An. Şt. Univ. Ovidius Constanţa Vol. 17(3), 2009, 131–148

A WEB SERVICES COMPOSITION WITH BPEL AND SATELLITE IMAGE PROCESSING FOR URBAN AND RIVER BED CHANGES ASSESSMENT

Cristina Gherghina

Abstract

The processing of satellite images can be a demanding task in terms of computer power. Most user workstations are common ones, with no special hardware or license for a commercial image processing tool. Web Services are applications that are made available from a business's Web server for remotely situated web clients. They can be a suitable solution to the satellite image processing domain, by deploying a rich set of functionality that runs on powerful servers to web clients that run on ordinary computers. In this paper, we introduce a Java Web Services composition for satellite image processing. Business Process Execution Language (BPEL) is used to orchestrate the Web Services composition. Among the image processing algorithms we mention enhance contrast, several filters for image correction and supervised image classification. All these are used to detect urban and river bed areas in two satellite images, taken in 1970 and 2000, and to highlight the urban area growth and the river bed changes over a period of 30 years. A client application was also developed, in order to show the composition functionality.

Key Words: BPEL, Web Services, composition, satellite image processing.
Mathematics Subject Classification: 68U10.
Received: April 2009
Accepted: October 2009
This work was supported by UEFISCSU, Romania, under Grant ID_262/2007

1 Introduction

Satellite images can be very useful in providing valuable information, but they cannot be processed on typical workstations, with no special hardware for image processing (a satellite image may include up to seven bands with about 15-40 MB per spectral band) and no license for a commercial image processing tool. In this context, a Web Service may be a solution.

Web services are a new breed of Web applications. They connect computers and devices with each other using the Internet to exchange data and combine data in new ways.

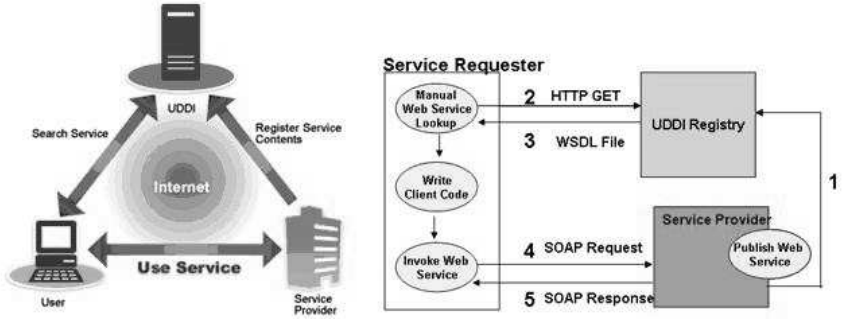

Figure 1: A simple Web Service invocation

A possible definition: Web Services are

- self-describing (services have a machine-readable description - WSDL (Web Service Description Language) file - that can be used to identify the interface of the service, and its location and access information),
- modular (service components are reusable and composable into larger components),
- implementation-independent (the service interface is available in a way that is independent of the ultimate implementation)
- interoperable (using SOAP (Simple Object Access Protocol), the services allow software running on disparate operating systems and in different environments to interoperate)

applications that can be published (service descriptions are made available in a repository - UDDI (Universal Description, Discovery and Integration) registry - where users can find the service and use the description to access the service), located, and invoked across the Web.

Web Services perform different kind of functions, which can be anything from simple requests to complicated business processes (e.g. currency conversion, weather reports or language translation). The key to Web Services is on-the-fly software creation through the use of loosely coupled, reusable software components: once a Web Service is deployed, other applications (and other Web Services) can discover and invoke the deployed service

Web Services might be a suitable solution for the digital processing and analysis of satellite images because the Web servers they run on may offer the required computational power to process satellite data in real time.

BPEL (Business Process Execution Language, also known as WS-BPEL or BPEL4WS) is the Web Services orchestration standard from OASIS [18]: it is an XML-based grammar for describing the logic to orchestrate the interaction between Web Services in a business process. BPEL has been well-received by many communities and applications, and as a result of this acceptance there are a number of evolving tools available for this technology. Acceptance of a common execution language allows for the portability of process definitions across more systems and hopefully easier acceptance at other data centers as the benefits are demonstrated.

