Investeşte în oameni !
FONDUL SOCIAL EUROPEAN
Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013
Axa prioritară 1: „Educaţia şi formarea profesională în sprijinul creşterii economice şi dezvoltării societăţii bazate pe cunoaştere”
Domeniul major de intervenție 1.5 "Programe doctorale și post-doctorale în sprijinul cercetării"
Titlul proiectului: „Q-DOC- Creșterea calității studiilor doctorale în științe inginerești pentru sprijinirea dezvoltării societății bazate pe cunoaștere”
Contract : POSDRU/107/1.5/S/78534
Beneficiar: Universitatea Tehnică din Cluj-Napoca

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Ing. Vasile-Dănuț Mihon

TEZĂ DE DOCTORAT

METODOLOGII DE DEZVOLTARE A APLICAȚIILOR INTERACTIVE PENTRU PRELUCRAREA ȘI VIZUALIZAREA DATELOR SPAȚIALE

Conducător ştiinţific,
Prof.dr.ing. Dorian Gorgan

Comisia de evaluare a tezei de doctorat:

PREȘEDINTE: - Prof.dr.ing. ***Liviu Miclea*** - decan, Universitatea Tehnică din Cluj-Napoca;

MEMBRI:
- Prof.dr.ing. ***Dorian Gorgan*** - conducător ştiinţific, Universitatea Tehnică din Cluj-Napoca;
- Prof.dr.ing. ***Valentin Cristea*** - referent, Universitatea Politehnica din Bucureşti;
- Prof.dr.ing. ***Vladimir-Ioan Creţu*** - referent, Universitatea Politehnica din Timişoara;
- Prof.dr.ing. ***Vasile-Teodor Dădârlat*** - referent, Universitatea Tehnică din Cluj-Napoca.

Investeşte în oameni !
FONDUL SOCIAL EUROPEAN
Proiect cofinanţat din Fondul Social European prin Programul Operaţional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013
Axa prioritară 1: „Educaţia şi formarea profesională în sprijinul creşterii economice şi dezvoltării societăţii bazate pe cunoaştere”
Domeniul major de intervenţie 1.5 "Programe doctorale şi post-doctorale în sprijinul cercetării"
Titlul proiectului: „Q-DOC- Creşterea calităţii studiilor doctorale în ştiinţe inginereşti pentru sprijinirea dezvoltării societăţii bazate pe cunoaştere”
Contract : POSDRU/107/1.5/S/78534
Beneficiar: Universitatea Tehnică din Cluj-Napoca

FACULTY OF AUTOMATION AND COMPUTER SCIENCE

Eng. Vasile-Dănuț Mihon

PHD THESIS

INTERACTIVE APPLICATIONS DEVELOPMENT METHODOLOGIES FOR SPATIAL DATA PROCESSING AND VISUALIZATION

PhD supervisor,
Prof.dr.eng. Dorian Gorgan

The PhD evaluation committee:

PRESIDENT: - Prof.dr.eng. ***Liviu Miclea*** - dean, Technical University of Cluj-Napoca;

MEMBERS: - Prof.dr.eng. ***Dorian Gorgan*** – PhD supervisor, Technical University of Cluj-Napoca;
- Prof.dr.eng. ***Valentin Cristea*** **-** reviewer, University Politehnica of Bucharest;
- Prof.dr.eng. ***Vladimir-Ioan Creţu*** - reviewer, Politehnica University of Timişoara;
- Prof.dr.eng. ***Vasile-Teodor Dădârlat*** - reviewer, Technical University of Cluj-Napoca.

Abstract

The availability of large volume of spatial data, broadband Internet access, high storage and processing capability devices, and the Web technologies accelerate the usage of geographic information into our daily lives. The Earth Science related domains (e.g. hydrology, meteorology, air and water pollution, etc.) offer standard services for storing, processing, analyzing, and visualizing spatial data of different types and formats. One of the main research directions of this thesis is to define a methodology for developing applications that provide solutions for all the previous mentioned issues. The theoretical concepts are inspired from the mathematical graphs theory that facilitates the flexible and interactive description of the complex use cases. The nodes represent specific geospatial algorithms while the graph's edges define the entire data flow of the scenario. The processing of large volume of spatial data is another research direction that was intensively studied within this thesis. In order to achieve this goal, a complex data model was defined that is able to represent large use case scenarios (experiments, phenomena) that take place in the real world or that are simulated for specific purposes. The processing term refers to the actual execution of algorithms that manage the spatial data in various forms, such as: the classification of geographic areas based on vegetation indices formula, the creation of thematic maps, the prediction of water quantity/quality for specific hydrological catchments, etc. Due to the fact that the outputs of the complex phenomena (experiments, scenarios) are usually visualized by various categories of users, it is important to present this information in an easy to understand manner, such as: results overlapped onto the interactive maps, dynamic charts, static images, statistics and text-based data, etc. The proposed solutions recommend the usage of standard services for improving the data visualization and analysis that also implies the concept of data sharing. The OGC standard provides the best methods for visualizing remote data, and retrieving them in a highly interactive manner. The spatial data visualization process also involves the identification of a common set of interaction techniques that should be integrated onto multiple platforms, in order to simplify the user related actions for accessing, retrieving, visualizing, and analyzing data, regardless of their storage location. The theoretical concepts proposed within this thesis are used for developing the GreenLand interactive platform that provides useful features for modeling, processing, and visualizing various formats of spatial data.

Acknowledgements

This thesis would not have been possible without the support and ideas of the people that were close to me during this entire period of time. I want to express my gratitude to professor Dorian Gorgan, for its motivation, encouragement, and coordination throughout this PhD thesis, and before that. I also wish to thank him for providing the proper work environment where I could create and validate these ideas.

I am also grateful to professors Valentin Cristea from University Politehnica of Bucharest, Vladimir Creţu from Politehnica University of Timişoara, Vasile Dădârlat and Liviu Miclea from Technical University of Cluj-Napoca, for beeing part of the PhD evaluation committee, and for their effort of expressing the suggestions for improving the quality of this work.

Many thanks go to my colleagues who helped me a lot with ideas, advices, and good thoughts. So, thank you Victor Bâcu, Teodor Ştefănuţ, Vlad Colceriu, Denisa Rodilă, Cristi Mocan, Adrian Sabou, Cornelia Melenti, Mihaela Ordean, and Adrian Coleşa.

Last, but not least, I want to express my sincere gratitude to my family who support me and encourage me to carry on this work, and to be the person I am today. Thank you very much!

Table of contents

List of Abbreviations

ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
CRS	Coordinate Reference System
DAG	Directed Acyclic Graphs
DEM	Digital Elevation Map
DLL	Dynamic Link Libraries
DOS	Dark Object Subtraction
EO1	Earth Observing 1
ERDAS	Earth Resources Data Analysis System
ESIP	Environment oriented Satellite Image Processing Platform
ETM+	Thematic Mapper Plus
EVI	Enhanced Vegetation Index
FTP	File Transfer Protocol
IFD	Image File Directory
IFOV	Instantaneous Field of View
INSPIRE	Infrastructure for Spatial Information in the European Community
ISO	International Organization for Standardization
GDAL	Geospatial Data Abstraction Library
GEMI	Global Environmental Monitoring Index
GEOSS	Global Earth Observation System of Systems
GGL	GearScape Geoprocessing Language
GIS	Geographic Information System
GML	Geographic Markup Language
GPP	Gross Primary Production
GPS	Global Positioning System
GRASS	Resources Analysis Support System
GSD	Ground Sample Distance
GSDI	Global Spatial Data Infrastructure
HDF	Hierarchical Data Format
HRVIR	High Resolution Visible Infrared
KML	Keyhole Markup Language
LAI	Leaf Area Index
MODIS	Moderate Resolution Imaging Spectro-Radiometer
MSI	Moisture Stress Index
MSS	Multi-Spectral Scanner
NDVI	Normalized Difference Vegetation Index
NTSG	Numerical Terradynamic Simulation Group
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium

OGR	OpenGIS Simple Features Reference Implementation
PDG	Processing Description Graph
QGIS	Quantum GIS
RGRI	Red Green Ratio Index
SDI	Spatial Data Infrastructure
SEXTANTE	Sistema EXTremeno de ANalisis TErritorial
SFS	Simple Features SQL
SG	Sum Green Index
SPOT	System Pour l'Observation de la Terre
SWIR	Shortwave Infrared
TIR	Thermal Infrared
TIFF	Tagged Image File Format
TM	Thematic Mapper
uDIG	User-friendly Desktop Internet GIS
VNIR	Visible Near Infrared
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WPS	Web Processing Service
WSDL	Web Service Definition Language

List of Figures

List of Tables

Chapter 1

Introduction

The availability of large volume of spatial data, broadband Internet access, high storage and processing capability devices, and the Web technologies accelerate the usage of geographic information into our daily lives. The Earth Science related domains (e.g. hydrology, meteorology, agriculture, air and water pollution, urban planning, etc.) offer standard services for storing, processing, analyzing, and visualizing spatial data of different types and formats.

Spatial data is a generic description that encapsulates information from different domain fields and can be classified into satellite images, aerial images, and ground-based measurements. Satellite imagery consists of images of the Earth captured by artificial satellites that orbit around the Earth's atmosphere. The ground-based data collection techniques complement the aerial images because they can inform surveys in areas where the airborne laser cannot penetrate (e.g. dense forests, building shadows, etc.). The resulting models are used in many applications, such as: vegetation growth, building and road measurements, river basin erosion, water quality, etc

There are several issues when talking about the spatial data usage within the Earth Science domains, such as: optimal execution of large volume of data, the flexible description of complex scenarios, data sharing among various categories of users, etc. The existing applications usually do not allow the processing of a large volume of data, and are more oriented on the small and medium use cases. On the other hand, to overcome this issue requires an intensive usage of hardware capabilities provided by the distributed platforms (e.g. Grid, Cloud, clusters, etc.).

The process of modeling the complex use case studies requires first the natural description of the inner functionality, and usually involves: a solid understanding of the background context, the collection of the adequate input data set, the syntactic and semantic description of the adopted solutions, the execution over distributed environments in order to speed up the entire process, optimized tools for partial results integration, and some special interaction techniques for visualizing and analyzing the final outputs. The main dificulty is to define a general data model that is able to represent complex algorithms that are connected by certain rules and policies.

The spatial data are currently used in a variety of experiments and studies that are performed by means of software platforms, involving specialists that are geographically distributed. One of the most common issues in such situations is the spatial data availability and data sharing between these researches' communities. The large number of formats and types complicates things even more and impose additional policies for accessing these data from specialized remote repositories. The mechanisms used for retrieving data from one storage does not provide access to other repositories, while the software platforms that process the spatial data tend to encode the results in a proprietary format. All these issues require the improvement of data sharing

and platforms communications.

The title of the thesis contains several keywords that are highly used throughout this entire paper: interactivity, methodology, processing, and visualization. Most of today's Earth Science applications do not provide easy-to-use features when modeling complex use case studies, and require the users to learn complex programming languages in order to fulfill this action. The interactive application refers to the development of features that overcome this issue and provides dynamic modules that can be use by any type of users.

One of the main research directions of this thesis is to define a methodology for developing applications that provide solutions for all the previous mentioned issues. The theoretical concepts are inspired from the mathematical graphs theory that facilitates the flexible and interactive description of the complex use cases. The nodes represent specific geospatial algorithms while the graph's edges define the entire data flow of the scenario.

The processing of large volume of spatial data is another research direction that was intensively studied within this thesis. In order to achieve this goal, a complex data model was defined that is able to represent large use case scenarios (experiments, phenomena) that take place in the real world or that are simulated for specific purposes. The processing term refers to the actual execution of algorithms that manage the spatial data in various forms, such as: the classification of geographic areas based on vegetation indices formula, the creation of thematic maps, the prediction of water quantity/quality for specific hydrological catchments, etc.

Taking into account the large size of spatial data (for some complex case studies it can reach a few hundred GB) the standalone applications do not provide enough computation power in order to process them in a reasonable amount of time. Instead, new platforms had to be developed that make use of the capabilities offered by the distributed infrastructures (e.g. Grid, Cloud, clusters, multi-core, etc.). The proposed methodology is flexible enough to allow the spatial data execution over different computing platforms, but the actual implementation of the theoretical concepts were experimented only on the Grid and standalone infrastructures.

The last step in processing the spatial data is related to the results visualization and analysis. Due to the fact that the outputs of the complex phenomena (experiments, scenarios) are usually visualized by various categories of users, it is important to present this information in an easy to understand manner, such as: results overlapped onto the interactive maps, dynamic charts, static images, statistics and text-based data, etc.

1.1 Thesis objectives

The development of interactive applications that allow the spatial data modeling, processing, visualization, and analysis represents the main research of this thesis. Based on these aspects, we can define the following list of objectives:

1. The definition of a general methodology for developing interactive applications that model, process, and analyze various formats of spatial data. At each step, the proposed methodology should provide the theoretical and practical solutions that solve the issues described earlier;

2. Provide optimal solutions for processing large volume of spatial data, regardless of their type and location. Usually, there are several public remote repositories that store these data, and implement specific policies for accessing them. This objective also takes into account the definition of an architecture that is able to retrieve the needed information;

3. The development of a flexible data model that is able to represent the complex use case scenarios from various Earth Science related domains. The theoretical concepts are inspi-

red from the mathematical graphs theory that facilitates the internal representation and organization of the dynamic elements;

4. Provide interactive techniques for modeling the complex use case studies. Most of today's applications do not allow the visual description of the experiments and use instead text based editors that leave room for syntactic errors, generated by the users that perform the scenarios description. The interactive solution intends to overcome this issue and provides visual modules that facilitate the description process;

5. Improve the process of spatial data sharing and platforms interoperability that use the Grid infrastructure in order to perform the execution process.

1.2 Thesis outline

The current thesis is organized in 9 chapters, with 162 bibliographic references. The contributions highlighted throughout this paper were published in 12 ISI and B+ scientific journals, 12 articles in national or international conferences indexed as ACM and IEEE, 1 book chapter, and 12 presentations given at different workshops and trainings.

Chapter 1 describes the general context of this thesis, together with the motivation and the proposed objectives that are further detailed throughout the paper.

Chapter 2 details and classifies the spatial data that are relevant for this study into three types: satellite images, aerial images, and ground based measurements. Due to the fact that the spatial data are more and more present into our daily lives, several initiatives were proposed in the last years that intend to provide optimal solutions for data sharing among different scientific communities and between specialized repositories from where they can be accessed by means of standard services.

The critical survey regarding the state of the art on spatial data related applications is the main subject of **Chapter 3**. The existing solutions provide similar services, but they address different types of requirements: the first ones are suitable for the execution of simple geospatial algorithms, while the second category uses the distributed computing environments in order to provide optimal solutions for processing a large volume of data. The current thesis aims to fill the gaps between the issues encountered when analyzing these applications. This is the main reason why this chapter also highlights the differences between the existing solutions and the ones adopted within this thesis. The theoretical concepts defined by the proposed methodology are going to be implemented as functionalities of the GreenLand platform that provides features for spatial data modeling, processing, and analysis. Based on this aspect, the current chapter generally describes the GreenLand platform as an example of interactive application, developed on the proposed methodology.

A parallel description of the methodologies used for developing the classical software products and the spatial data related applications, is presented in **Chapter 4**. The first one is based on 5 methodological steps, while the other one is a 7-phase process that starts with the context problem definition and the identification of the requirements, and continues with modeling the system related architecture, specifying the methods that can be used for processing the spatial data over different computing infrastructures, and identifying the most suitable presentation techniques that can be used in the visualization and the analysis process. The last two steps of the proposed methodology describe the use cases that can be developed for testing and validating the application, together with the guidelines for installing it on different platforms.

The main theoretical concepts related to the abstract representation of the complex use case scenarios are presented in **Chapter 5**. The proposed solution uses the concepts of basic

operators and complex workflows for describing the experiment's functionality. These concepts are inspired from the mathematical graphs theory and allow the data flow description and organization as a graph, where each node encapsulates the behavior of a specific geospatial algorithm (as basic operator) or groups of algorithms (as complex workflows). The connections between these items are specified as uni-directional edges that also have the role of transferring the output results of one item as input to other elements. The abstract definition of the workflow needs to be encoded into a digital format that can be executed over different computing infrastructures. This stage involves its transcription into a high-level programming language, by taking into account all the concepts defined earlier. The XML internal format is recommended, due to the fact that it allows the definition of entities closely related to the graph notion. In cases of complex experiments, the size of these files is large and can be managed only by computer scientists, but usually these scenarios are developed by the Earth Science domain field specialists. The proposed solution is based on the OperatorEditor and WorkflowEditor tools that provide interactive techniques that simplify the development process to the level of specifying the operators and workflows in a visual manner, avoiding this way the direct management of the text-based files.

One of the main research directions of **Chapter 6** is the execution of a large volume of spatial data in a reasonable amount of time. This means that the geospatial algorithms should be processed on powerful machines that provide parallel and distributed capabilities (e.g. Grid, Cloud, clusters, etc.). On the other hand, the proposed methodology also addresses the small scale scenarios that are not so computing intensive. This chapter starts with a general overview of the most important computing environments (e.g. Grid, Cloud, clusters, multi-core, etc.) and particularizes the execution process onto the Grid and standalone platforms. Another important aspect is the fact that the entire processing is divided into 4 stages: pre-processing, actual execution of data, monitor, and results analysis. Each level describes the features of the methodology that are representative, and provides details about the GreenLand platform's mechanisms for partitioning the workflows into tasks and execute them onto the Grid worker nodes. The resources sharing improvement between the applications that use the Grid infrastructure is also described within this chapter. The OGC standard seems to provide the best solutions for invoking geospatial algorithms that are resident on external platforms. In the standalone environments the default solutions are working just fine, but when dealing with large data processing that takes a lot of time to complete, those methods do not provide the proper guidelines. This methodology recommends the extension of the WPS service to include a monitor operation that is able to periodically query the status of the execution.

Chapter 7 describes the theoretical concepts that are used in order to visualize and analyze the spatial data, regardless of their type and storage location. It identifies the proper presentation techniques of the results obtained during the execution process. Based on the categories of users that are going to utilize these applications, the spatial data results should be visualized as: text files, dynamic charts, static images, statistics, interactive maps enriched with the actual image result, etc. The Earth Science related applications usually exchange a large volume of spatial data, in various formats. The execution results can also be included into this category, and represent important resources for many scientific communities. Based on these aspects, the proposed solutions recommend the usage of standard services for improving the data visualization and analysis that also implies the concept of data sharing. The OGC standard provides the best methods for visualizing remote data, and retrieving them in a highly interactive manner. The spatial data visualization process also involves the identification of a common set of interaction techniques that should be integrated onto multiple platforms, in order to simplify the user related actions for accessing, retrieving, visualizing, and analyzing data, regardless of their storage location. Otherwise, the users have to utilize multiple tools

(that provide similar functionalities) with proprietary interaction mechanisms that have to be learnt in order to be able to perform a complete visualization and analysis process.

The theoretical concepts proposed within this thesis define a methodology for modeling, processing, and visualizing various use cases from the Earth Science activity domains. Even though these concepts were partially validated in the sections they were described, **Chapter 8** proposes two complex experiments in order to test all the functionalities implemented throughout this methodology. These two scenarios represent real use case studies that were formulated by domain field specialists and reflect the current problems they are confronting with, in terms of water quality/quantity and land cover/land use aspects.

The main contributions of this thesis are presented in **Chapter 9**, together with the conclusions and the future work research directions.

Chapter 2

Spatial data characteristics

Spatial data represent the information that identifies geographic locations, features, and Earth surface boundaries. It is also known as geospatial data, and it makes use of the geographic coordinates and surface topology to identify all these features [1]. The location attributes contained within are formally expressed by means of points, lines, and polygons. This referencing model of observations and analysis is part of the Geographical Information System (GIS) [2].

2.1 General overview

A Geographical Information System (GIS) could be described as a collection of software tools that allows the creation, visualization, and analysis of spatial data. Map production is one of the most common functions of geospatial applications. This process renders the data in a viewable format that is displayed on a computer screen or a printed paper. Such applications generate static or dynamic maps that could be easily customized and integrated in complex analysis algorithms.

2.1.1 Applicability domains

In the modern society, the spatial data has been used in a wide range of studies, such as:

1. **Land use/land cover mapping**: Knowledge of land use and land cover is important for many planning and management activities. Land cover maps are continuously developed from local to national to global scales. The use of panchromatic (grayscale) data to map land use, has been an accepted practice since 1940. More recently, small-scale aerial photography and satellite images have been used in order to perform the classification processes.

   The term land cover relates to the type of features present on the surface of the Earth (e.g. lakes, trees, vegetation, urban areas, etc.). It also relates to the human activity or economic function associated with a specific region. As an example, a piece of land of an urban area may be used for family housing. Depending on the level of mapping detail, the land use can be described as urban use or residential use.

   An even more detailed classification can be obtained by taking into account the agriculture land (cropland, orchards, confined, etc.), urban or build-up land (residential, industrial, and mixed), forest land (deciduous, evergreen, mixed forest land), water (lakes, reservoirs,

bays), barren land (beaches, sandy areas, salt flats, etc.), snow or ice (snowfields and glaciers), etc. [3];

2. **Geological and soil mapping**: Persons who seek to describe and explain the Earth structures, must be able to recognize the surface expressions of various materials and soil types. Through the processes of geologic and soil mapping, these materials will always require a considerable amount of field exploration, but the mapping process can be greatly facilitated through the use of visual data interpretation.

   Geologic mapping involves the identification of landforms, rock types, and rock structures. Because most of the surface and near-surface mineral deposits have been found, the current emphasis is on the location of deposits far below the Earth's surface. Geophysical methods that provide deep penetration into the Earth, are needed to locate potential deposits. However, much information can be provided by interpretation of surface features, described in the spatial data images. The soil mapping process involves the field examination and the classification of soil types. Spatial data has been used for almost 80 years to facilitate these kinds of processes [4];

3. **Environmental assessment**: Environmental impact assessment is defined as the systematic identification and evaluation of the potential impacts of projects, plans, programs relative to physical, chemical, biological, and cultural environments [5]. Such assessments will engage expertise from multiple domains: forestry, landscape architecture, land planning, geology, environment, ecology, hydrology, air pollution, etc. Many of the spatial data interpretation techniques can be used to assist the analysis of such assessments. These techniques may include characterizing changes in the land surface over time, identifying the location of landfills, detecting water quality in different watersheds, etc.;

4. **Water resources**: Whether for irrigation, manufacturing, or power generation, water is one of the most critical resources. Spatial data interpretation can be used in multiple ways to monitor the quality, quantity, and geographic distribution of this resource.

   In general, most of the sunlight that enters a clear water body is absorbed within a 2 meters of the surface. The degree of absorption is highly dependent on wavelengths types. Near-infrared wavelengths are absorbed in only a few tenths of a meter of water, resulting in very dark satellite images tones [4]. The analysis of underwater features is often permitted by using imaging systems sensitive to wavelengths of 0.48 µm up to 0.60 µm. The groundwater location is important for both water supply and pollution analysis. Available spatial data interpretation techniques cannot be used directly to map the depth of water in a groundwater system. However, vegetation types have been successfully used as indicators to approximate this feature. There are several sub-domains, where the water resources are used for modeling different Earth phenomena: water pollution, water quality, floods, hydrological watershed assessment, shoreline erosions, snow cover, etc. [6];

5. **Agricultural and forestry applications**: When considering the components involved in studying the worldwide supply agricultural products, the applications of remote sensing are many. Classifications performed within this domain are based on their spectral response patterns. These patterns are influenced by human values and economic, political, and social systems. On the other hand, forestry is concerned with the management of forests, vegetation, water, and wildlife. Visual interpretation of spatial data provides a feasible mean of monitoring many of the world's agricultural and forest conditions [7];

6. **Archeological investigations of ancient settlements**: The archeological investigations deal with obvious monuments of earlier societies. Visual interpretation of spatial

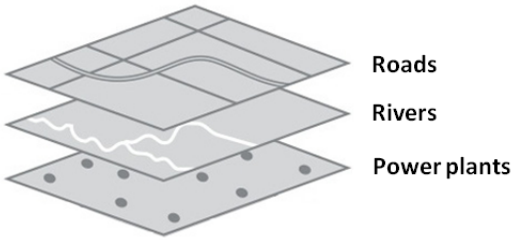


Figura 2.1: Digital map layers overlapped within GIS systems
(Source: `http://makingmaps.owu.edu/gisatowu/gisatowu.html`)

data has prove to be useful in locating sites whose existence has been lost to history. Surface features include visible ruins, rocks, and other surface markings. Subsurface features identify covered building, canals, and roads. When such elements are beneath agricultural fields or vegetation, they may be revealed on satellite images by tonal anomalies, resulting from difference in soil moisture or crop growth [8];

7. **Urban and regional planning**: Requires nearly continuous acquisition of data to formulate governmental policies and programs that range from the social, economic, and cultural domains to the context of environmental planning. The current research directions of urban and regional planning refer to: population estimation, traffic and parking studies, site selection processes, and urban change detection [9];

8. **Etc.**

2.1.2 GIS systems

A GIS system is a computer tool capable of capturing, storing, analyzing, and displaying geographically referenced information (data identified for a specific location). One of the most important functions provided by GIS systems is the ability to relate different information in a spatial context and to reach a conclusion about their relationship.

GIS applications are software tools that can be manipulated using the mouse and the keyboard input devices. A common feature among all these applications is to display the information as different layers over interactive maps. Each layer is stored as files, or as records in a database and it is represented with different symbologies (points or pixels, lines, polygons, or other icons that identify real world objects) (Figure 2.1).

Spatial data used in GIS applications is not only about the interactive maps creation. Instead, it could be used to apply a large set of analysis methods for computing distances between geographic locations, to determine what features overlap other features, to establish the number of locations within a certain radius, etc. These may seem simplistic, but can be applied across multiple disciplines. The analysis results may be shown on a map, or tabulated into database tables, reports, or files (Table 2.1). Any feature that can be located spatially can be classified as spatial data. The location attribute may be determined by (x, y, z) coordinates, latitude and longitude measurements, or by other systems such as ZIP codes [10].

If the data in not digitized into a form recognized by GIS applications, various techniques should be applied in order to overcome this impediment. Electronic scanning devices can be used to represent geometrical primitives (e.g. points, lines) as map pixels. One of the most time consuming components of a GIS system is the data digitization process (of the information collected by hand) into recognizable formats supported by the related software tools[11].

No.	Power plant name	Location	Latitude	Longitude
1	Cernavoda Nuclear Power Plant	Romania - Cernavoda	44.3203°N	28.0593°E
2	MZFR	Germany - Leopoldshafen	49.1043°N	8.4325°E
3	KWG	Germany - Grohnde	52.0348°N	9.4097°E
4	Almendra Dam	Spain - Almendra	41.1614°N	6.1912°W
5	Langage	England - Plymouth	50.3884°N	4.0117°W

Tabela 2.1: Example of tabular organization of geospatial data

In order to assure a data level interoperability, the GIS platforms should provide specific conversions algorithms [12]. All these issues (regarding the data digitizing process, data projection, storage, and conversion) are part of the GIS system and will be presented in more details in the following chapters.

2.2 Data measurement and acquisition

Remote sensing is the science of obtaining information about an object, area, or phenomena through the analysis of data acquired by a device that in not in a direct contact with that feature. This domain is similar to a reading process. Data is remotely collected by sensors that use various measuring techniques: acoustic wave distribution, electromagnetic energy distribution, or natural force distributions [4].

Generally speaking, the remote sensing includes aerial, satellite, and spacecraft observations of the surfaces and atmospheres of the planets in our solar system. In a more restrictive concept the remote sensing discipline detects and measures electromagnetic energy (e.g. visible light, ultraviolet, infrared, microwaves, and audio waves) that reaches the Earth's surface and atmosphere. The Earth is naturally illuminated by the Sun electromagnetic energy that emits visible light as wavelengths between 0.4 - 0.7 µm ranges (represents the visible spectrum that consists of the red, green, and blue color channels). Other solar energy fractions are in the form of ultraviolet and infrared wavelengths. Remote sensing platforms generate these types of waves and measure the characteristics of the portion of the signal that returns to the sensor after it hits the Earth's surface.

Space photography is the result of the sunlight that passes through the full thickness of the Earth's atmosphere. On the other hand, an airborne sensor detects energy emitted directly from objects on the Earth's surface without the need to cross the atmospheric field. Because of the various nature of atmospheric effects the following sections describe the scattering and absorption mechanisms that are closely related to the accuracy of spatial data measurements [13].

Atmospheric scattering is the unpredictable diffusion of radiation by particles in the atmosphere. The scattering increases with the degree of humidity and the number of particles load in the atmosphere. Most of the time, the path radiance produced by the scattering effect is not taken into account for near infrared wavelengths. On the other hand, atmospheric absorption results in the effective loss of energy. The most efficient absorbers of solar radiation are the water vapor, carbon dioxide, and ozone. The wavelengths ranges in which the atmosphere is particularly transmissive of energy are referred to as atmospheric windows [14].

In order to better understand how the interaction processes influence the spatial data acquisition let's analyze the reflected Electromagnetic Radiation (EMR) mechanisms (Figure 2.2). When entering the Earth's atmosphere, the sunlight encounters small particles of gas, dust,

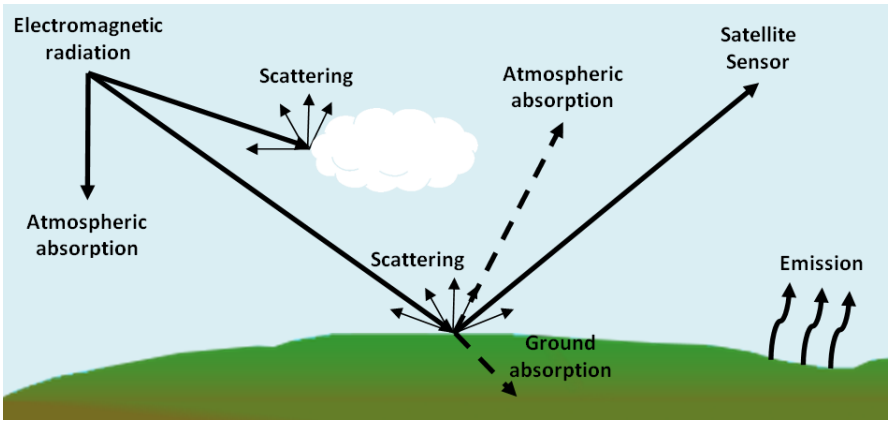


Figura 2.2: Interaction processes in remote sensing
(Source: `http://remote-sensing.net/concepts.html`)

and aerosol that scatter a fraction of the incoming radiation in all directions. In the next step, the remaining light source is absorbed by other atmospheric gases (e.g. the ozone layer absorbs the ultraviolet radiations). All these effects filter the sunlight that reaches the Earth's surface. At this stage it may encounter soil, rock surfaces, vegetations, lakes, etc. that will absorb a fraction of this radiation.

The amount of absorbed energy depends on the material's physical properties, creating a sort of spectral signature of that geographical area. Most of the not absorbed radiation is reflected back into the atmosphere where it is detected and measured by the artificial sensors. Based on the sensor attributes, this information is stored in a standard format, known as satellite images [15]. Imaging in remote sensing can be carried out from satellites, aircraft platforms, or ground based sensors. They have similar characteristics, but the differences in their altitude and stability can lead to very non-common image properties.

Remote sensing involves the data acquisition about an object or a geographic area, by reflecting and absorbing electromagnetic energy. Panchromatic (grayscale) and color (e.g. red, green, blue) images provide this information from the visible wavelengths domain of the electromagnetic spectrum. More recently, panchromatic images evolved, and now they encompass many bands (several hundreds or even more) by covering the rest of the electromagnetic spectrum, in the Near Infrared (NIR) and Shortwave Infrared (SWIR) bands.

Based on these measurements, the resulting spectral signature can be used to identify the materials and the properties for the geographic area. After collecting all the data, each pixel contains a sample value (called spectral signature) that can be interpreted in order to determine the properties of the materials present in that area. Usually this information is related with soil, water, and vegetation that emit specific wavelengths that can be directly measured by the satellite's sensor.

Multiple techniques were developed for data collection. The push-broom imagining sensor is the most common one, where a scan line of spatial pixels is decomposed in K spectral bands (Figure 2.3). The area coverage rate is the *swath* multiply by ground velocity (v). The area of a pixel on the ground is the square of the Ground Sample Distance (GSD) [16]. The ground truth data, or reference data, stores the information about objects, areas, or natural phenomena that are collected by field measurements. Usually, Global Positioning System (GPS) sensors are used to accurately determine the position of these observations. Reference data might be used to serve one or many of the following purposes:

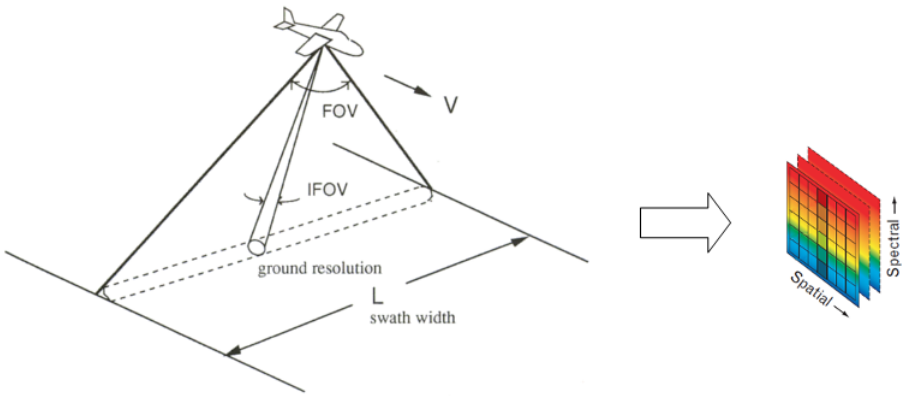


Figura 2.3: Multispectral imaging system
(Source: `http://wtlab.iis.u-tokyo.ac.jp/~wataru/lecture/rsgis/rsnote/cp6/cp6-2.htm`)

- To calibrate an artificial sensor;
- To verify the information extracted from remote sensing data;
- To aid in the analysis and interpretation of remotely sensed data.

The reference data collection can be time consuming and consists of time-critical and time-stable measurements. The time-critical measurements are performed in cases where ground conditions change rapidly in time (e.g. water pollution event). Time-stable measurements are involved when these conditions are stable over a longer period. For example, the geologic field observations can be conducted at any time and would not change appreciably on long periods of time [4].

One of the most important outcomes of remote sensing data is the role played by these technologies in representing the Earth as a system. The satellite systems operate within the electromagnetic spectrum that extends from 0.3 µm up to 14 µm. This range includes: ultra-violet, near infrared, mid infrared, thermal infrared, microwaves, and radio waves.

The remote sensing discipline is rapidly changing due to the fact that new products are developed and launched into the Earth's orbit. In this dynamic environment it is important that the potential users understand the basic characteristics of these systems and what are the trades-offs that recommend a type of sensor for a particular application.

The satellite's orbit is one of these characteristics and is related to the altitude, period, inclination, and equatorial crossing time elements. For example, a 900 inclination is referred to as near polar, because the satellite will pass close to the South or North poles. Spatial, spectral, radiometric, and temporal resolutions are also important characteristics that influence the data quality. The spatial resolution can be defined as the smallest feature detected by remote sensing imagery and it depends on the Instantaneous Field of View (IFOV). The ground area generated by the IFOV is called cell and determines the sensor's maximum resolution (e.g. 1:10000 means that one image pixel would actually represent an object with 1 Km in length). Images where only large features are visible have low resolution, while the high resolution is allocated to images that offer a clear display of the small objects.

Band	Name	Description
1	Blue	Useful for describing the water quality and quantity
2	Green	Measures the green reflectance of vegetation
3	Red	Designed to sense in a chlorophyll absorption region
4	Near infrared (NIR)	Used for identifying the vegetation types and biomass content analysis
5	Shortwave infrared (SWIR)	Gives an accurate estimation about the soil and vegetation moisture content. It is also useful for snow and clouds differentiations
6	Thermal infrared	Used by the thermal mapping applications
7	Shortwave infrared (SWIR)	Useful for identifying the mineral and rock types

Tabela 2.2: Landsat spectral bands description

The spectral resolution of a satellite includes the number and the width of the spectral bands generated by its sensor. The extreme cases are panchromatic sensors (contain a single grayscale band) and hyper-spectral sensors (contain hundreds of spectral bands).

The radiometric resolution describes the number of bits used to record the remote data. The finer the radiometric resolution of the sensor, the more detailed ground objects it could record. Usually, satellite images use positive digital numbers that can be expressed as a power of 2. If a sensor uses 8 bits within its measurements there will be 2^8 digital values available, starting from 0 up to 255. These kinds of representation have a big impact over the accuracy and the size of the resulting image.

Temporal resolution refers to the time period needed by the satellite to complete one entire orbit cycle. Usually this period is several days long; therefore the absolute temporal resolution of a remote system to record the same ground area at the same IFOV angle is equal to this period. Not in all cases this value is preserved between orbit stages. In fact it depends of a variety of factors: satellite/sensor capabilities, the *swath* overlap, latitude, IFOV, etc.

There are multiple remote sensing platforms that can be classified based on their low, medium, and high spatial resolution. The most common are:

- **Landsat**: The Landsat satellites are about 3 m tall, 1.5 m in diameter, and they were launched in 1972 into the Earth's orbit at an altitude of 900 Km. They cover the entire surface every 16 days, about 20 times per year. The best nominal ground resolution is 30 m, and the spectral sensitivity includes the following infrared ranges: 0.47 - 0.57 µm (Green band), 0.58 - 0.68 µm (Red band), 0.69 - 0.83 µm (Blue band) that represent the first three bands of the Landsat images [17].

