entitle it as the prototype of geoportal or geoportal interface designed to provide publication and access capabilities for HGWSs.

6. CURRENT RESULTS

To provide the client-side execution of the HGWS processes, the client (desktop) runtime environment application was developed that can interact with HGWS server. This application implemented using the Python programing language and the PyQt4[14] program library.

When the user selects in the Web interface one of the processes that is suitable for execution on the client side, and sends a command to execute it on the client side, the RE application is downloaded and executed on the client automatically.

The process input parameters specified by user in the Web interface are transmitted to the client RE interface.

Additionally the RE application can be started by the user manually, if the corresponding executable file was previously saved. In the case of direct execution of client RE, user may select a server to connect indicating two URLs. The first is the WPS server URL, and the second is the URL of the HGWS server. After that, the RE application executes getCapabilities request,

[9] http://www.qgis.org
[10] http://195.70.211.131
[11] http://en.wikibooks.org/wiki/JavaScript
[12] http://jquery.com
[13] http://php.net
[14] http://www.riverbankcomputing.co.uk/software/pyqt/intro

using PycURL[15] program library, and shows the list of geoprocesses that are marked as desktop_available in the getCapabilities response.
For example, when you select the process that converts Digital Number values into Reflectance values for Landsat 8 remote sensing imagery (which is implemented using the SAGA command line) these steps are performed as follows.
RE application executes the describeProcess request, and generates the graphical user interface using PyQt. Requested process needs the input parameters to be passed as a links to a single-channel raster in GeoTIFF format and metadata text file. Also, the channel item number should be pointed. Accordingly to these input requirements, the RE application generates three file input fields (i.e. group of Label, lineEdit and pushButton graphical interface elements) to select the input files and the output file name and saving path, as well as a field (i.e. group of Label and lineEdit) for entering channel item number as a text (Figure 8).

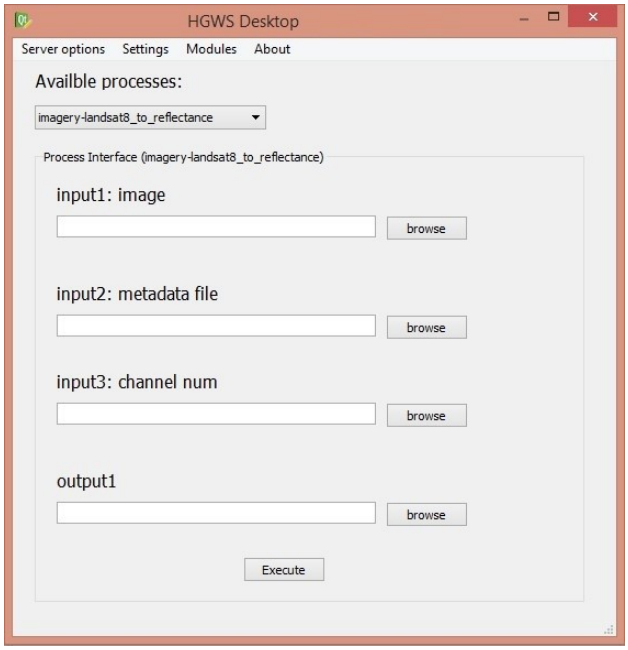

Figure 8. Graphical UI of the HGWS client RE for desktop

The Python program code fragment, which generates these graphical user interface components, is as follows:

```
lineEdit_3 = new QLineEdit(Dialog);
lineEdit_3->setObjectName(QString::fromUtf8("input3:
channel_num"));
lineEdit_3->setGeometry(QRect(30, 160, 211, 20));
pushButton_3 = new QPushButton(Dialog);
pushButton_3->setObjectName(QString::fromUtf8("browse"));
pushButton_3->setGeometry(QRect(260, 160, 75, 23));
```

Then the base interface is generated (the process input function), which connects graphical interface elements and software components that will be downloaded later. Base interface name is equal to the name of the executed process (in this case imagery_landsat8_to_reflectance). It stores the data specified in the user interface of the RE application and then transmits these data to the process as input parameters.
On the next step, the RE application executes a getProcess request:

```
http://host/server?
 Request= getProcess&
 Service=HGWS&
 Version=1.0.0&
 Platform=Win64
 Identifier=imagery-landsat8_to_reflectance
```

HGWS server processes the request and generates the XML response:

```
<complexType name="fModules">
 <complexContent>

<name>Imagery_Landsat8_to_reflecatnce_Fmodule</name>
   <annotation>Python module, launching SAGA CMD for
Landsat 8 DN to reflectance transform</annotation>
   <compability>Win64</compability>
   <type>PythonPlainText</type>
   <usedExtraApps>
    <complexContent>
     <name>SagaCMD</name>
     <annotation>OpenSource portable SAGA GIS command
environment</annotation>
     <compability>Win64</compability>
     <type>PortableCMDApp</type>

<downloadLink>host.domen/hjhfSFHjashsd</downloadLink>
    </complexContent>
   </usedExtraApps>

<downloadLink>host.domen/ljsdbnSmujhHJFjshjsf</download
Link>
 </complexContent>

 <complexContent>
  <name>SAGACMD_imagery_tools</name>
  ...
 </complexContent>

 <complexContent>
  <name>SAGACMD_io_gdal</name>
  ...
 </complexContent>

</complexType>
```