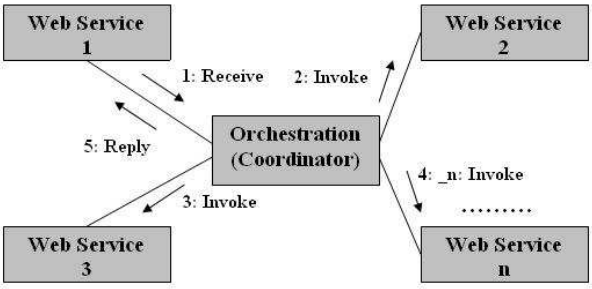

Figure 2: Orchestration: A central process takes control over the involved Web Services and coordinates the execution of different operations. The involved Web Services do not know that they are involved into a composition

A BPEL process describes how tasks are orchestrated, what component performs them, what their relative order is, how they are synchronized and how information flows to support the tasks. With BPEL, we can express conditional behavior, construct loops, declare variables, copy and assign values, define fault handlers, and so on. By combining all these constructs, we can define complex processes in an algorithmic manner. We can describe deterministic as well as non-deterministic flows and address advanced features such as support for long-running transactions, fault and event handling, compen-

sation, message correlation, etc. This standardized composition description is eventually deployed on a BPEL engine, required to process the instructions described in BPEL. Being able to expose a composition of a set of Web Services, makes the solution itself a Web Service and thus reusable.

In what follows, we describe a BPEL process, in which each phase is viewed as a Web Service. The Web Services provide various functions of image processing: contrast enhancement, several image filters, supervised image classification and a simple image comparison algorithm. The composition aims to highlight the urban area growth and the river bed changes over a period of 30 years.

The Web Services are programmed in Java, using Java Advanced Imaging (JAI)[3] for image processing and analysis. We choose Oracle BPEL Process Manager as our BPEL engine, and JDeveloper as the visual process composer.

A client application has been also created, in order to show the composition functionality.

2 Related Works

Currently, there are several image processing research projects and commercial products implemented as Web Services and as BPEL processes.

[4] and [5] are implementations of a Web Service-based image processing platform for providing medical image processing techniques remotely. Several medical image processing algorithms were implemented and deployed at a server site, at the client side the developer being able to invoke the remote processing methods to build an efficient medical image processing application.

[6] is a collection of Web Services that include a wide range of tools, including image file conversion (150+ file formats including JPEG2000, PDF, CMW, and DICOM) and image processing.

[7] provides the steps for creating a simple thumbnail service built on Amazon Web Services (AWS). This is a straightforward solution that would allow users to upload an image and then process that image with AWS.

The remote sensing information processing combined with Web Services is addressed in [8]. The paper describes in detail a flood remote sensing image creation service. Using semantic Web Services, a semantic service environment is created, the final result being a picture with different colors showing the water area of Poyang Lake in different periods of time.

The GeoBrain project [9], started in 2005, is a NASA research project that uses NASA EOS data and information through Web Service and knowledge management technologies for higher-education teaching and research. The system will allow users to dynamically and collaboratively develop interoperable, web-executable geospatial service modules and models, and run them

online against any part of the petabytes of archived data. One of the project products is BPELPower–Service Chaining Engine, in which WSDL-based web services can be deployed and executed.

[10] introduces the concept of Simple Services, and provide results of a prototype chaining distributed services to handle the processing of large data sets. The Simple Service architecture may contain services that are autonomous (can be used outside of any particular architecture) and can be used with data mining, image processing, image/map generation and other spatially oriented data applications. The prototype's main application is implemented using Java Server Pages for the user interface components. The individual image processing services are implemented as a distributed mix of Java servlets, Perl scripts and C programs with Perl wrappers to handle the HTTP communications, on both Windows and Linux systems.

Orchestra [11], a major European project, developed a service-oriented architecture for risk management based on open standards, together with a software infrastructure for enabling risk management services. Several Orchestra pilots were organized to provide practical tests and demonstrations of what could be accomplished using this architecture: forest fire and flash flood risk assessment in the Tordera basin, risk assessment for the road network in the French-Italian border region, environmental risk assessment of ship traffic in the German Bight, etc. One goal is to define workflows that combine several services into one value-added service chain that achieves a certain goal.

The Web-based geoprocessing, including chained Web services that operate on images, is addressed in OWS-5 [12], by Open Geospatial Consortium (OGC). The OWS-5 Geo-Processing Workflow bound together a couple of OGC Web Services standards for geospatial data sharing and processing in BPEL scripts to model data conflation. This successful prototype signals the utility of this approach for a host of geospatial tasks that require the combination of many different geospatial services and operations.

[13] introduces a distributed image processing system, each algorithm being implemented as a Web Service. Depending on user's requirements, the Web Services are dynamically linked using event condition action paradigm.

[14] describes an evolvable and composable framework for Web-Services-based medical image processing, with a three-tiered architecture. Its composability is achieved through the use of the BPEL, in its middle tier, for the choreographing of medical image processing Web Services.