  The satellite has 3 types of sensors: Multi-Spectral Scanner (MSS), Thematic Mapper (TM), and Enhanced Thematic Mapper Plus (ETM+). Currently, the ETM+ is the state of the art that exports the collected data as 7-bands products (Table 2.2).

  The Landsat images are used in multiple domain fields, such as: agriculture, cartography, environmental monitoring, forestry, geography, land use planning, water resource analysis, etc. Usually these types of images are used in combination with low altitude aerial data sets, in order to improve the accuracy of the analyzed results. The common format of the Landsat data is GeoTIFF that allows geo-referenced information to be embedded in a regular Tagged Image File Format (TIFF) file. The geographic region, type of projection, and the measurement description techniques are stored within the file header [18], while the actual data represents the image body content;

Bands	Spatial resolution	Description
1, 2	250 m	Useful for land cover classification process, including leaf area index mapping
3 - 7	500 m	Identify the land, cloud, and aerosol properties
8 - 16	1000 m	Provide information about the biogeochemistry, ocean color, etc.
17 - 19	1000 m	Useful for atmospheric water and vapor analysis
20 - 23	1000 m	Sea, cloud, and surface temperature
24, 25	1000 m	Store information related to atmospheric temperature
26 - 28	1000 m	Include data about the troposphere temperature and humidity
29	1000 m	Cloud properties
30	1000 m	Deep analysis of the ozone layer
31, 32	1000 m	Useful for interpretation of the surface and cloud temperature
33 - 36	1000 m	Provide cloud altitude related information

Tabela 2.3: MODIS spectral bands description

- **Moderate Resolution Imaging Spectro-Radiometer (MODIS)**: The first MODIS sensor was launched in 1999 on board of the Terra satellite, at an altitude of 705 Km. Three years later the second sensor was installed on the Aqua satellite. The coverage of the same geographic area is completed every 1-2 days. The MODIS products contain 36 spectral bands, with a spatial resolution of 250 m (channels 1 and 2), 500 m (channels 3 - 7), and 1000 m for the rest of the bands. The wavelengths range starts from 0.62 µm and goes up to 14.38 µm, due to the large number of bands [19]. The full bands description is provided within Table 2.3.

  The resulted images are available free of charge for a large suite of products that process and analyze the vegetation indices. MODIS products are organized in different processing levels. Level 0 provides raw digital data, level 1 includes radiance values, and the second and third layers are derived from the first two.

  The MODIS products improve the understanding of the global dynamics related to land, water, and atmospheric domains. They are also used in the development and validation of complex Earth models that are able to predict global changes, and to assist the specialists in their decisions regarding the environment.

  Hierarchical Data Format (HDF) is the most common file format that contains MODIS sensed data. It allows the information definition in a hierarchical manner, based on data descriptors and data elements. The descriptors contain metadata information, an offset inside the file to indicate the location of the object, and the length of the data. Data elements represent the actual sensed information, stored as a collection of objects [20];

- **Advanced Spaceborne Thermal Emission and Reflection-Radiometer (ASTER)**: Is one of the five sensors on board Terra satellite that was launched in 1999, at an altitude of 705 Km. Its main applicability domains are related with cloud cover, glaciers, land temperature, land use, vegetation patterns, etc. The temporal resolution of this sensor is 16 days, and for ground based geographical areas it provides data at 15, 30, and 90 m spatial resolution. The wavelengths of the electromagnetic spectrum vary from visible (0.52 µm) to thermal infrared light (11.65 µm).

The ASTER images contain 15 bands, grouped in three main categories: Visible Near Infrared (VNIR), Shortwave Infrared (SWIR), and Thermal Infrared (TIR) [21]. The first 3 bands are part of the VNIR category that provides data at a 15 m spatial resolution, and represents the best multispectral data available, excepting IKONOS and QuickBird. The bands indexed from 4 up to 9 are contained within the second category, with a spatial resolution of 30 m. The remaining bands are classified in the thermal infrared spectrum and they offer the lowest resolution for the ASTER products (90 m);

- **System Pour l'Observation de la Terre (SPOT)**: It is used for obtaining Earth imagery for land use, geology, and water resources and it is part of the commercial remote sensing platforms, operated by the French Space Agency. The system weights 1750 kg, with 3.5 m tall and 2 m in diameter. The first version (SPOT 1) was launched in 1986 at an altitude of 832 Km. Meanwhile SPOT 6 product was released that provides even more accurate data. It has three types of sensors: High Resolution Visible (HRV), High Resolution Visible Infrared (HRVIR), and Twin HRV.

  There is a waveforms range difference (compared to Landsat products) when collecting data for the Blue (0.45 - 0.52 µm), Green (0.53 - 0.59 µm), Red (0.62 - 0.69 µm), and near infrared (0.78 - 0.89 µm) bands. By combining the four SPOT products, it is possible to generate high spatial resolution data at 20, 10, 5, and 2.5 m [22].

  They offer two imaging modes: panchromatic (P) and multispectral (XS). The IFOV, ground sampling interval, ground *swath*, and pixel resolution are the main differences that occur between the two functional states;

- **Earth Observing 1 (EO1)**: This satellite was launched in the year 2000, at an altitude of 700 Km, and offers continuous data with a repeating cycle of 16 days. Its main objective is to develop and validate technologies for observing the Earth features with unique spatial, spectral, and temporal characteristics. Such a unique feature is the hyper-spectral sensor (called Hyperion) that captures high resolution data in 242 continuous spectral bands. The Hyperion sensor provides data at 30 m spatial resolution, and covers the range of 0.36 µm up to 2.6 µm electromagnetic spectrum [23];

- **QuickBird**: It provides the highest resolution satellite imagery currently available to the public domain. It was launched in 2001, at an altitude of 450 Km. The average revisit time is 1 to 3.5 days, depending on the latitude and image collection angle. It functions in two modes: panchromatic and multispectral (contains 4 bands - Blue, Green, Red, and NIR). The spatial resolution varies from 0.6 m up to 1 m (for single band images) and from 2.5 m up to 4 m, in the case of multispectral images. The QuickBird high resolution data are comparable with aerial acquired information, and they are used for accurate large-scale mapping applications [24];

- **IKONOS**: Is the first successful launch of a commercial satellite platform that provides very high resolution images. It was deployed in 1999, at an altitude of 682 Km, with a ground coverage delay of 11 days. It produces panchromatic images (black and white) and multispectral ones (contains 4 bands - Blue, Green, Red, and NIR). These two types of images can be combined in order to produce pan-sharpened imagery, with 1 m spatial resolution [25]. The GIS applications that use the IKONOS products are in the fields of agriculture monitoring, resource management, or urban planning.

2.3 Data types and formats

Spatial data is a generic description that encapsulates information from different domain fields (e.g. topography, meteorology, hydrology, land cover, etc.) and can be classified into satellite images, areal images, and ground-based measurements.

Satellite imagery consists of photographs of the Earth made by means of the artificial satellites that orbit around the Earth's atmosphere. There are four attributes that characterize the satellite images accuracy: spatial, spectral, temporal, and radiometric resolution. Spatial resolution represents the pixel size that corresponds to a specific geographic area. The spectral resolution specifies the number of bands contained within the collected data. Temporal resolution represents the time interval required by the artificial satellite to measure the same geographic areas over the Earth's surface. The radiometric resolution identifies the number of grayscale levels of the satellite image, and it is usually represented on 4, 8, or 16-bits [26]. Most used products perform these measurements on a daily basis or over an 8 or 16-days time range interval.

Aerial images are generated by using aerofotogrametric cameras, installed on board of airplanes. They scan the Earth's surface in more detail then satellite images (with meters values for the spatial resolution) but require a longer period of time for data collection. The surface topography is distorted in aerial images, and until corrections are applied these measurements are not accurate [27]. The following list provides the most common characteristics used for identifying objects in aerial images:

- Hue or color tone: offers the most basic interpretation of elements, based on their relative brightness;
- Size: useful to determine the visual impact of objects (e.g. a small stream or a large river);
- Shape: provides a fast way to identify objects in aerial images (e.g. regular geometric shapes usually represent human settlements);
- Patterns: it is also known as spatial arrangement. For example, there is a difference between the random patterns generated by an unmanaged area of trees, and evenly arranged parks;
- Associations: there are cases when objects are always found in association with other objects (e.g. crop fields are usually located along a river stream).

The ground-based data collection techniques complement the aerial images because they can inform surveys in areas where the airborne laser cannot penetrate (e.g. dense forests, building shadows, etc.). The resulting models are used in many applications, such as: vegetation growth, building and road measurements, river basin erosion, water quality, etc. Generally speaking, the ground-based data though is easy to collect, often requires post processing. Not all sources of spatial data are available as pixel oriented image. In some cases the data will be available as analog maps that require digitization. This is particularly the case with line and area data types that must have a pixel-value representation.

Even if Remote Sensing Products (RSP) could not replace ground based methods to provide high quality information, their advantages include the provision of large scale datasets with large temporal resolution. Therefore RSP could be a valuable source of information for many applications related to Earth observations and could eficiently complement data gaps. The measured parameters include: station position (elevation, latitude and longitude), daily

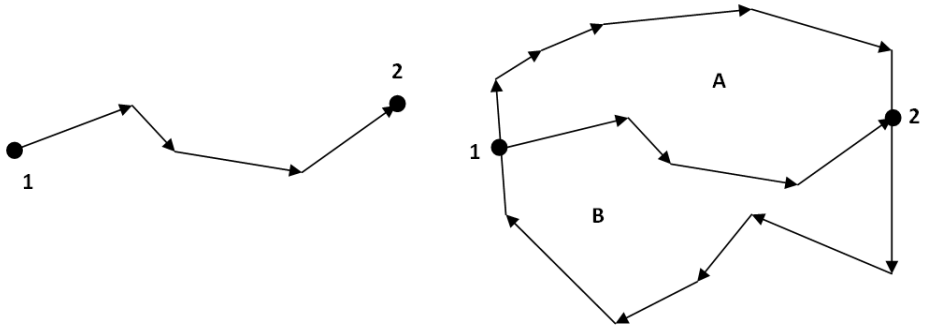


Figura 2.4: Polyline (left) and polygon (right) vector data features

discharge rates, climate characteristics, station, air temperature (e.g. temperature at set times every 3 hours), wind speed, cloud altitude, etc

Depending on the type of features and the purpose to which the data will be applied, spatial information can be also classified into vector and raster categories. Depending on the type of problem that needs to be solved, even a combination between them could be used.

Vector data is stored as a series of (x, y) coordinates within the GIS system. It is used to represent geometric primitives (points, lines, and polygons) and provides a simple way to represent different features (e.g. roads, trees, rivers, houses, etc.) that exists in the real world. For better features management, they are enriched with metadata, called attributes [28].

The point feature is map scale dependent. For example, for a small scale map this feature can be used in order to represent a city location. However, when zooming in, the map resolution increases, and it makes more sense to display the city boundaries as a polygon. For a correct representation, each point feature has to be provided with (x, y) coordinates, and optionally with the z value. One of the most common types of representation is the latitude/longitude one. This will position the point feature at certain location on the map, based on the 180 latitudinal lines (starting from the Equator and extending to the South and North Poles) and 360 longitudinal axes (covering the horizontal Earth's surface, from East to West).

When dealing with multi-points features, they can be described as polylines that represent a continuous path throughout each vertex (Figure 2.4) and identifies roads, rivers, open contours, flight paths, etc. There are a few rules for polylines that have to be taken into account. For example contour lines may touch, but they never overlap each other. The attributes attached to a polyline describe some of its properties (e.g. the surface color of the road, the traffic direction of a street, the trees height, etc.). The polygon features identify enclosed areas (e.g. country borders, islands, etc.) and they follow the same principle as the one explained for the polyline case [10].

In its simplest stage of representation, the vector data can be used as a normal map. But when combined with the spatial analysis processes, it allows the user to find answers to questions like: what houses are within 100 Km range?, what are the boundaries of Romania?, etc.

There are also some disadvantages when working with vector data sets. The first one is the correct representation of the same information at different scales (e.g. maps have different scales, and when importing the vector data from a map into a GIS system, the digital information will have the same scale as the original map. This process should be corrected by applying different spatial transformation algorithms).

When representing the real world features as polylines and polygons, slivers (there is no proper intersection between the areas of two polygons), overshoots (two line features do not

meet in an inflexion point, and instead the second line ends beyond this point), and undershoots (two line features that should intersect do not meet each other) effects may occur [10].

While vector features use geometry (points, polylines, and polygons) for the real world representation, the raster data is composed of pixels matrix that contains numerical values. While any type of geographic data can be stored in a raster format, these data are especially suited for the representation of continuous information (e.g. vegetation growth, elevation maps, water quality, water quantity, etc.). Raster data pixels may have or not have proper values (null symbol means that data was not collected properly or is outside of the area of interest).

When working with raster information, there is the need to establish the geographic location of the dataset. This process is commonly known as geo-reference that stores coordinates for the top left corner of the raster image and the number of pixels in the X and Y directions. The vector data can be visualized as multiple layers over interactive maps. The same principle applies to the raster data containing multispectral bands, like RGB and satellite images.

A major disadvantage when using raster data is its cell size related with the spatial resolution at which the information is represented. Processing the associated data attributes may be tricky if large amount of information exists. Raster maps reflect only one attribute or feature for a geographic area. Since most input data has a vector shape representation, the vector to raster conversions are frequently used.

Raster format is recommended for the representation of multispectral image data. Moreover, most image forming devices (e.g. digital cameras, satellite images) operate on a raster display, compatible with digital data acquisition and storage. Raster format is also appealing from a processing point of view, since the logical records for the data are the pixel values and adjacent relationship are easy to establish by means of the pixel addresses. In contrast, vector format does not offer the features.

2.4 Relevant initiatives and standards

The initiatives promote the use of open standards, concerning data access, interoperability, standardization, and services integration. These processes involve different policies, rules, data, technologies, and human and financial resources that are required in order to fulfill their objectives [29].

Standards are documented agreements used in international trades that provide technical specifications, rules, and guidelines in order to ensure that products, processes, and services fulfill their purposes [30]. Even though standards implementation is a complex process that involves a large number of organizations, it is recognized that standards are important for productivity improvement, lowering costs, and facilitating the maintenance process [31].

2.4.1 Global Earth Observation System of Systems - GEOSS

The Global Earth Observation System of Systems (GEOSS) initiative started its activity in 2005 and represents a worldwide effort related to the development of a system of systems. It consists of international organizations and governmental agencies that fulfill the objectives of GEOSS.

What differentiate GEOSS from other initiatives is that it focuses on existing systems and tries to update their rules and guidelines in order to improve them, rather than create new ones. All these are included into a portal that connects the end-users with the data producers in order to facilitate public access to near-real time data, collected through various Earth observations mechanisms.

The processes for data access and data sharing are described in the 10-year Implementation Plan where information providers have to implement "a set of interoperability arrangements, including technical specifications for collecting, processing, storing, and dissemination shared data, metadata and products. GEOSS interoperability will be based on non-proprietary standards, with preference to formal international standards. Interoperability will be focused on interfaces, defining only how system components interact with each other and thereby minimizing any impact on affected systems" [32].

The organizations involved in the GEOSS community are required to implement the following principles:

- All data and metadata should be made public as soon as possible, even in near-real time related to their acquisition;
- The data exchange should respect all the national and international policies within GEOSS;
- Free of charge data encouragement is promoted, otherwise a no more than reproduction cost should be required.

2.4.2 Infrastructure for Spatial Information in the European Community - INSPIRE

The INSPIRE consortium [33] started its activity in 2007 with the goal of creating a Spatial Data Infrastructure (SDI) that allows organizations across Europe public access to geospatial data. The data repository will be updated periodically with the information provided by these organizations, agencies, and other consortiums.

The following issues influence in a negative manner the data access and data exchange across European states:

- Incomplete geospatial data;
- Data duplication and incompatibility;
- Data level interoperability issues;
- Lacking of documentation;
- Cultural, linguistic, and financial barriers in data sharing.

In order to overcome these issues new legislative rules should be adopted based on sets of requirements, standards, processes, and European states activity coordination. The main goal of this initiative is to define an "infrastructure for spatial information, meaning: metadata, spatial data sets and spatial data services; network services and technologies; agreements on sharing, access and use; and coordination and monitoring mechanisms, processes and procedures, established, operated or made available in accordance with this directive" [34].

The spatial infrastructure development is a slow process, because it involves all the European member states. It provides several implementation rules regarding the elements of the system, based on the following principles:

- Geospatial data should be collected only once, to avoid information duplicates;
- Data should be collected from different sources across Europe and they should be available for public usage;

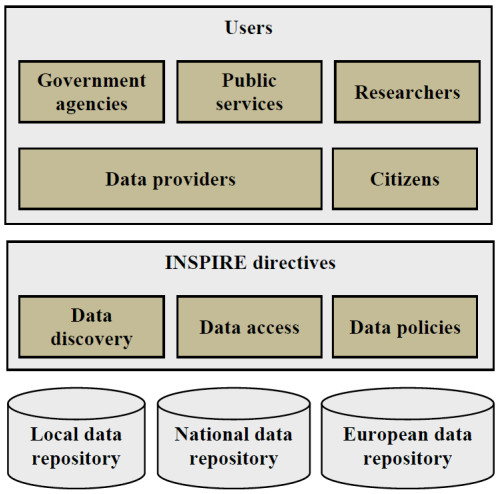


Figura 2.5: Data flow within the INSPIRE framework

- Data discovery software products proved to be useful in satisfying the users' needs when searching for specific information.

Figure 2.5 highlights the data flow within the INSPIRE framework, facilitating this way the users' efforts when searching for specific information. All data are stored in local, national, and international data repositories from where the users may access it by the means of existing standards. The spatial data regards different activity domains, such as: hydrology, land use, land cover, networks, urban planning, etc. All this data sets are accessible throughout a common gateway that aggregates multiple instances of SDIs, improving this way the data loss issues.

According to INSPIRE, all member organizations (private or public) should provide access to the following categories of services:

- Discovery services: allow data discovery, evaluation, and use throughout specialized software applications;
- View services: provide innovative techniques for displaying data to the users (e.g. using interactive maps, generating data legends, showing metadata content, etc.);
- Download services: all these public data can be downloaded free of charge. In case of highly important information, an arbitrary price may be required;
- Transformation services: focus on the usage of standards for providing data interoperable services.

2.4.3 Global Spatial Data Infrastructure - GSDI

It is a worldwide group of organizations, agencies, and people whose main goal is to "promote international cooperation and collaboration in support of local, national, and international spatial data infrastructure developments that will allow nations to better address social, economic and environmental issues of pressing importance" [35].

Its main goals are to provide spatial data access, based on the help of research, education, and communication agencies. The GSDI community relies on international standards and guidelines in order to assure the following purposes:

- Facilitate data access: based on Web services and metadata description, the users are able to extract the needed information;
- Promote and develop awareness exchanges: implies the global level interaction with policy makers in order to influence other Earth Science domains;
- Promote SDI development: they are useful for storing data in common repositories, and offering public access for various categories of users;
- Partners involvement: find partners with common goals and include them in GSDI program. This is useful in order to accomplish the GSDI visions and goals.

A specific set of tools is required to handle the pre-processing of spatial data, such as: download and storage, geo-referencing, or conversion algorithms between multiple data formats. Another challenge is the large size of the datasets which makes it inconvenient for the users to download and store the entire sets at multiple locations. One possible solution is to collect them into a global data repository and provide users with routines for data access and extraction.

Regular users do not have direct access to raw satellite and aerial data. Instead, these data are pre-processed (e.g. removing national strategic locations) and then stored into a large repository from where the users are able to access relevant information. In order to achieve this, an extra step is required: downloading the data from the repository onto the user's machine. This scenario is working in cases in which the user tries to process a small data set of information.

Usually spatial data are stored in large repositories that offer remote access to the information, via File Transfer Protocol (FTP) and HyperText Transfer Protocol (HTTP) protocols. New measurements are added periodically by means of satellite sensors that provide raw data that is pre-processed and then inserted into these repositories. The time cycle is identical with the temporal resolution of the sensors that provide this information. In consequence, the users are able to remotely access the data and use it in their own analysis studies.

In some cases, such simulations require a large volume of data, over a long period of time. One experiment was about extracting hydrological data from the entire Black Sea catchment, starting with 10 years-ago measurements and ending in the present. The total size of data was about 550 GB. It was therefore necessary to store these datasets within one place and provide users with spatial and temporal sub-setting functions.

Spatial data is organized differently across remote repositories. First of all, the packaging mode is of a certain type (e.g. Landsat, MODIS, ASTER, etc.). Because the methodology described in this paper proposes a unique internal data representation (GeoTIFF) all these information have to be converted into GeoTIFF. Most of the times, the satellite image contains multiple bands of information. Based on this aspect, the software application that implements this methodology should be able to extract each band into a different GeoTIFF file.

Another issue is related with the time series datasets organization. These data repositories are updated periodically (e.g. every 8 days), so each of them should have an indexing algorithm. For example, the Numerical Terradynamic Simulation Group (NTSG) repository stores multiple MODIS products with a proprietary data organization. The MOD16 products are organized in different folders available worldwide via FTP protocol. As can be seen from Figure 2.6, the MOD16 data is organized in years, started from 2000 until the current time. Because it is an 8-days temporal resolution product, each year contains day-folders (e.g. D001, D009, D0017) starting from 1 and increasing up to 361.

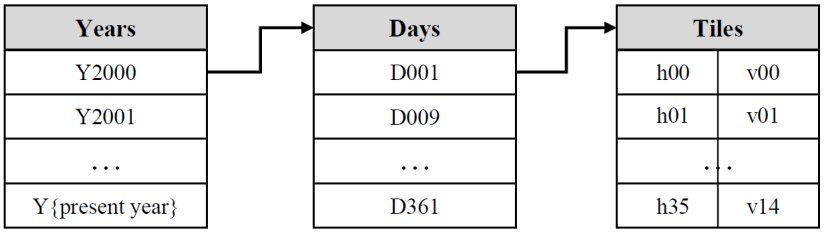


Figura 2.6: MOD16 data representation in NTSG repository

For an easier identification of the MOD16 global level data, this consortium partitioned the Earth's surface into horizontal and vertical tiles (e.g. the Central Europe region is between h18-h21 and v03-v05). Each day from Figure 2.6 contains all combinations of horizontal and vertical tiles. Taking into account all these aspects, an example of accessing one of the MOD16 images could be generated: ftp://ftp.ntsg.umt.edu/pub/MODIS/Mirror/MOD16 /MOD16A2.105 _MERRAGMAO/Y2000/D001/MOD16A2.A2000001.h00v08.105.20103551538 35.hdf.

Spatial data from other repositories are organized differently. It is up to the software application to differentiate between these representation styles and to allow users the possibility to extract relevant data, by means of semi-automated mechanisms.

2.4.4 Open Geospatial Consortium - OGC

The rules and guidelines provided by the previous mentioned initiatives represent the basis for several standards organizations that put them into practice in the GIS domain fields. According to these issues, they manage to increase the interoperability between systems and geospatial data. There are several standards that are related with the GIS system (e.g. Open Geospatial Consortium - OGC [36], International Organization for Standardization - ISO [37], Spatial Data Transfer Standard - SDTS [38], Organization for the Advancement of Structured Information Standards - OASIS [39]), but the most important ones are OGC and ISO.

The Open Geospatial Consortium (OGC) is a non-profit organization that consists of more than 380 companies, governmental agencies, and universities, and it is the most representative standard within GIS activity domains. Its goals are to provide guidelines, rules, and software API, recognizable throughout the entire geospatial community. Based on these standards, the developers are able to create interoperable services for information access and information exchange, but also complex data structures that could be accessible to a large number of applications [36]. These standards are based on HTTP protocol, SOA architectures, Web Service Definition Language (WSDL), and REST protocol for Web services.

The goal of the OGC organization is to provide guidelines for developers and users in order to:

- Produce services for accessing spatial data;
- Assure the geospatial interoperability;
- Allow the creation of standards, integrated into daily processes, regarding spatial Web computing;
- Facilitate interoperability between GIS applications;
- Facilitate the implementation of open architectures.

Without interoperability and standardization, data access and data exchange is really difficult among organizations. In general, any Web service must have the ability to describe its own capabilities, enabling this way other services and products to interoperate based on its standard functionalities. There are several OGC standards that are commonly used for accessing, visualizing, analyzing, and processing data throughout Web services:

- **Web Map Service (WMS)**: defines a Web interface that allows geo-referenced data retrieval as map layers. For security reasons the WMS service does not give access to the real data, but instead it creates different layers representation (e.g. JPEG, PNG, TIFF) of this data. The WMS interface is a three-step process that consists in the following operations: *GetCapabilities* (exposes the service functionalities and lists all its available layers), *GetMap* (produces a map based on the selected layers set), and *GetFeatureInfo* (returns information about the generated map content).

  For each WMS request there are lists of mandatory and optional parameters. For example, a basic *GetCapabilities* request should include at least the WMS service version (e.g. http://<server_host>/service=WMS&request=GetCapabilities&version=1.1.1). The response is an XML file that contains information about the service and the data availability. A simplified type of response is presented bellow:

```
<Capability>
   <Request>
      <GetCapabilities>
         <Format>application/vnd.ogc.wms_xml</Format>
      </GetCapabilities>
      <GetMap>
         <Format>image/png</Format>
         <Format>application/atom xml</Format>
      </GetMap>
      <GetFeatureInfo>
         <Format>text/plain</Format>
         <Format>application/vnd.ogc.gml</Format>
      </GetFeatureInfo>
   </Request>
   <Layer queryable="1">
      <Name>preview:snow_classification</Name>
      <Title>Snow classification</Title>
      <KeywordList>
         <Keyword>snow</Keyword>
      </KeywordList>
      <SRS>EPSG:4326</SRS>
      <LatLonBoundingBox minx="-180.0" miny="-56.0"
                         maxx="179.989" maxy="84.011"/>
      <Style>
         <Name>snow_classification_style</Name>
         <Title>Raster</Title>
         <LegendURL width="20" height="20">
            <Format>image/png</Format>
         </LegendURL>
      </Style>
   </Layer>
</Capability>
```

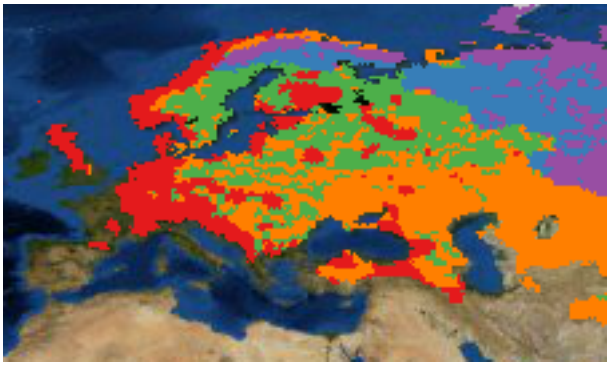


Figura 2.7: GetMap service response

The *GetMap* request returns a map (Figure 2.7) of the selected layers in the format supported by the WMS service (PNG, in this case). There are several mandatory parameters that the user must include in its request: *bbox* (bounding box coordinates for the area of interest), *format* (the graphical format of the returned map data), *layers* (a list of layer names), *width* (map width in pixels), *height* (map height in pixels), *SRS* (the projection type), *styles*, and *version*.

The *GetFeatureInfo* service is used to get information about the selected layer or about a specific feature (e.g. by clicking on a map point, its geographic coordinates and color can be displayed to the user). As in the case of the previous two requests, there are several mandatory parameters: *bbox*, *format*, *layers*, *query_list*, *x*, *y*, *SRS*, *styles*, and *version*;

- **Web Feature Service (WFS)**: it allows features retrieval and management using the Geographic Markup Language (GML) [40] format. This service is intended to be used only for vector datasets, while the WCS works mainly on raster images. The WFS interface is a three-step process that integrates the following operations: *GetCapabilities* (similar to the one described for WMS), *DescribeFeatureType* (defines the structure of the feature), and *GetFeature* (the response is encoded based on a GML schema). There are two implementations of this standard: basic WFS and transactional WFS. The first one is used to query and to retrieve features, while the WFS-T provides services for features creation, deletion, and update;

- **Web Coverage Service (WCS)**: the Web Coverage Service provides standardized access to raster datasets. Like the rest of the OGC services, the WCS interface consists in three types of requests: *GetCapabilities*, *DescribeCoverage*, and *GetCoverage*. Based on the XML result, generated by the *GetCapabilities*, the user is able to select and download data for a specific area of interest. The area's geographic coordinates, image format, width, height, and projection are a few of the mandatory fields required by this service;

- **Simple Features SQL (SFS)**: it is an open standard that offers rules and guidelines for storing, querying, updating, and retrieving geospatial features from SQL databases. SFS establishes an architectural framework for features representation, provides syntaxes for defining geometric attributes attached to those features, and describes a set of geometry types in order to ease the data exchange processes [41];

- **Web Processing Service (WPS)**: besides data accessibility, the OGC standard also provides geospatial data processing services through the WPS interface. It allows users

to know what processes are available, to select the proper input data, to create and run different models, and to manage the results of the computation. As any OGC standard implementation, it includes three types of operations: *GetCapabilities* (returns a list of all available processes), *DescribeProcess* (for each selected process it provides a general description of its parameters), and *Execute* (performs the process execution. It does not offer tools for monitoring the execution progress, and once the result is available it is send to the user).

The Keyhole Markup Language (KML) [42] and Geographic Markup Language (GML) [40] are both working with an XML schema, allowing the description of geographic objects that could be visualized in a Web based viewer. The KML is implemented by Google Earth and represents a simpler version of the GML standard (e.g. the GML can also be used as an interoperable data exchange format within Web applications, not only in the information visualization process). The GML Application Schemas offer a powerful environment for defining specific features, in a format similar to this one:

```
<gml:Point gml:id="point" >
    <gml:coordinates>45.67, 88.56</gml:coordinates>
</gml:Point>
```

2.4.5 International Organization for Standardization - ISO

It is a non-governmental organization consisting in more the 160 representative countries that activate as promoters of international standards in different activity domains (e.g. social, academic, scientific, governmental, etc.). Within ISO there is a special committee who focuses in geographical information area of interest. It provides more the 50 standards, including services and tools for data management (e.g. acquiring, accessing, processing, analyzing, and visualizing digitized data) [37]. The ISO 19115/19139 and ISO 19119 are two of these standards that address metadata and XML schema implementation for the Geographic Information System (GIS).

The ISO 19115/19139 [43] contains rules for describing geospatial information based on more than 400 metadata elements. This standard can be used to catalogue and describe spatial datasets and to identify their geographic features. Data are defined using an XML schema that can be also used in the geospatial information exchange processes. The ISO 19119 [43] defines patterns for service interfaces in GIS platforms and specifies the structure of the XML document that is used for storing the capabilities of geospatial Web services (similar to *GetCapabilities* from the WMS interface).

Recently, the OGC organization started to interact with ISO throughout a partnership in order to promote their standard at a global level, and to integrate it in the ISO 19100 series.

2.4.6 Organization for the Advancement of Structured Information Standard - OASIS

It is worldwide consortium with more than 5000 members that provides support for open standards for the global information society, and focus on the following categories: Web services, computing management, XML processing, etc. It also offers some guidelines for the Earth Science related domains, but only at a general level [39].

Chapter 3

State of the art on spatial data related applications

The availability of high performance measuring platforms, broadband Internet access, high storage and processing capability devices, and the Web technologies accelerate the usage of geographic information into our daily lives. GIS applications are widely spread across Earth Science domains: hydrology, meteorology, agriculture, air and water pollution, urban planning, etc. They offer standard services for storing, processing, analyzing, and visualizing spatial data of different types and formats.

This critical survey identifies the main issues of the existing spatial data processing and visualization platforms, and provides theoretical and practical solutions that are further described within this thesis. The state of the art analysis covers the functionalities provided by these systems, the computing infrastructures used for optimizing the overall data processing, the spatial data management, and the human-computer interaction techniques.

3.1 Standalone platforms

This critical survey is conducted on Earth Science related applications that work in distributed or standalone environments. In this context, the distributed environment term integrates all the computing resources (e.g. Grid, Cloud, clusters, multi-core, etc.) that provide parallel and distributed capabilities in processing various types of spatial data. On the other hand, the standalone platforms are identified as single-core machines that sequentially perform the execution of geo-spatial algorithms. The multi-core infrastructure can also be integrated in this second category if the related application does not perform parallel computations on all its CPUs. A list of the most known platforms that use the standalone environments for spatial data processing is given bellow:

3.1.1 Sextante toolkit

Sistema EXTremeno de ANalisis TErritorial (SEXTANTE) is an open source spatial data analysis library that allows the integration with other GIS applications. It was design to work on several operating systems, like Windows, Linux, and Mac OS. It contains more than 300 modules and algorithms that handle raster and vector data types, providing rich common geoprocessing functionalities useful for the entire GIS communities. The remaining paragraphs highlight the main objectives and characteristics of this application [44]:

- Algorithms development: any GIS algorithm that process spatial data is based on specific mathematical formulas that represent its main core. By using this approach the algorithm can be easily extended by other specialists, and parts of it may be used in order to create more complex functions. After the validation phase these algorithms can be used by researchers in the GIS domain fields, through the provided API packages.

  The SEXTANTE algorithms are divided into several groups, based on the functions they provide: raster and vector layers analysis, vegetation indices, statistical computations, distances and area identification, hydrological analysis, etc. The output of an algorithm could be saved as a local file, as a database entry item, or uploaded as a remote resource by using the OGC WFS service;

- Flexibility: in order to achieve this goal the algorithms were implemented to work with most of the existing data types, including raster and vector information, shapefiles, various projection formats, etc;

- Algorithms re-usage: in some cases, complex geo-spatial algorithms can be applied on specific GIS domains. The algorithm re-usage is important in these situations, when someone wants to maintain the same basic functionality, but applicable for other Earth Science disciplines.

Even though it was developed by a large number of specialists in computer science and GIS domains, the SEXTANTE library still encounters some limitations and issues in the following areas of work [45]:

- Topology: in its current version the application does not offer support for topological information, even though there is a small set of operators related with this domain (e.g. correcting the topological structure of lines and polygons layers);

- 3D vector data: SEXTANTE integrates different libraries that handle 3D data representation (e.g. JTS [46]), but in reality all the operations are performed as bi-dimensional computations;

- Incomplete semantics: some of the data processing modules are highly dependent on specific input data types and cannot be easily extended for other purposes. For example, there was an attempt to integrate these modules into the GearScape Geoprocessing Language (GGL), but some algorithms failed to be included due to their data type interconnectivity issues [47];

- Processing large data: several limitations may occur when working with large raster datasets that are based on the erroneous behavior of the current Java GIS functions that are part of the SEXTANTE internal structure. Not the same could be expressed for large vector data types, since special routines were optimized to work specifically with these kind of information.

Figure 3.1 describes the system related architecture, where graphical components link the algorithms implementation with the application interface. The data binding and GUI binding modules are needed because they connect the data processing results with the application viewer system (in this case an interactive map). Besides the predefined set of functions developed as SEXTANTE's core module, some external algorithms were also integrated based on the WPS standard and the GRASS library.

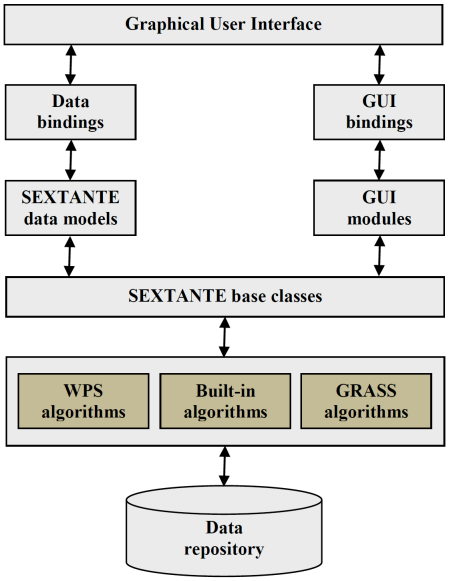


Figura 3.1: General architecture of the SEXTANTE library

The major functionalities provided by SEXTANTE were integrated in several open source platforms, such as OpenJUMP [48] and gvSIG [49]. Recently, this application has been ported as a plugin for the gvGIS system, enriching its spatial data processing algorithms. The 52North WPS integrated the SEXTANTE library of predefined functions[47], offering this way the possibility to access all these geo-spatial algorithms through OGC standards.

There are four basic GUI modules that are used to run the internal functions in order to analyze the results and to display the spatial data as raster and vector map layers:

1. **The toolbox**: Is the most frequently used module that provides quick access to all the available algorithms. Due to the length of this list, a tree based grouping mechanism was implemented. In respect with this feature a search option could be used that offers a faster way of finding the desired functions. It accepts as inputs a word or a complex phrase, and will perform the searching in the help file associated to each algorithm.

   Depending on the data availability for the current user, some of the algorithms may not be accessible. For example, if a function requires raster layers as input, but the user hasn't loaded them yet, this function will be disabled and displayed in grey color. The generated result can be stored as local files or uploaded to remote repositories by using the WFS-T service.

   The algorithms are easy to use. After selecting such a function a pop up window will be displayed, containing the inputs, the outputs, and the options section. The data types recognized by SEXTANTE are: raster layer, vector layer, table data, numerical value, strings, band index when dealing with multispectral images, and files. Based on the inputs and outputs used for the selected algorithm, the application will apply a filter to all these data types, allowing the selection of the proper information from the SEXTANTE's data repository.