In this XML response the detailed descriptions are presented for all of the uploadable functional modules and extra applications, which are used by the selected process (these are the Python script containing the process main function, the SAGA libraries, e.g. io_gdal and imagery_tools, and all dependencies of these libraries).
All these modules refer to SagaCMD as the extra application, which is portable applications able to execute different SAGA modules in console mode. This application is transmitted as the archive containing the executable file and the required dependencies, without external libraries and modules. The main function in the Python script calls the SagaCMD, which calls all other libraries when need.
RE application saves all loaded modules and applications locally, so when recalling by other processes, this components will not be loaded again.
After downloading of process program components, the process can be executed independently of the HGWS server connection and Internet connection at all.

[15] http://pycurl.sourceforge.net

7. OUTLOOK

At the current stage of the project, we had formulated the HGWS concept and offered its possible implementation.
The implementation of the client software was elaborated, which combines elements of Web-based and desktop user interface. This software is convenient for HGWS processes execution on the server side and on the client side by the user's choice. A testing of the selected implementation is conducted on the example of the tools for satellite imagery processing.
Possibilities of the getProcess request implementation into the WPS standard, or development of a separate WPS-compatible standard, require further study and discussion.
Other promising areas of investigations are the research of more useful technical implementations of the HGWS conceptual approach and discussion over the more convenient HGWS XML schema.
An important issue is also a study of the possibilities of using various development tools for HGWS processes. Above all, the study of limitations for the safe running of the executable code transmitted to the client from the external source, and verification techniques of the source of executable files, as well as the other security issues.
Finally, some study required to develop approaches to atomization and chaining of the HGWS processes for the implementation of the complex algorithms and processing techniques.
Despite these issues, which require further study, the first step towards the development and implementation of HGWSs can be considered successful. The elaborated software and data exchange schemas indicate the principal possibility of implementing a relatively simple technique of geoprocessing Web services execution on the server or client side by the user's choice.

ACKNOWLEDGEMENTS

The study was partially supported by Russian Foundation for Basic Research (RFBR), research project No. 13-05-12079 ofi_m.

REFERENCES

Coene Y., Marchetti P.G., Smolders S., 2007. Architecture and Standards for a Distributed Digital Library of Geospatial Services. Third Italian Research Conference on Digital Library Systems (IRCDL 2007) Padova 29–30 January 2007 Proceedings pp. 52–60.

Governor J., Hinchcliffe D., Nickull D., 2009. *Web 2.0 Architectures: What entrepreneurs and information architects need to know*. O'Reilly, 276 p.

Kazakov E.E., 2013. Approaches to Implementation of Web Services for Pro-cessing and Analysis of Remote Sensing Data. In Proceedings of the Earth Remote Sensing From Space: Algorithms, Technology, Data - Young Scientists Workshops (Altai State University, Barnaul, Russia, October 2 - 6, 2013). pp. 82-86. Issued in Russian, accessible at http://elibrary.asu.ru/xmlui/bitstream/handle/asu/203/read.7book

Keens S. (ed.), 2007. Discussions, findings, and use of WPS in OWS-4. OGC Discussion Paper. OGC 06-182r1. Version 0.9.1, 2007-05-10.

Panidi E.A., 2013. Geoservices for Online Remote Sensing Data Processing. In Proceedings of the Earth Remote Sensing From Space: Algorithms, Technology, Data - Young Scientists Workshops (Altai State University, Barnaul, Russia, October 2 - 6, 2013). pp. 74-81. Issued in Russian, accessible at http://elibrary.asu.ru/xmlui/bitstream/handle/asu/203/read.7book

Panidi E.A., 2014. Towards Client-Side Web Processing Services. Proceedings. OSGeo's European Conference on Free and Open Source Software for Geospatial, Independent Innovation for INSPIRE, Big Data and Citizen Participation (FOSS4G-Europe 2014) July 15-17 2014 Jacobs University, Bremen, Germany. Issued online, accessible at http://europe.foss4g.org/2014/content/toward-client-side-web-processing-services.html

Schut P. (ed.), 2007. OpenGIS Web Processing Service. OpenGIS Standard. OGC 05-007r7. Version 1.0.0, 2007-06-08.

Software & Information Industry Association, 2001. Software As A Service: Strategic Backgrounder. Washington, D.C., USA.

Revised March 2015