[15] presents the use of Web Service compositions in providing a mechanism where remote users will be able to deploy custom data analysis solutions at data repositories. The project is based on the Algorithm Development and Mining (ADaM) toolkit to provide a rich set of analysis tools and the Earth Science Markup Language (ESML) to programmatically interpret heteroge-

neous data formats. ADaM components are implemented as Web Services using standard C++ for portability across platforms, and composed as a process with BPEL. The paper describes a demonstration Web Service workflow that uses the k-means clustering algorithm to create a cloud mask in satellite images.

3 Approach

3.1 Web Services For Image Processing

In what follows, we present the Web Services invoked by the BPEL process proposed here:

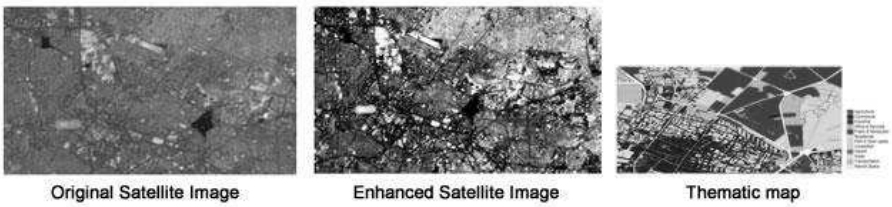

Figure 3: The image processing Web Services results

- Enhance Contrast (Fig.4): this service enhances image contrast by using histogram equalization. First, it creates a color histogram - a histogram of an image is a graph showing the number of pixels in an image at each different intensity value found in that image. Next, the histogram is equalized: the pixels are distributed evenly over the whole intensity range. Due to the discrete character of the intensity values, the histogram is not entirely flat. However, the values are much more evenly distributed than in the original histogram and the contrast in the image is essentially increased.

- Filters - several services for image correction were implemented:

  – ConvolveFilter (Fig.5): does a convolution which emphasizes edges in an image. Convolution is a spatial operation that computes each output sample by multiplying a kernel matrix with the corresponding matrix of source pixels surrounding a particular source pixel. The resulting product represents the destination pixel value.

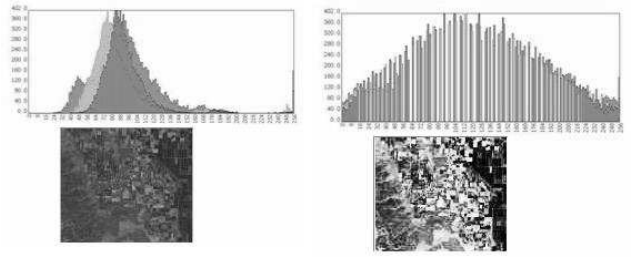

Figure 4: Left original image and its histogram; Right - equalized image and its histogram

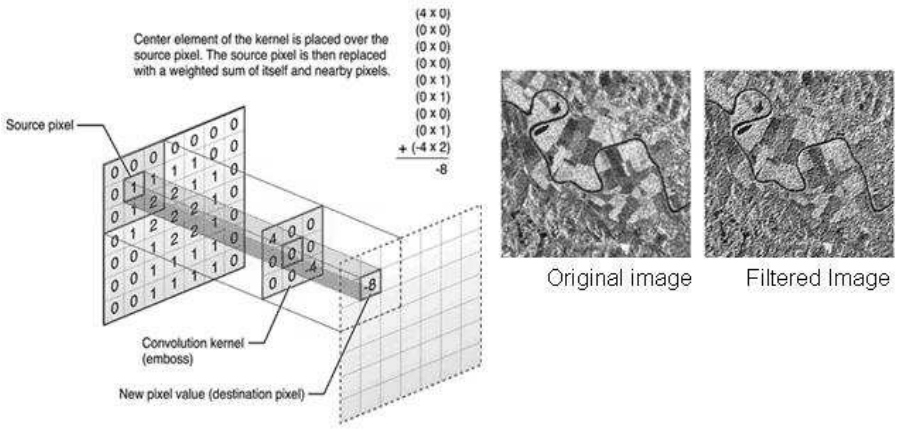

Figure 5: The convolution filter

- MedianFilter (Fig.6): filter for noise reduction. It replaces each pixel by the median of the input pixel and its eight neighbours. Each of the RGB channels is considered separately.
- ReduceNoiseFilter (Fig.7): reduces noise in an image. It compares each pixel with its eight neighbours and if the pixel is larger or smaller in value than all eight, replaces it by the largest or smallest of the neighbors. This is good for removing single noisy pixels from an image.
- SmartBlurFilter: performs a blur which only takes into account surrounding pixels which differ from the source pixels by more than a given threshold. This has the effect of blurring flat areas of an image while preserving sharp edges.