   There are cases in which the inputs do not have the same geographical area extent. By applying data resampling and cropping mechanisms this impediment is eliminated, and

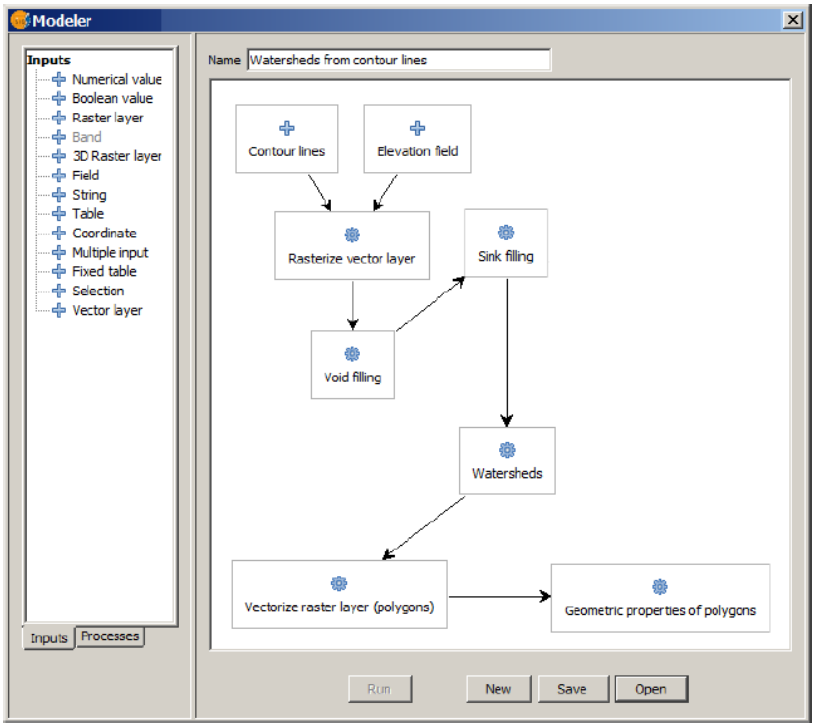


Figura 3.2: SEXTANTE's graphical modeler

uniform data type may be created.

Iterative execution of algorithms is possible only from command line instructions, without the possibility to interleave results between executions (e.g. to use as input the result generated by a previous iteration). The execution order and parameters information are written into a table-like structure, where each row represents one processing iteration;

2. **The graphical modeler**: Allows the creation of complex workflow of algorithms. All these actions are done interactively, by placing the nodes on a virtual canvas. Describing the relations between these nodes is also possible through the use of mouse devices. Each node embeds an algorithm, but for visual purposes it is represented as a rectangular shape (Figure 3.2).

   The graphical modeler does not support the sub-workflows concept, meaning the integration of one graph within other ones. The process of enumerating the inputs values for each node of the workflow is similar to the one presented above, where a modal window contains the inputs, outputs, and options fields.

   Updating the position of the elements is done interactively by dragging them to different (x, y) coordinates, while the corresponding relations are updated automatically. Enriching the database of geo-spatial algorithms (left side column from Figure 3.2) is also possible by saving the content of the workflow as an XML representation;

3. **The command line interface**: It is recommended for advanced users that are developing complex operations and need to use the full capabilities of the SEXTANTE application. The GUI has several limitations (e.g. lack of looping commands and conditional sentences)

that are not encountered when describing the data processes as command line instructions. This interface extends the BeanShell [50] Java source interpreter with new mechanisms for geo-algorithms execution. BeanShell allows users to develop functions that can be called within a script and provides support for Perl and JavaScript objects and methods definition;

4. **The history manager**: The parameters list, execution status, and results information are saved within the history manager module. This way it is easy to supervise the entire execution and interfere when errors occur.

3.1.2 uDIG platform

The User-friendly Desktop Internet GIS (uDIG) is a Java open source desktop application, compatible with the most common operating systems: Windows, Linux, and Mac OS that integrates the following capabilities [51]:

- WMS, WCS, and WFS support: allows access to data that can be displayed over interactive maps (as WMS layers) and it provides read and write information privileges via transactional WFS (WFS-T). Usually, the data is stored on a MapServer [52] or GeoServer [53] instance that implements the WCS service, allowing information retrieval in a standard manner, compliant with the latest OGC requirements;
- Standard GIS file formats support: currently the uDIG works only with shapefiles and GeoTIFF data types that could be viewed, edited, or printed on paper support;
- Database access support: allows a direct connection to PostGIS, ArcSDE, and MySQL databases;
- Modularity: provides the possibility to integrate new features as external plugins, or to easily extend the existing capabilities.

The main goal of the system is to fill the functional gaps between the geo-spatial standards (e.g. OGC) and open source communities, as described in the uDIG system related architecture (Figure 3.3). For the open source community this application has a similar role as ArcView [54] for the proprietary GIS software tools. It also offers the possibility to represent database information in a visual manner, just like any other raster image. On the other hand, uDIG was designed as an internet services consuming client that integrates perfectly with the latest OGC standards.

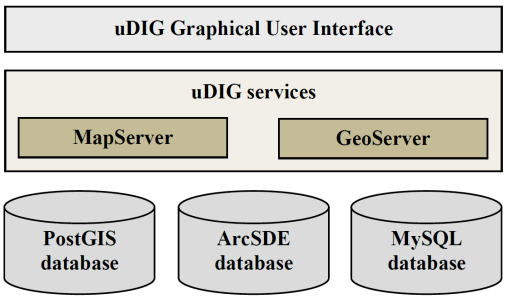


Figura 3.3: uDIG system related architecture

One of the most important aspects about the uDIG system is related with the attempt to integrate as much existing technology as possible, without the need to re-implement all their functionalities. As a result several tools were successfully added as parts of the uDIG platform [55]:

- OpenJUMP: an spatial data interface framework that implements an advanced data editor tool. It also adds some extra enhancements such as: advanced attribute query, SVG data export, classification methods for generating thematic maps, geometry editing tools, etc. [48];
- GeoTools: is an open GIS library that provides advanced data access, but lacks of a graphical user interface. It integrates a large set of data types and formats, including PostGIS information. The project began in 1996 at Leeds University and its main goal was to provide a toolkit of resources to enable the creation of interactive geographic data visualization layers. The GeoTools represented a major improvement in the overall uDIG functionality, because it gave access to algorithms that take a lot of time and effort to implement [56].

  Its core architecture is highlighted in Figure 3.4. The interface level provides an API for external tools that access its main functionalities. The Java Topology Suite (JTS) offers a set of spatial data operations through the use of complex geometric algorithms, and optimized queries related to spatial datasets [46]. JTS implements Simple Features SQL (SFS) from the OpenGIS standard, and in cases when this standard does not offer clear specifications, JTS adopts a consistent alternative;
- gvSIG: is a user friendly open source desktop GIS application developed by European GIS communities. It recognizes the most common raster and vector data formats, and can integrate local and remote information based on WMS, WFS, WCS services and database sources. The gvSIG project is developed based on the Spatial Data Infrastructure (SDI) paradigms that offers users the possibility to share and integrate spatial information. The latest released version adds new raster functions, such as: large raster images visualization, transformation functions, DEM generation, time and spectral analysis, best route computation, data geo-codification, etc. Editing, analyzing, and viewing GIS data is a

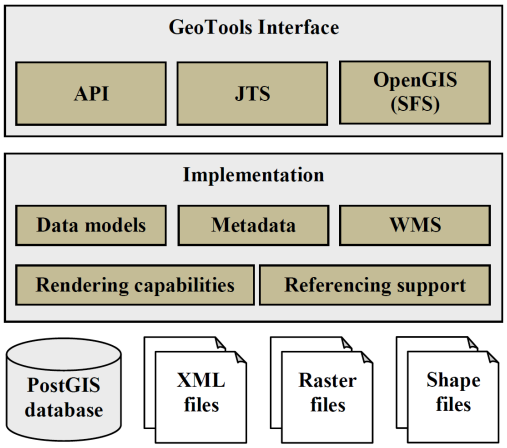


Figura 3.4: GeoTools system related architecture

common task within the gvSIG framework. The high quality maps are enriched with pie and bar charts, and support for raster and vector layers.

Initially, both uDIG and gvSIG followed common development directions with different implementation solutions, without knowing one of each other. For example uDIG started to integrate GeoTools library, while gvSIG rejected this technology and implemented a new graphical interface using Java Swing. Also a major issue is related with license usage, neither of them willing to change it [49].

3.1.3 GRASS library

The Geographic Resources Analysis Support System (GRASS) is one of the oldest open source GIS applications, implemented by the U.S. Army Construction Engineering Research Laboratories in 1985. It can be described as a desktop application, portable on Windows, Linux, and Mac OS operating systems. It is written in POSIX, C++, and Python programming languages, with all changes tracked in public source repositories, along with real time notification system.

Data interoperability is assured through the Geospatial Data Abstraction Library (GDAL) and OpenGIS Simple Features Reference Implementation (OGR) libraries [57]. GDAL is an open source collection of algorithms useful in reading and writing various raster data formats. The latest version includes over 200 types, where the most frequently used are: GeoTIFF, DEM, ESRI, JPEG, PNG, ascii, and RasterTIF.

The OGR is a C++ open source library that provides reading access to more than 40 vector data formats. The most common formats handled by this library are: ESRI shapefile, geodatabase, PostGIS, MapInfo, etc. In general, the OGR tool is similar with GDAL, the main difference between them is the data types (raster and vector) they are working with [57].

Additionally, GRASS offers the possibility to access and process spatial data based on OGC services (e.g. WMS, WCS, and WFS). It also provides 3D visualization techniques based on the NVIZ [58] engine and can be linked to various databases, such as: PostgreSQL, MySQL, and SQLite.

It is highly modularized, allowing its execution even on mobile devices [59]. Currently, the GRASS library consists of more than 400 data processing modules that are organized in several categories, based on the functionality they provide: vector, raster, image processing, database management, etc. GRASS can also be used on high performance clusters when large volume of data needs to be processed (e.g. satellite images). The distributed and parallel executions are not provided by default. Instead, it requires explicit specification of the modules that are needed at runtime. Some of these modules are highly dependent one to another, so the identification of all dependencies is another subject that must be taken into account when using GRASS within the distributed infrastructures.

GRASS proved to be a powerful tool in many environmental science domains. For example, it was used to manage weather information and to classify the clouds formation based on radar collected data [60]. It is used in terrain modelling, leading to a rich framework for large data analysis. In geology domain it proved to be useful in 3D modelling of the relationships between the surfaces and geologic units. This tool was applied to develop a new groundwater flow model that takes into account sediment transport, water quantity, and terrain representation [61].

GRASS algorithms can be accessed based on command line instructions. This way it is very easy to integrate them with other GIS applications and to wrap these functionalities into their internal specifications. On the other hand, many GIS specialists were not too satisfied in using the GRASS script commands, and wanted a more lightweighted mechanism. All these issues represented the basis for creating a graphical interface where each user action is automatically

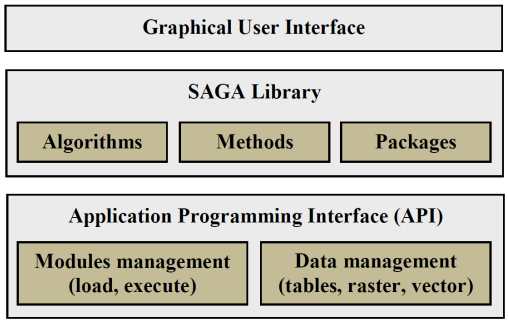


Figura 3.5: SAGA system related architecture

linked to a proper command line instruction [62]. There are two main Graphical User Interface (GUI) components:

- Layer manager: allows users to execute GRASS functions by selecting them directly from the application's menu. It is also possible to open command line terminal, database content, or the map layer management tools;
- Map display window: contains an interactive map that provides zooming, panning, data querying, charts integration, etc. Different raster and vector layers could be embedded within this module.

System for Automated Geoscientific Analyses (SAGA) is another application that provides similar GRASS functionalities. It is an open source platform (developed at the Department of Physical Geography from University of Gorringen, Germany) useful in the editing process of spatial data. It is written in C++ and includes 48 libraries and more than 300 algorithms for spatial data processing and visualization. Data exchange formats is assured by the GDAL and OGR libraries [57], while data re-projection is based on the PROJ.4 library [63].

Besides the raster and vector data processing functions, SAGA provides other useful modules for image manipulation and analysis, including: sharpening, edge detection, filter algorithms, classification techniques, distance and area computation, statistical operations, terrain analysis based on DEM data, etc.

SAGA's system related architecture (Figure 3.5) is modular and includes a large number of classes and functions exposed as an API. They are generated as Dynamic Link Libraries (DLL) and have to be accessed through a graphical interface. This interface is defined on a simple structure that allows working with many different data sources, while keeping them organized correctly. There are five main GUI modules: menu bar, status bar, messages window, toolbar, and project window. They all work together in order to provide the user the current status of its project and to assist it when needed.

An alternative to the graphical user interface is the command line interpreter. It is intended especially for experienced users, and it has a major advantage over the same functionalities provided throughout the interface: it enables the looping execution of modules, by using bash script files [64], [65].

3.2 Distributed environments related platforms

Earth Science applications that process large amount of data require intensive usage of hardware capabilities provided by the distributed platforms. The description and processing of natural

phenomena and experiments, from different domain fields, is a complex process that usually involves: a solid understanding of the background context, the collection of the adequate input data set, the syntactic and semantic description of the adopted solutions, the execution over distributed environments in order to speed up the entire process, optimized tools for partial results integration, and some special interaction techniques for visualizing and analyzing the final outputs.

Taking into account the large size of spatial data (for some complex case studies it can reach a few hundred GB) the standalone applications do not provide enough computation power in order to process them in a reasonable amount of time. Instead, new platforms had to be developed that make use of the capabilities offered by the distributed infrastructures. The following sections highlight the current adopted solutions in order to provide meaningful results to the end-user, even in cases of processing large volume of data.

3.2.1 Quantum GIS

Quantum GIS (QGIS) is an open source Geographic Information System that offers standard functionalities based on predefined packages or plug-ins. The project began in 2002 and today it is available for UNIX, Windows, and Mac OS platforms. Its main goal is to provide common functions and features that are easy to use by the GIS community.

Even though it started as a spatial data viewer, QGIS quickly implemented new functionalities that support data processing and interpretation [66], [67]. Just like any other desktop software, QGIS can be used for mapping operations of spatial data, display data layers over interactive maps, zooming, panning, perform distance measurements, customize map legend, create symbologies, data re-projections, support for a large number of data types, GPS integration, geo-referencing, etc. The following paragraphs describe its main features:

- Map creation: the QGIS software offers the possibility to display both raster and vector data. The raster layers can be rendered as a single band (individual band within the same spatial data stack) or as three colored bands (represent the red, green, and blue layers of the colored image).

  It also provides support for displaying each layer with a certain transparency degree, by using a slider to indicate to what extent the layer should be visible. This feature is useful when several bands need to be overlapped or when the final result requires a 3D look-like effect.

  When using large resolution images, the QGIS creates a pyramid-like structure of lower resolution copies of this data. The rendering performance is improved especially on zooming actions. The pyramids are computed based on two resampling algorithms (average and nearest neighbor). There is the possibility to display the data color histogram that can be generated for each band individually or for the entire spatial image.

  The QGIS offers support for more than 60 vector data types (e.g. ESRI, MapInfo, GML, etc.) based on its OGR library integration. In general a vector data file contains information about the geometrical features that are going to be represented on the map, the attributes attached to these features, projection information, coordinate reference system, etc.

  Another important functionality related with the map creation process is the possibility to display PostGIS layers, stored within PostgreSQL databases [68]. Importing and exporting PostgreSQL data is handled by predefined QGIS functions. Once the data is displayed on the map, the user has the possibility to apply different types of symbols: font

markers, Scalable Vector Graphics (SVG) markers, line decoration, polygon fill, gradients, etc.

In some cases there is the need of labeling the map features, for optimizing the search mechanisms. This way, QGIS provides smart labeling support for vector points, lines, and polygons layers. Because they are displayed as plain text, several placements and attributes settings could also be applied;

- Data conversion: the GDAL and OGR libraries [57] are integrated parts of the QGIS system that allow the conversion between raster and vector data types. The GDAL provides raster management tools to query, re-project, create contour layers, and merge different data formats. This library consists of a set of command lines instructions, and a graphical interface that exposes only the most popular operators. The OGR library offers read and write access to more than 60 vector data types, using command line instructions;

- Geographic projection support: QGIS allows users to setup global Coordinate Reference System (CRS) projections that apply on layers that do not provide a predefined one. All the CRS definitions are stored in SQLite database, and offers access to more than 2700 known projections types defined by European Petroleum Group (ESPG) and Institut Géographique National de France (IGNF).

  On the fly projection is also supported for both raster and vector data. If the data you own is not recognizable as a standard QGIS format, there is the possibility to specify a custom projection that is compliant with the PROJ.4 standards [63];

- OGC implementation: the Open Geospatial Consortium (OGC) [36] is an international organization that provides services for geospatial data processing. Currently, the QGIS offers support only for the Web Mapping Service (WMS), Web Feature Service (WFS), and Simple Features SQL (SFS). An interesting feature is the ability to search for available WMS servers, by using the geopole [67] frontend Application Programming Interface (API).

  The WMS server handles client requests and generates a raster data map, typically in Joint Photographic Experts Group (JPEG) or Portable Network Graphics (PNG) formats. The entire client - server communication is implemented as Representational State Transfer (REST) services.

  Another feature of the WMS standard is the possibility to visualize raster layers over interactive maps. This is a three-step process that begins by requesting the server capabilities and the list of the available layers. For the selected layer the user performs an extra call, in order to obtain detailed information about metadata, projection type, and data format. The last step consists in displaying the actual data. The WFS service has a similar behavior as WMS, but instead of displaying data to the users, they are able to download this data as JPEG or PNG images;

- Query, digitize, and edit data: the users have the possibility to search for data by implementing complex queries. Currently, the QGIS supports four types of databases: Microsoft Access, PostgreSQL, MySQL, and SQLite. An advanced feature of the QGIS software is the shapefile creation that allows users to create and edit vector features and the attributes attached to them.

  The editing process implies the configuration of the snapping tolerance and search radius. The snapping tolerance represents the distance used for searching the closest vertex (or segment) that can be connected with the current feature. The searching radius adopts

a similar concept, but is used when translating the feature instead of connecting it with other ones. Both these concepts work at pixel level and is important to set them with proper values, otherwise it will reach to the case of snapping the wrong vertices, or finding results that are not in the area of interest.

By default, QGIS loads the layers with a read-only property and does not offer any write privileges, avoiding this way the users misguided actions. The most common digitizing tools are related with: moving, deleting, copying, saving, adding new points, lines, or polygons features, and toggle editing;

- GRASS integration: one key aspect of the QGIS software is related with the access to the GRASS algorithms [57]. In order to create a new GRASS vector layer it is enough to use one of the QGIS's menu options, provide a proper name and start digitizing points, lines, and polygons features. The digitization toolbar offer the possibility to work directly with geometric primitives, and to specify for each of them an attribute that will be useful when querying the layer for specific information.

  The access to the GRASS list of functionalities is done by command line instructions or by graphical dialog boxes. The latest was intended for users that are not specialized in working with GRASS, and it has its drawbacks regarding the lack of expressiveness while processing spatial data is such way;

- Plugins support: the QGIS has been design to support predefined and external plugins that will allow an easy integration of new functionalities. The predefined modules are developed internally, by using the C++ and Python programming languages [67]. Currently, there are over 120 available plugins and the most important ones contain the following features: visualize and edit OpenStreet Map and Google Map data, database synchronization while performing offline editing, adding projection information to raster files, GRASS integration, tools for processing and visualizing GPS data, data analysis algorithms, and geo-processing tools provided by the fTools software package [10], etc.

QGIS provides simple, yet powerful tools to work with. Complex features (e.g. creating a thematic map, using querying capabilities) take a longer time for a regular user to understand, but once this learning process completes they become easy to use. One important development direction is represented by the ease in creating, editing, and deleting shapefiles' features. QGIS offers remarkable capabilities also for raster data management, including a variety of formats.

3.2.2 ERDAS Suite

Earth Resources Data Analysis System (ERDAS) is the world largest computer-based system for generating, managing, processing, and displaying geographic data. Its development started in 1978 as a commercial GIS platform that provides several tools (ERDAS Imagine, ERDAS ER Mapper, ERDAS Extensions for ArcGIS, GeoMedia, etc.) that are used in different Earth Science domains [69] [70]. From all these software platforms, the ERDAS Imagine will be described in a more detailed manner, because a comparative analysis can be performed with other GIS tools that are presented throughout this chapter.

ERDAS Imagine has several strong features against other software-like applications. The first one is related with the geometric correction, spectral analysis, classification and interpretation that are performed much faster on regular standalone machines. Secondly, this platform supports much more data formats then open source applications, which recommends it for

highly trained specialists that perform complex analysis in different GIS domains. It also contains a range of classification algorithms, change detection, support for ESRI ArcSDE, OGC standards, etc. The most important features provided by the ERDAS Imagine are listed below:

- Virtual GIS: useful when creating 3D interactive environments. It allows querying large databases with real time performance in order to display the data as a virtual environment, created by overlapping raster and vector layers. The performance is based on an optimized multi-resolution rendering algorithm that displays the farther objects at a lower resolution than the objects closer to the observer.

  In order to generate a 3D environment, ERDAS Virtual GIS uses Digital Elevation Map (DEM) data. This information represents the basis for the 3D terrain generation, but raster and vector data can also be used for environment enrichment (use of symbology to represent different features).

  All the features are displayed in a two-dimensional perspective view. It also offers support for 3D images by the use of anglyph, and requires no equipment rather than a pair of colored glasses [71]. There are several navigations modes in virtual GIS: position mode (pivots the image around the current user's position), target (the terrain gets pivoted around a target specified by the user), dashboard (performed with the help of two joysticks that allows the world navigation), and terrain (involves the ground-level movement);

- Imagine Vector: these capabilities are designed to provide a complete GIS set of libraries for raster and vector data processing, including data types conversions, re-projections, full editing tools, etc. These capabilities are divided into two levels: native (allows the querying and processing of vector attributes) and vector module (an external ERDAS Imagine add-on that allows the creation and editing of vector layers, together with their conversion to different data formats).

  When the data is loaded, the user is able to edit it, by assigning visual symbols to certain map layer features. Translating the points, lines, and polygon shapes is also possible within the Imagine vector framework;

- Radar data suite: it offers accurate information that cannot be extracted using traditional photogrammetric methods. For this purpose it uses specialized tools for: radar data processing, data geo-rectification, calibrating radar images, surface changes or displacement within centimeters level of accuracy, etc. One of the main advantages in using radar data is their independency of sunlight, and that they may be collected in day or night time. This represents a significant improvement over satellite and aerial imagery, when dealing with mapping situations in areas with high cloud cover.

  The Radar mapping suite includes four types of modules: OrthoRadar, Radar Interpreter, StereoSAR DEM, and SAR Interferometer. The first one performs geo-referencing and ortho-rectification for Synthetic Aperture Radar (SAR) images. These results are highly accurate and do not include distortions due to radar imagery. The Radar Interpreter module is useful for pre-processing radar data (e.g. converting them into a standard format that could be further used by other software tools). The StereoSAR module produces advanced algorithms for extracting terrain height information, while the last module includes state of the art algorithms for processing SAR images, and offers a user-friendly interface to apply all these features [69].

  Radar images provide important information in a variety of GIS domain fields, and they are used in a large range of applications: defense and intelligence surveillance, oil exploration, agricultural and environmental monitoring, water quality, etc.;

- Imagine AutoSync: in most cases GIS images need to be geometrically corrected in order to be further processed. This correction must be very accurate otherwise the features misalignment could produce improper results. There are frequent situations that need to mosaic several independent images in order to generate a new one whose extent covers their reunion. Unfortunately, manual point measurements could not provide such accuracy, and instead the specialist has to use different interpolation techniques to generate the extra points that fill this gap.

  The AutoSync module overcomes this impediment by aligning two or more images of different types and resolutions, over the same areal extent. This method can be used to improve the registration between these data sets or to correlate new generated image to an existing geo-referenced image base. For this purpose the AutoSync module generates thousands of points (pixels) to gap the geographic space between these sets of images.

  It is an easy to use module, due to all of its features: wizard process guide, automatic point matching algorithm for mass points generation, automatic sensor detection, iterative model refinement and results rectifications, assures database consistency in respect with the new generated point's dataset, etc.;

- Imagine DeltaCue: it offers useful information about the differences between two images of the same geographic area, acquired at different times. Includes a large set of change detection algorithms and provides flexible tools to highlight the differences of these areas. Based on the analysis and interpretation of these processes, decision makers and domain field specialists can focus on the most important elements;

- Interoperability: extends the ERDAS Imagine's native vector format, and integrates additional GIS and CAD data. In order to offer interoperability support, existing data has to be provided in different formats. Using this module, simple conversions may be performed, together with data re-projection, attributes filed mapping, measurement units transformations, etc.

  Data transformers enable the generation of new information, based on a series of processes that are applied on pixels, vector features, or attributes. Grouping these processes into categories simplifies the user's efforts in performing such transformations. The most common categories include the following algorithms: collectors (useful for merging attributes of the collected data, update a subset with new information, etc.), filters (removing duplicates and detecting feature changes), geometric (implements overlap, clipping, and buffering operations), infrastructure (includes the translation processes between different module implementations), Web services (request for specific information by making the results available to the infrastructure transformers), etc.

3.2.3 GreenView system

It is a Grid based environmental application that can be used in various Earth Science domains. Even if for now it is limited to the refinement of surface and vegetation parameters in specific geographic locations, it could be easily extended to new research directions, like water and heavy metal detection or air pollution analysis and prediction [72], [73].

This platform provides algorithms for vegetation detection, satellite measurements calibration, and temperature analysis and prediction. In particular it is used in one case study (vegetation classification in the South Eastern European region) that requires the MODIS products as input data sets and involves three main algorithms: coarse to fine interpolation, fine

to coarse interpolation, and the ecosystem model calibration by computing the Gross Primary Production (GPP) value [74].

The *coarse to fine* algorithm requires as input one MODIS satellite image with a low spatial resolution and tries to generate an finer resolution output of the same area, by using different interpolation techniques. Both satellite images contain similar information: coordinates of the area (given as metric coordinates or as latitude and longitude values), the temperature values, vegetation indices values, etc.

It is necessary to have data fields of different environmental and biophysical variables on the same grid in order to efficiently use them in complex studies or models. There is a variety of interpolation methods available in the literature, developed specifically for certain applications. In order to choose the most appropriate one, issues like precision, computation resources, parallelization capabilities and others must be taken into account.

As the main goal of the process is to obtain temperature data samples of the fine resolution satellite image, a nonlinear cosine function based on distance weighted interpolation has been chosen. In this study for demonstration purposes, we used the 0.5°x 0.5°resolution climate dataset supposing a hypothetic climate change impact study performed by the end-user. The generated output is useful when performing a more detailed analysis over the vegetation growth in the SEE geographic area, due to the fact of its higher resolution.

The *fine to coarse interpolation* functionality from the GreenView application has the opposite effect as the previous mentioned method. This means that from a higher resolution temperature image, a lower resolution one is obtained, based on a pixel averaging approach in order to compute the final output.

The third algorithm of the SEE vegetation classification process is the *ecosystem model calibration* that is used to determine the presence or absence of vegetation over a specific geographical area, or to determine in what quantity it appears. For a better analysis process, the algorithm generates a pseudo-colored map where every pixel represents a Gross Primary Production (GPP) value that can be defined as the total amount of carbon taken up by the vegetation via photosynthesis during a specific time interval.

The calibration process is based on multiple comparisons between the satellite and ground based measurements. In each step a new GPP output dataset is computed that is compared with the ground measurements based on a merit function. If the calibration is repeated n times, a number of n merit functions are obtained from which the minimum value is selected. This minimum value represents the best calibration that can be obtained by using this third GreenView algorithm.

The GreenView system relies on the gProcess [75] platform and the Grid infrastructure [76] in order to speed-up the entire execution process. The client-side contains the previous mentioned algorithms and a graphical interface that allows interactive access to their functionality. The gProcess platform is used at runtime, when the entire processing can be partitioned into smaller tasks that are submited and executed in parallel over different Grid machines (Figure 3.6).

The SEE vegetation classification use case requires a large input data set, in order to obtain the best classification results. Executing the algorithms on standalone infrastructures is not the most suitable solution because of the large overhead of the total processing time. Instead, the GreenView platform uses the Grid environment for parallel and distributed spatial data processing. Following this approach, the total execution time reduces significantly.

The interactive input data selection, the results visualization over dynamic maps, online editing of the calibration parameters, the Grid based execution management are only some of the main features provided by the GreenView platform that increase the users satisfaction and the system usability.

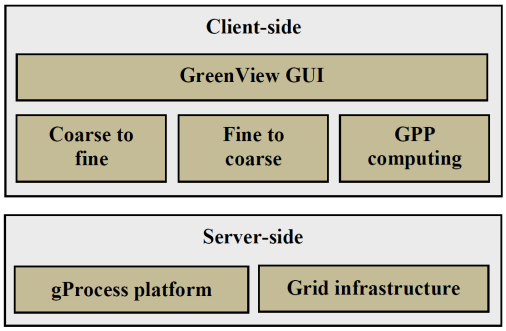


Figura 3.6: GreenView system related architecture

3.2.4 GreenLand platform

GreenLand is a Grid based software application that operates in the Geographic Information System domain, and it is used for geospatial data management and analysis, satellite image processing, graphics/maps generation, spatial data modeling and visualization, etc. In particular this application offers support for two major case studies: Black Sea catchment hydrologic modeling and the land cover/land use analysis of the Istanbul geographic area [77] [78]. [79]. Part of this research was performed within the FP7 enviroGRIDS project [80], funded by the European Commission.

By geospatial data we mean raster inputs such as satellite images in different formats (e.g. MODIS, Landsat, Aster, etc.) and vector inputs (e.g. ESRI shapefiles). These data are retrieved from remote repositories or from the user's local machine and converted into a GeoTIFF internal representation.

The GreenLand platform is developed based on the methodology proposed within the thesis and tries to provide solutions to the issues encountered in the previous mentioned spatial data related applications. This section generally describes the GreenLand features and characteristics, while the following chapters give more details about their theoretical and practical implementation.

Modeling large scale environmental use case scenarios is most of the times a challenging task, due to the multitude of conditions, restrictions, and algorithms that need to interconnect in order to provide the desired output. Regarding this aspect, the new GIS applications try to provide advanced interaction techniques that facilitate the end-user work and increase the usability of the entire platform.

In order to overcome these issues, the adopted solution consists in representing the entire use case as a workflow, where each node identifies one of the algorithms (functions) of the main process. The uni-directional edges of the graph specify the interaction between these algorithms and how they communicate in order to generate the output results.

The new research directions from various Earth Science domains impose new requirements regarding the spatial data computation speed-up and the gain degree of the user satisfaction. In order to obtain better execution times, the complex use case studies are partitioned into smaller tasks that are interconnected based on a certain pattern that corresponds to the natural language description of the use case. The manual identification of the relationships between these tasks slows down the entire development process. The GreenLand platform proposes a solution based on a semi-automated method that provides interactive techniques for tasks description and relationship definition.

Even though the complex use case development is recommended to be performed by domain

field specialists, certain variations of these models can also be created by novice users. Regarding this aspect it is worth mentioning that the GreenLand platform addresses several categories of users: regular or non-expert persons (they do not have solid knowledge about the Earth Science domains and are more interested in visualizing the output results for educational or informational purposes), persons from the urban planning sector (they focus on describing and analyzing the experiments that have a direct impact to the assets they manage), data providers, GIS domain field specialists, etc.

Among the previous mentioned features, the GreenLand platform provides an optimized graphical user interface that targets to increase the system usability and users satisfaction. In this context, the optimization term takes into account the accomplishment of user-tasks in a shorter amount of time than usual and with less effort from the user point of view. These features provided by the GreenLand application can be grouped in three main categories:

1. **Interactive development of the use case studies**: This platform offers support in developing complex scenarios that usually: simulate natural phenomena, model hydrological catchments, classify characteristics of certain geographic areas, etc. The workflow-based representation proves to be the best solution for describing such experiments, where each node identifies an algorithm (function or method) while the uni-directional edges specify the relationships between these nodes. The graph acyclicity is another important characteristic that prevents the system from going into infinite looping when executing the scenarios developed by the users.

   This approach simplifies the user-tasks accomplishment by reducing the entire process to interactive specification of the items of the graph, instead of hand-writing them in some markup languages (XML for example). In this case the system can automatically apply certain validation rules against all these items and perform proper adjustments, without the user interference. Let's consider for example the case when the user tries to connect two independent nodes. After the output of the first item is specified, the system can automatically filter the rest of the elements (by applying the input-output type match validator) and display to the user only those nodes that are potential candidates.

   Some of the previous mentioned platforms (e.g. Sextante, Quantum GIS) are also using the XML representation for describing the GIS use cases, but do not provide enough flexibility when dealing with large scenarios development (e.g. the Sextante tool does not allow the creation of a sub-workflow within a node of the graph). The manual update of this format is a hard task to accomplish and it is recommended only for highly trained computer science specialists.

   The GreenLand solution is based on the same XML representation concept, but instead it provides an interactive development environment (called WorkflowEditor) that can be access by any categories of users, regardless of their domain of expertise. Instead of manually updating the XML file, this dynamic solution allows the users to position and move the workflow's items with the help of the mouse device. This approach reduces significantly the number of errors encountered when working directly with the XML language and diminishes the time required by the user to complete the description of the scenarios.

   The following list contains the most important four constraints used by the GreenLand platform in order to validate the workflows description:

   - Nodes overlap: there should be a minimum distance (d) between each two adjacent nodes. If the user tries to position a node nearby another item that does not comply with this property, the current node is automatically repositioned in the correct region;

- Nodes interconnection: each node of the workflow encapsulates an algorithm (function or method) that contains multiple input and output variables that correspond to the inputs and outputs of the node item. The interconnection process is available only between the output and the input of two independent nodes that have the same data types. The link between these elements is done by uni-directional edges that also have the role of describing the entire sequence flow of the general use case scenario;
- Graph acyclicity: the mathematical definition for the cyclic graphs assumes that there is at least one sequence of nodes (that partial of fully describe the entire workflow) where the first and the last item are identical. The graphs that have this property can easily introduce at runtime infinite looping. By taking into account this remark, the solution adopted by the GreenLand platform is to use only acyclic workflows;
- Non-empty graph: it assures that the workflow contains at least one node;

2. **Internal representation of the workflows**: The XML representation of the workflow is different from the technique used to generate the internal data model. This structure relies on database attributes that allow recurrence calls and multiple imbrications levels.

   After the description of the workflow, it becomes available for execution over the Grid infrastructure. Due to the various combinations between these workflows (e.g. same graph processed multiple times, the embed of one workflow within another one, etc.) it was necessary to develop a proprietary data model that is flexible enough to support these structures.

   Taking into account all these aspects a new solution was adopted that simplifies the process of specifying the way of combining the workflows, their instantiation with different input data sets, the execution feedback display, and the interpretation and the visualization of the results. This solution is based on the concept of virtual container (called project) that stores all the essential information that are required throughout the entire user session;

3. **Spatial data management, execution, and visualization**: The GreenLand platform offers support for three types of spatial data: satellite images, aerial images, and field measurements. The main characteristics of the satellite images are that they have a wide coverage range of the geographic areas, with the temporal resolution between 1 and 8 days. They are extremely useful in natural phenomena modeling. On the other hand, the aerial data have a better resolution (1 - 50 meters) and provide more accurate information than the satellite sensors. The values obtained for the third type of data are measured near the Earth's surface and they provide information in dense areas where the artificial sensors cannot reach.

   The spatial data management process takes into account information sharing, local and remote data handling, dynamic techniques for instantiating the workflows with the proper input data set, collaborative analysis and visualization of the results, etc. The following paragraphs offer more details about these features implemented at the GreenLand platform level:

   (a) Data sharing: the spatial data storage usually involves specialized repositories that implement proprietary access policies. Because there are several such remote storages, the users from Earth Science domains have to adhere to all these rules which slow down the entire data sharing process.

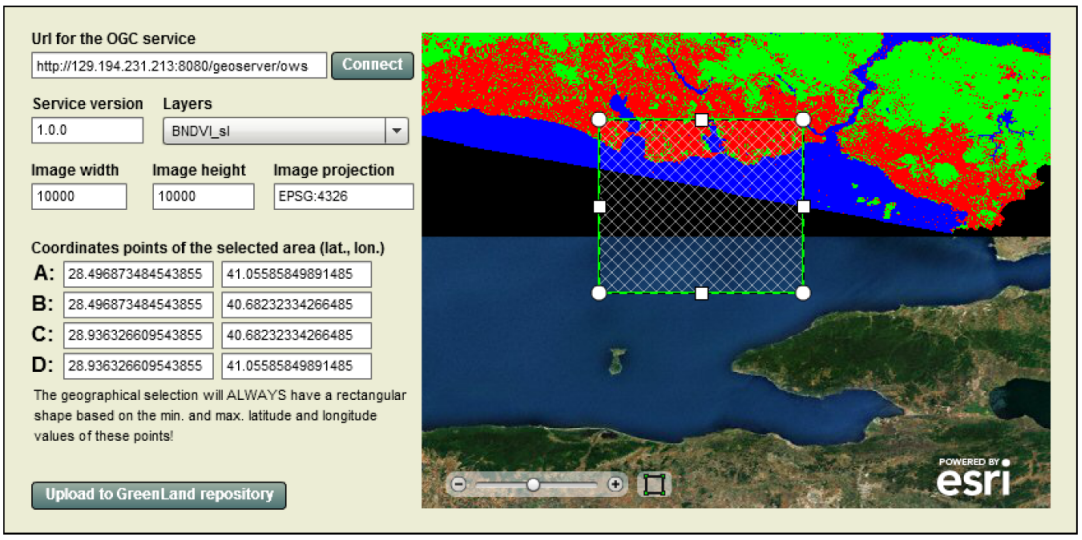


Figura 3.7: Spatial data visualization and extraction within the GreenLand platform

One of the possible solutions is to use standard access to all data repositories, regardless of their type and location. The OGC provides an easy to use set of guidelines about accessing, querying, fetching, and executing remote spatial data.