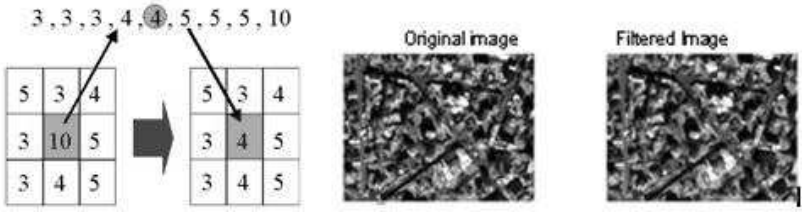

Figure 6: The median filter

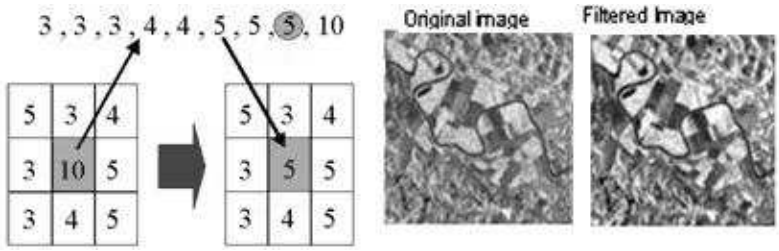

Figure 7: The reduce noise filter

- Image Classification (Fig.8): The supervised image classification method used is the parallelepiped algorithm or the "box decision rule classifier". The parallelepiped classifier uses intervals of pixels' values to determine whether a pixel belongs to a class or not. Classes are given by an image analyst, and represent an area of known identity delimited on the digital image, usually by specifying the corner points of a rectangular or polygonal area using the line and column numbers within the coordinate system of the digital image.

  First, some numerical descriptors (signatures) are calculated for each class, showing the minimum and maximum bounding coordinate for each class.
  Each pixel in the image is then compared numerically to signatures of each class, and labeled with the name of the class it looks most like.
  The final result is in the form of a thematic map (Fig.9) where each pixel has a fixed label of class assigned to it.
  More about the parallelepiped algorithm is to be found in [16].

- Image Comparison (Fig.10): this service implements a simple image comparison algorithm, which takes as input the filenames of the two images that are to be compared, the result being an image file which shows the

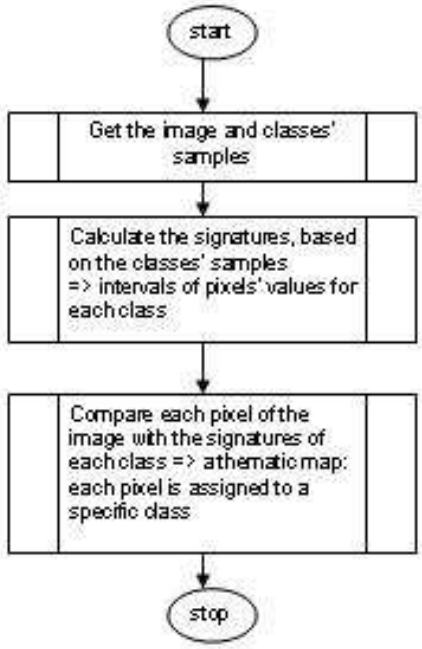

Figure 8: The parallelepiped algorithm

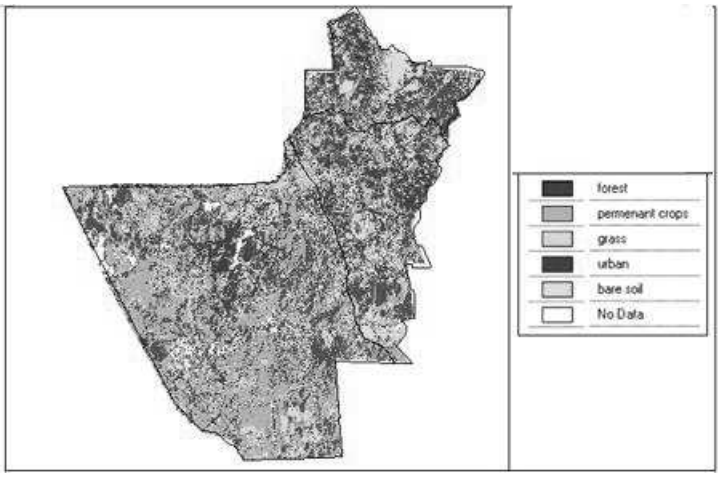

Figure 9: A thematic map

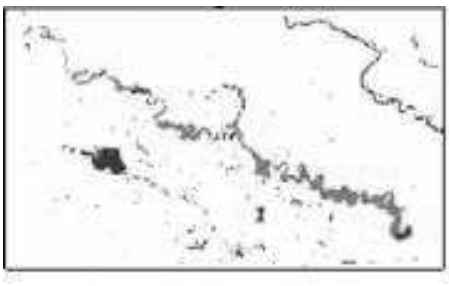

Figure 10: The image comparison method result

differences between them (changes are marked with specific colors).