The GreenLand platform adheres to the latest OGC directives and implements specific interaction techniques that allow users to perform these operations. Even though the solution will be fully described in the following chapters, it is worth mentioning the GreenLand approach used for relevant spatial data extraction. This process is highly interactive (Figure 3.7) and requires from the user only the specification of a rectangular geographic area ($ABCD$) while the system automatically performs the adequate OGC algorithms in order to extract the required information. The area specification is a three-step process that implies:

- The starting point (A) selection by using the left mouse click;
- The end point (D) identification that is performed by moving the mouse cursor over the interactive map. Throughout this entire process the geographic section is marked as a green rectangle;
- The final representation of the area of interest, by releasing the left mouse button. If this selection is not properly configured, the system automatically applies several filters in order to obtain its correct shape;

(b) Input data set specification process: there are three mechanisms that allow the upload of external files into the GreenLand platform: file by file, OGC based data extraction, and large dataset upload by using the FTP protocol. Using these three techniques the system provides support for a large variety of user requirements, related to the specification of the input data.

The first method is useful for scenarios that contain a relatively small number of inputs (e.g. at most 5 items). The second mechanism can be used in similar situations and provides functions for remote data extraction.

Simplifying the user-related actions of assigning valid inputs to the complex scenarios is one of the graphical interface optimization that is taken into account by the Gre-

enLand platform. There are cases (e.g. natural phenomena prediction) that require a large number of input dataset and involve some time-consuming user operations. The solution proposed within this paper is based on the OGC and FTP automatic data extraction features that proved to be useful in reducing the duration and the complexity of these tasks and improving the overall system usability.

The *Mosaic12* use case (see Chapter 9) is adequate in validating the automatic data extraction from various remote repositories. This scenario predicts the water quality and quantity within the Black Sea catchment and requires more than 500GB of relevant data that covers this entire geographic area. The manual specification of the inputs is time consuming and slows down the entire use case development. Instead, the GreenLand platform requires from the user only some basic information (the name of the satellite images bands and the processing time interval) while the system automatically downloads the data and assigns them to the corresponding Grid processes;

(c) Results visualization: once the output results become available, the users are interested in analyzing them in order to obtain relevant feedback. Their integration as layers over the interactive maps represents a good indicator related to the system usability and the user's satisfaction degree. This approach increases the visualization process quality and diminishes the analysis complexity by avoiding the usage of intermediate analysis tools.

The integration of the OGC standard within the spatial data visualization process allows users to share and analyze data in different environments without the need to learn new interaction techniques. The GreenLand platform implements the guidelines proposed by this standard and provides data level interoperability support with external platforms.

3.3 Conclusions

The new Earth Science research directions recommend the sharing of data and the usage of common interaction techniques for accessing the remote information. This way, new standard guidelines are required that allow interoperability at data and platforms level.

Even though the next GIS applications are using standards (OGC for example) in order to overcome these issues, they address the small and medium complexity use case studies, whose execution does not require powerful computation resources. This category of applications was highlighted in the previous sections and includes: Sextante [44], uDIG [51], GRASS [59], etc.

These platforms provide a large number of algorithms used for spatial data analysis and visualization. On the other hand all of the cited tools provide support for OGC related services useful in accessing, processing, and visualizing various types of spatial data. Filling the gap between these issues represents the main goal of this thesis that proposes a methodology for developing spatial data processing applications. The theoretical concepts are put into practice within the GreenLand platform, a system that differentiates from the previous mentioned applications through:

- the ability to process complex use case scenarios that cannot be executed on standalone environments;
- the processing optimization by using parallel and distributed data execution over the Grid infrastructure;

- a better data security management, by using the Grid certificates for spatial data access, execution, and visualization [81];
- the implementation of a monitor mechanism that provides feedback about the execution status and allows users to interfere with the current on-going processing;
- a more complete workflow editor (in comparison with Sextante) that supports: the sub-workflow concept (embedding a workflow as node within another graph), collaborative description of the workflows, syntactic validators, auto-layout algorithms, etc.

This chapter identifies and describes the most important platforms that are used for managing, processing, and visualizing various types of spatial data. Among these applications there are several tools [82], [83], [84] that use proprietary methods for spatial data analysis and interpretation that also work on distributed environments (Grid and Cloud in particular). These semi-opened platforms provide smart solutions when they are used in isolated environments that do not require information exchange or automatic mechanisms for remote data retrieval.

The system described in [85] highlights the theoretical concepts for executing the geospatial algorithms on different backend infrastructures. In order to validate these concepts, a NDVI experiment was conducted for measuring the vegetation growth in several geographic areas. Depending on the spatial data quantity (specified as input of this experiment) the system was able to redirect the execution to standalone or Grid infrastructures.

Other platforms [86] facilitate the standardized Cloud-based processing of spatial data, but the results analysis and visualization are performed in the traditional manner. This limits the system only to the data execution process and requires from the users' communities additional tools in order to perform the results interpretation. The GreenLand platform can be classified in the same category with the last two mentioned systems. It uses the Grid infrastructure for parallel and distributed execution of the geospatial algorithms, providing in the same time support for the majority of the OGC related services. The most important features that differentiate the GreenLand platform from these applications are:

- standardized support for all stages (description, execution, monitor, analysis, and visualization) of a use case development process;
- integrates an interactive environment for scenarios description, avoiding this way syntactic errors that may occur when presenting the interactions between the algorithms in certain text-based editors;
- automatic data fetching mechanisms from various remote repositories. Once this information is available, the system (without the user interference) assigns it at runtime to the corresponding processes;
- the ability to easily extend to other domains [87], beside the Earth Science related ones. This is possible through the workflow-based description of the use cases and to the modular development of the application (a fresh installation of the platform may be performed by combining these modules in various ways);
- the interaction techniques are not focused on a single user category. Instead they address various categories of users, regardless of their background knowledge in the Earth Science or computer science domains.

Chapter 4

A general comparison related to the classical software development methodology

In the last years there was a powerful migration towards Web based Earth Science applications. The main advantage of this approach is the platform availability to the users, regardless of the physical machines or locations. In general, these types of systems adopt the client-server architecture, and the communication between the users' requests and the backend functions is provided through Web services. The latest OGC [36] standards encourage the usage of Web services for spatial data access, processing, and visualization.

Earth Science related platforms resemble the traditional software applications only to a certain point, meaning that they can be developed by following the same methodological steps. Data acquisition, user demands, the variety of data formats that have to work together, the visualization techniques involved when analyzing the results, the standards implementation, or improvement of the user interaction techniques are only a few of the challenges of the Earth Science applications over traditional software systems.

There is a big difference when implementing systems that process large or small volume of data. In the first case, there is the need to make use of the execution and storage capabilities of the parallel and distributed infrastructures (e.g. Grid, Cloud, computer clusters, or mainframe machines), while the applications that process less complex algorithms are not that computing intensive.

Most of today's Earth Science related applications focus only on processing small to medium amount of data. For this purpose, the usage of standalone (or multi-core) machines is a feasible solution. However, this thesis analyzes a general methodology that allows the development of Web based applications from the Earth Science domains that also function over distributed infrastructures, Grid in particular.

At this moment there are several platforms that offer these types of functionalities, but they could only be used by highly-trained specialists. So, another key aspect that is described throughout this paper is the possibility to allow non-technical persons to use these applications as easy as any types of specialists.

4.1 Classical methodology for software applications development

Regarding all these aspects, the following sections describe the classical methodology used for developing the software applications that is based on the Waterfall [88] and Agile techniques [89].

The Waterfall model represents a sequential approach to software development, where each sequence is implemented by a specific team. Until one iteration finishes, the next one could not start. Due to this limitation, the entire project development takes a longer time to complete, compared to the Agile methodologies. The Waterfall model is a 5-step process that involves: the analysis of the users' requirements, system design, implementation, validation, and maintenance.

The Agile based software development can be described as a group of iterative and incremental development methods that involves several teams of self-motivated and self-organized members (usually 7-10 people) who plan, code, execute, test, and deliver their work to the clients. Their work is divided into weekly or monthly cycles (called sprints) that are evaluated at the end of this period. There are several benefits of using this basic approach, such as: reducing overheads, keeping the documentation and meetings as simple as possible, dividing the products into smaller and independent modules, better teams monitoring, easier integration of the new users' requirements, etc.

The Agile methodologies are based on the following principles: customer satisfaction by delivering the intermediate product versions as soon as possible (weekly or monthly), late requirements change acceptance, project modularization, close communication between users and developers, documentation overhead reduction, and simplicity.

The Extreme Programming (XP) and Scrum are two of the most known Agile development methodologies. The first one is designed to reduce meetings and documentation overheads, and keep things as simple as possible. Scrum methodology is used to manage complex software products, using iterative and incremental approaches. Each iteration (or sprint) is one to four weeks long, and provides rapid product delivery and allows the users' requirements change. There are several roles within the scrum methodology, such as: product owner, scrum master, and development teams [89].

The main advantage of the Agile techniques over the Waterfall methodology consists in system modularization and parallel development, without the need to wait for the previous stage to be completed, before starting the next one. The following paragraphs describe other benefits of Agile methodology over the Waterfall method:

- The Waterfall method does not allow inverse iterative steps, meaning that once a stage is completed there is no possibility to go to the previous one. This is a huge impediment when customers require features redesign or the implementation of new issues. On the other hand, Agile methods deliver product versions before a cycle completes. This way it is much easier to integrate the new users' requirements;

- Early delivering of modules allows better problems management, reducing the overall costs and efforts;

- Due to the project modularization technique, the developing teams work on smaller functionalities, rather than on stage based project modules, like in the Waterfall method. Such an approach minimizes the development delays, because modules that assure partial customer specifications are also accepted.

The following sections describe the steps of the methodology used for software applications development. It is based on the Waterfall and Agile methods, but generalized for Web based systems that perform data processing over distributed and standalone infrastructures (Figure 4.1).

4.1.1 Requirements analysis

When developing software applications it is mandatory to know their domain of applicability, and to create several use cases that describe the interaction between actors, systems, and platforms. Before collecting the users' requirements, a general overview of the application should be established. After that it is recommended to build multiple use cases, in order to ease the identification and documentation of later specifications required by the users. A use case is initiated by a user with particular goals in mind, and usually:

- Represents the basic technique for collecting and describing the users' requirements;
- Allows developers and users to adopt a common description language related to the problems they want to solve.

Each use case should be described in an easy to understand language, in such way that it can be read by both developers and customers. A more technical document should also be created (for the developers who are involved in the implementation process) that relates to aspects like:

- Use case name, description, and general purposes;
- Basic events flow, indicating in each step what are the requirements and the constraints;
- Alternative events flow;
- Connections with other use cases.

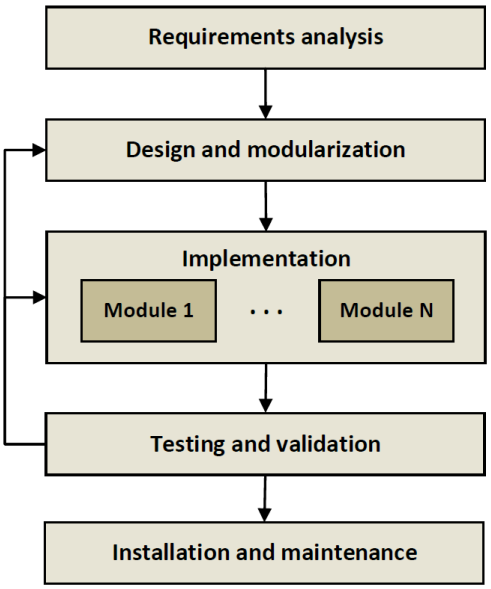


Figura 4.1: Classical methodology for software applications development

Depending on the functionalities provided by the system, there are several users' categories (e.g. data providers, regular users, etc.) that interfere with the software product. It is important to correctly identify from the beginning these categories, because each of them has specific requirements that need to be addressed by the system.

For software applications, the requirements collection process can be performed through questionnaires that contain general information about the system that is going to be implemented, and help the developers in better understanding the users' needs. On the other hand, it is recommended to apply other collection techniques, in order to obtain more detailed information (e.g. interviews, direct observations, etc.). The most common issues encountered within checklists documents are related to: spatial and temporal aspects, multilingual and cultural issues, data interoperability, data conversion, metadata availability, data collection, data storage, data processing and analysis, etc.

The requirements specification process should be complete and unambiguous, and has to provide useful information related to:

1. **Functional requirements**: describe the systems capabilities and the actions that must be provided to the users;
2. **Non-functional requirements**: imply the constraints imposed to the system (e.g. maximum time wait for a process to complete, the scalability factor regarding the number of users, etc.);
3. **Design directives**: represent the constraints imposed by the users related to the system architecture;
4. **Implementation directives**: same as design directives, but regarding the solutions used in the system implementation.

4.1.2 Design and modularization

The Agile methods propose a project partition into independent modules that provide specific functionalities. This stage of the development methodology describes the modularization issues and its conceptual architecture, and exemplifies the methods based on the Grid processing infrastructure, in order to provide concepts as close as possible to the Earth Science applications that process a large volume of data that requires parallel and distributed computing environments.

The only way for the user to access the application functionalities is by authenticating with a username and password at the Web application graphical interface level (Figure 4.2). The user actions at this level correspond to different server-side operations, based on Web and Grid services.

Depending on the application's complexity, the second block from Figure 4.2 exposes other services (e.g. data conversion mechanisms, algorithms management and execution, support for collaborative sessions, etc.) compliant with the application requirements. The significance of the mechanisms provided by the *Backend services* layer are described in the following paragraphs:

- Process decomposition tool: in order to execute complex algorithms over distributed infrastructures they have to be divided into atomic components, called tasks. This is a mandatory step in order to benefit of the parallel execution capabilities provided by the Grid infrastructure;

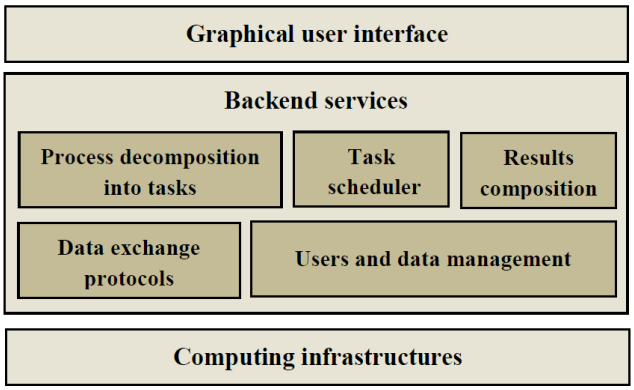


Figura 4.2: System related architecture

- Task scheduler: the Grid infrastructure only provides the distributed environment where all the processing takes place, but it does not schedule these tasks by default. This means that a proper scheduler tool has to be integrated within the application architecture;
- Each task is executed on a specific Grid worker node and generates a result that is transfer as input to other processes. These intermediate results are then combined in order to generate the final output. This is known as the results composition concept;
- Because an algorithm is divided into smaller processes, called tasks, there is the need to implement a proper data exchange protocol, in order to assure the communication between the Grid nodes that perform the execution;
- Because users have to identify themselves within the system, it is useful to provide services for users management (add and delete users, update their profile, etc.). Similar management tools should also be available for handling the internal data resources.

The Grid infrastructure represents the distributed environment where the tasks execution takes place. It communicates directly (or throughout the Grid services level) with different data repositories. This type of communication allows the storage of intermediate results in shared memory locations, called Storage Element (SE). An extra module could be added to this infrastructure in order to redirect the process execution towards the Grid infrastructure or towards other platforms (e.g. Cloud, clusters, or standalone machines), depending on the process complexity.

Based on the architecture presented in Figure 4.2, the following modules can be identified: graphical user interface, task scheduler, protocols implementation, user management, data management, Grid based execution platform, monitoring the Grid processing, result analysis and interpretation, and data storage and conversion. These general modules could be further partitioned into sub-modules, and assigned as separate tasks to the development teams.

In the design phase, the project manager and team leaders agree upon the technologies that are going to be used throughout the system implementation. If additional tools are needed they are embedded at this stage of development. The usage of standards services is also important for allowing the application to communicate with other systems. Due to this fact, the project leader should also agree upon the standards they need to implement in order to be compliant with the users' requirements.

4.1.3 Implementation

After the requirements analysis phase, the project manager estimates the human, financial, and hardware efforts that are needed in order to successfully complete the system development. The technical teams are formed, and parts of their members are involved in the system design stage. This way the team leader is aware of the requirements of the modules that are going to be assigned to each team, and facilitates the understanding of the problems that need to be solved.

Each team is assigned with different modules (identified in the previous stage). Figure 4.1 describes the methodological development steps related to the overall software platform, while Figure 4.3 presents insights within the developer's team, who implement a specific software module.

The Agile methods use sprints for module implementation. Each sprint is one up to four weeks long, and at the end of this period a preliminary version is delivered to the clients. If the users are satisfied, this version represents the final module release, otherwise a new iteration begins.

The team leader (who assisted at the general discussion in the first two phases) describes

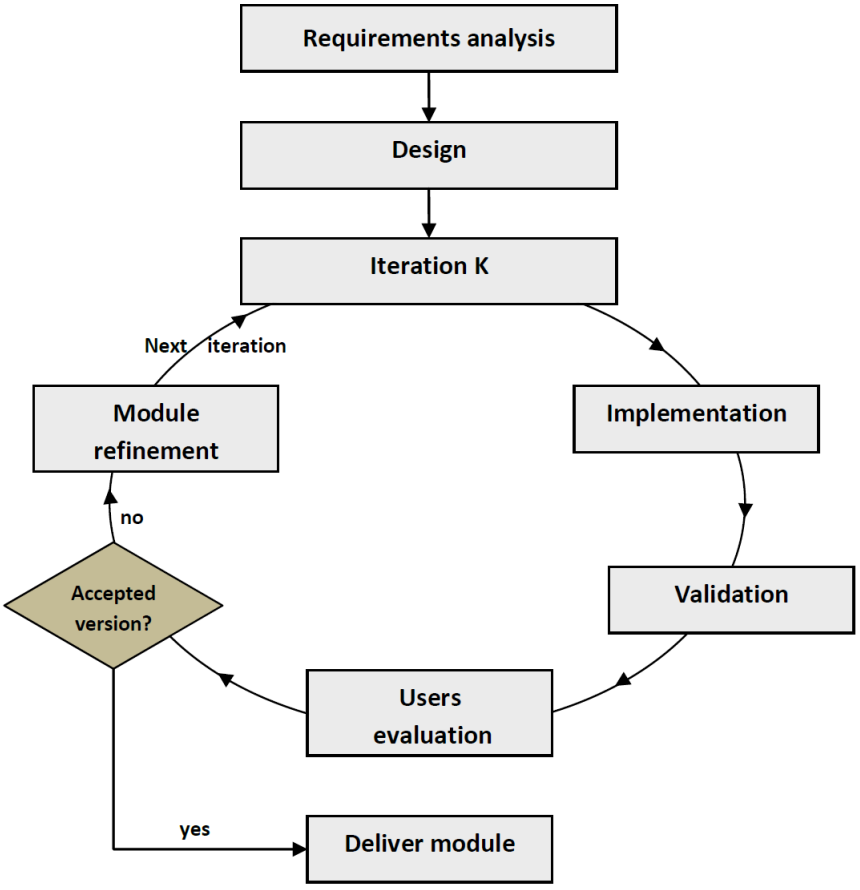


Figura 4.3: The K iteration of the development process, related to a specific module implementation based on Agile methodology

the problem to the rest of the team. A quick analysis of the requirements is necessary at the beginning, then the team leader together with the development team architect build the module architecture. The next phases embed an unknown number of iterations for implementing and delivering the current module.

These sections describe the development methodology for the software applications that run over the Grid infrastructure. Due to this issue, the technical team must be aware of the problems raised by such an approach, and try to minimize its implication by: creating a data model that can be processed in a parallel and distributed manner, integrating a monitoring mechanism that provide feedback to the end-user about the status of the Grid data processing, implementing useful interaction techniques that enable the results analysis, provide easy to use and easy to learn graphical modules, etc.

Because each module is developed by different teams, there is the need to integrate them within the software application and to offer to the users the product as a whole. If the modules, generated in second development stage, were independent one from another, integrating them will not raise difficulties.

The main assumption proposed by this methodology is that if the independent modules work correctly, the final product should also accomplish the users' initial requirements. Due to this issue, a good communication mechanism should exist between the development teams. This way, they can share their experiences, co-work when modules are not that well individualized, and perform different validation tests.

4.1.4 Testing and validation

The validation phase described in Figure 4.3 is related to a single module that is currently in the implementation process. On the other hand, the testing and validation step proposed in Figure 4.1 is performed at the application level. This means that the Agile methods recommend a permanent validation that takes place during the entire implementation process with the goal of identifying the potential errors as quickly as possible, and to fix them accordingly.

Throughout this phase, the application functionalities are tested with real data, and with the support of the domain field specialists. In some cases the data acquisition is hard to accomplish, but even though it is mandatory to have a minimum amount of information, in order to validate the system functionality and to assure that the users' actions will generate a correct result.

This is a two-step process that starts with the implication of the testing team. Even though it resembles with the Waterfall model approach, it is very different from it. The majority of the problems are identified in the implementation process, due to the early sprints delivered by the development teams. Any change of requirements are included within this stage. This leads to a small number of functional errors that will propagate up to the testing and validation methodological step.

The second validation is performed by the system users, by comparing the functionalities provided by the application with their initial requirements, documented in the early development stages. If they agree with the current version, this represents the final product, otherwise it is required to go back to the implementation stage and add or update the current features proposed by the users.

4.1.5 Installation and maintenance

The last stage integrated within the software applications development methodology is related to system maintenance. This step occurs after the installation process, and includes platform improvements by altering different modules and internal properties. Generally speaking, each

maintenance cycle could be described as a new product release, because it is performed when new users' requirements arose or when the system performance needs to be improved.

4.2 Applications development methodology for processing and visualizing spatial data

There is a tight correlation between the software application development methods and the methodology proposed within this thesis (Figure 4.4 highlights these similarities). As can be seen, one module from Figure 4.4a corresponds to one or more phases of the methodology presented in Figure 4.4b, while some steps are shared between consecutive phases of the software development techniques. The following paragraphs describe in more details the significance of the concepts introduced during this study.

4.2.1 Context definition and requirements analysis

This first step is about defining the context of the problem that needs to be solved by the use of geospatial algorithms. The preliminary sets of requirements are important, due to the fact

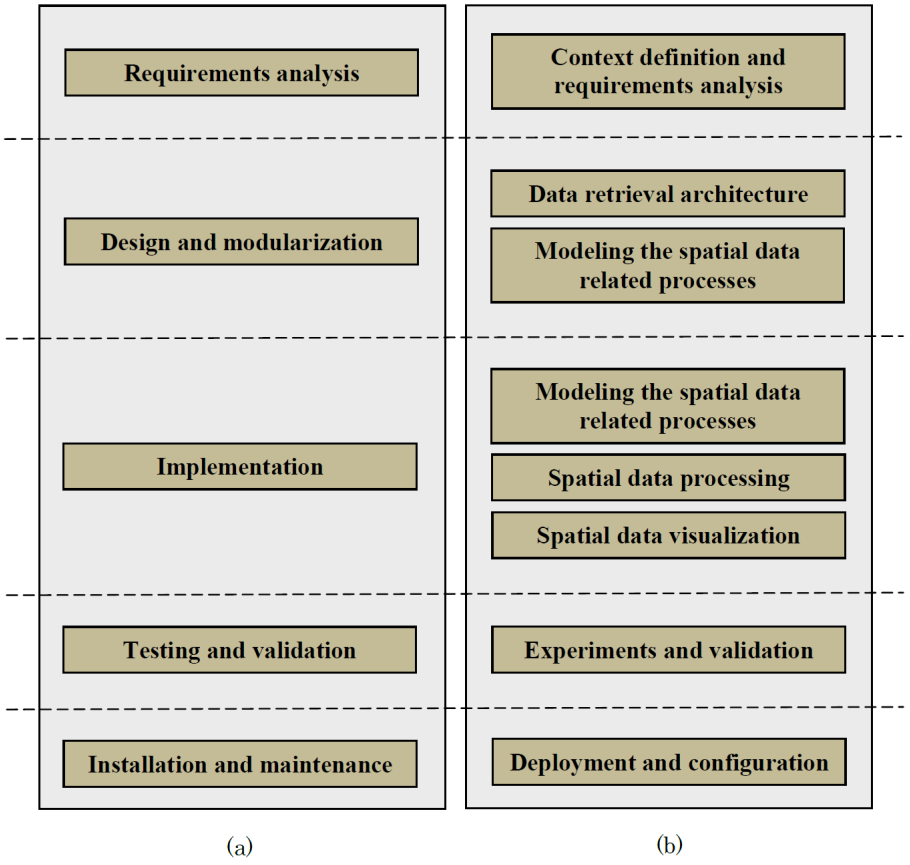


Figura 4.4: Parallel between the methodologies used for developing (a) software products and (b) spatial data related applications

that they will provide policies, constraints, and guidelines for the next development phases.

The proposed methodology is not addressing only the Earth Science domains, but it is flexible enough in order to be extended to other disciplines (e.g. physics). This is why the correct identification of the activity domain is important, due to the fact that each such filed has its one particular issues that have to be taken into account. For example the hydrological experiments require a large volume of satellite measurements that are not that easy to collect, while the identification of a new substance can be performed based on theoretical formulas that do not require additional tools for data retrieval.

The requirements specification should be performed into a common language that can be understood by the Earth Science specialists that define the problem, as well as by the computer scientists that are involved in the software development process. Usually the requirements are written in natural language, and periodical meetings are established with the purpose of clarifying any questions that arose from both sides.

The last issue related to this methodological step is the identification of the categories of users that are going to experiment the final products. For the Earth Science related domains, there are several types of users that can be taken into account:

- Filed specialists: have the role of formulating the requirements of the problem they want to solve. They are involved throughout the entire development process, because they are qualified to answer any potential questions and to reformulate (if needed) the initial requirements;
- Data providers: most of the times identified as governmental agencies or private companies that provide data that are used in testing the preliminary versions of the applications. When needed they are also able to support information for real use case studies;
- Decision makers: they are involved only in those actions that provide certain prediction results and require the identification of rules, policies, and decisions about the assets they manage;
- Regular users: they use the experimental results in educational or informative purposes, and usually they are not involved in the use case development process.

Depending on the types of users, this methodological step should also establish the interaction techniques that best fit onto the users' needs. Features like: interactive modeling of the use cases, identifying the results presentation techniques, simplifying the data retrieval procedure in terms of users-actions, etc. should be taken into account and provide specific guidelines for developing them.

4.2.2 Data retrieval architecture

The types of experiments influence the architectural design of the entire application. The previous mentioned example highlighted the need of developing a data retrieval mechanism from remote repositories, in order to be able to perform the hydrological use case scenario. We've assign this step to the *Design and modularization* item (Figure 4.4) due to the fact that it has a major influence in modeling the entire software platform.

The initial set of requirements should allow an estimation about the data quantity that is required in conducting the experiment, and also their type and storage location. It is important to know if we need digital data (e.g. satellite images) or hand written information that are harder to collect and use. The next step is to further classify the requested data, based on their spatial resolution and the information contained within each layer (e.g. MODIS satellite

images with 36 bands, Landsat with 7 layers, etc.). The temporal resolution is also important, due to the fact that it determines the frequency at which the experiment is repeated.

For situations that involve a small amount of data, they can be stored on the users' local machines and uploaded manually into the system. But in general, for the applications that require a large volume of data, this information should be automatically retrieved from specialized repositories that are updated periodically with the latest information. The spatial data applications that are developed based on this methodology should provide several options for instantiating the geospatial algorithms with:

- Remote data that are collected automatically or manually from specialized storages. In this case, the modeling of the processes should be carefully performed and include the possibility of extracting at runtime the relevant information;
- Local data that are uploaded manually by the users. This method is recommended for experiments that require a small number of inputs.

The concept of data sharing is very important, especially in the Earth Science related domains. This methodology promotes the usage of standards (e.g. OGC [36]) in order to achieve this goal and to assure a constant interoperability between platforms that activate in the same domains. A more detailed description of these two concepts is going to be presented in the following chapters of this thesis.

4.2.3 Modeling the spatial data related processes

The modeling of the spatial data processes is divided between two consecutive phases (*Design and modularization* and *Implementation*) of the classical software application development methodology. In the first place, this step has the role of identifying the geospatial algorithms (methods, formula) that are involved in solving the previous defined problem. In cases of complex experiments, it is recommended that these modules are less interconnected, in order to facilitate their parallel and distributed execution.

The proposed methodology uses the concepts of basic operators and complex workflows for describing the experiment's functionality. These concepts are inspired from the mathematical graphs theory and allow the data flow description and organization as a graph, where each node encapsulates the behavior of a specific geospatial algorithm (as basic operator) or groups of algorithms (as complex workflows). The connections between these items are specified as uni-directional edges that also have the role of transferring the output results of one item as input to other elements.

The modeling process should be performed based on the concepts of basic operators and complex workflows, meaning that the entire use case scenario should be partitioned into atomic entities that cannot be divided into smaller sub-problems. The functionality of these entities should be mapped as basic operators. Let's consider the case where we need to add a constant to all the pixels within a Landsat satellite image, and then interpolate the result in order to obtain a finer resolution image. This experiment can be described by the use of two basic operators:

- The first one (*OP1*) takes two inputs (the Landsat satellite image and the numerical constant) and provides the result obtained by adding the constant to each inner value of the image;
- The second basic operator (*OP2*) takes as input the previous result and applies the interpolation algorithm in order to generate the final output of the experiment.

Without adding the proper connections between the identified operators, they remain in an isolated environment (Figure 4.5a), where the results cannot be reused as the original requirements demand. To continue the previous example, the connection should be from *OP1* to *OP2*, like in Figure 4.5b.

This way the entire use case scenario can be translated into a graph structure. For complex experiments, situations of grouping the modules into a multi-level hierarchy may be encountered, and can be solved through the use of workflows. This entity allows the definition of a set of basic operators into a single structure that can be shared across various use case scenarios.

If we add to the previous example the feature of creating classes of colors for the interpolated image (e.g. green nuance for the values between [-1; -0.33]) it is best to encode the first experiment as *W1* workflow and to reused it in this second scenario (as exemplified in Figure 4.6). When multiple workflows are used to model the same experiment, the obtained structure is called hyper-workflow.

The step of modeling the spatial data related processes also corresponds to the *Implementation* stage from the classical software application development (Figure 4.4). This means that the abstract definition of the workflow needs to be encoded into a digital format that can be executed over different computing infrastructures. This stage involves its transcription into a high-level programming language, by taking into account all the concepts defined earlier.

The XML internal format is recommended, due to the fact that it allows the definition of entities closely related to the graph notion. In cases of complex experiments, the size of these files is large and can be managed only by computer scientists, but usually these scenarios are developed by the Earth Science domain field specialists. The proposed solution is to provide interactive techniques that simplify the development process to the level of specifying the operators and workflows in a visual manner, avoiding this way the direct management of the text-based files.

4.2.4 Spatial data processing

The title of this thesis contains the keyword methodologies, but until now we talked only about one methodology that aims to provide guidelines for developing interactive applications that process various formats of spatial data. Three more methodologies are presented within this thesis:

1. The first one is related to the processing of spatial data that divides the entire process into 4 phases: pre-processing, actual data execution, monitor the status of the execution,

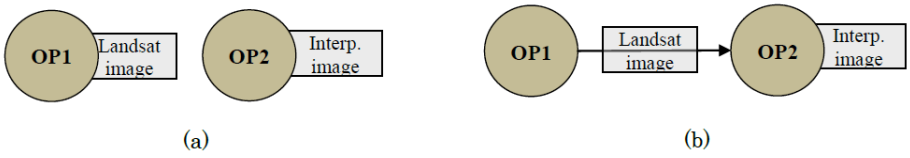


Figura 4.5: Modeling the spatial data related processes as basic operators

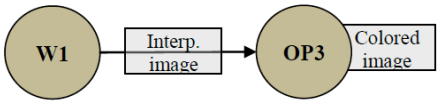


Figura 4.6: Modeling the spatial data related processes as workflows

and the results analysis and visualization. This is similar to the general methodology, but resized only to the level of spatial data execution;

2. The development of the basic operators is also based on 5 methodological steps. It starts with the abstract definition of the algorithm, and continues with the identification of the proper data resources, the formal description of the operator, its transcription into a high level programming language, and its execution over different infrastructures;

3. The third methodology is related to the process of mapping the theoretical concepts onto the features provided by the workflow concepts. This way we have a 4-steps methodology that starts with the description of the use case scenario in a natural language, and ends with the workflow processing, results analysis, interpretation, and visualization.

One of the main research directions of this thesis is the execution of a large volume of spatial data in a reasonable amount of time. This means that the geospatial algorithms should be processed on powerful machines that provide parallel and distributed capabilities (e.g. Grid, Cloud, clusters, etc.). On the other hand, the proposed methodology also addresses the small scale scenarios that are not so computing intensive.

In order to solve both issues, the proposed solution is based on the Grid and the standalone infrastructures. The experiments conducted throughout this thesis were manually redirected for execution to one of these two environments. Several studies were performed outside the scope of this thesis [90] in order to automatically find the switching point at which the Grid infrastructure performs better. These studies are based on determining the complexity of the experiment and comparing it with a threshold value, computed from statistical measurements. If the complexity is higher than this default value, the Grid infrastructure should be taken into account, otherwise the standalone environment is recommended.

The parallel and distributed data execution is a complex process, and involves several phases: identify the available computing machines, partition the entire workflow into tasks or groups of tasks, transfer the input dataset and the additional resources (e.g. external libraries), perform the local execution on each worker node, recompose the general output based on the results obtained from each task. The spatial data related applications should take into account all these stages and implement the proper scheduling tools that allow an optimal data execution.

In any processing it is important to know all the time the status of the execution, especially in applications that involve several categories of users. This methodology recommends the usage of an interactive monitor mechanism that is able to interpret the technical notifications and to translate them into more easy to understand messages. Offering the users the possibility to interfere in the overall execution process is also important. This is why they must be able to stop a process that takes too long to execute, or to restart it when needed.

One of the main objectives of this thesis is the improvement of resources sharing between the applications that use the Grid infrastructure. The OGC standard seems to provide the best solutions for invoking geospatial algorithms that are resident on external platforms. In the standalone environments the default solutions are working just fine, but when dealing with large data processing that takes a lot of time to complete, those methods do not provide the proper guidelines. This methodology recommends the extension of the WPS service to include a monitor operation that is able to periodically query the status of the execution.

4.2.5 Spatial data visualization

This step is about identifying the proper presentation techniques of the results obtained during the execution process. Based on the categories of users that are going to utilize these applications, the spatial data results should be visualized as: text files, dynamic charts, static images,

statistics, interactive maps enriched with the actual image result, 3D representation of objects, etc.

The Earth Science related applications usually exchange a large volume of spatial data, in various formats. The execution results can also be included into this category, and represent important resources for many scientific communities. Based on these aspects, the proposed methodology recommends the usage of standard services for improving the data visualization and analysis that also implies the concept of data sharing. The OGC standard provides the best methods for visualizing remote data, and retrieving them in a highly interactive manner.

The spatial data visualization process also involves the identification of a common set of interaction techniques that should be integrated onto multiple platforms, in order to simplify the user related actions in accessing, retrieving, visualizing, and analyzing data, regardless of their storage location. Otherwise, the users have to utilize multiple tools (that provide similar functionalities) with different interaction mechanisms. This way the users are required to learn all these techniques in order to be able to perform a complete visualization and analysis process.

4.2.6 Experiments and platform validation based on real and simulated use case studies

The concepts presented throughout this methodology should be validated with real or simulated data before launching the final version of the application. Actually, the testing process should take place at each stage of development, in order to identify the possible errors as quickly as possible.

The creation of testing scenarios is the main goal of this methodological step. The users that describe these experiments should have solid background knowledge about the Earth Science domains and about the functionalities of the application. The data sets used throughout these experiments should have (if possible) the same type and size as the information from the real scenarios.

The proposed interaction techniques for accessing, modeling, processing, and visualizing the spatial data should also be tested by taking into account the users' feedback, collected through questioners and usability metrics. The obtained results have an important role in improving these techniques and to make them available in future releases of the application.

4.2.7 Deployment and configuration

This is the last phase of the methodology used for developing the spatial data related applications, and it refers to the installation of the products on platforms with different configurations. It is recommended having a centralized repository of workflows (that contains algorithms only for the activity domains that they were developed for) and multiple distributed storages that provide solutions for other disciplines (this way the application can be easily extended to other domains).

The integration of functionalities from external platforms should be taken into account, and provide configuration files that can easily apply these features. The continuous development of new operators and system maintenance are also recommended by this methodology. This means the involvement of other Earth Science specialists and the collaborative work between them.

4.3 Conclusions

This chapter describes in parallel the methodologies used for developing the classical software products and the spatial data related applications. The first one is based on 5 methodological steps, while the other one is a 7-phase process that starts with the context problem definition and the identification of the requirements, and continues with modeling the system related architecture, specifying the methods that can be used for processing the spatial data over different computing infrastructures, and identifying the most suitable presentation techniques that can be used in the visualization and the analysis process. The last two steps of the proposed methodology describe the use cases that can be developed for testing and validating the application, together with the guidelines for installing it on different platforms.