All these Web Services will be called by the BPEL process in various phases of the composition, being implemented as BPEL partner links.

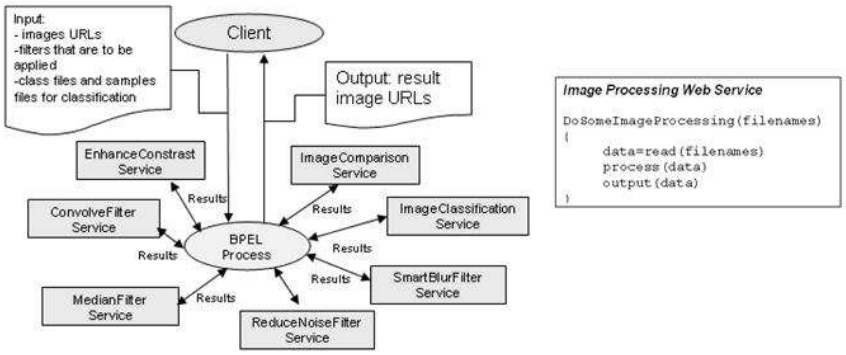

Figure 11: The architecture of the workflow considered

3.2 The BPEL Process

The BPEL process we developed meets the requirements of a user who wants to apply some complex image processing techniques to extract particular information from specific satellite images, the available computing power and storing resources not allowing him to process them locally. First, he uploads the satellite images on the remote server (the one which provides the BPEL process). Then, he invokes the BPEL process through a designated client ap-

plication. When the process ends, the user downloads the processed images on local computer, using the URLs returned by the process.

```
<message name="filterMessage">
     <part name="imageName1" type="xsd:string"/>
      <part name="imageName2" type="xsd:string"/>
    <part name="reduceNoiseFilter" type="xsd:string"/>
    <part name="medianFilter" type="xsd:string"/>
    <part name="convolveFilter" type="xsd:string"/>
    <part name="smartBlurFilter" type="xsd:string"/>
    <part name="enhanceContrast" type="xsd:string"/>
</message>
<message name="returnFilterMessage">
     <part name="resultImage1" type="xsd:string"/>
    <part name="resultImage2" type="xsd:string"/>
</message>

<portType name="processingPort">
    <operation name="initiateClassify">
        <input message="tns:classifyMessage"/>
        <output message="tns:returnClassifyMessage"/>
    </operation>
    <operation name="initiateFilter">
        <input message="tns:filterMessage"/>
        <output message="tns:returnFilterMessage"/>
    </operation>
</portType>
```

Figure 12: The WSDL description of the BPEL process

To develop a BPEL process, we must go through the following steps:

Step 1: Get familiar with partner Web Services (e.g. the ImageComparison Web Service, the ImageClassification Web Service)

Step 2: Define the WSDL for the BPEL process (Fig.12). Being a composition of Web Services, the BPEL process becomes itself a Web Service, so it must be exposed as a Web Service; therefore we have to write its WSDL file. Typically, a BPEL process waits for an incoming message from the client, which starts the execution of the business process.

Step 3: Define Partner Link Types (Fig.13): Partner link types represent the interaction between a BPEL process and the involved parties, which include the Web Services the BPEL process invokes and the client that invokes the BPEL process

Step 4: Develop the BPEL process (Fig.14):

- Define partner links

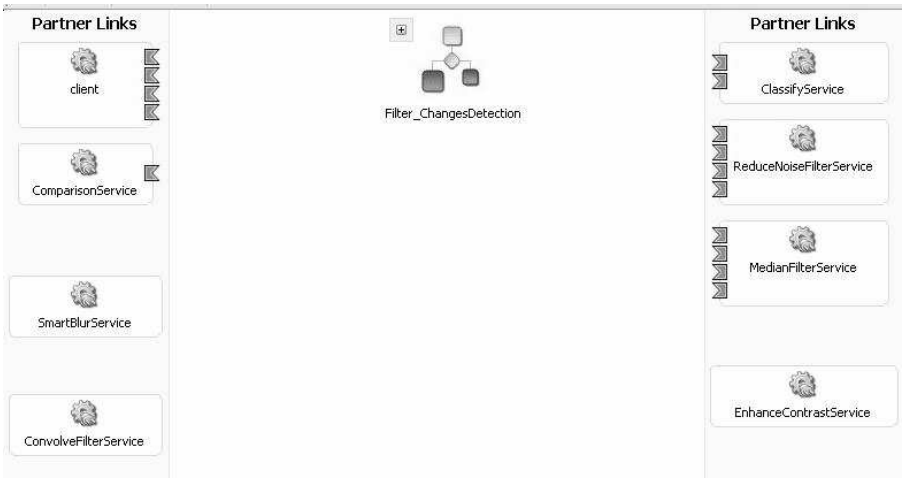

Figure 13: The BPEL process and partner links

```
Example:
  <process name="ImageProcessingService"... >
    <partnerLinks>
      <!-- The declaration of partner links -->
    </partnerLinks>

    <variables>
      <!-- The declaration of variables -->
    </variables>

    <sequence>
      <!-- The definition of the BPEL business
process main body -->
    </sequence>
  </process>
```