One important thing that is worth mentioning is that the *Modeling the spatial data related processes* phase provides features for both the *Design and modularization* and the *Implementation* stages from the classical methodology of software products development. The identification of the geospatial algorithms and their interconnection represent the design phase, while the encoding of their functionality into a format that can be used at runtime is related to the implementation process.

The following paragraphs summarize the main contributions there were described throughout this chapter:

1. The definition of a methodology used for developing interactive applications that model, process, and visualize various formats of spatial data;

2. The parallel description of the methodologies used for developing software products and spatial data related applications.

Chapter 5

Modeling the spatial data related processes

This chapter describes the main concepts related to the possibility of representing the entire execution process as complex graphs. The development methodology is build upon the fact that the resulting system handles large volume of data. Even though these definitions, rules, and concepts can be used to implement applications regardless of the computation platforms, the methodological steps are exemplified by taken into account the distributed infrastructures (Grid, in particular).

5.1 Types of processes

A process is an abstract representation of an algorithm, method, experiment, phenomena, or simulation. The physical execution of the process is called processing that can be defined as a multi-variable mathematical function P that produces, in a finite amount of time, a valid result, based on a specific input data set. It contains sub-processes (sp_i), and operators combined in a specific order that corresponds to the process description flow. From this methodology point of view, the mathematical definition of such a process that contains n sub-processes is given bellow:

$$P = (SP, IN, OUT, C) \tag{5.1}$$

where,

- P - the main process;
- $SP = \{sp_1, sp_2, \ldots, sp_n\}$ - the sub-processes list, generated by partitioning the definition of P. The dividing mechanism should take into account the infrastructure type (distributed, parallel, multi-core, or single-core) that will execute them. Based on this remark the core process P could be partitioned differently, optimizing this way the entire execution. Each one of the sub-processes contains an abstract representation of a unitary algorithm, operator, or function. It should be mentioned that a sub-process is the smallest division level (e.g. an atomic unit) that can be computed on physical machines;
- $IN = \{in_1, in_2, \ldots, in_m\}$ - the available inputs that represents a list of data resources stored on local or remote data repositories;

- $OUT = \{out_1, out_2, \ldots, out_u\}$ - the variables that are going to be computed;
- $C = \{c_1, c_2, \ldots, c_k\}$ - a finite set of connections that specify the order in which the sub-processes are related between them.

No.	Name	Description
1	Dark Object Subtraction (DOS)	Reduces the noise in satellite images, generated by natural phenomena or by shadow and electrical gain in the sensor device. A dark object represents an area with the lowest reflectance, and it is assimilated as a dark region. For accuracy purposes it is best to remove such spots before moving on to further computations [91].
2	Atmospheric correction	Represents the process of recreating the satellite image without distortions generated by means of absorption and scattering. It was proven that the atmospheric correction results have a major impact in Normalized Difference Vegetation Index (NDVI) usage, as basis for the land cover classification and analysis [92].
3	Geometric correction	It usually takes place in geographic areas where topographic maps are not available, and it is hard to find the desired map projection. Most of the times the correction is performed with the help of field measurements that allow a coordinate transformation that will further be applied to all pixels in the satellite image [93].

Tabela 5.1: Pre-processing operators

No.	Name	Description
1	Raster to raster	Represents the ability to import different raster data formats (e.g. GeoTIFF, MODIS, Aster, SPOT, etc.).
2	Vector to vector	Same as in the case of the first operator, with the difference that both input and output data sets represent vector data (e.g. ESRI shape file, Mapinfo file).
3	Raster to vector	It is suited when bounding the borders of classified objects (e.g. greenness, water, urban buildings). Instead of representing the information as pixels inside an irregular shape, the border is defined as a closed polygon described by points and lines.
4	Vector to raster	Produces the opposite effect as in the raster to vector transformation.
5	Data reprojection	Projections are customized for a certain geographic area. In some cases it is useful to be able to have the same data in different projections, especially when visualizing them over interactive maps.

Tabela 5.2: Conversion operators

The classification of processes can be performed from two points of view. The first one is related to the complexity of their data model structure. In this case they could be grouped into basic operators and complex workflows. A detailed description of both is going to be offered in the following sections.

Another classification type takes into account their usage in different activity domains. Examples of such processes that are supported by the development methodology proposed in

this thesis are going to be described shortly. Although the data model is flexible enough to support other types of processes, the following categories were identified from the experiments that were conducted in order to prove the efficiency and validity of the system:

1. **Pre-processing operators**: useful when the satellite images are influenced by different atmospheric and land cover noises (e.g. clouds, fog, sensor difficulties in providing the correct measurements, sensor calibration, etc.). Experiments were conducted on three types of operators, described in Table 5.1;

2. **Conversion operators** (Table 5.2): due to the large number of spatial data formats there is a permanent need to apply convertion algorithms between these types. In case of multi-band images it is best to divide them into corresponding bands, and apply all these algorithms on individual bands. In order to create a uniform reference model, an internal data structure should be adopted. This means that all incoming information is converted into the GeoTIFF standard, while all further processing will be using this format;

3. **Image bands manipulation** (Table 5.3): this category contains some of the algorithms that could be directly applied on bands of the satellite images. Usually they involve

No.	Name	Description
1	Extract band	In case of multi-band images there is the need to extract the pixel by pixel information from a specific frequency domain, and generate a regular single-band image.
2	Subset	Extracts a sub-region from a band of a satellite image, based on a shape file (vectorial geometry described by a set of points and edges). The result is an irregular raster image.
3	Combine bands	Works in opposition with the subset algorithm, and it is used to generate multi-band images from single-band data sets.
4	Mosaic	Allows the possibility to generate a single-band image, based on individual adjacent tiles. These tiles represent geographical areas that are closed one to another. The mosaic algorithm contains the flow of grouping the tiles together, representing this way the enlarged region, under the shape of a single image file.
5	Pseudo coloring (density slicing)	Generates a pseudo-colored image, based on different combinations between bands that are outside of the visible frequency domain.
6	Arithmetic operations	Represent a list of arithmetic operations (add, subtract, multiply, divide) that are applied on spatial data. Frequent processes involve adding a numerical constant to each pixels from single-band images.
7	Statistics	• In some cases the area computation of a specific geographic region is a powerful factor in remote sensing domains (e.g. generating reports about the area covered by greenness, water, and urban buildings); • Statistics about the classification accuracy.

Tabela 5.3: Operators used in spatial data bands manipulation

operators that are used for extracting relevant information, cropping the image, generate statistic data, etc. Some of these methods can be found in the spatial data retrieval process, described in *Spatial data characteristics* section;

4. **Classification operators**: allows the classification of land regions, by estimating the presence or absence of specific vegetation attributes. They offer useful information about the vegetation density across small/large geographic areas, and operate mainly on multi-band satellite images. The most important vegetation indices are highlighted in Table 5.4.

No.	Name	Description
1	Normalized Difference Vegetation Index (NDVI)	It is the most known VI that combines the Red and NIR bands in order to classify the land cover [94].
2	Enhanced Vegetation Index (EVI)	Improves the NDVI estimation by using data from the Blue reflectance band. This algorithm reduces the atmospheric influence, and it is the most used VI for areas covered by forests [95].
3	Global Environmental Monitoring Index (GEMI)	Another vegetation index that is optimized to work under varying atmospheric conditions, regardless of the illumination factor [96].
4	Sum Green Index (SG)	Detects the changes in the vegetation greenness, based on the fact that the light is strongly absorbed by the green areas [97].
5	Red Green Ratio Index (RGRI)	It is used in order to estimate the development of foliage in canopies. It indicates the density of leaf and flowering production for different geographic areas [97].
6	Moisture Stress Index (MSI)	It measures the water content of leaves, and it is intensively used in studies regarding the ecosystem physiology [[98].
7	Supervised classification	Represents the mechanism for extracting quantitative information from remotely sensed image data [99]. By using this method, the domain field specialists know the pixel values and could generate representative parameters for each class of interest. This step is also known as training. In the end, the objective is to label each pixel with a specific category (e.g. greenness, water, urban buildings, etc.). Usually the supervised classification involves a series of steps: data correction, data subset, apply vegetation index algorithms in case of land cover classification, generate statistics, etc.
8	Unsupervised classification	It doesn't require human intervention and it uses clustering algorithms in order to classify the image data [100]. The final aim is to label each pixel to a specific cluster (class) based on natural grouping of the spectral properties of the image.

Tabela 5.4: Classification operators

5.2 Basic operators

As seen in the previous chapter, the development methodology involves two types of processes: basic operators and workflows. The complexity of the data model that is encapsulated within the two processes plays an important role in the performed classification.

5.2.1 Theoretical concepts

The operator is the smallest unit [101], [102] that can be processed, without the possibility to divide it into atomic modules. Due to this aspect it is also referred to as basic operator. It integrates the description of an algorithm, under the form of executable files. In other words the operator represents a specific piece of code that implements that algorithm. The software technology is not important in this case, as long as it generates files that could be executed over different platforms.

The embedded algorithm exposes only the inputs and the output data set as a list of triplets $< name, value, type >$ while the implementation sequence is hidden under the shape of executable files. In order to change the algorithm it is necessary to modify the source code and to recompile again. The significance of the triplet $< name, value, type >$ is the following:

- *name*: name of the input/output that matches the variable used within the algorithm;
- *value*: a particular value, expected as input. Depending on the type of the input, the value field could be: numerical (e.g. 1; 1.3; -10.89), string (e.g. "second band", "[-1;0.25], [0.25;1]"), spatial data file that identifies the physical path to that resource (e.g. "/resources/Landsat/2013.01.20"), etc.;
- *type*: the majority of programming languages require that each attribute (constant or variable) should have a data type. A list of possible types that were identified within this development methodology are given in Table 5.5.

Technically speaking the operator can be defined as a function $O(IN, OUT, DATA)$, where:

- $IN = \{in_1, in_2, \ldots, in_n\}$ - the available inputs data set of the operator;
- OUT - the output of the operator;
- $DATA = \{d_1, d_2, \ldots, d_m\}$ - a set of data resources that are assigned to each input.

The definition of O depends on n inputs ($in_k, k = \overline{1,n}$), one output (*out*), and a set of data resources ($d_p, p = \overline{1,m}$). Each parameter in_k is a triplet $< name, value, type >$ that has a name, a value (or a resource), and an associated type. It's worth mentioning that the order in which the parameters $\{in_1, in_2, \ldots, in_n\}$ are given has a major impact on the final result. There should be a perfect match between the n arguments and the inputs of the algorithm represented by the operator.

The circle was chosen as graphical representation for the basic operator, while the rectangle is the symbol of workflows. It can easily be observed (Figure 5.1) how the operator exposes its inputs as a list of triplets, a concept introduced earlier. Similar, the output of the operator has the same structure as the one of the inputs. The vertical line (that divides the inputs from the output) is an abstract representation of the operator. Even though the circle is the graphic shape of the operator, the vertical line will be used in other complex figures, where adding extra primitives would complicate the entire schema.

No.	Type	Description
1	geotif	It represents the GreenLand internal format for all raster spatial data, regardless of their original type (e.g. Landsat, MODIS, Aster). Usually requires a conversion from all these types into the GeoTIFF format.
2	raster	Well-known formats for today's graphics (e.g JPEG, PNG). They are used especially when the user requests to download the result generated by the process execution.
3	vector	Used to store vector data formats, represented by a set of points that compose the shape of the image (shp), an index of the vector geometry that improves the seeking mechanism (shx), and a table-like representation of each shape in data base format (dbf).
4	int	Integer values, within [-32768; 32767]. Compatible with integer domain from different programming languages.
5	float	Floating-point values, within $[-3.4 \cdot 10^{38}; 3.4 \cdot 10^{38}]$.
6	double	Double precision floating-point values, within $[-1.8 \cdot 10^{308}; 1.8 \cdot 10^{308}]$.
7	string	Text representation of data. The density slicing operator represents one common place where the string type is required. In this case the color palette is given as an array of intervals that could be specified as a string value.
8	proj	In case of reprojecting the satellite image to a different geographic area, the corresponding algorithm needs a projection file that stores all the necessary information. Until now only the PROJ4 file types [103] are supported.
9	metadata	Useful for storing the metadata in a standardized format. Information about the author, creation time, type of spatial data, original projection, and geographic coordinates are a few attributes that are stored in this file. Optimizing the spatial data search mechanism is one possible feature where the metadata can be used.
10	text	Manages data that are stored inside ascii files. For example, this type can be used for representing the ground-based measurements.
11	archive	Useful when data is compressed in .zip, .rar, or .tar.gz files. For example, the multi-band Landsat satellite images are often exposed as a big archive file.

Tabela 5.5: List of input/output types

As can be seen there are n inputs (indexed from top to bottom) and only one output of type: image, file, metadata, text, or archive (see Table 5.5). In case we need to generate data that requires a more complex structure (e.g. a GeoTIFF image and a metadata file), we could package them into an archive.

There is a tight correlation between function O (that mathematically describes the operator) and Figure 5.1. In both places we have n inputs $(in_1, in_2, \ldots, in_n)$, m possible data resources to assign to each input $(d_1, d_2, \ldots, d_m)$, and one output (out). Based on their interaction the following validation rules could be generated:

- $value_k, k = \overline{1,n}$: takes values from the data resources set $\{d_1, d_2, \ldots, d_m\}$. There are cases in which the same resource is shared by multiple inputs;
- $value_{out}$: is generated at runtime, when executing the operator with the given input data

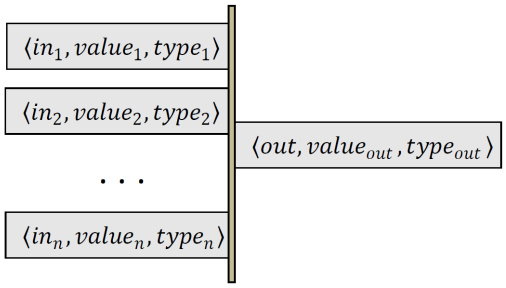


Figura 5.1: The abstract representation of the basic operator

set. $value_{out}$ is not part of the $\{d_1, d_2, \ldots, d_m\}$ resources list;

- $type_k, k = \overline{1,n}$: takes one value from the 11 possible types described in Table 5.5. As in the case of the first rule, multiple inputs could have the same data type;
- $type_{out}$: takes one value from the 11 possible types described in Table 5.5;
- $n \geq 1$: the operator must have at least one input;
- $m \geq 1$: there should be at least one resource to choose from.

The following paragraphs highlight a 5-steps methodology (process definition, data resource identification, formal description of the operator, its transcription into a high level programming language, and the execution of the algorithm with different input datasets) [73] required to develop a new operator. The exemplification is performed on the Global Environmental Monitoring Index (GEMI), described in Table 5.4 that works with GeoTIFF satellite images, and tries to compute the greenness area based on the Red and Near InfraRed (NIR) spectral bands.

1. The first step identifies the mathematical definition of the operator. The entire process of vegetation classification, based on GEMI index, can be reduced to the following formulas [104]:

$$k = \frac{2 \cdot (NIR^2 - Red^2) + 1.5 \cdot NIR + 0.5 \cdot Red}{NIR + Red + 0.5} \tag{5.2}$$

$$GEMI = k \cdot (1 - 0.25 \cdot k) - \frac{Red - 0.12}{1 - Red} \tag{5.3}$$

2. The second step involves the conversion of the mathematical formulas into a graphical representation of the GEMI operator. This includes the correct identification of inputs and output, and does not concern with the actual implementation of the formulas. There are several inputs required in order to complete this step:
   - NIR: the Near InfraRed band of the satellite image;
   - Red: the red band of the satellite image;
   - Integer values: 1 and 2;
   - Float values: 0.12, 0.25, 0.5, and 1.5.

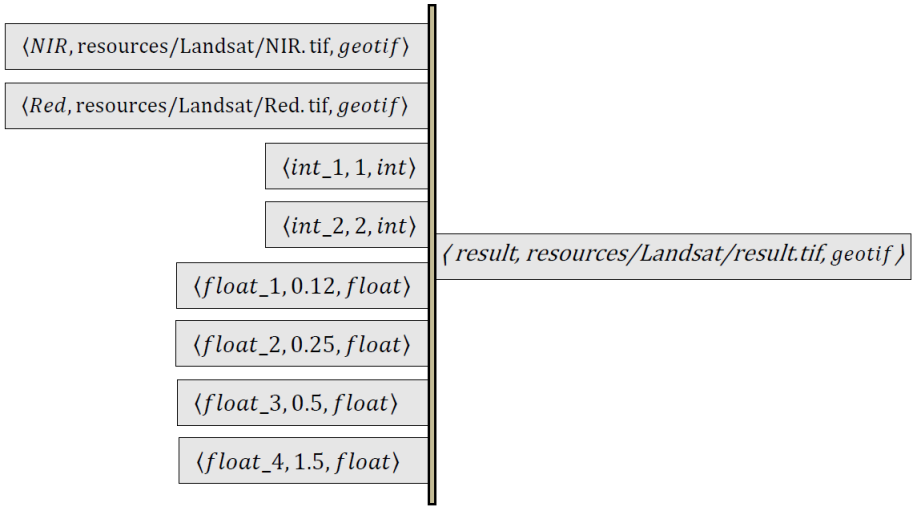


Figura 5.2: The graphical description of the GEMI operator

Based on this information, we can generate the graphical representation of the operator (Figure 5.2). The inputs position starts from the top and moves down to the bottom of the operator. So, the *NIR* band is the first input, while the *float_4* is the last one. This order should be maintained when implementing the algorithm into a high level programming language (step 3);

3. Even though we've identified the inputs types and values, it is important to know the order in which they are send to the algorithm. Otherwise the implementation result will not match the two formulas. The third step is related with the implementation of the GEMI functionality, using the pseudo-code language to describe the two mathematical functions, like in Algorithm 5.1.

   The algorithm is a general representation of the GEMI index that is particularized in this case with the input data set specified in step 2. In the first line of the algorithm, an array (*algIn*) was defined whose number of items is equal with the number of input parameters defined in the second step of the GEMI's development methodology. We also need two extra variables (k and *GEMI*) that will help us to compute the value of the mathematical functions, described in the first step.

   The first validation rule is to check if the number of input parameters passed to the algorithm (*passIn*) is equal to the size of the *algIn* array. If not, the algorithm will stop and no output result will be generated (lines 6-8). Next, we have to assure that the external output parameter (*passOut*) matches the output defined within the algorithm (lines 10-12). The same approach is performed on the inputs received by the algorithm. In case of success, the value of these inputs will be assigned to the *algIn* array (lines 14-20). Lines 22 and 24 computes the GEMI index, based on the mathematical formulas specified in the first step. The algorithm ends with generating the final output (*algOut*);

4. This step concerns with the transcription of the pseudo-code algorithm into a high level programming language (e.g. Java, C, C++) that is able to generate the outputs as executable files. The Algorithm 5.1 represents a general description of the GEMI vegetation index that is instantiated at runtime with different inputs data sets. At this stage, only

Algorithm 5.1 Implementation of the GEMI operator

var algIn[8]
var algOut
var k
var GEMI

if $length(passIn) \neq length(algIn)$ **then**
  exit algorithm
end if

if $\neg typeMatch(passOut, algOut)$ **then**
  exit algorithm
end if

for $i = 1 \rightarrow length(passIn)$ **do**
  if $typeMatch(passIn[i], algIn[i])$ **then**
    $algIn[i] \leftarrow passIn[i]$
  else
    exit algorithm
  end if
end for

$k \leftarrow \frac{algIn[4] \cdot (algIn[1]^2 - algInputs[2]^2) + algIn[8] \cdot algIn[1] + algIn[7] \cdot algIn[2]}{algIn[1] + algIn[2] + algIn[7]}$

$GEMI \leftarrow k \cdot (algIn[3] - algIn[6] \cdot k) - \frac{algIn[2] - algIn[5]}{algIn[3] - algIn[2]}$

$algOut \leftarrow GEMI$
return algOut

the executable file is needed, and a list of parameters (*passIn*) that store the path of the physical data resources;

5. The last step of this development methodology is related with the operator's execution. The processing is possible based on the executable file, generated in the previous phase. Using this flexible approach in developing new operators, gives the possibility to apply the same operator multiple times. For example, in the same execution process we could have 2 (or more) GEMI operators, instantiated with different input values. Monitoring the execution and results management are other features that could be included at this stage of development.

5.2.2 Practical implementation over the GreenLand platform

The previous sections described the theoretical concepts of the basic operators, by highlighting their characteristics and how they can encapsulate the functionality of a specific algorithm (method, formula, or function). In order to validate these concepts, the operators were integrated within the GreenLand platform and then executed over different computing infrastructures. The results analysis confirmed or infirmed that the inner functionality of the operators was properly setup.

The methodology of developing new operators (see section *5.2.1 Theoretical concepts*) requires five steps, starting with the description of the algorithm and going up to the implementation and execution of this operator. The fourth step regards the transcription of its functionality into a high level programming language (Java, C#, C++, etc.) and it is the main subject of this section. The following paragraphs highlight how the users are able to create and customize their own operators and what is the procedure of integrating them within the GreenLand platform, from where they can be used in further processing.

5.2.2.1 General overview of the OperatorEditor tool

In order to ease the development of basic operators, an API was provided to the users that contain the core implementation of a default operator that has to be extended with the new functionality. Even though the operator's data model supports the usage of any kind of programming language (as long as it generates outputs compatible with the operating system installed on the computing infratructure) the exemplification is going to be performed on the Java based API.

As can be seen, the implementation of new operators is not that simple and requires good knowledge of the Earth Science (in order to be able to describe the algorithm's functionality) and computer science (necessary for implementing the functionality into a high level programming language) domains. Taking into account all these aspects it is recommended that the methodology of developing new operators involves specialists from both domains.

It is mandatory to implement valid operators, because they are going to be used in different use case scenarios within the GreenLand platform. This is another remark that encourages the involvement of domain field specialists that have solid knowledge about the context of the problem they try to solve.

Once the algorithm (operator) is correctly implemented it is time to integrate it within the GreenLand platform. It is worth mentioning that this system provides to proper tools in order to easily instantiate, manage, and execute various types of operators over local or distributed environments. If the development process requires highly-trained domain field specialists, the integration within the GreenLand framework can be performed by any type of users. This is possible through the OperatorEditor module that is capable of translating the operator's functionality into an internal language recognized by the GreenLand platform.

This module provides interactive and intuitive techniques that facilitate: the inputs/output specification, the description of the inner functionality of the algorithm, the setup of certain privacy roles, the update of new features, the history track of all the versions of that operator, etc. An alternative option for the OperatorEditor tool is to allow the user to manually input all these information as database entries or to describe them within a proprietary text format.

Both these options do not provide enough flexibility and interactivity as the OperatorEditor does. On the other hand this module leaves no room for syntactic errors and assures the user that at the end of this integration process the new operator is valid and ready to be used in further processing. Even though the OperatorEditor tool is able to perform some syntactic validations, the correctness of the algorithm's implementation is the user responsibility.

5.2.2.2 Extension of the default operator

The previous mentioned theoretical concepts allow the development of the operator in any programming language, as long as it generates executable files compatible with the operating system that is installed on the infrastructures that perform the execution. For exemplification purposes this section describes the implementation process using the Java language.

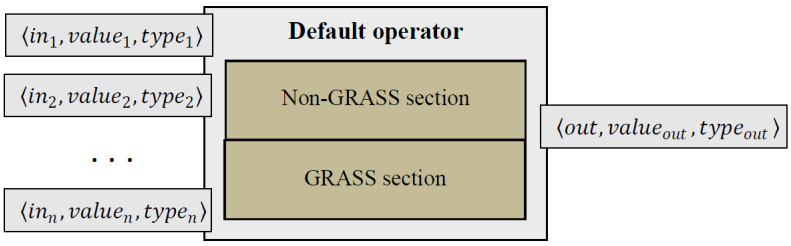


Figura 5.3: The generic description of the default operator

There is a default basic operator class that needs to be extended in order to develop new algorithms. This functionality is packed as an API archive that includes:

- the core features that intermediate the communication between the GreenLand platform and the processing infrastructures;
- the GRASS distributable library that is useful when processing various types of spatial data;
- the default operator class;
- documentation files that contain the instructions about the implementation process and the requirements that need to be taken into account.

Figure 5.3 contains the generic description of the default operator, while Figure 5.4 highlights the corresponding Java source code embedded within the API. As can be seen the default operator contains n input datasets and a single output (their significance was already presented in the theoretical concepts paragraphs). An important aspect to take into account is the fact

```
1  public class DefaultOperator extends Operator {
2
3      @Override
4      public void Execute() {
5
6          //Get the operator's parameters
7          String in1 = this.getParams().get(0);
8          int in2 = this.getParams().get(1);
9          //. . .
10         Proj inn = this.getParams().get(n-1);
11
12         generic("grassExecute");
13         generic("nonGrassExecute");
14     }
15
16     @Override
17     public void grassExecute(StringBuilder builder) {
18
19         //Implement the GRASS functionality
20     }
21
22     public void nonGrassExecute() {
23
24         //Implement the regular functionality
25     }
26 }
```

Figura 5.4: The source code related to the default operator's functionality

the content is divided in two regions that process regular data (non-GRASS section) and spatial information (GRASS section).

The first area is dedicated to operations that process primitive data (e.g. adding two integers, string formatting, numerical conversion, etc.). The GRASS library provides specialized algorithms for spatial data management, such as: relevant information extraction from satellite images, vegetation index computation, thematic map creation based on a specific color palette, etc. When needed the two sections may complete each other, meaning that the first one is able to provide certain functionality for the GRASS area.

Figure 5.4 describes the Java code that implements the default operator's functionality by customizing the *n* inputs to specific Java types (e.g. the *in1* input has the String type). The parameters list can be accessed by using the *getParams.get* method that is already defined in the *Operator* class. The type and order of each parameter have to be known before actually retrieving its value (this rule is validated by the OperatorEditor tool and requested when integrating the algorithm within the GreenLand platform).

The *nonGrassExecute* and *grassExecute* methods implement the functionality of the two corresponding blocks, described in Figure 5.3. Because the output of the operator needs to be persistent over time and it takes various forms (e.g. numerical values, strings, files content, metadata, etc.) the adopted solution was to always store the result within an external file.

The following paragraphs describe two examples of developing new operators, based on the concepts presented earlier. The first experiment (Figure 5.5) proposes the addition of two integer values and the output result storage as content of a text file. The two numbers and the name of the file are part of the input parameters list. The computed sum is stored within the *result* variable that is then written into the output file (line 26). After executing this operator, the user is able to download the file directly from the GreenLand graphical interface.

```
1  public class Add2Integers extends Operator {
2
3      @Override
4      public void Execute() {
5          generic("nonGrassExecute");
6      }
7
8      @Override
9      public void grassExecute(StringBuilder builder) {
10
11     }
12
13     public void nonGrassExecute() {
14
15         int in1 = Integer.parseInt(this.getParams().get(0));
16         int in2 = Integer.parseInt(this.getParams().get(1));
17         int result = 0;
18         Writer out = null;
19         File f = new File(this.getParams().get(2));
20
21         result = in1 + in2;
22
23         try {
24             f.createNewFile();
25             out = new BufferedWriter(new FileWriter(f));
26             out.write(result.toString());
27         } catch (Exception e) {}
28
29         out.close();
30     }
31 }
```

Figura 5.5: Example of adding two integer values

The second example (Figure 5.6) uses the GRASS library in order to compute the NDVI value for specific geographic locations. This operator requires as input two satellite image bands (Red and NIR) and the name of the output result (in this case another satellite image). The GRASS commands are initially written inside a string variable whose content is evaluated as individual commands and executed at runtime. The final result is computed for each pixel of the two bands by using the instruction from line 22 that actually corresponds to the formula:

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{5.4}$$

5.2.2.3 Operators integration within the GreenLand system

Extending the behavior of the default operator is an independent process from the OperatorEditor tool, and it allows the creation of new functions that match the features desired for the new algorithm. Once the operator is implemented, the recommendation is to test it on the user's local machine. If its execution generates errors there is a good chance that it will also fail when running it from the GreenLand application. Otherwise, if the algorithm is processed correctly and generates the desired output, the user is encouraged to integrate it with the GreenLand environment.

The benefit of this action consists in the fact that this platform provides specific tools for: optimized execution over distributed infrastructures (Grid in particular), specialized monitor mechanisms of the entire execution, interactive integration within other use case studies (described as workflows), and availability setup to the entire users' communities.

The OperatorEditor module facilitates the integration of the new developed operators within the GreenLand platform. Figure 5.7 highlights this process for the NDVI example, described earlier. First of all the owner of the operator has to provide a proper name that can be distinguished among the list of other algorithms, and that intuitively describes its functionality. In this case we chose the name *NDVI*.

```
1  public class NDVI extends Operator{
2
3      @Override
4      public void Execute () {
5          generic("grassExecute");
6      }
7
8      @Override
9      public void grassExecute(StringBuilder builder){
10
11         String output = this.getParams().get(2);
12
13         //Import the Red band
14         builder.append("r.in.gdal input=").append(this.getParams().get(0));
15         builder.append("output=in1");
16
17         //Import the NIR band
18         builder.append("r.in.gdal input=").append(this.getParams().get(1));
19         builder.append(" output=in2").append("\n");
20
21         //Compute the NDVI result
22         builder.append("r.mapcalc\"output\"=\"(float(in2-in1)”+”/float(in1+in2))\"\n");
23         builder.append("r.out.tiff input=output output=" + output);
24     }
25 }
```

Figura 5.6: Example of computing the NDVI index

Due to the fact that only the owner has access to its inner functions, the OperatorEditor tool requires a short description of the operator. It is also recommended to provide an extended description (*Extra description* field) under the shape of document files (.doc or .pdf formats are supported). This file should explain the inner functionality of the algorithm, the mathematical formula (is necessary) used to implement the operator, the significance of each input, and a use case scenario with real input data sets. Following this procedure it is most likely that other users will fully understand the characteristics of this operator and will reused it (instead of creating a new one from scratch) when developing various workflows.

The owner of the operator has the possibility to keep it private or to share it with the rest of the GreenLand users, by setting up the *Privacy* attribute. Next, the OperatorEditor requires some technical information about the implementation of the operator, such as: the name of the class that extends the functionality of the default operator (*NDVI* in this case) and the API archive that contains the full description of the algorithm.

The third category of attributes displayed by the OperatorEditor module is related to the specification of the input types. In the previous example (Figure 5.4) we've seen that the n inputs were provided with specific data types (String, int, ..., Proj). These types are known by the owner that developed the operator, but hidden from other users. At this stage, the OperatorEditor tool requires the specification of these types. The *Plus* button has the effect of inserting a new input, while the *Minus* component deletes the current data entry (Figure 5.7). For this example we need two satellite image input types (encoded as TIFF) that correspond to the Red and NIR bands.

Another important aspect that needs to be taken into account is the order in which these data entries are specified. It is worth mentioning that there should be a perfect match between the input variables defined at the source code level and the position and type of the entries described in the OperatorEditor module. Otherwise, even though the operator is correctly implemented the final output result will be distorted.

Once these attributes are completely specified, the next step is to upload the operator within the GreenLand repository. At this stage, the algorithm is going to be analyzed by the GreenLand support team in order to find if it is compatible with the requirements requested

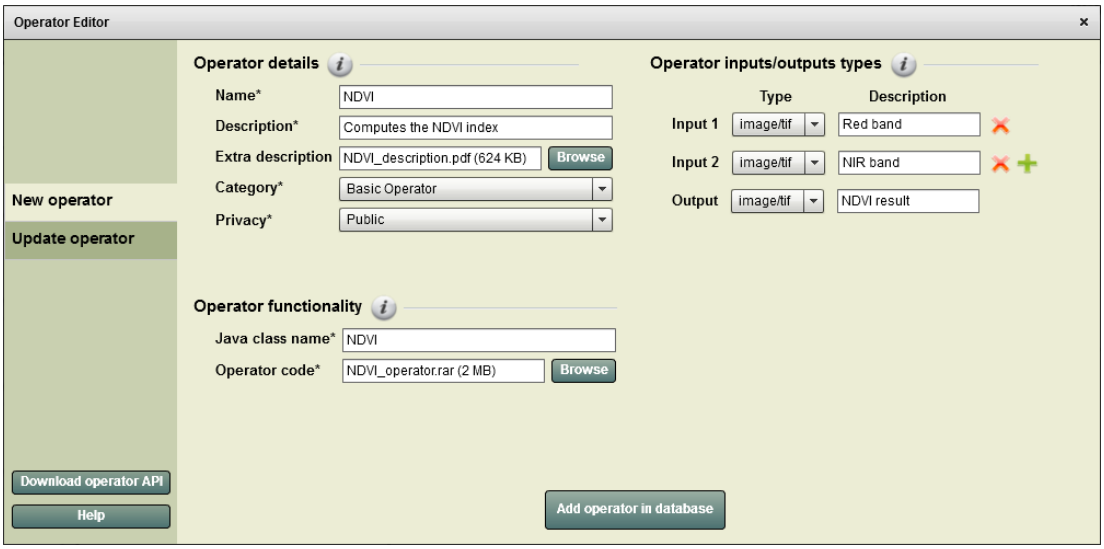


Figura 5.7: The integration process of the NDVI operator through the OperatorEditor module

for executing it over the available computing infrastructures. This task includes the following verifications:

- the algorithm should be errorless, from syntactic point of view;
- the processing infrastructures should be able to interpret the executable files generated by the programming language used for developing the operator;
- the content of the archive should contain all the required information for running the operator on the Grid or multi-core machines.

If the validation tests are passed the other stage is to verify it the execution of the operator completes in a finite amount of time and provides the proper results (as specified in the documentation). If errors occur at any steps, the status of the operator updates to *Error*, otherwise is will be setup to *Valid*. In the first case the support team will notify the users that implemented the algorithm about the encountered issues. These problems are due to the requirements' violations (e.g. using a programming language that is not supported by the computing infrastructure) or to erroneous implementation of the inner functionality. In both cases the owners are able to update the operator and to comply with the received recommendations.

The *Valid* status assures that the operator is correctly implemented and that it can be used in further processing. Depending on the privacy attribute, it can also be integrated in various workflows that describe the functionality of complex use cases. It is worth mentioning that throughout the entire validation process the state of the operator displays the *Pending* value.

Figure 5.8 highlights the updating process of the NDVI operator, developed earlier. Depending on the status attribute the user is able to update specific characteristics of the algorithm. During the validation process, only the name, description, and privacy attributes are allowed to be changed, while the inner functionality should remain the same in order for the support team to proceed with the validation rules.

When the owner performs major changes to the algorithm's behavior (e.g. changing the number of inputs, adding new mathematical formulas) it is recommended to create other versions

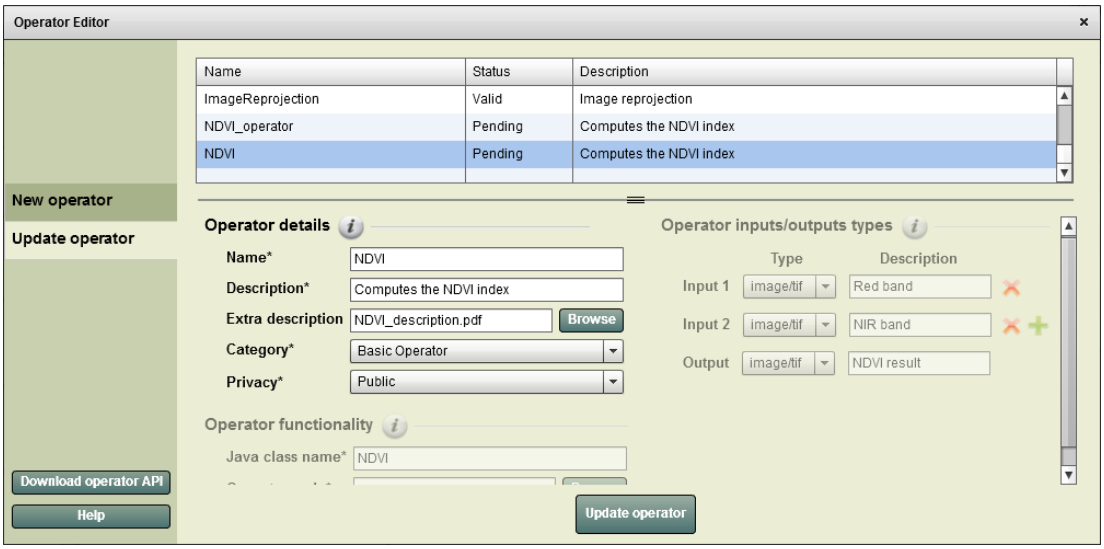


Figura 5.8: Updating the attributes of the NDVI operator

of operators, and to leave the current one as it is. This is strongly related with the fact that the operator might be already integrated as a node within complex graphs, and changing its default functionality will affect the entire workflow.

5.3 Complex workflows

The description of natural phenomena, of real or simulated experiments, from different activity domains, is represented by complex processes that involve a solid understanding of the phenomena, the syntactic and semantic description of the proposed solutions, the experimental data collection, and the analysis and interpretation of the results. Such a use case scenario (natural phenomena, experiment) is modeled as a collection of operators, connected in a workflow (graph) that is able to generate in a finite amount of time a valid output, based on a range of inputs data sets [105].

The main goal behind the workflow concept is the optimal representation of use cases and the data model organization. Using the graph representation method, the use case is easier to visualize and analyze by different categories of users [78].

5.3.1 Theoretical concepts

Workflows represent the most complex entities that can be executed over distributed infrastructures or standalone machines. They usually describe, in an abstract way, real case studies or experiments from different Earth Science domains, and take the shape of acyclic graphs that indicate the processing flow of these case studies [74], [106].

Technically speaking the workflow can be defined as a function $W(IN, OUT, DATA, N, C)$ that depends on:

- $IN = \{in_1, in_2, \ldots, in_q\}$ - a set of input parameters. Each input $in_k, k = \overline{1,q}$, is a triplet $< name, value, type >$. The significance of the triplet was already described in the previous section (*5.2 Basic operators*). The *value* attribute of the inputs is retrieved from the data resources list $\{d_1, d_2, \ldots, d_m\}$;

- $OUT = \{out_1, out_2, \ldots, out_s\}$ - a set of outputs. This is one of the main differences between basic operators and workflows. While the first structure allows only one output/unit, the workflow may have multiple outputs of different types. Each output $out_p, p = \overline{1,s}$ share the same triplet data model $< name, value, type >$.

  The *value* attribute is generated at runtime, when executing the workflow with the given input data set, and it is not part of the $\{d_1, d_2, \ldots, d_m\}$ resources list;

- $DATA = \{d_1, d_2, \ldots, d_m\}$ - a set of data resources used to instantiate the workflow at execution time. The path attribute is common for all items, and it represents the local or remote physical location of the resource. FTP and HTTP protocols could be used for data retrieval from these locations;

- $N = \{n_1, n_2, \ldots, n_u\}$ - a list of nodes that determine the graph's structure. In its simplest form the node $(n_k, k = \overline{1,u})$ is identified as a basic operator, and it keeps the same concepts defined earlier. A more complex node, called hiper-workflow, stores other graphs within;

- $C = \{c_1, c_2, \ldots, c_v\}$ - a finite set of uni-directional connections (edges) that link the graph nodes. Such a connection $(c_p, p = \overline{1,v})$ is defined as a tuple $< n_i, n_j >$ that links the two nodes (n_i, n_j) through a direct edge.

The workflow is graphically represented by means of rectangles, and stores a finite set of basic operators (*OP1, OP2, ... , OPu*) (Figure 5.9).

There is a tight correlation between function W (that mathematically describes the workflow) and Figure 5.9. In both places we have q inputs $(in_1, in_2, \ldots, in_q)$, m possible data resources to assign to each input $(d_1, d_2, \ldots, d_m)$, s outputs $(out_1, out_2, \ldots, out_s)$, u nodes (*OP1, OP2, ..., OPu*), and v edges. Based on their interaction the following validation rules could be generated:

- $value_{ik}, k = \overline{1,q}$: takes values from the data set $\{d_1, d_2, \ldots, d_m\}$. There are cases in which the same resource is shared by multiple inputs;
- $value_{ok}, k = \overline{1,s}$: it is generated at runtime, when executing the workflow with the given input data set, and it is not part of the $\{d_1, d_2, \ldots, d_m\}$ data resources list;
- $type_{ik}, k = \overline{1,q}$: takes one value from the 11 possible types described in Table 5.5. As in the case of the first rule, multiple inputs could have the same data type;
- $type_{ok}, k = \overline{1,s}$: takes one value from the 11 possible types described in Table 5.5. Multiple outputs could share the same type;
- $q \geq 1$: the workflow must have at least one input;
- $s \geq 1$: the workflow must have at least one output;
- $u \geq 1$: there should be at least one operator node within the workflow;

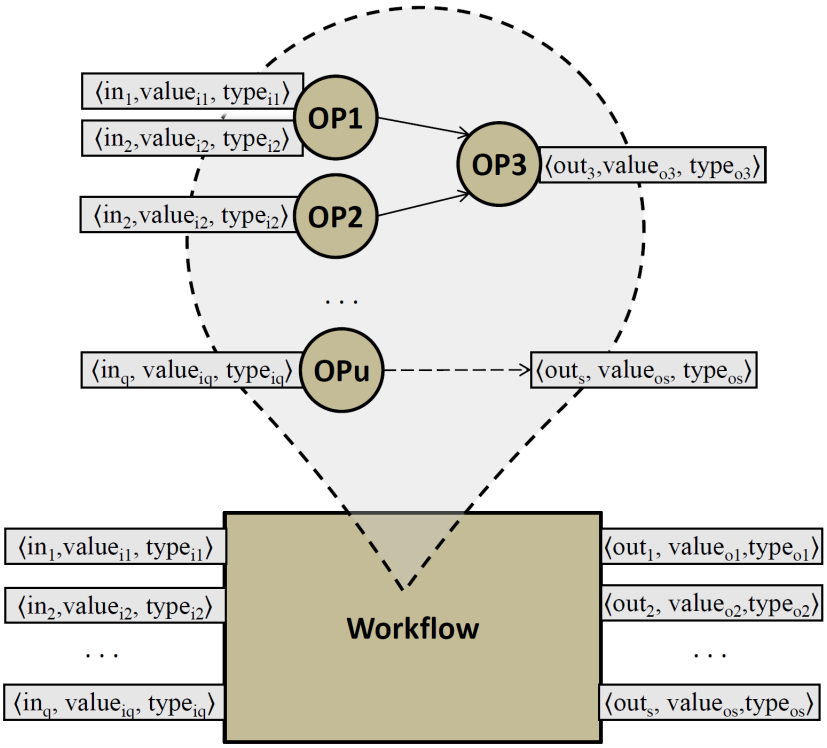


Figura 5.9: The abstract representation of the workflow

- $v \geq 0$: regardless of the number of nodes, there are cases in which the workflow does not contain any edges (connections).

The workflow described in Figure 5.9 contains u operators *(OP1, OP2, ..., OPu)*. The inputs of the workflow are individually distributed as inputs for these operators. As can be seen, the first node takes in_1 and in_2 as input, while the second node requires a single input (in_2). The general rule is that all inputs of the workflow have to be assigned to the list of operators (same for the list of outputs).

Due to the constraint that each operator must have a single output, the operators *OP3, OP4, ..., OPu* are the ones that generate the final result of the workflow. There are two more outputs (from *OP1* and *OP2*), but these have only an internal impact over the entire structure and could not be defined as primary outputs.

By means of symmetry between the function W (used for describing the workflow) and the structure presented in Figure 5.10, there should be v edges. From this graphical representation only two of them are visible: (*OP1, OP3*) and (*OP2, OP3*). The rest of them were not displayed, in order to simplify the schema.

There is a strong correlation between workflows and graphs, regarding the data structure. A mathematical graph can be defined as an ordered pair *G=(X, U)*, where X is a non-empty and finite set of nodes ($n_k, k = \overline{1,u}$), and U is a non-empty set of item pairs from X (called uni-directional edges – $c_p, p = \overline{1,v}$). It has to be mentioned that all these workflows are acyclic graphs, avoiding this way the situation of infinite looping [107].

The items from X can be defined as nodes of the graph, whose functionality is implemented through high level programming languages, such as Java and C++. There are three types of nodes:

1. **Operator**: similar with the ones presented in the previous section;
2. **Workflow**: name given to a data structure that contains only basic operators;
3. **Sub-workflow**: this property is given only to nodes that are part of a hyper-workflow. The main difference between workflow and sub-workflow is the fact that the latest has a parent data structure, while the workflow identifies as the root of this structure;

The hyper-workflow is the most complex data structure that stores operators and sub-workflows, with unlimited number of nodes. Mathematically, such a structure can be described as an ordered pair $G_H = (X_H, U_H)$, where at least one node from X_H contains another ordered pair *G=(X, U)*. Multiple imbrications levels are allowed. This means that one node (n_k) may contain an inner workflow node (n_{k_1}) that in his turn stores another workflow structure ($n_{k_{11}}$), and so on.

The items from U represent uni-directional edges (c_p) that describe the processing flow of the entire case study (experiment) modeled through the workflow. Syntactically speaking, one edge may connect only two nodes that share the same output-input types.

Figure 5.10 highlights the general structure of a hyper-workflow that contains three basic operators *(OP1, OP2,* and *OP5)*, an inner workflow (*SW1*), 5 inputs for the *OP1* and *OP2* and two outputs. In order to simplify the figure, only the types of the inputs/outputs were defined (see Figure 5.9 for a full description of these attributes). The *SW1* node includes other graphical elements, connected through edges. One important thing is the fact that the two inputs of *OP3* and *OP4* correspond to the outputs of the *OP1* and *OP2*. Also, the "File" input of *OP5* represents the output for the *SW2* [78]. The rectangle is the graphical symbol of the workflow, while the basic operators have the circle as geometric primitive.

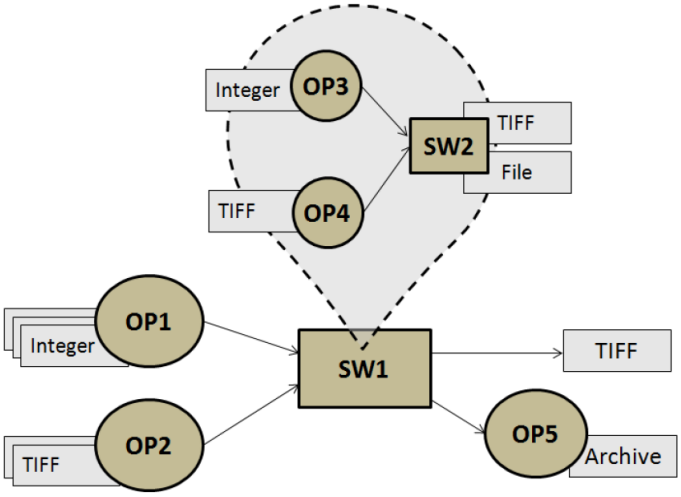


Figura 5.10: The abstract representation of a hyper-workflow

5.3.2 Data model description

"A model is an abstraction of something for the purpose and understanding it before building it. Because a model omits nonessential details, it is easier to manipulate than the reality" [108]. The creation of such models usually involves a set of attributes and a set of rules that describe the usage of these attributes.

One important issue is the fact that the workflow concept represents an acyclic graph that doesn't allow looping executions. By conducting a set of experiments we've noticed that the processing failures were due to the infinite executions, introduced by the users. Working with cyclic workflow is a complex task, so instead we've concluded to assign to each graph the acyclic property.

The graph is a complex structure of nodes and edges. Each node is an operator or a sub-workflow. The edges represent the virtual connections between these nodes, and highlight the processing flow of the use case scenario described by this complex structure. In order to represent such a hyper-workflow, there is the need of a dynamic data model, capable of:

- Storing all the information, regardless of the number of imbrications levels;
- Allowing easy access to inner elements (nodes and edges);
- Auto-updating: when a sub-workflow (*SW1* from Figure 5.10) is updated, the data model should propagate this change into all workflows that contain *SW1* as inner node;
- Allowing the possibility to duplicate a workflow as many times as needed, and to have the possibility to instantiate them with different input resources.

Regarding all these aspects, we propose a recursive data model in which the information is connected by means of references (pointers). A pointer is a unique identification of an operator or a workflow. Based on this concept, the links between the nodes could be established by storing this unique identifier. Figure 5.11 highlights the attributes that were identified for the node and the edge elements of the hyper-workflow structure.

The *nodeID* and *edgeID* represent the unique identifiers (at the main workflow level) that differentiate one element from others. They have a numerical pointer value that increases each

time a new item is added to the workflow. The *nodeID* also identifies the memory location path that stores the actual information of the data model. The *name* and *description* store important information about the functionality of the workflow, its domain of activity, etc. Because a workflow contains several types of nodes, it is mandatory to establish this aspect. This is the reason why the *type* attribute was introduced, and it could take one of the following values: "operator" or "sub-workflow". The type attribute is very useful when navigating the entire workflow, because it is an indicator regarding the expandability of that node (a node is expandable if it contains one child with the "sub-workflow" type).

The last two attributes described in Figure 5.11 are similar with the ones presented in Figure 5.1 and Figure 5.9. Both the *inputs* and *outputs* arrays represent the interface of the node and provide a powerful method of instantiating the node with different resources (values).

Generally speaking, an edge is defined as a direct connection between two nodes of the workflow, regardless of their types. Because each node contains multiple inputs/outputs there is the need to refine the edge definition, to also include the following feature: an edge represents a uni-directional connection between an output (out_i) of the node n_i and an input (in_j) of the node n_j, where $n_i \neq n_j$, and out_i and in_j have the same type. Due to this constraint, the edge must connect two different nodes. Another aspect to take into account is the fact that the edges must be defined in such way that the main workflow keeps its acyclic property.

The *nodeStart* and *nodeEnd* attributes from Figure 5.11 store the identifiers of the two nodes that are connected by the edge. The *outputIndex* and *inputIndex* are related with the index of the *outputs* and *inputs* attributes, specified in the node definition.

Two examples are provided just to highlight how the data model is populated with information about the workflow structure. The first example (Figure 5.12) is a simple one that describes

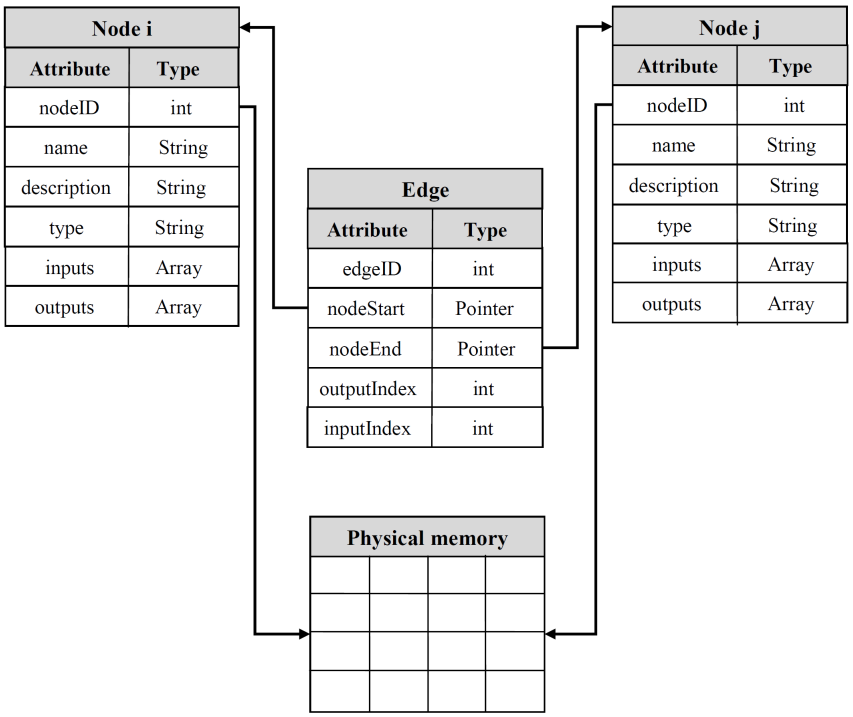


Figura 5.11: Data model attributes for the node and edge elements

Figura 5.12: Data model attributes exemplification of a simple workflow

a workflow that contains only two basic operators (*OP1* and *OP2*) and one edge. In order to simplify the figure, we've chosen to highlight only the inputs/outputs names, and defined their triplet structure outside of the drawing. For this example the attributes are highlighted in Table 5.6 and 5.7.

	OP1	**OP2**
Attribute		
nodeID	1	2
name	OP1	OP2
description	First operator	Second operator
type	operator	operator
inputs	$< in_1, OP1in.tif, geotif >$, $< in_2, 1, int >$	$< in_3, OP1out.tif, geotif >$, $< in_4, 3, int >$
outputs	$< out_1, OP1out.tif, geotif >$	$< out_2, out.tif, geotif >$

Tabela 5.6: Node attributes table exemplification of a simple workflow

	edge1
Attribute	
edgeID	3
nodeStart	1
nodeEnd	2
outputIndex	1
inputIndex	1

Tabela 5.7: Edge attributes table exemplification of a simple workflow

Let's consider the case in which the previous generated workflow is labeled with *nodeID=4* and the hyper-workflow from the second example (Figure 5.13) integrates it as an inner node (*SW1*). As can be seen the *SW1* exposes only three of its four inputs in_1, in_2, and in_4. The in_3 input is generated automatically by the output of *OP1*, so there is no need to require it in the list of inputs (similar for the inputs in_5 and in_6). Expanding the *SW1* sub-workflow will generate the structure from Figure 5.12.

When a workflow is used inside a graph (as an inner node), a new copy is created, maintaining the same attributes' values, excepting the *type* that changes to "sub-workflow". For this example the nodes and edges attributes are presented in Table 5.8, and 5.9.

The development of hyper-workflows is based on a 4-steps methodology that starts with the description of the use case scenario in a natural language, and ends with the hyper-workflow processing, results analysis, interpretation, and visualization [102], [81], [109].

1. The first step consists in identifying the use case (experiment), defining the scenario

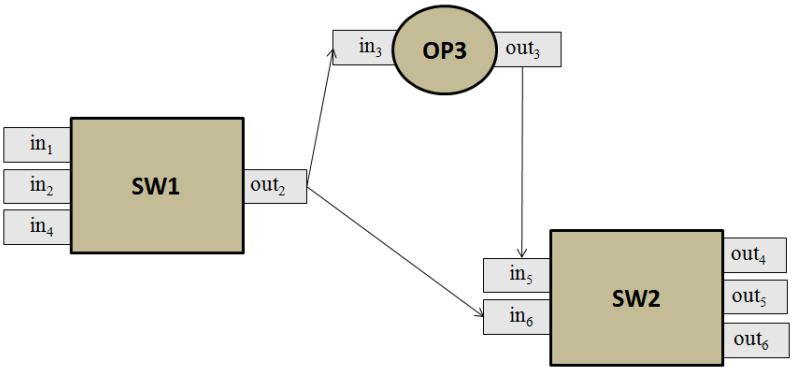


Figura 5.13: Data model attributes exemplification of a hyper-workflow

in natural language with all its implications, identify its domain field and the area of applicability, and specify the data resources needed as input (e.g. what types of spatial data are required);

2. The identification of the operators is the second step of this methodology. The specialists have to analyze the use case, and the already existing operators' data set. Then, they have to identify what data models can be reused and to mathematically describe the new operators that are needed to finalize the use case implementation;

3. The third step involves the graphical representation of the use case scenario (from the previous steps) by means of operators, workflows, and edges. All the previous concepts,

	SW1	**SW2**	**OP3**
Attribute			
nodeID	4	5	6
name	SW1	SW2	OP3
description	First sub-workflow	Second sub-workflow	First operator
type	sub-workflow	sub-workflow	operator
inputs	$< in_1, SW1in.tif, geotif >$ $< in_2, 1, int >$ $< in_4, SW2in.tif, geotif >$	$< in_5, OP3out.tif, geotif >$ $< in_6, SW1out.tif, geotif >$	$< in_3, SW1out.tif, geotif >$
outputs	$< out_2, SW1out.tif, geotif >$	$< out_4, out4.tif, geotif >$ $< out_5, out5.tif, geotif >$ $< out_6, out6.tif, geotif >$	$< out_3, OP3out.tif, geotif >$

Tabela 5.8: Node attributes table exemplification of a hyper-workflow

	edge1	**edge2**	**edge3**
Attribute			
edgeID	7	8	9
nodeStart	4	4	5
nodeEnd	6	5	6
outputIndex	1	1	1
inputIndex	1	2	1

Tabela 5.9: Edge attributes table exemplification of a hyper-workflow

regarding the data structure can be applied.

The workflow's development starts by using the existing resources (e.g. operators and workflows). As new structures are added to the data model, they became available for later usage. Navigating in the hyper-workflows could become a problem, due to the fact that we have to expand each node down to the operators' level. In order to be able to do this, the reference (pointer) concept was introduced that allows the storage of these imbricate structures. Each node has an attribute (*nodeID*) that points to the content of that node. This process continues as long as the *nodeID* points to a non-empty memory location.

From the previous two examples (Figure 5.12 and Figure 5.13) we can see that after developing the first workflow it could be reused in the creation process of the second one. At this point a really big problem arise: when adding an existing workflow (e.g. *SW1*) as an inner node within a hyper-workflow what should be done:

(a) Store a reference to the *SW1* structure?

(b) Create a copy of *SW1* and use it in further processing?

Avoiding residual data, diminish the memory space for storing this information, and optimizing the data model queries are the pro-arguments for adopting the first solution. The easiness of updating all the hyper-workflows that contains *SW1* is another pro-argument. On the other hand, the users may want fractions of *SW1* content and to combine it with other elements. Solution (a) offers no support in this case, and the users need to recreate the entire workflow structure by scratch.

The second solution facilitates the editing process of *SW1* inside a hyper-workflow, but requires the duplication of its entire content. Due to the fact that the development methodology proposed in this thesis is used for describing natural phenomena, and complex use case studies, the second solution was adopted, because there are a lot of changes in the hyper-workflow structure until it reaches its final stage.

In order to ease the workflow's development, a collaborative interactive environment was setup. This way the domain field specialists can create the entire structure by working together, and they could establish what are the inner nodes and the relations between them.

When reusing a node resource within the hyper-workflow, a new copy of that resource needs to be created. This gives the user the possibility to update the new added item without affecting the structure of the original workflow. The change/delete operations of the initial workflow does not influence its copies, and has only a local effect.

Most of the times the complex workflow development is conducted by a group of specialists, usually from the same domain of activity, but geographically separated. In order to facilitate a close interaction between them, a new collaborative environment was setup that allows the online development of these data structures. The inter-specialists collaboration is related with the implementation of operators and their relationship, as elements inside the hyper-workflows.

The online collaborative platform must offer shared access to resources (e.g. operators database, data inputs and outputs, etc.). This means that each workflow node must have some extra attributes:

- **lock**: an indicator that symbolizes if the node is currently in use. If true, the attribute contains the id of the user, or a null value otherwise;

- **history**: a dynamic list where the user could store the previous node's development versions. When needed he could go back to one of the implementations and continue the development process from that point;
- **coordinates**: because the content of the workflow is displayed on a whiteboard, each node has a specific position in the XOY space, stored within the coordinates attribute.

Figure 5.14 proposes a general architecture related to the implementation of these concepts. The *clients management* module is responsible with registering the new users that connect to this environment. At this stage the participant has access to the whiteboard that displays the content of the hyper-workflow, and to the data resources exposed by the *resources management* module. When the user tries to update an existing node he must set its *lock* attribute. At this moment the node in no longer available for the other connected users, until the owner unlocks it. Saving successive versions of the node is possible due to the session data set memory location, available on the shared resources space.

All users have access to the same data resources. Each update of the nodes (e.g. moving them on the whiteboard, adding/changing/removing edges) is visible to all users, due to a notification system that listens to these changes on a given interval. Each new update is encoded into a format known by all parties. Once the notification reaches the users, it is decoded and applied to the proper element.

Different screen resolutions utilized by the participants, generate different sizes of the whiteboard. This issue requires a normalization of the coordinates and an identification of a unique reference point (the upper-left corner of the whiteboard situated at *(0, 0)* coordinates).

In order to apply the conversion algorithm we need some extra arguments: the whiteboard's dimensions of the user that performs the updates (initiator), and the width and height of the whiteboard of the user that receives those updates (receiver).

The initiator sends the node position $P(x, y)$, the width (w) and the height (h) of its whiteboard, measured in pixels. The receiver tries to compute the node position $P^{'}(x^{'}, y^{'})$,

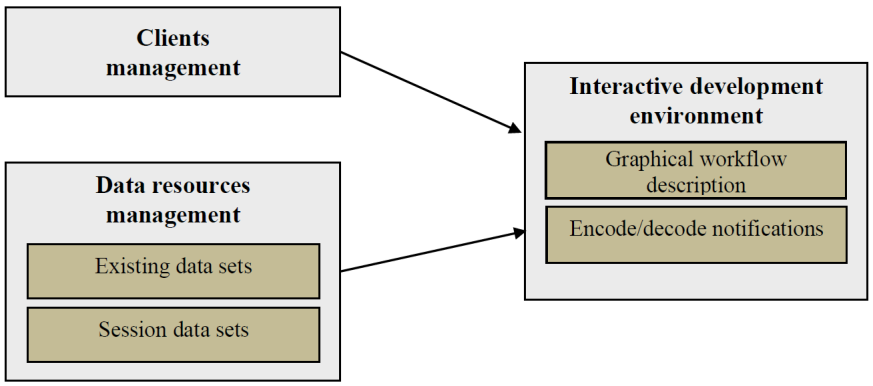


Figura 5.14: Conceptual architecture for collaborative development of workflows

related to its screen dimensions $(w^{'}, h^{'})$. The conversion formula is:

$$\begin{cases} x^{'} = x \cdot \frac{w}{w^{'}} \\ y^{'} = y \cdot \frac{h}{h^{'}} \end{cases} \tag{5.5}$$

4. After describing the hyper-workflow in a graphical manner, it is time to export it into a specific data format that could be used at runtime. This step involves the data processing concept that will be described in the following chapter. To summarize this action: an XML file is generated from this data model that contains the attributes for all the hyper-workflow nodes, and also contains information about the edges (connection) that were established at the hyper-workflow level.

   The XML file acts like a guidance mechanism at runtime, by packaging the data to be executed in a distributed manner, and by indicating the connections between these inner nodes of the hyper-workflow.

   This last step of the methodology involves the execution and the monitor of the processing. Based on the generated XML file, the use case scenario is divided into sub-processes (called tasks). Each task is executed on a specific machine (called worker) from the distributed environment. The monitoring phase identifies the status of all the machines involved in the execution process. We consider the hyper-workflow execution as completed when all workers dispatch the complete status. The difficulty in monitoring such a process is related with the dynamic number of stations, the management of large amount of spatial data, and the extraction of relevant data to display to the users. The visualization and interpretation of results are also part of this step.

5.3.3 WorkflowEditor - an interactive use cases development tool

The interactive modeling of complex use cases (experiments) facilitate the involvement of various categories of users that do not necessary have solid understandings about the scenarios related activity domains. Most of today's applications do not allow the visual description of the experiments and use instead text based editors that leave room for syntactic errors, generated by the users that perform the scenarios' description.

5.3.3.1 General overview

The solution proposed in this section tries to overcome these issues and provides user-assistance throughout the entire development process. The WorkflowEditor tool [78] is part of the GreenLand platform and has been developed for similar purposes as the OperatorEditor, but involved in the workflows management process instead of facilitating the implementation of new operators. The benefits obtained from using this solution are listed below:

- faster development of new complex workflows;
- the usage of interactive techniques (instead of text based markup languages) for describing the use case functionality;
- development of multi-level workflows, where one node is able to contain other workflows;
- syntactic errors reduction by applying automatic validation rules onto the nodes and edges of the workflow;

- collaborative environment for use cases development that provides specialized mechanisms for data sharing;
- direct linkage to the GreenLand platform that simplifies the execution process of the new created workflows.

In its initial state, the WorkflowEditor's repository contains a default set of operators, available for all the GreenLand users. Starting from this default dataset, new scenarios can be generated by using the existing resources or by creating new ones. The following paragraphs describe the theoretical and practical concepts that were used for the development of this module.

The WorkflowEditor can be defined as a middleware layer between the natural language description of the use cases and the GreenLand platform. It allows the interactive specification of the scenarios' functionalities by using visual components, and then automatically converts this description into an internal format, recognized by the GreenLand platform. At runtime, this format provides useful information about: the structure of the workflow, the interaction process between the algorithms involved in the use case description, the input/output types, etc.

5.3.3.2 Workflow layout algorithms

The modeling and simulation of complex phenomena involves a large number of operations that require the usage of workflows with tens of nodes and multiple interconnections. Specific algorithms were implemented for a better visualization and organization of such structures. The following paragraph describes the most important visual aesthetics [78] that were taken into account when implementing this solution:

1. **Minimizing the graph's layout surface**: it is strongly related to the requirement of the correct and intelligible layout of the workflow in a surface whose size is as smaller as possible. The solution is based on identifying the position and the dimension of the smallest rectangle that fully covers all the nodes and edges of the graph. When performing a maximum scale of the graph's content this rectangle is similar to the display viewport;
2. **Creating edges of similar length and angular resolution**: tries to improve the first aesthetics, by creating symmetric positions between the graph's elements;
3. **Storing the blueprint of the workflow**: it is based on the solution proposed in [110] that stores the position, dimension, and order of the nodes, regardless of their previous state. This aesthetics has the effect of minimizing the users' efforts regarding the action of memorizing the entire data structure. By identifying the relative coordinates of all visual elements, the system is able to fulfill this requirement, regardless of the previous actions (e.g. reposition of nodes, updating the length of the edges, etc.);
4. **Symmetry**: it is related to the symmetric display of the nodes against a pivot point that usually corresponds to the center of the graph. Various algorithms that perform the visual representation of the graphs could be taken into account. The adopted solution uses the hierarchic algorithms that display on the first level the leaf nodes, followed by their parents. The process continues recursively until all nodes of the graph are placed on the drawing surface;
5. **Minimizing the number of intersections between the edges**: the usage of elbow arrow connectors is a possible approach for avoiding un-useful intersections. For the

moment the WorkflowEditor tool allows only straight uni-directional edges, due to the high complexity that is required for implementing the above solution. Neither approach is capable of handling workflows with a large number of nodes and edges. In conclusion, this aesthetics is an NP-complete problem that cannot be completely satisfied [111].

There are situations where these visual features exclude each other and require for a compromise solution. The approach used within this section assigns each aesthetics a priority based on which it displays the content of the workflows. The priority level is decreasing from the first aesthetics down to the last one.

Depending of the workflow's category an optimal layout algorithm can be identified that satisfies as much aesthetics as possible. The category is obtained by analyzing the graph from different perspectives: if it is cyclic or not, if it is oriented or not, if a node is able to have multiple parents, etc.

Usually the rendering rules are related to the representation of the edges of the graph (Figure 5.15) as: straight segments, elbow lines, virtual lines and columns that partition the entire surface as matrix cells. The reduction of the number of intersections between the edges of the graph represents an immediate effect of using the elbow segments. Placing the nodes on the same position as the virtual matrix's cells satisfies the second aesthetic (creating edges of similar length and angular resolution).

There are four classes of algorithms (circular, orthogonal, force based, and hierarchic) that can be applied for automatic layout of the nodes and edges of the workflows. The first category [112] allows the nodes positioning around a virtual ellipse. They are recommended in case of small networks, otherwise they do not provide an intelligible vision of complex graphs.

The orthogonal display of the nodes [113] is based on repositioning the elements within the center of the virtual matrix until the majority of the edges become parallel to one of the coordinate axis. This representation manner diminishes the number of intersections between the edges of the graph.

The force based algorithms [114] satisfy the requirements of symmetry and the similar length and angular resolution edges. The solution consists in periodical reposition of the nodes based on the forces that exists between them. At each step a local optimum is computed, based on the previous mentioned heuristics. The process continues until the global optimum solution is achieved that fulfills as much visual aesthetics as possible.

The solution adopted within the WorkflowEditor tool uses the hierarchic algorithms [115] for visual layout of the graphs. The process starts from identifying the leaf nodes and then tries to reposition the parent elements on consecutive levels. The algorithm continues until all the graph's items have a specific position. The performed experiments showed that this solution

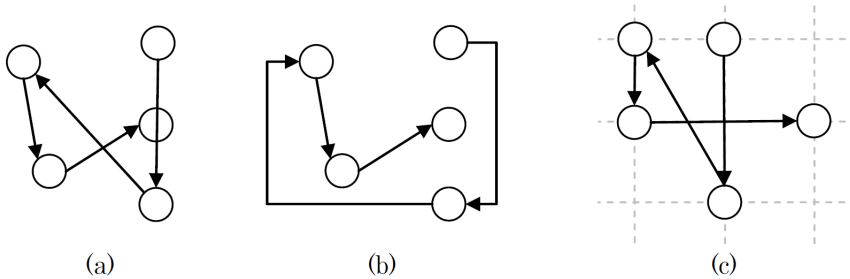


Figura 5.15: Workflow rendering as (a) straight segments, (b) elbow lines, (c) nodes positioned within the cells of the virtual matrix

provides reasonable results even when dealing with complex workflows, with a large number of nodes and edges.

The WorkflowEditor has a predefined set of constraints and rules that are automatically applied onto the graphs' items in order to reduce the number of syntactic errors that might be introduced by the users that model the workflow. These validations assure the data model correctness that is further used for executing the use case over different computing infrastructures. The most important constraints and rules imposed by the system are:

1. **Nodes overlap**: it is not allowed to have multiple nodes at the same position. There should be a minimum distance (d) between the adjacent items and in case that the user overlaps two nodes, the system automatically reposition the second one in the nearest area that satisfies this constraint;

2. **Nodes uniqueness**: each new item that is added to the workflow's content should be uniquely identified. When the experiment (scenario) requires multiple instances of the same operator, they should have different names, even though internally they perform the same functions;

3. **Outputs connection**: each node has a finite number of input and output parameters. This constraints is related to the fact that the user should not leave any output unconnected, and instead it should specify proper connections between them and the rest of the inputs;

4. **Specification of the edges**: they represent an uni-directional communication between the output of one node and the input of another one. The action of connecting items from the same node is considered incorrect;

5. **Types matching**: each edge should connect inputs/outputs of the same type;

6. **Acyclic graph**: currently there is no support for modeling such kind of data structures, due to the fact that they need advanced management mechanisms. On the other hand there is a high probability of introducing at runtime infinite looping;

7. **Not-empty graph**: each workflow should contain at least one node whose inputs and outputs are properly connected to data resources.

One important characteristic of the WorkflowEditor tool is the fact that it provides a collaborative environment where users (from various geographic locations) are able to contribute to the development of the scenarios. Most likely they are using monitors with different resolutions that need to display the correct position of the items. There are two possible solutions in order to overcome this issue: the usage of normalized coordinates or the screen coordinates for positioning the graph's elements on the drawing surface. In both cases we need a reference point (the upper-left corner that has the (0, 0) coordinates) that is used to compute the rest of the points.

The first approach needs normalized values within the [0; 1] range. The main disadvantage of this method consists in converting large sets of coordinates and storing more data quantity (the floating numbers require more memory space than the integer ones). Based on these observations the WorkflowEditor tool works with screen coordinates in order to notify the users about the changes that were performed during the development of the use case. For rendering the same information at different resolutions this module uses the formula 5.5 and the approach described in that section.

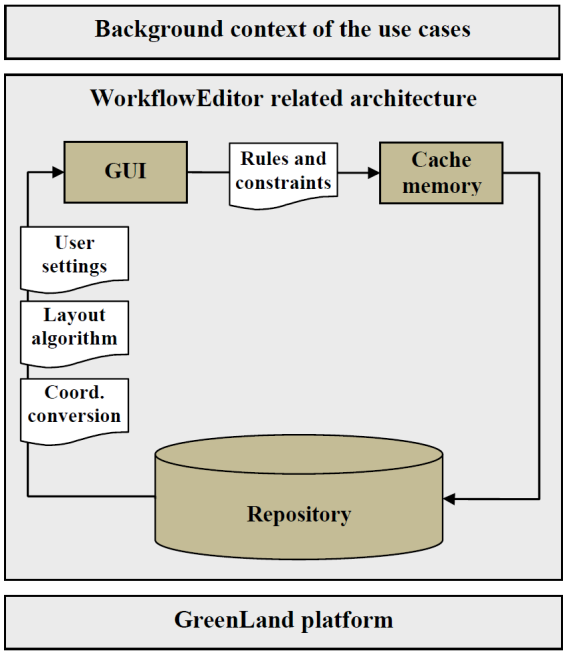


Figura 5.16: WorkflowEditor system related architecture

5.3.3.3 Basic functionalities

This section describes the most important features provided by the WorkflowEditor module. The concepts introduced earlier require the presentation of the system related architecture (Figure 5.16) in order to make a general idea of where to place this module within the GreenLand platform. As said before, the WorkflowEditor tool acts like a middleware between the natural description of the use cases and the GreenLand framework that performs the actual execution of the workflow. In other words, it provides specialized techniques that facilitate the workflows' development and avoids the need to describe them in a proprietary markup language (as the majority of today's spatial data related applications do).

The background context level from Figure 5.16 contains documented information about the use case that is going to be implemented: mathematical description, the input and output parameters and their type, datasets that are used for testing the experiment, the connections between the algorithms that are going to be integrated as inner functionalities, etc.

This information is then parsed to the second architectural level that contains the actual features of the WorkflowEditor. The repository module stores the compact information about the existing basic operators and workflows, and it can be seen as a specialized database that is extended to support data from multiple activity domains (e.g. hydrology, archeology, physics, etc.). The content of this repository is available to the users throughout the entire development operations. The graphs rendering is a 4-step process that involves:

1. The extraction of relevant information (from the WorkflowEditor repository) about the workflow that is going to be drawn;

2. The conversion of the coordinates of each item in order to be properly displayed on the user's screen, regardless of its resolution;

3. The usage of hierarchic algorithms for positioning the items in an intelligible manner. Another alternative is to use the same settings as established in the last working session, and to load the graph's elements at their original position without the need to apply the automatic layout techniques;

4. The creation of customized graphical elements by applying certain fill colors, transparency, styles for drawing the edges, etc. The user settings option (Figure 5.16) stores all this information and applies it when loading the current workflow.

All the four steps are performed before the actual display of the workflow. Once the content is available in the graphical interface, the user is able to update it by adding existing resources (operators or workflows) in order to match the natural description of the use case. Each user-action is validated based on the constraints and rules described earlier. The intermediate data are stored in the cache memory of the system that is available throughout the entire current working session. When saving the content of the workflow, the information from the temporary buffer is transferred to the database repository, from where it can be reused at any time.

The WorkflowEditor module provides all the necessary tools in order to facilitate the workflows development, but it does not contain the mechanisms for executing these use cases. This is the main reason why the WorkflowEditor module was integrated within the GreenLand platform that is capable of providing such processing services.