Figure 14: The basic structure of a BPEL process

- Declare variables
- Write the process logic definition

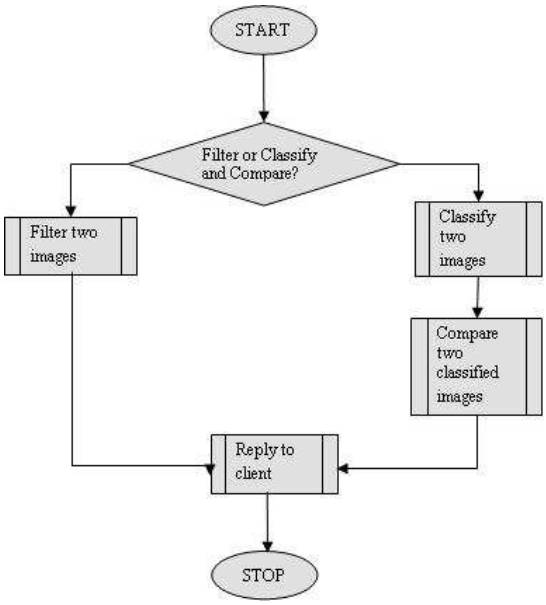

Figure 15: The logical flow of the BPEL process

An instance of the BPEL process is created by a BPEL Pick Activity, through which we can specify that the process should await the occurrence of one event in a set of events. We used message events handled with the $< onMessage >$ activity. For each event, we specified a set of activities that should be performed (Fig.15):

- Event: filter - a set of filters are applied on two images. The client specifies, through the BPEL process "initiateFilter" method arguments, the filters and the images URLs. Setting of the filters to be applied is made by a BPEL Switch Activity.
- Event:classifyCompare - the image classification and comparison algorithms are applied on two images, who's URLs are specified by the client through the BPEL process "initateClassifyCompare" method arguments.

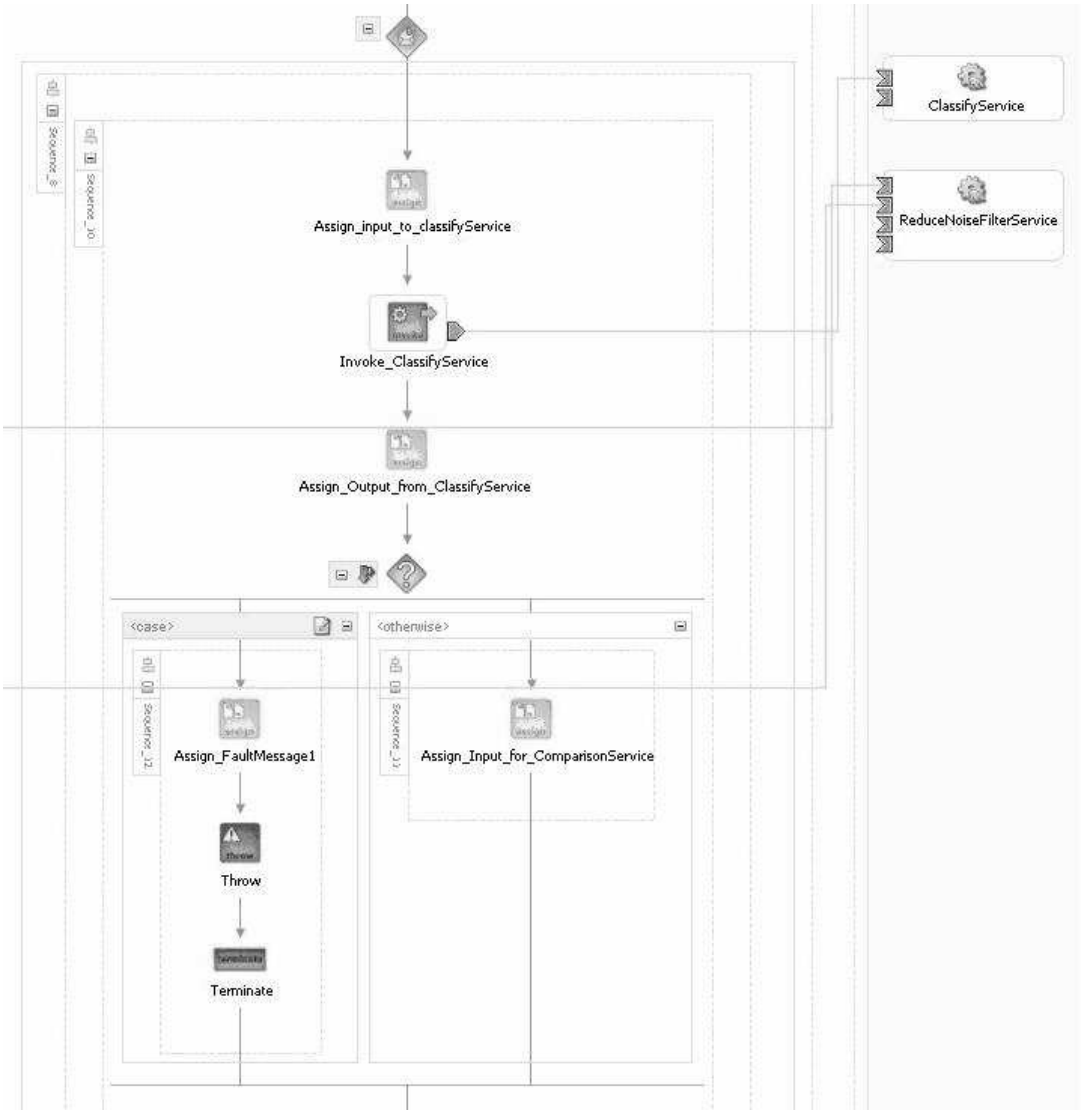

Figure 16: The logical flow of the Classification Web Service calling

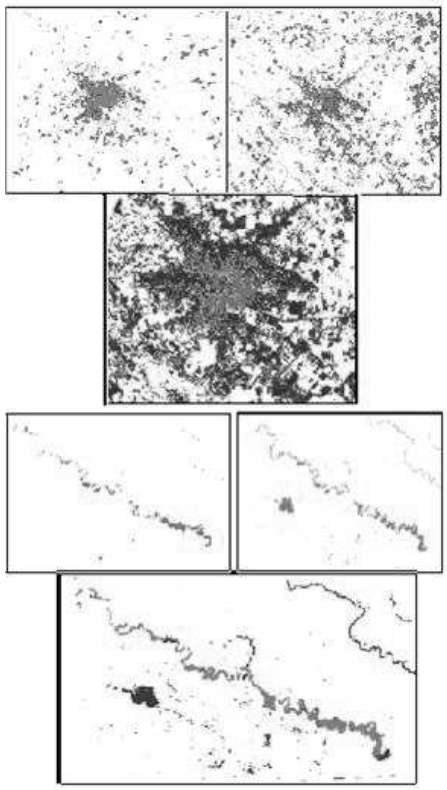

Figure 17: Classified images: a. Bucharest and surroundings 1970, 1 class: urban area; b. Bucharest and surroundings 2000, 1 class: urban area; c. Overlapped Fig. a and Fig. b: light grey - Bucharest 1970, dark grey - Bucharest 2000; d. Bucharest and surroundings, 1970, 1 class: water; e. Bucharest and surroundings, 2000, 1 class: water (Dambovita River, "'Lacul Morii"' Lake); f. Comparison algorithm: black - Bucharest 2000, dark grey - Bucharest 1970, light grey - common area (same in 1970 and 2000)

The arguments passed when the BPEL process calls the Classification Web Service (Fig.16) are the images URLs, the class description files (text files) and the samples files (text files) for both of the images. The text files are created by the end-user and uploaded on the remote server prior of process invocation. The Web Service returns a string message, based upon the process is terminated (if the message is an error one), using a BPEL Throw and Terminate Activities, or the process continues with the calling of the ImageComparison Web Service, passing as arguments the URLs of the classified images. The service applies the comparison algorithm on the two images; the result is an image file which shows the differences between them (changes are marked with specific colors). The service returns a string message that will be forwarded to the client. The message contains the URL of the resulted image, or an error message.

The images used in our test cases are Landsat 1 MSS (Multi-Spectral-Scanners) and Landsat 7 TM type of images, downloaded from [17] and are referring to the Bucharest urban area, Dambovita River and surroundings. The years that were considered are 1970 and 2000. The urban growth and the river bed change along 30 years are explicitly shown in (Fig.17). The town surface in 1998 was 228 km2 and the population was 2.029.899; in 1966 its population was 1,452,000.