The graphical interface of the WorkflowEditor framework is highlighted in Figure 5.17. The available resources are classified in two lists: *Basic operators* and *Complex workflows* (the left hand side of the interface). The first category contains operators that process various types of satellite images (e.g. NDVI, DOS, GEMI, etc.) while the existing workflows are presented as content of the second list (e.g. Density slicing).

The drawing surface represents the main area of the WorkflowEditor and displays all the inner elements of the graph. The inputs are marked with small circle primitives and are placed

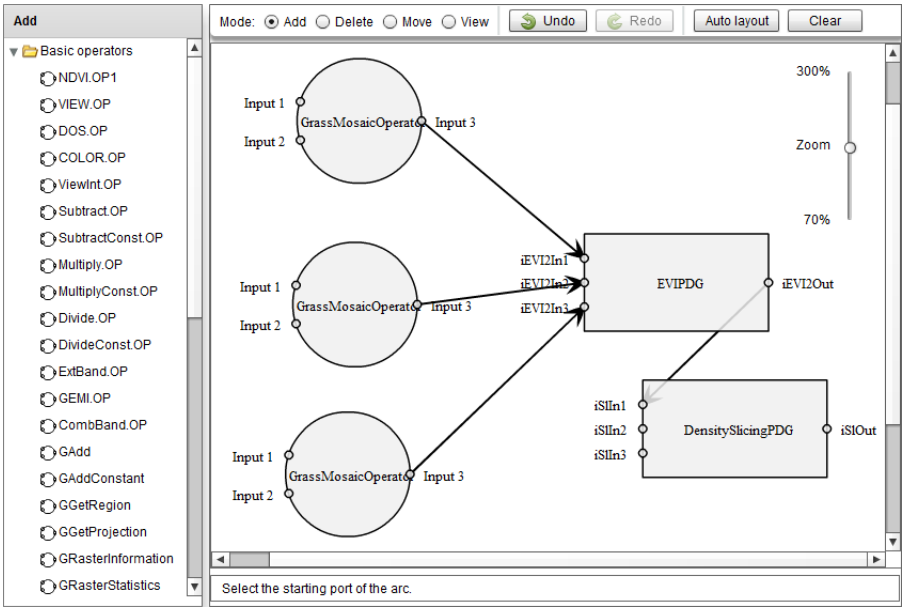


Figura 5.17: The graphical interface of the WorkflowEditor tool

on the left side of each node. Similarly, the outputs have the same geometric shape and are positioned on the right side of the items. Based on the theoretical concepts described in the previous sections, the graph contains uni-directional edges that connect the output of one node to the input of another one. The most common operations performed by the users are:

- Adding new graphical elements onto the drawing surface: the users are able to create new data structures based on the existing resources provided by the two left hand side lists. These structures are called workflows or hyper-workflows, depending on their inner content. Adding new elements is done by selecting the desired algorithm and then placing it (with the help of the mouse device) onto the editor's surface. At each user-action the system automatically verifies if the nodes are overlapping, and performs proper adjustments in reposition them is safe regions;

- Edges specification: after selecting the output of a node, the system automatically highlights the compatible inputs from all other nodes, while the non-candidate entries are disabled so that the user cannot introduce flow related errors. At different zooming levels, the selection of a specific input is hard to accomplish. According to this aspect, the system defines an extended area for them and even thought the mouse cursor did not accurately reaches the item, the algorithm deduces the user's intention and automatically performs the connection;

- Removing the graph's items: the WorkflowEditor has multiple working modes (Figure 5.17). One of them (*Delete*) is used for removing the existing items from the graph. This action can be performed by clicking with the left mouse button onto the element that is going to be removed. The redo and undo options are available throughout the entire working session;

- Updating the position of the nodes and edges: each graphical item has a specific (x, y) position on the drawing surface. When the user updates the placement of these elements, new sets of coordinates are computed. By using the layout algorithms, the system is able to automatically position the graph's elements in such way that satisfies as much visual aesthetics as possible. It is important to take into account the fact that the performance of these algorithms is strongly related to the complexity of the use case;

- Visualizing the information about a specific item: there are cases when one node contains other embedded workflows. The WorkflowEditor module provides specific tools for handling these situations, by opening the content of that node in another active window. The system is able to manage multiple windows at the same time, while the user can navigate them by using the mouse or the directional keys. At any time the user has the possibility to visualize the details about the selected node: the types of the input and output parameters, the inner content of the node, its name and description, etc.

5.4 Conclusions

Modeling the spatial data related processes is one of the first stages proposed by this methodology. It comes before the data execution because it provides abstract models that only need to be instantiated at runtime. The efficient communication between the entities of these models can significantly improve the execution time of the use cases over different infrastructures.

This chapter describes the theoretical concepts that were used for representing various scenarios (experiments, natural phenomena) as mathematical graphs whose description is closely

related to the natural human understanding. For this purpose, two main concepts were introduced: basic operator (encapsulates the functionality of an algorithm, method, or function) and workflow (used for describing the complex use cases that require the interconnection of multiple operators).

The formal description of the two concepts allows the modeling of scenarios from different activity domains, such as: hydrology, meteorology, physics, biology, etc. Possible limitations occur only when integrating these solutions within software applications, due to the lack of:

- technology;
- specialists that have solid understandings about these domains;
- finding the proper experiment's description that matches the concepts proposed within this chapter;
- relevant input datasets used during the testing phase of the applications.

Two procedures were defined in order to model the real use case scenarios as operators and workflows. Both of them start by identifying the requirements and by describing the problems in natural language. The next step is about giving the use case a formal interpretation and to find the relationships between the algorithms (formulas, methods) that were described in the previous step. Each algorithm should encapsulate the functionality of a group of requirements. Implementing these features into high level programming languages is also part of these procedures, because it facilitates the integration of the use case with the GreenLand platform that provides specialized tools for executing them over different computing infrastructures.

In order to facilitate the modeling of the real use cases, these theoretical concepts were implemented within two software applications: OperatorEditor and WorkflowEditor. They provide interactive and intuitive techniques for the scenarios development and do not require the users to handle text based editors (based on certain markup languages) for describing the same concepts.

The concepts proposed within this chapter represent the solutions that were found for the limitations that exist in the spatial data processing related domains. The following paragraphs describe the most important issues related to modeling the spatial data processes:

- Most of today's software applications are limited to a specific activity domain, and do not provide extensions towards new research directions. This situation requires the users to distribute their work onto multiple applications. For example: one platform is used for data retrieval, another one is able to perform the data processing, while the third tool facilitates the analysis and visualization of the results. As the number of frameworks increases, the users' activities become harder to accomplish, because they have to learn how to operate the mechanisms for each one of these applications;
- The state of the art analysis highlighted the fact that there are not so many solutions that provide interactive description of the complex use cases, and instead require the users to perform this description in proprietary text based markup languages. This limits the usage of these techniques only to the highly-trained specialists that have solid understandings on both Earth Science and computer science domains.

The following paragraphs highlight the most important solutions (contributions) that were proposed in order to partially or fully overcome the issues encountered when modeling the large volume of spatial data related processes:

1. The introduction of two concepts (basic operators and complex workflows) that can be used for modeling the use cases from various activity domains. This contribution (in combination with the GreenLand services) can be used in order to solve the domain activity extension issue;
2. The abstract description of the basic operator and complex workflow;
3. The identification of a predefined set of types, rules, and constraints that can be used for modeling the inputs and outputs of the operators and workflows;
4. The development of two applications (OperatorEditor and WorkflowEditor) that implement these theoretical concepts and provide interactive techniques for developing complex use case studies.

Chapter 6

Spatial data processing

The processing of large volume of spatial data is one of the research directions that were intensively studied within this thesis. In order to achieve this goal, a complex data model was already defined in the previous chapter that is able to represent large use case scenarios (experiments, phenomena) that take place in the real world or that are simulated for specific purposes.

One important characteristic is that fact that the processing of spatial data requires valid resources for instantiating at runtime these data models. Depending on the activity domain that the application addresses, finding the proper information is a hard thing to achieve. Usually, these data are stored on the local computers' of the domain field specialists or on remote repositories. In both cases, the methodology proposed within this thesis must be able to provide specific techniques that facilitate the extraction of relevant data that are going to be used as inputs for the complex scenarios (experiments) described as workflows.

The remote data repositories are updated periodically with new information from various artificial sensors. This way, they offer enough resources that can be used in different prediction experiments. The spatial data execution takes into account this possibility and provides the proper mechanisms for near-real time data processing from these remote repositories, all the information transfer being done transparently from the end-user.

This chapter describes the execution of the use case scenarios (represented through hyper-workflows) over distributed and parallel infrastructures. The entire execution is based on the XML file, generated on the fourth step of the workflow development methodology.

The data processing concept is related with the execution of an algorithm, written in a high-level programming language that is capable of generating platform independent executable (bash) files that could run over different operating systems (e.g. Linux, Windows, iOS). Grid infrastructure is used for exemplifying some of the introduced concepts.

Besides these considerations, that were only for experimental purposes, the methodology proposed in this report addressees all sorts of distributed infrastructures (e.g. Grid, Cloud, clusters, mainframes), and does not limit to a specific operating system.

The processes taken into account by this methodology were listed in the beginning of the chapter (section *Types of processes*). These examples are exclusively from Earth Science domains (e.g. land cover, land use, hydrology, etc.). Due to the flexible structure of the data model used for representing these algorithms, operators and workflows from other domains (e.g. medicine, physics, archeology, etc.) could be easily integrated [116], [117], [118].

Currently, the operators and workflows support only 11 data types (listed in Table 5.5). Among these we could find: numbers, strings, raster and vector data formats, projection and

metadata files, and satellite images (converted to GeoTIFF format). For the moment, the digitized data field measurements and the satellite images are the only ones that could be included into the spatial data category [119]. But the proposed framework allows easy integration of other data types: airborne images, radar images, and other formats.

6.1 Processing infrastructures

The data processing is possible on different distributed, parallel, and standalone infrastructures. Most of the times, the processing performance is closely related to the hardware components. A general rule could be derived from this remark: the more powerful the computation resources are, the better execution times are obtained. The following paragraphs describe the main platforms that are supported by this methodology:

1. **Single-core infrastructure**: Single-core machines are represented by ordinary computers that do not provide high power computation capabilities, but instead the data executes as a monolithic block of information. Most GIS systems may adopt this approach, especially if the execution time is not relevant (this is the case were independent data processing takes place, and their execution does not exceed a few seconds).

   The hyper-workflow data structure is dedicated to complex use cases that involve large time consuming algorithms, and large input data sets. Executing these algorithms on a single-core machine is not a feasible solution, due to its limited computation and storage capabilities. Even though the proposed methodology addresses this type of infrastructure, it is not recommended to use it when experimenting with complex case studies.

   When we talk about implementing a simple algorithm (stored within an operator data model) that performs simple computations, the single-core machines represent one solution to be taken into account. At this point no parallelism is necessary, because the entire algorithm is executed on a single processor, in a sequential order, established when implementing it into a high-level programming language;

2. **Multi-core infrastructure**: It is represented by a monolithic hardware component that integrates two or more Central Processing Units (CPU) that are called cores. These cores execute at the same time multiple instructions (e.g. add, mode, branch) in a parallel manner, increasing this way the processing speed of the entire execution. Nowadays, the CPUs integrate two (e.g. Intel Core Duo [120]), four (Intel i3, i5, and i7 [120]), six (e.g. Intel Core i7 Extreme Edition [120]), or eight (Intel Xenon [120], AMD FX-8150 [121]) cores connected through a high-speed bus that assures the data transfer between them. A multi-core processor may use one shared memory (a common memory shared by all cores) or more distributed memories (each core has its own local storage).

   There are two types of multi-core processors: general-purpose and application-focused. The first class provides the same computing resources for all the processing needs, regardless of the application type. On the other hand, the application-focused type provides different cores for different means in processing the application. For example, if there are two available cores, one could be used to process the audio data, while the other one is specialized on processing the video information [122].

   When implementing applications that process large volume of data, standalone machines cannot perform these executions in a decent amount of time. That is why a parallel approach should be taken into account. Multi-core architectures are based on many

processors and associated caches and memories. This infrastructure leads to performance enhancement, reduces power consumption, and provides better tasks management [123].

Not so long ago, all software applications were design to function on single-core stations. By introducing the new multi-core architecture, a new problem arose: a large part of those applications needed to the re-implemented in order to benefit from the advantages brought by this new solution. New software techniques were defined (such are multi-threading) which allows the shift from the sequential to parallel technology.

The C/C++ programming languages didn't scale that well for multi-core purposes, due to their sequential nature of design. New solutions had to be discovered. One of them is the Open Multiprocessing (OpenMP) [124] that provides an API that offer support for parallel processing in different programming languages (e.g. C, C++).

There are several issues regarding the development of applications that efficiently use the multi-core processors:

- **Partitioning**: represent the stage of partition the algorithm into smaller tasks that work in parallel on the cores of the processor. In our case we may want to divide the entire hyper-workflow into processes (comparable in complexity) and execute each one onto a different core. A partitioning algorithm is required in order to divide the workflow into equally complex structures.

  One solution [125] could start from partition the graph into smaller groups that have the same number of nodes. After that an approximation function could be used to compute the complexity (weight) of each group. A re-organization of groups is required when found large weight differences. The main idea is to achieve a load balance of cores' CPUs performance, while minimizing the inter-cores communication costs;

- **Communication**: most of the applications could not be divided into fully independent tasks, but instead they need to share data in order to compute the entire algorithm. In our case inter-nodes communications are assured by the edges of the hyper-workflow that redirects the data output-resources of a node to the proper inputs of other nodes;

- **Agglomeration**: represents the phase that tries to re-organize the tasks (e.g. to combine them in order to reduce their number and to reduce the data transfer rate) in order to achieve the best performance from all the cores of the processor. Data replication is also one of the issues that have to be address at this stage of development. In our case we need to replicate the outputs of each group of tasks between the cores that are connected by workflow edges to this group;

- **Mapping**: it is an optional phase that does not take place on stations that provide automatic tasks schedulers. In all other cases the developers of the application need to specify were each task is going to be executed.

  In our case we start to assign tasks to each core, until all of them are busy. If after this first iteration there are tasks that were unassigned, the mapping algorithm should wait until one of the cores finishes the processing and assigns it with another task. The mapping algorithm continues until there are no more tasks in the waiting queue. The final output of the hyper-workflow is a combination between the intermediate results generated by each core, based on a specific formula that is described through the edges of the hyper-workflow.

3. **Cluster infrastructure**: Until a few years ago the parallel supercomputers represented the state of the art related to high-performance computing. Their performance rapidly increase at the beginning, but began to decline in the new stages of development. The future improvements are constrained by different physic, thermodynamic, and financial principles. One possible solution to these problems is to have multiple stations (not necessarily as powerful as these ones) connected into a high-speed network and try to coordinate their efforts by using different scheduling algorithms [126].

   By computer cluster we understand a set of connected computers that are working together, as an integrated resource, or individual computers. They are mainly used to improve computational performance and also the data availability. Cluster technology provides a processing boost for high computation applications. The most powerful cluster (Titan) has 560640 processors, 261632 accelerator cores, and 17.59 petaflop/s computation power [127].

   Two common characteristics are related with the load-balancing and high-availability techniques. In load-balancing clusters all the nodes share the same workload in order to improve the performance. On the other hand, the high-availability clusters have redundant nodes that store the same information in multiple places. This is useful especially when dealing with important data and the need to eliminate information loss issues [128].

   One of the most important aspects in the cluster architecture is the interconnection of technologies, based on memory storage and communication protocols. There are four types of such interconnections: I/O attached memory - message based, I/O attached memory - shared storage, memory attached - message based, and memory attached - shared storage.

   Choosing the right combination requires a compromise between the latency and bandwidth. Latency measures the time required by a data package sent from one computer to reach another one, including the overhead introduced by preparing this data package. Bandwidth represents the amount of bits that can be transferred between the two computers in a second.

   At the first glance the Transmission Control Protocol/Internet Protocol (TCP/IP) [129] is the most adequate solution for clusters architecture. It is easy to use, applicable on wide-area networks, and reduces the number of resends of a message in case of errors on the network. On the other hand it has one major drawback: latency incensement for local-area networks.

   At a closer look the cluster infrastructure has few network errors and the distance between stations is small, comparable to the wide-area that uses the TCP/IP protocol. Regarding all these aspects, new cluster-based protocols were implemented (e.g. Active Messages [126] or Fast Messages [126]).

   The Cluster Management Software (CMS) is used to offer management solutions regarding the jobs (tasks) submitted to the stations of the cluster. There are six types of requirements specified for this software:

   - **Efficient resource management**: concerns with the station nodes management: host name, IP address, memory type and size, etc. It needs to provide solutions for adding/removing nodes in/out of the system;
   - **Load cluster configuration**: the cluster infrastructure contains a group of at least two machines. One of them is the main station, were information about the role of all other machines are stored. Based on this information, the CMS system should

be able to assign roles to each station or to a group of stations, and to periodically interrogate their status;

- **High availability**: it is one of the main features of the cluster architecture, and it should provide an alternative to fault-tolerant services. In other words, if one station fails, the CMS should try to recover it by using the configuration data. The cluster reconfiguration, in such situation, is usually reduced to replacing the station that failed with a new one, and transfer all the processes to the new added computer;
- **Security**: data security is another important issue that needs to be addressed by the CMS system. When the volume of information is not that high, data encryption is enough. When dealing with huge amount of data, the encryption algorithms generate large overhead in the inter-stations communication protocol. Several solutions were given to this problem by securing the data that reaches the station processor or the Network Interface Card (NIC) [130]. An additional method is to request authentication mechanism for data access;
- **Load balancing**: offers support for distributing the processes on the hosts of the cluster, based on the process complexity, load of the host, network traffic, and number of users. In order to use this advantage the software applications need to support task and data parallelism.

When developing applications that are executed on clusters, most of the times there is the need to debug and monitor the entire implementation process. This requires specialized tools that must implement different facilities, as the ones described in High Performance Debugging Forum (HPDF) specifications [131]. TotalView [132] is one of such tools that offer support for debugging on clusters that use the MPI technology;

4. **Cloud infrastructure**: Cloud infrastructure is a paradigm that focuses on sharing data and computations over a network of computers, and provides services over the Internet on a pay-as-you-go basis. It is a type of parallel and distributed system, integrating a collection of virtualized computers that could be described as multiple resources based on Service Level Agreements (SLA), established between the service provider and consumers.

   The general architecture of Cloud computing (Figure 6.1) integrates the server infrastructure level hosted within the Cloud, and the client applications that consume the services provided throughout the Application layer.

   The Infrastructure as a Service (IaaS) is the first-service model of the Cloud architecture that delivers physical and virtual machines, and other IT modules, based on virtualization.

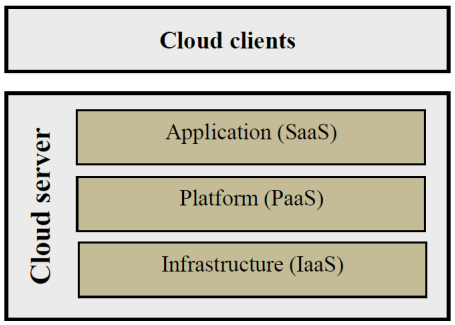


Figura 6.1: Cloud computing conceptual architecture

The virtualization concept allows the placement of hardware modules into independent and scalable environments, regarding the CPU and memory storage devices. The IaaS integrates: server, virtual machines, load balancer stations, and interconnection hardware equipment. The Cloud clients "consume" these services by requesting the installation of different operating systems, firewall systems, and other software applications that fulfill their needs. The Amazon EC2 [133] or Windows Azure Virtual Machines [134] could be included in this category of service providers.

The PaaS (Platform as a Service) level delivers a computing platform that provides software applications, such as: programming languages, web servers, database tools, etc. This layer acts like a middleware that manages different types of services used in the request mechanism for the bottom and the upper levels. From users point of view, the PaaS is used as a hosting infrastructure for their application. They have access to the applications installed on this infrastructure, but they do not control the IaaS. The Google App Engine [135] is an example of platform that could be included in the PaaS category.

The Software as a Service (SaaS) layer provides services that could be used in different types of applications: desktop, web, or mobile. The main advantage of this approach is that the client does not need to install and configure on its machine all these services, reducing this way the software maintenance and support. Another advantage is the scalability of the applications, by setup additional virtual machines whenever is the case, while the load balancers are capable of redistributing the tasks to the new virtual machines. The consumers of the SaaS layers use the services provided by these software applications, but they do not have access to change them or to change the hardware or network infrastructure that hosts these applications. The Google Apps [135] is a good example of SaaS applications.

The Cloud clients represent the users that consume the services provided throughout the three levels. The users-services connection is possible based on client devices (e.g. computers, tablets, smart phones, etc).

There are four types of Cloud models: public, private, community, and hybrid. The first one offers its services to the clients from third party providers only via Internet (the direct connectivity is not possible). The private Cloud offers its services only for organizations, and acts like a proprietary infrastructure that provides services to a private network (usually accessible behind a firewall). When a group of communities share the same infrastructure for specific goals they could use the community Cloud. The hybrid version represents a combination of the previous ones, and offers support for both companies and individuals [136].

The Cloud computing has the following characteristics:

- On demand self-service: refers to the fact that a user could access the Cloud services without interacting with the Cloud provider;
- Scalability: it is one of the main characteristics of the Cloud computing and is related with the ability of up-scaling and down-scaling the hardware resources used by the consumers. From the user's perspective the Cloud infrastructure provide unlimited computation power and storage resources. In order to achieve this, the scalability has an important role;
- Measured services: used by the Cloud provider for billing the users as a response to their utilization of services;
- Network access: offers different access means (e.g. Internet, firewalls, etc.) to the services exposed by the Cloud provides;

- Easy implementation: the users' requests in offering Cloud services, regardless of their type, are easy to fulfill, without the need to purchase hardware and to install different software applications;
- Quality of service: usually the Cloud providers offer support and maintenance in all situations. These quality agreements are established between the clients and the providers based on a Service Level Agreement;
- Security: usually the Cloud infrastructures provide strict security policies, including data encryption and users authentication.

The main difference between the Cloud and the cluster computing is the Cloud dynamic nature regarding the ability to up-scale and down-scale whenever the situation imposes it. Usually in the cluster environments the hardware resources are pre-configured. The applications development for these two distributed infrastructure is based on the same concept of parallel programming. In our case the hyper-workflow data model has to be partitioned into groups of tasks, where each task is executed on a different machine. The scheduling tools perform the same steps on both the Cloud and the cluster infrastructure;

5. **Grid infrastructure**: It could be described as a worldwide computer network that offers support for storing and processing large volume of data. Built over the Internet layer, this infrastructure uses the existing Internet communication protocols, but it adds some new features like: GridFTP [137] and Grid services [138]. The difference between the Grid and the cluster infrastructures is that the first one is more heterogeneous, geographically spread, and more dynamic regarding the integration of new working stations.

   For certain applications that support partitioning in multiple tasks, the Grid acts like a parallel computing. In this case the following question arise: what is the difference between the Grid and the supercomputers (mainframes)? To answer this question we have to take into account that these tasks are executed on different Grid worker nodes (e.g. physical stations with their own CPU, storage systems, and network interfaces). In case of supercomputers, these tasks are processed on different CPUs from the same physical machine. The high-speed data transfer protocols that connect these CPUs, is the main advantages over the Grid computing.

   The Cloud and the Grid computing have the same vision: reducing the cost of data processing, increase performance, and address applications that process large volume of data. Differences between these two concepts arose at different levels [76]:

   - **Business level**: the Cloud infrastructure adopts a pay-as-you-go system that provides certain services, consumed by the users by paying an amount of money related to the SLA document. The business model in Grid is project-oriented. There are some available stations that offer free access to all the Grid community. In order to be a part of that community, a VO membership is required, and a contribution with hardware resources to that VO;
   - **Architecture level**: the Cloud services may be accessed directly, by using existing protocols (e.g. SOAP) and paradigms (e.g. REST). The Grid computing defines a set of services build on standard protocols and middleware tools;
   - **Security level**: the Cloud security is based on the Web forms model that allows users to create an account with a personalized username and password. The Grid security is more complex. In order to have access to resources, the user must be recognized by a Certificate Authority (CA) that creates a certificate available only for that user.

The general architecture of the Grid computing (Figure 6.2) highlights three interconnected levels:

(a) The Grid infrastructure integrates farms of computers, with different computation and storage capabilities, and different software applications (e.g. operating systems, programming languages, etc.). The storage nodes are called Storage Elements (SEs), while the computational stations are called Computing Elements (CEs).

Information security is another important aspect that targets the Grid platform. The Grid Security Infrastructure (GSI) becomes a standard security issue for all the Grid middleware tools. GSI contains a set of protocols that allow users and applications to access the Grid's shared resources. As extra features, the Grid implements Ipv6 communication protocols and GSS-API [139]. The authentication is based on Grid certificates, emitted by an authorized Certificate Authority.

If two entities contain valid Grid certificates, emitted by an authorized CA then these entities could access one's another resources. This mechanism is called mutual authentication. In other words if entity A communicates with entity B, then A must authenticate B and B must be able to authenticate A.

In order to reduce the number of user authentications, the GSI implements the Single Sign-On (SSO) mechanism. If the Grid execution process requires multiple resources or if other entities need to access shared resources on the behalf of this process, than the proxy mechanism could be used. The inconvenient of multiple authentications could be avoided by using the SSO mechanism and the proxy certificate.

The proxy has the same basic principles as the X.509 certificate, excepting the fact that it is signed by the owner entity and not by the CA. The lifetime of the proxy certificate is limited (usually 24 hours). This way even if this proxy is hacked, due to this restriction the damages have a smaller impact that in the case of the Grid certificate [139].

Each computer is called worker node, and it interacts with the other nodes based on communication protocols. When a node tries to access resources from different locations, the SSO concept is applied by using the rights delegation generated by the proxy service;

(b) The middleware layer represents the inner level between the Grid infrastructure and the software applications that utilize hardware resources from this platform [140]. In other words, the middleware provides:

- Support for integrating these software applications into the Grid environment;

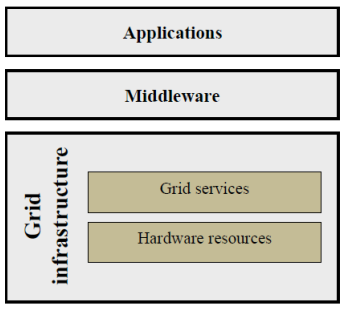


Figura 6.2: Grid computing conceptual architecture

- Access to hardware and software resources within the infrastructure;
- Rapid application development.

For interacting with the Grid infrastructure, the most used middleware systems are gLite [141] and GlobusToolkit [142]. These tools handle the security issues, data management, execution management, schedule tasks to run in the parallel and distributed environment, monitor of these tasks, etc.;

(c) The top level of Grid computing is represented by software applications from different domains of activity, such as: Earth Science, education, medicine, archeology, etc. This thesis describes the environmental applications (represented as models) that simulate or predict specific Earth phenomena, based on some already known input data set.

In order to obtain valid results, a large volume of data needs to be processed. In fact we could establish a direct relationship between the data volume and the predictions accuracy: as the volume of processed data increases the prediction model offers more accurate results.

The applications from this level benefit of the Grid infrastructure computational and storage capabilities in order to process large volume of data, in a distributed and parallel manner. A direct connection between levels 1 and 3 is not possible, but only through the middleware layer that listens to the application requests, interprets them, and sends them to the Grid infrastructure for further processing.

After highlighting the most important characteristics of the five infrastructures it is worth mentioning that the experiments regarding this development methodology were mostly conducted over the Grid environment, but also over powerful multi-core machines. The Grid offers the most complex computer networks that do not require any financial efforts in order to perform the system implementation. The hardware resources are part of the envirogrids.vo.eu-egee.org VO that integrates one Computing Element (ce01.mosigrid.utcluj.ro) and one Storage Element (se01.mosigrid.utcluj.ro). The computational resources consist of 128 physical CPUs, with a total of 1024 logical cores. The storage capacity is approximately of 13 TB.

The general execution of data over the distributed and parallel infrastructures involves four main stages (Figure 6.3): pre-processing of data, data execution, monitoring the execution status and provides feedback to the client applications, and result interpretation and analysis by certain user's categories.

6.2 Pre-processing phase

The pre-processing phase integrates a set of operations over spatial information and data models that are performed before the actual processing, and allows users to: specify the description of the experiment within the GreenLand platform, retrieve the relevant input resources from local or remote repositories, setup the additional dependencies that are needed at runtime, etc. Beside the theoretical description of the solutions, some GreenLand examples are also highlighted, in order to better exemplify these concepts.

6.2.1 Offline adjustments of the workflows

The workflows from this title refer to the GreenLand entities (contain the general use cases description) that can be executed over the computing infrastructures. This is not a mandatory step and should be apply only when the natural description of the experiments was not defined

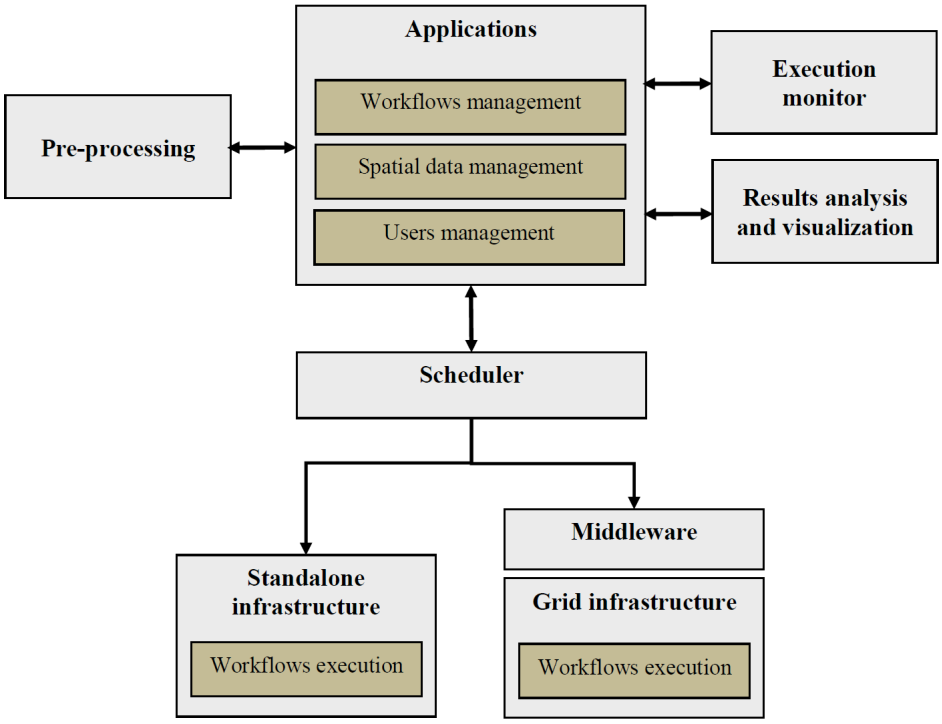


Figura 6.3: Data execution flow over distributed and parallel infrastructures

as GreenLand workflows. The offline adjustments of these workflows contain the operations that are performed before integrating them within the framework:

- The definition of the data models (see the previous chapter) that store the description of the use cases that are going to be executed;
- The implementation of the basic operators into high level programming languages and the linkage of the generated executables to the GreenLand platform (by using the OperatorEditor tool);
- Using the existing resources in order to model the behavior of the experiments as complex workflows through the WorkflowEditor application. Once this phase is completed, the workflows can be integrated within the GreenLand system;
- Setting the specific accessibility attributes that have a global effect over the entire users' communities registered in this framework.

6.2.2 Spatial data retrieval process

A specific set of tools will be very useful to download and store the required global datasets, allowing users to extract them within required geographic extents and temporal intervals. The following paragraphs describe a range of features that need to be taken into account to successfully retrieve these data:

1. Large storage requirements (terabytes of data for each global data set);

2. Automation of processing (to avoid manual handling of data processing steps);
3. Wide variety of raw data formats (each sensor stores data in specific format and needs specific codes);
4. Reproject various data formats into common projection;
5. Spatial sub-setting of global datasets in specific areas of interest (e.g. Black Sea catchment);
6. Temporal sub-setting of data;
7. Data resampling in order to deliver information at certain intervals (e.g. 1-3-6-9-houly sampling intervals).

Usually, remote sensing products are distributed via FTP servers and provided in a compilation of multiple MODIS bands. There are two methods that allow data retrieval from these servers:

- **Manually**: first of all the user has to select the data, in a file by file process, from the remote repositories. The process continues with downloading the images from the FTP server, extracting the bands from the multi-band images, apply sub-setting algorithms in order to get the relevant information for the interest geographical area. After all these steps are manually performed, the final result could be used in other purposes or it could be shared among the scientific communities. In both cases the result should be uploaded to another server (computer) from where it could be retrieve by external users, systems, and platforms.

  When working on a small scale, this method is quite practical. But when involving complex use cases, performing all these steps manually is totally inefficient and time consuming. Because the methodology proposed in this report, offers guidelines for developing software applications that process complex case studies, we could focus on the large scale processing, and say that this method is not efficient in retrieving large datasets from remote repositories;

- **Automated**: it overcomes the issues of extracting large sets of data, and offers elegant solutions in developing complex case studies, such as Earth modelling in different domains. This solution involves minimal efforts from the user's point of view, and has several advantages. The first one is related with optimizing the data search, extraction, and download by means of developing mechanisms that automatically perform these tasks.

  On the other hand, no particular remote sensing or computational skills will be required in order for the users to utilize these tools. All computations and data transfer will be done in the background, without the need of the users to monitor the entire process. Moreover, an automatized export of processed data into other repositories is possible. This will encourage the reutilization of data that have already been prepared by domain field specialists.

The process of data retrieval from remote repositories consists in a succession of steps (Figure 6.4) that should be performed automatically by the software applications implemented on the basis of this methodology. The user intervention should be minimalistic, the only required input is the Web address of the remote server and the types of data that he needs in order to complete its experiments.

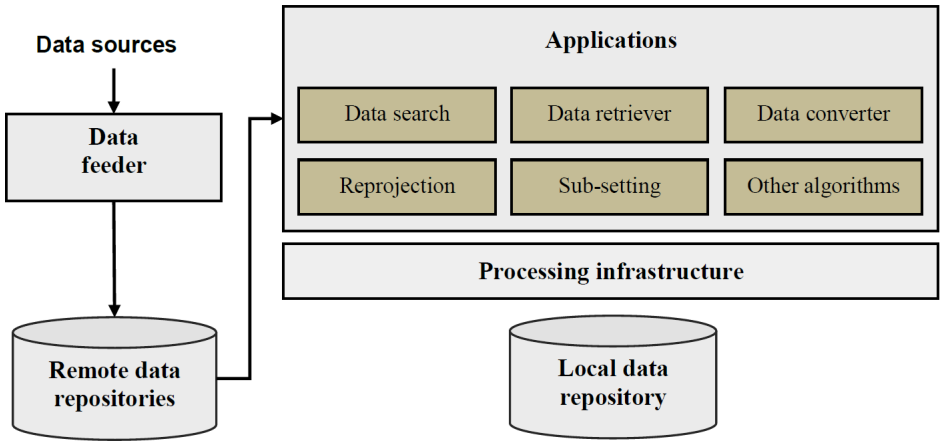


Figura 6.4: Spatial data retrieval process

The general flow starts with the *data feeder* module that continuously inserts spatial data into the global data repository. This is possible due to the fact that artificial satellites that orbit around the Earth provide new data at specific timestamps. Data types vary across multiple activity fields, depending on the purposes that emerged to launch that satellite: land cover, land use, atmospheric pollution, etc.

The main idea is that from multiple data sources, information is collected into large data repositories that could be accessed remotely by any types of users. The main goal of the *data feeder* is to pre-process these raw data before offering them to the worldwide audience. For some geographic areas, the pre-processing also means the removing of national and international strategic points that are not offered to regular users. Instead they become part of the governmental agencies.

By certain communication protocols (FTP and HTTP) spatial data, stored in the remote repositories, could be accessed and copied to external databases. Most of the times, the users are interested in specific information around a limited geographic area. Searching for data is the first step in the general workflow proposed by Figure 6.4. As seen in the previous section, spatial data are packed into files. Each such file has a metadata description, provided in XML format.

The geographic area coordinates, time when the image was created, type of projection, type of the sensor, and other useful information are coded within the metadata. When the user specifies that he is interested in retrieving all data for the Central European region, for the last 10 years, the *data search* module should be able to query the metadata catalogue provided by the remote repository and to select only those files that correspond to the search criteria.

A major issue is related with the fact that there is no standard representation of data within repositories. As seen in the previous section, each data provider could organize its information under a proprietary format. The search module should be aware of the folders/files structure and be able to recognize their internal representation.

The *data retriever* module is implemented at the application level and consists of transferring the information from the remote to the local data repository, by means of different protocols allowed by the data provider side. Usually, this step is performed after searching throughout the metadata structure. In order to avoid excessive downloads the remote server limits the number files that could be transferred simultaneously.

Until this point the type of data was not relevant for none of the so far involved modules. Because the methodology proposes a unique format of data on the local repository (GeoTIFF) there is the need to be able to provide such conversion features. The *converter module* is optional depending on the type of information stored in the remote repositories (e.g. MODIS, Landsat, ASTER, etc.). In case in which the two types are different, the transferred data must be converted. When multi-band images are involved, a new GeoTIFF file should be generated for each band, containing the same information as the original one.

The usage of the three modules (search, retriever, and convertor) is enough for successfully transfer spatial data from remote to local repositories. The main disadvantage of this "simple" transfer is that the copied image keeps the same characteristics as the original one (e.g. geographic area, projection, and information values). In order to alter these attributes other algorithms have to be implemented at the application level.