Three versions of the client application were implemented, using different technologies: RMI (Remote Method Invocation), JSP (Java Server Pages) and HTTP. The user interface is minimal, in the sense that it provides the ability to specify up to five filter services, the original images files, the class description files and the samples files used by the classification algorithm, with no way to specify other service parameters.

The testing environment contains 2 PC nodes (Intel P4, 2 GHz, 1 GB RAM) connected at 100 Mbps. The images' size was up to 40 MB. We performed the following tests: upload the images on the server, apply/don't apply the enhance contrast algorithm, apply 1/2/3/4/no filters, and finally apply the classification and the comparison algorithms. Processing time depended on the images' size and the processing algorithm invoked. The tests proved that the presented application is efficient and easy to use.

4 Conclusions

In this paper we evaluate the suitability of the Web Service composition as possible solution for satellite image processing. We present an example of detecting urban growth and river bed changes over a long period of time.

The service chain is represented as a BPEL document and can be executed in a Web Service orchestration engine, such as Oracle BPEL-engine. The

BPEL process can then be exposed as a Web Service for other researchers to use, resulting in potential for a rich set of functionality that researchers can deploy against large amounts of data without the overhead of local storage or processing resources.

The applications presented here perform complex image processing techniques on large satellite images. In order to diminish the time needed in calculations, our future work will focus on remote parallel image processing, using Grid Computing: the image to be classified will be first split into sub-images; the number of sub-images will depend on the number of available computing nodes in the Grid cluster. Also, the integration of BPEL and Grid computing is to be discussed, along with the problems arising with orchestrating Grid Services.

References

[1] C. Mindruta, *Arhitecturi, Tehnologii si programare in Web*, Ed. Matrix, 2005

[2] D. Ciurchea, *Resurse si programe pentru stiinte experimentale*, http://download.academic.ro/ebook

[3] Java Advanced Imaging, https://jai.dev.java.net/

[4] C.Chrysovalantis, G. Pantelis, S.Konstantinos, D. Antonis, D.Nikos, C.Dionisis, *Development of a web service platform for remotely designing medical image processing applications*, 2nd International Conference From Scientific Computing to Computational Engineering 2nd IC-SCCE Athens, 5-8 July, 2006

[5] David Prez del Rey, Jos Crespo, Alberto Anguita, Juan Luis Prez Ordez, Julin Dorado, Gloria Bueno, Vicente Feli, Antonio Estruch, Jos Antonio Heredia, *Biomedical Image Processing Integration Through INBIOMED: A Web Services-Based Platform*, Lecture Notes in Computer Science, Volume 3745/2005, ISBN 978-3-540-29674-4, pg. 34-43, October 24, 2005

[6] LEADTOOLS Web Service, http://www.leadtools.com/SDK/Web-Services/default.htm

[7] J. Fronckowiak , T. Myer, *Processing Images with Amazon Web Services*, http://developer/amazonwebservices.com, 2008

[8] Qian Li, Haigang Sui, Yuanyuan Feng, Qin Zhan, Chuan Xu, *Research On Remote Sensing Information Processing Services Based On Semantic*

Web Services, The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII, Part B4, Beijing 2008

[9] GeoBrain, http://geobrain.laits.gmu.edu/index.html

[10] K. Keiser, R. Ramachandran, J. Rushing, H. Conover, S. Graves, *Distributed Services Technology for Earth Science Data Processing*, AMS 19th International Conference on Interactive Information Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, Long Beach, CA, 2003

[11] Orchestra project, http://www.eu-orchestra.org/overview.shtml

[12] OWS-5, http://www.opengeospatial.org/projects/initiatives/ows-5

[13] T. Hemalatha, G. Athisha, S. Jeyanthi, *Dynamic Web Service Based Image Processing System*, Advanced Computing and Communications ADCOM 2008, pg. 323-328, ISBN: 978-1-4244-2962-2, 14-17 Dec. 2008

[14] T. Mitsa, P. Joshi, A*n Evolvable, Composable Framework for Rapid Application Development and Dynamic Integration of Medical Image Processing Web Services*, Proceeding of Web Technologies, Applications, and Services , 2005

[15] S. Graves, R. Ramachandran, K. Keiser, M. Maskey, *Deployable Suite of Data Mining Web Services for Online Science Data Repositories*, AMS 23th International Conference on Interactive Information Processing Systems (IIPS) for Meteorology, Oceanography and Hydrology, Long Beach, CA, 2007

[16] ENVI Tutorial, www.ittvis.com/Envi/docs/tutorials/$Classification_Methods$.pdf

[17] Landsat Imagery, http://www.landsat.org/ortho/index.htm

[18] OASIS, http://www.oasis-open.org/home/index.php

Cristina Gherghina
Faculty of Civil Engineering
"Ovidius" University of Constanţa, România
cgherghina@univ-ovidius.ro