Reprojecting the original image is one of these algorithms. The sensor that records data for a certain geographic area uses the best projection for representing this information. But in some cases there is the need to distort the original projection just to be able to insert data into a limited-type system. For example the GeoSever [53] tool recognizes a few projections, and makes it impossible to feed it with data that is represented in other projection. A possible solution is to use the reprojection algorithm.

One of the recommendations of this methodology is to avoid reprojection as much as possible. The data from the remote repository uses the correct projection of the geographic area from where the data was collected. By using this algorithm the resulted image is going to be distorted, regardless of the performance of the reprojection.

There are cases in which the user wants to extract inner regions of the original data file. This is possible due to the sub-setting algorithm that is exposed at the application level, as described in Figure 6.4. Sub-setting does not mean extracting certain types of information (e.g. hydrological data from a land cover image). It means cropping the image against a vector shape given by the user. The result contains the same data as the original image, but only for the specified sub-region.

Depending on the users' needs, other algorithms could be integrated and exposed as application services. But these five methods are enough for selecting and retrieving any types of data, from any remote repositories. The time required to perform this action involves several factors: size of data that need to be transferred, bandwidth, remote and local server loads, etc.

The processing infrastructure is the lowest level of the data retrieval process, and it is used to perform the algorithms and computations described earlier. Local data repository is a database located on the application server that stores the results generated after applying different combinations of algorithms.

In order to implement the data retrieval process, an internal data model is required. Such a solution is proposed in Figure 6.5, where each image is considered to have multiple bands attached. The file from the remote repository is processed at the application level by certain algorithms. The resulted image must comply with the recommendations proposed by this methodology:

- Partition the image into multiple bands;
- Use GeoTIFF as a unique data format;
- When executing workflows over distributed infrastructure, use the bands as input resources, instead of the entire multi-band image;
- Allow the users to rename their uploaded images instead of the real name which in some case is difficult to remember (e.g. MOD16A2.A2000001.h00v08.105.2010355153835).

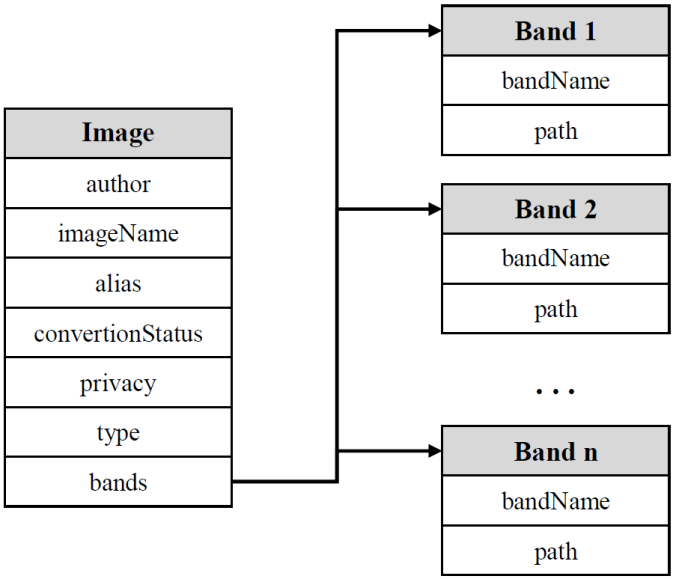


Figura 6.5: Data model for storing external spatial data

The attributes significance from the data model, represented in Figure 6.5, is the following:

- **author**: each user must be uniquely identified at the application level. When importing a file to the local repository, the ID of the user that initiated the request is stored in the *author* field;
- **imageName**: it is the same with the original name of the file, stored within the remote repository. In some cases this name is long and hard to remember by users (e.g. MOD16A2.A2000001.h00v08.105.2010355153835), but useful when organizing information at the repository level;
- **alias**: an easy to remember name, given by the user to the imported spatial image. In all processing the *alias* field is the one displayed to the users, while in the background the system will continue to work with the real name of the image;
- **conversionStatus**: the conversion algorithm applies on all bands of the original image. In cases in which the algorithm fails to generate the corresponding band, the entire image should be marked as not ready for usage. The available statuses are: pending, success, and error. The first one is displayed throughout the entire conversion process. Once this process ends, one of the last two statuses is assigned. Only images that share the "success" state are available to be used as input resources when executing workflows over distributed infrastructures;
- **privacy**: when fetching external datasets the user should be able to define data access rights for the rest of the members of the system. There are two possible privacy types: private (only the current user could process the uploaded image) and public (the resource is available for the entire users community). When trying to alter the data (i.e. delete it) the application will verify if the author of the image is the same as the user that made this request. In case of perfect match, the file will be deleted, otherwise a warning message should notify the user that it is not able to alter the data own by other users;

- **type**: in the next chapter of this report, a table is provided with all possible data types supported by the software applications that implemented this methodology. The data model from Figure 6.5 works only on multi-band spatial images. For the rest of the cases, the *convertionStatus* and *bands* attributes will be removed from the first table, while the *Band1*, *Band 2*, ... ,*Band n* are not needed;
- **bands**: an array of pointers that indicate the path to the bands extracted from the original image. Each *Band* table contains the name of the band and the physical path on the local repository that stores these bands;
- **bandName**: the real name of the band extracted from the original file. For example, if the image has 7 bands, than the k band is automatically renamed by the following rule: *<parent image name>_Bk.tif*;
- **path**: the physical path on the local repository that stores these bands.

6.2.3 Storage data model

In order to be processed by software applications, spatial data has to be digitized, and formatted in a proper way. Because there are many types of data, one special issue in developing such applications is the converter component that is able to parse data of different formats, without loss of information. The solution proposed in this thesis is based on using single-band Landsat (GeoTIFF) images as an internal representation of raster data. There were several reasons for choosing the Landsat standard:

- Cross-platform portability: the Geo-referenced Tagged Image File Format (GeoTIFF) type of the Landsat images is supported by all operating systems and hardware platforms;
- Different level of precision in data representation: depending on the type of data, this format is able to generate bands of information with simple precision (8, 16, and 32 bites) or double precision (32 and 64 bites);
- Easy to write and read;
- Supported by a large number of GIS software applications, such as: ARC/INFO [143], ERDAS [69], and GRASS [58];
- High compression rate, by using lossy (e.g. Joint Photographic Experts Group (JPEG) [144]) and lossless (e.g. Lempel–Ziv–Welch (LZW) [144]) packaging algorithms.

The spatial data representation in GeoTIFF [18] format is presented in Figure 6.6. It contains three main layers, described as follows:

1. File header: stores basic information about the image, such as the owner, file type (simple or double precision), creation date, etc.;
2. Image File Directories (IFDs): it consists of 12-bytes blocks array that point to the real data. The first two bytes represent the tag area, where the GeoTIFF stores geo-referencing information, projections types, etc. The next two bytes represent the type of data (e.g. integer, float, or double). As mentioned before the GeoTIFF is able to represent data in different formats. The count field is 4 bytes in length, and it gives the number of values stored in the data field. The last position of the IFD is occupied by the data field, with 4 bytes in length that stores the actual value or a pointer to that value;
3. Dataset of values.

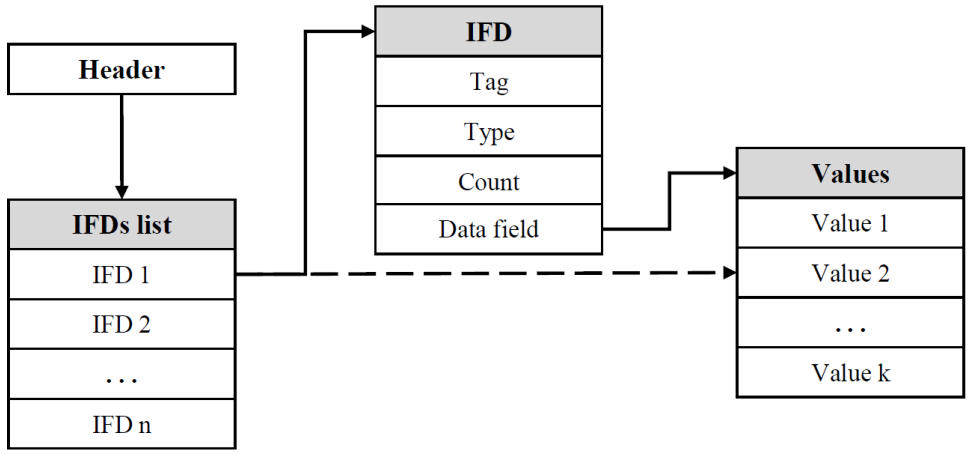


Figura 6.6: Internal representation of the GeoTIFF format

6.2.4 Workflows instantiation techniques

Once the GreenLand workflows properly describe the functionality of the use case scenarios, the next step consists in specifying the input data resources that are going to be used at runtime. This process is called instantiation and it a mandatory phase in the entire data execution flow. The GreenLand system provides specialized mechanisms for linking the inputs of the workflows to local or remote spatial information. All the resources that are resident of the same sever machine as the application, are considered to be local.

There are two concepts related with the workflow term: abstract workflows and instantiated workflows [101], [104]. The main difference between them is that the second one integrates the first one, but it also contains the physical data resources (e.g. satellite images, integer values, strings, text files, etc.) specified for the inputs list. Basically, the abstract workflow is only a template that could be instantiated multiple times with different resources, and each time it generates another workflow that is executed over the distributed infrastructures.

The abstract workflow contains the nodes of the graph, the connections between them (as edges), information about inputs and outputs (name and type), and information about how the workflows are nested within the hyper-workflow. As can be seen one important argument is missing: the value of the inputs and outputs. In the workflow data model the input/output was described as a triplet $< name, value, type >$, but in this case we have knowledge only about the name and the type of each input/output. If no resource value is attached to the inputs, there is no possibility to perform the execution phase, and the workflow keeps it abstract property.

In order to solve this issue, a new concept was defined: instantiated workflow. It contains the same information as the abstract workflow, but adds specific resources to the inputs.

When installing a fresh copy of the GreenLand platform, its data repository is populated with a predefined set of satellite images. In order to allow the repository enrichment, some special upload techniques were implemented that provide support for:

- Local upload: relevant only for the data that are resident on the user's local machine. This operation performs a regular upload, by transferring the corresponding byte array to the server that hosts the GreenLand application. From here the new information can be used to instantiate the workflows before their actual execution;
- FTP upload: it is recommended to be used only when a large volume of spatial data needs

to be retrieved from various remote repositories. By using the FTP protocol, data can be transferred automatically to the stations that process the workflow, without the user interference. This kind of upload is usually connected to complex workflows that predict certain phenomena and require data from the past, in order to calibrate the models that generate future events;

- OGC upload: implements the latest OGC WCS services for accessing and retrieving relevant information (in a standardized manner) from remote databases. This type of operation can import data only from sources that also implement the OGC related mechanisms. From the system's point of view, this approach simplifies the entire transferring process, because all the repositories that implement this standard have the same internal organization. On the other hand, the FTP resources require proprietary accessing rules, due to the fact the each consortium organizes its data as it desires.

One important characteristic about the GreenLand workflows is the fact that they have a well defined list of input and output parameters. The instantiation process is related to linking these items to physical data resources that can be accessed by one of the three methods described earlier. Even though the content of the spatial data cannot be validated until runtime, the GreenLand application automatically verifies if the specified information corresponds to the type of the inputs/outputs of the workflow.

6.2.4.1 Manual specification of input data

The GreenLand platform allows two modes of specifying the input datasets: manual and automatic. The manual technique is dedicated to small to medium workflows (with no more than 15-20 entries) and requires that all the data are resident on the same machine as the workflow is. The instantiation process is highly interactive and automatically filters the information in order to display for an input only the resources whose formats match the type of the input.

Figure 6.7 exemplifies how the users are able to manually specify the inputs of different GreenLand workflows. The first example (Figure 6.7a) displays the information for the NDVI algorithm that requires two Landsat satellite images that correspond to the Red and NIR bands. As can be seen each input item contains a short description, the type, and a value attribute. In case of multiple choice solutions, a drop down component displays all the necessary information from where the user can select the one he is interested in. There is also the possibility to input certain information in text fields that store numerical value, array of characters, etc. (Figure 6.7b). Before the actual saving operation an automatic validator check the correctness of these data.

The GreenLand platform is flexible when dealing with inputs specification process. Its types table (Table 5.5) can be easily extended to support new customized data formats (Figure 6.7c can be used to highlight this aspect). The workflow described in this figure is often coupled with the NDVI operator in order to produce classified images. Each interval of values are colored with a specific nuance, in this case the red is used for values between [-1; -0.33], the green for [-0.33; 0.33], while the [0.33; 1] range corresponds to the blue nuance. In reality these groups of colors represent geographic areas covered with urban buildings, vegetation, and water.

6.2.4.2 Automatic specification of input data

There are cases that require a large number on input parameters that cannot be specified entirely by hand. For these exceptions, the GreenLand platform provides automatic mechanisms that connect at runtime these inputs to the corresponding data resources, by using the rules defined at the graphical interface level.

Usually, the automatic instantiation process involves the data transfer from remote repositories by using the FTP protocol. The Mosaic workflow is such an example that is used in order to predict the water quality and quantity from the Black Sea catchment. It works with MOD15 and MOD16 satellite images that are periodically retrieved from two specialized remote repositories: NTSG (ftp://ftp.ntsg.umt.edu/pub/MODIS/Mirror/) and USGS (ftp://e4ftl01.cr.usgs.gov/MOTA).

In order to correctly retrieve information from these database resources, the GreenLand platform has to know their internal organization. Without the Spatial Data Infrastructure (SDI) standard guidelines, each such repository implements its own data access policies. This complicates the data sharing process, because the systems that connect to them for information retrieval have to implement these proprietary policies. The *2.4.3 Global Spatial Data Infrastructure – GSDI* section describes the internal organization of the data repositories that are used in the case of the Mosaic workflow.

The results generated by the execution of this workflow have a temporal resolution of one year. Internally, the Mosaic algorithm takes 12 inputs that together cover the entire Black Sea catchment. The MODIS products are collected every 8 days, so in a year we have to process 45 Mosaic workflows, instantiated with the available data resources. This means that there are 12x45=540 entries that have to be specified for a single processing year. Furthermore, in order to obtain accurate results this use case study has to take into account multiple years which only increases the number of input parameters.

Figure 6.8 highlights the graphical interface that displays the information required for the automatic specification of the inputs of the Mosaic workflow:

- The processing year that is used like a filter in order to retrieve the corresponding data from the NTSG and USGS remote repositories;
- The bands for the MOD15 and MOD16 products. Even though the MODIS satellite images have multiple bands, only the ones displayed in Figure 6.8 provide useful information for this case study.

Once these three sets on metadata are specified by the user, the GreenLand platform automatically transfers at runtime all the remote spatial data (that match the selected criteria)

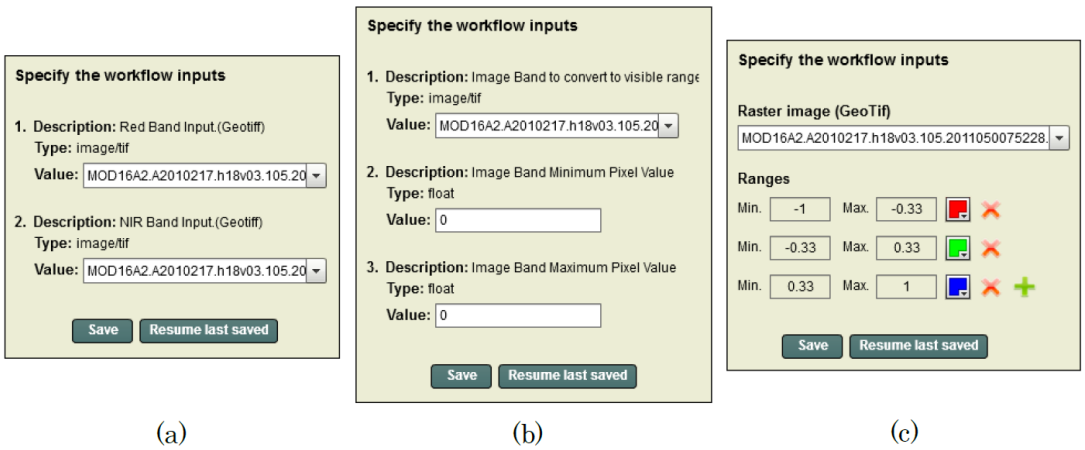


Figura 6.7: Manual specification of the inputs of the workflows

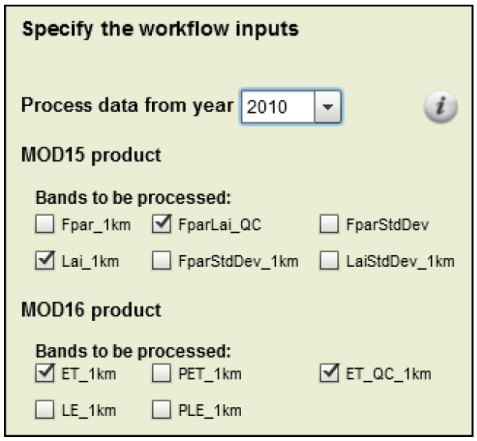


Figura 6.8: The automatic instantiation process for the Mosaic workflow

and assigns them to the corresponding inputs of the Mosaic workflows. The main advantage in comparison to the manual instantiation of the entities, is the fact the when transferring the information it reaches directly onto the machines that process the Black Sea modeling scenario, and does not pass through another intermediate storage (GreenLand in this case).

Another thing that is worth mentioning is the fact that the regardless of the original type of data (e.g. MODIS, Landsat, ASTER, etc.) the GreenLand platform converts it into an internal format (GeoTIFF). This policy was implemented in order to simplify the overall types table only to the generic name of satellite image (that actually refers to the GeoTIFF format). Otherwise, a new type had to be defined for each new spatial data format that was uploaded to the GreenLand platform.

6.3 Processing phase

Once the use case studies are integrated as GreenLand workflows and they are instantiated with proper data resources we can move on to the processing phase that computes the functionalities described within these entities. The development methodology of this thesis provides optimized solutions for processing large volume of spatial data. Taking this into account, the processing of workflows should be done on powerful computing environments that are able to produce valid results in a reasonable amount of time. Even thought the theoretical concepts of this methodology support various backend infrastructures (e.g. Cloud, Grid, clusters, etc.) their implementation within the GreenLand platform was tested only on the Grid and on standalone environments.

The standalone term refers to the unitary execution of the workflows, without using any parallel and distributed computation capabilities. Even thought these infrasturctures may contain multi-core machines that support parallel processing, the GreenLand workflows are executed as unitary modules.

The GreenLand platform provides all the necessary tools for executing the workflows over the two environments (Figure 6.3). Today's GIS research directions promote the data interoperability between multiple platforms that deal with the spatial data processing. For this reason, the GreenLand system implements the WPS standardized execution of data that is used to achieve this interoperability. Further details can be found in the following sections of

this chapter.

Even though the processing phase requires only the graphical modules from the GreenLand framework, in the background the system uses the gProcess platform [75] in order to perform the actual execution of data. It acts like a middleware layer that interprets the processing requests and converts them into commands that are understood by the computing machines.

The execution phase is the main component of the architecture proposed in Figure 6.3. It integrates the application level and the computing infrastructures (multi-core and Grid) that perform the actual processing [145]. The concepts developed in this methodology are not limited to the exemplified infrastructures, but only these ones were mentioned due to the fact the all the experiments were conducted on the multi-core and the Grid environments.

This multi-infrastructures layer is possible due to the fact that the workflow related data model is flexible enough to support data and task parallelism. Once the workflow is partitioned into tasks, a scheduler mechanism is able to send them to each of these computing platforms.

The *applications* block stores the client applications and the server-side mechanisms that link the clients' requests with the distributed infrastructures' services. These applications should offer support in developing hyper-workflows, and also some guidelines and tutorials about the implementation of basic operators.

At this level the workflow's data model must have an internal representation in form of database tables and XML files that are used for communicating with the distributed environments. The spatial data is another aspect that is handled at this stage, in terms that all input data is converted into the GeoTIFF representation, regardless of its original format.

The databases that handle all these information are application-oriented and they are not shared among same-level applications. This means that information from one place could not be replicated so easily to other locations. And it is the right decision, because each application should have its own structure. For example: application A monitors the vegetation evolution over a specific geographic area, based on vegetation indices. Application B addresses issues related with atom combination in the physic research domains. The two applications are not related, and should have their own databases.

On the other hand there are cases in which two or more applications activate in similar domains. In this case there should be a connection between their information. So, the question that arose is: keep the information separated or shared among all applications? The solution adopted in this report is somewhere in the middle. We propose a centralized solution with a common database that stores a predefined set of basic operators that could be accessed by all applications that were developed based on this methodology. At each application side, there should be other databases that contain specialized operators and workflows. The communication between these systems could be done by using the XML standard. The synchronization should be viewed as a semi-automated mechanism, enriched with data resources from all other applications, and it may update its database with only a subset of this information. The advantages of this solution are:

- Better management of local databases;
- Data consistency and data reutilization;
- The XML files generated at this stage are also used when partitioning the hyper-workflow into tasks and executing it over the parallel and distributed infrastructures;
- Offline and online synchronization between databases.

6.3.1 GreenLand - gProcess resources mapping technique

The gProcess [75] platform is an interactive toolset that supports the flexible description, instantiation, scheduling and execution of the Grid processing. The gProcess platform can be defined as a collection of Grid services and tools that acts like a middleware between the computing infrastructures and the GreenLand application. Even though it provides basic functionalities of the spatial data management, the GreenLand platform uses it only for tasks scheduling, execution, and monitor.

Internally, the gProcess represents the workflows as Directed Acyclic Graphs (DAG) that have a similar structure with the one described in the theoretical concepts sections. In this case, each node of the graph is considered to be a task. At runtime all tasks are stored within a queue and dynamically deployed to the Grid CPUs. When the number of tasks is greater than the number of available hardware resources, the gProcess scheduler waits until one of the worker nodes finishes, and assigns it with another task. The execution of the entire workflow is completed when all its inner nodes (tasks) are successfully processed.

The gProcess is part of the Environment oriented Satellite Data Processing Platform (ESIP). The ESIP provides a specialized repository of geospatial algorithms for the GIS domain. On the other hand the gProcess is responsible for executing these algorithms over different computing infrastructures. This means that the two platforms can function as a whole in order to provide similar functionalities as the GreenLand does. Actually, the GreenLand platform is built over the ESIP one and has the advantage of extending the repository to other activity domains (e.g. biology, physics, etc.) and provides far more advanced interactive tools that facilitate the description of the use case studies.

Figure 6.9 highlights the ESIP related architecture that is developed over the client-server model. One important aspect is the fact that it can be easily integrated within external platforms (such as GreenLand) and configured to expose only certain functionalities that are required by these applications. It has different modules (e.g. editor, manager, executor, and viewer) that have a Web service interface and a user interaction component.

The Web services (exposed by the server-side related modules) communicate with the computing infrastructures in order to perform the data execution. Also they have the role of sending periodic feedback that is displayes into the graphical interface components.

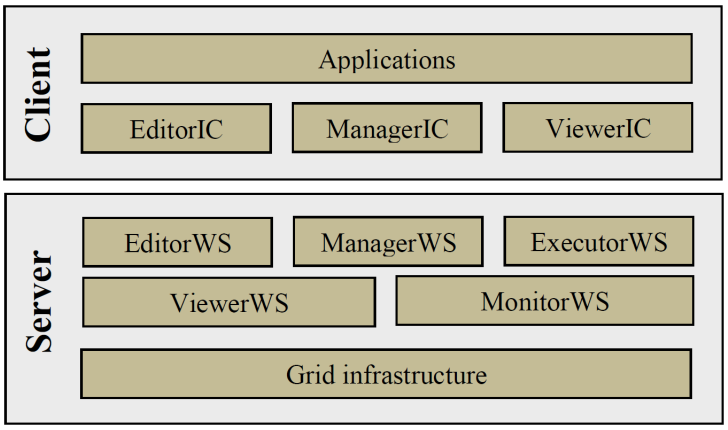


Figura 6.9: ESIP related architecture
(Source: `http://cgis.utcluj.ro/applications/gprocess`)

The EditorWS provides information about the types of satellite images, types of temperature files, list of operators, etc. that are involved in the input specification process. On the other hand it is not enough to have knowledge about the available resources, if the execution workflow that uses them is unknown. This means that the EditorWS is also benefic for providing information about the general characteristics of the workflow: name, description, statistical data, etc.

The *ManagerWS* is used to find out all kinds of information about the current workflow, or about other ones (e.g. what kind of resources, services, and operators are involved in that workflow). The *ExecutorWS* component is invoked when starting the Grid processing or when monitor this execution.

The modules that are resident on the client-side facilitate the development and management of DAGs. The *EditorIC* supports the user's editing operations for workflow development. The *ManagerIC* instantiates workflows with particular satellite data and manages the model resources (e.g. operators, services, sub-graphs, satellite data, etc.). The *ViewerIC* displays the output results in a graphical representation that facilitate the visualization process. It is also used in order to state the feedback retrieved from the monitor mechanism.

One of the main functionalities provided by the gProcess platform is the ability of partitioning the workflows into tasks and to schedule them over various computing machines. Based on this feature, the gProcess tool can be identified as the *Scheduler* module from Figure 6.3 that directly communicates with the multi-core and Grid environments.

When the workflow content is parsed to the gProcess platform it is converted into an internal representation that is further used in the execution process. For example, the following data structure corresponds to the workflow defined in Figure 5.10.

```
<Workflow>
<Nodes>
   <Resource id="1_1" name="in1" >
        <ValueResource value="1"/>
                <PostConditions>
                       <Output idTypeDB="4" />
                </PostConditions>
          </Resource>

   <Resource id="2_1" name="in2" >
        <LocalResource path="resources/images/in2.tif" />
            <PostConditions>
                     <Output idTypeDB="1" />
            </PostConditions>
   </Resource>

   <!--  The other 3 inputs of the hyper-workflow  -->
                         . . .

   <Operator id="1" name="OP1" >
        <Preconditions>
                <Input id="1_1"/>
                <Input id="1_2"/>
                <Input id="1_3"/>
        </Preconditions>
   </Operator>

   <Operator id="2" name="OP2" >
```

```
        <Preconditions>
                <Input id="2_1"/>
                <Input id="2_2"/>
        </Preconditions>
   </Operator>

   <Operator id="3" name="OP3" >
        <Preconditions>
              <Input id="1"/>
        </Preconditions>
   </Operator>

   <Operator id="4" name="OP4" >
        <Preconditions>
              <Input id="2"/>
        </Preconditions>
   </Operator>

   <--  Describe the SW2 sub-workflow as a list of inner operators -->
                      . . .

   <Operator id="5" name="OP5" >
        <Preconditions>
              <Input id="6"/>
        </Preconditions>
   </Operator>
   </Nodes>
</Workflow>
```

The structure of the graph is described between the <*Workflow*> tags. The hyper-workflow from Figure 5.10 has 5 inputs, 2 outputs, 5 operators, and a sub-workflow (*SW2*). The XML structure begins by defining the resources of the inputs. There are two types of resources: primitives (e.g. integers, strings, etc.) and complex (e.g. satellite images, text files, archives, etc.). The first type is identified by the *ValueResource* tag, while the other one contains the *LocalResource* tag that stores the path to the physical location of the file. The type of each input is retrieved from the database (Table 5.5) and it is represented by the *idTypeDB* tag. In this example only the first two inputs were defined, while the other 3 could be described in the same manner.

An operator contains a set of pre-conditions that describe the connection of their inputs with outputs from different operators. For example, *OP1* has 3 inputs connected directly to the specified resources, while *OP3*'s input is linked to the output of *OP1*. The XML file expands the hyper-workflow structure down to the operators' level. That is why *SW1* and *SW2* are represented as a set of inner operators.

6.3.2 Execution management over different infrastructures

The processing of large volume of data is one of the main research directions of the methodology proposed within this thesis. Because the execution of the complex use cases may take tens of hours, it was necessary to adopt a solution that provides parallel and distributed capabilities, such as the Grid infrastructure. At the same time, the GreenLand platform also process small-scale algorithms that do not require the full capacity of the Grid. For these situations, the standalone environment can be used (Figure 6.3).

Even though the theoretical concepts introduced by this methodology are flexible enough to support a variety of computing infrastructures, the experiments were conducted only on the standalone and Grid environments. The standalone term integrates machines that are not connected one to another and do not facilitate the parallel or distributed execution of the workflows. The multi-core machines are also part of the standalone category, because the workflows are executed as unitary entities that cannot be partitioned into tasks (like in the case of the Grid infrastructure).

The gProcess platform is the layer that connects the GreenLand user-related execution actions to the computing infrastructures (Figure 6.3). Each such infrastructure is unique in terms of data processing and requires new gProcess versions. Until now only the Grid and standalone related tools were integrated within the gProcess.

Due to the fact that the standalone execution does not involve parallel and distributed features, the following paragraphs describe the processing steps of the spatial data over the Grid infrastructure. These phases correspond to the Web services that are resident on the server and client side of the ESIP related architecture (Figure 6.9) and provide support for:

1. **Workflows partitioning**: At runtime, the XML based representation of the workflow is flatten to a single level of nodes. This way the gProcess platform is able to create groups of nodes (called tasks) in order to load balance the execution over the machines that work in parallel (see the experimental results described in the following section). Usually, the Grid execution of simple data takes less than the allocation of the machines and the input parameters transfer process. This is why it is important to obtain a balanced execution between the Grid worker nodes and to diminish as much as possible the extra communication;

2. **Tasks scheduling**: Once the workflow is decomposed into tasks, the gProcess invokes specific services from the Grid middleware (e.g. gLite [141], Globus Toolkit [142], etc.) in order to allocate the necessary hardware resources. If the number of available machines is less than the number of tasks, the gProcess uses a queue and waits until one worker completes and then assigns it the first item from this queue;

3. **Tasks execution**: The geospatial algorithms that are part of a specific task are executed on the Grid machines were they were scheduled. Because the workflow has the structure of a graph, the result of one task corresponds to the input of another one (as defined by the edges of the workflow). If the task execution fails then all the next steps are going to generate errors at runtime. Otherwise, if the task processing succeeds, the gProcess takes the next item from the waiting queue and assigns it to this Grid machine;

4. **Execution monitor**: During the entire workflow execution the gProcess platform provides feedback about the status of the processing that is displayed to the user in the graphical interface of the GreenLand platform. Based on the state of the execution, the user is able to interfere and stop (if necessary) the Grid processing. Once the workflow execution stops it has no resuming option, so the process must start from the beginning;

5. **Results composition**: There are two types of workflow results: intermediate (the output generated by a node that is connected to the input of another node) and final (the results match the outputs of the main workflow). The gProcess has to redirect these intermediate results to the proper tasks, and to use them in order to generate the final outputs of the entire execution.

In cases of workflows with multiple outputs, the gProcess creates a separate file for each one of them (e.g. the numerical values are written as text inside this file, the satellite images

pixel values create the content of this output, etc.). If the same output generates multiple files, a new archive is created that stores all these data. In other words it is not allowed to have multiple files attached to the same output.

When referring to the general Grid execution of data, all these five steps are part of the process. Now it can be seen more clear that in case of simple workflows (that require a few seconds to complete their execution) the time required to finish all the steps is greater than the actual processing. In conclusion, only the complex workflows (that require a few hours processing) are worth to be executed over the Grid, otherwise the standalone infrastructure is most suited.

This methodology aims to provide solutions for processing complex use case scenarios, but in the same time it facilitates the development of interaction techniques that can be easily used by various types of users, without the need to have domain related background knowledge. Due to the fact that these users are not involved in the complex phenomena development, execution, or analysis, they usually handle small data. In conclusion, this methodology is also related to the processing of small amount of data.

We know that the Grid infrastructure is recommended for the execution of large volume of information, but it does not provide optimal solutions for processing less complex algorithms. This is the main reason why this methodology proposes multiple computing platforms that can handle all these situations and improve the execution process.

One of the main goals is to be able to identify the point in which the Grid provides better execution times than the standalone environments. Two approaches were used in order to solve this issue:

- A particular solution (that works only for a specific list of workflows) that is based on the manual identification of the switching point between the two infrastructures. In other words, for each workflow several experiments were conducted that indicate the complexity at which the Grid infrastructure performs better;
- An automatic solution that is able to decide (based on the complexity of the workflow, the size of the inputs, the availability of the Grid resources, etc.) the type of the infrastructure that produces the best results at that time.

The first solution uses the statistical results, obtained from multiple simulations, in order to manually decide if the algorithm is executed on the Grid or standalone infrastructures. This is useful in case of workflows that we know that require either complex (e.g. Mosaic experiment [77]) or simple (e.g. NDVI algorithm) computation resources and these pre-conditions do not change over time.

For these cases, after developing the workflow the next step is to determine its complexity. This way, the system can use this indicator in order to further execute the workflow, regardless of its input data and of its number of instances.

1. The Black Sea catchment modeling experiment

The analysis of the water quality and quantity in the Black Sea catchment (Mosaic use case study) is one of the most time consuming geospatial algorithms contained within the GreenLand repository. It works with MODIS products that have a temporal resolution of one year. The GreenLand algorithm (workflow) that is used to process this case study is also called Mosaic. In order to distinguish between the Mosaic use case and the Mosaic algorithm, within this experiment the algorithm will be called Mosaic12 because it needs 12 inputs.

Internally, the Mosaic12 algorithm takes 12 inputs that together cover the entire Black Sea catchment. The MODIS products are collected every 8 days, so in a year we have to process 45 Mosaic12 workflows, instantiated with the available data resources [77].

Due to the fact that the Mosaic12 workflow is able to provide results for both MOD15 and MOD16 satellite images, a total of 45x2=90 products have to be processed in a year. The range period of the case study is for 10 years (between 2000 and 2010) and involves more than 100000 satellite image bands with a total size of 560 GB.

In order to determine the complexity of the Mosaic12 workflow several experiments were performed. The first one was on the standalone environment with two quad core processors at 2.13 GHz frequency, and 32 GB of RAM memory. Each of the 90 MODIS products was represented by a different instance of the Mosaic12 workflow. This local execution was done in sequence, meaning that after the completeness of the first Mosaic12 algorithm, the second one stated its execution, and so on until all the 90 instances were finished.

The total execution time took about 9 hours and a half and included the 2000 processing year with 90 MODIS products. This means that the entire Black Sea catchment modeling (for 10 years period interval) will require around 95 hours of processing on the standalone environment.

The next step was to run the same experiment, with the same input data set, on the Grid infrastructure. Due to the fact that the input images have to be downloaded at runtime from the NTSG remote repository, then entire use case was divided into 18 groups, where each group contains 5 Mosaic12 workflows. This partition was required due to the fact that the NTSG repository limits the number of simultaneous downloads to 5.

The Grid machines were less powerful than the standalone server, but provide parallel and distributed capabilities that speed up the entire execution process. One processing year was executed simultaneously on 9 different Grid nodes, where each worker node has a quad core processor that receives one group of 5 Mosaic12 workflows. The total execution time (including the tasks scheduling, data transfer, and final result generation) was 2 hours and 5 minutes, while for the 10 years processing scenario took around 20 hours.

From the two sets of experiments it can be seen that the Grid execution is better than the standalone one. Because the performance ratio is approximately the same (regardless of the number of processing years) we can conclude that the Black Sea catchment modeling use case should always be executed over the Grid infrastructure.

2. The NDVI experiment

The less complex algorithms (e.g. satellite images pseudo-coloring, vegetation index computation, etc.) are recommended to be executed over the standalone infrastructure rather than the Grid environment. The following experiment validates this aspect for the NDVI algorithm that is used to determine the vegetation areas for a specific geographic location [146]. It is based on a simple formula (formula 5.4) that is applied at pixels level for the Red and NIR input bands.

The first experiment consists in executing the NDVI workflow on the same standalone infrastructure as in the previous experiment, with two Landsat satellite images of 1200x1200 pixels resolution. The total execution time was 0.5 seconds. In order to see the behavior of this algorithm at different input sizes, we've obtained a processing time of 3.09 seconds for 7200x7200 resolution images. This means a linear variation of the execution time related to the input complexity, in either case the NDVI processing time does not exceed the seconds level.

The NDVI experiment was than simulated on the Grid environment, with the 1200x1200 resolution input data set. The total execution time was 11.5 minutes. There is a huge difference between the two experiments that is due to the fact that the Grid execution time includes the time needed to:

- partition the workflow into tasks;

- schedule the available worker nodes and to assign each task to a specific CE;
- transfer the input data set and additional dependencies;
- perform the actual execution;
- recompose the final result based on the intermediate outputs that are linked through the edges of the workflow.

The Grid execution time is also influenced by the nodes availability and the network traffic at that time. The experiment that contains the 7200x7200 resolution inputs was not performed because the execution time difference between the first set of experiments was relevant that for small algorithms (e.g. NDVI) the Grid infrastructure is inefficient and it is recommended to use the standalone environment.

The previous experiment took into account only one instance of the NDVI workflow. The question that arises is: how about the case of processing multiple NDVI algorithms on the same time? This is another type of measurement that will be described in the following paragraphs.

3. The execution time variation over different Grid instances

Another interesting fact is that the Grid based processing is highly dependent of the current status of the context of the Grid. In other words, the same experiment will always generate different execution times (with seconds or minutes amount difference) due to the issues mentioned for the NDVI scenario. Based on this remark, the third category of experiments was focused on establishing the Grid reliability when executing multiple times the same types of algorithms.

This analysis took into account a geospatial algorithm that has the effect of improving the resolution of a coarse satellite image in order to obtain a better one, by using a nonlinear cosine function based on a distance-weighted interpolation method [101]. The purpose of the use case is to analyze the evolution of the temperature for the South Eastern European continent, based on inputs with one month temporal resolution.

The experiment consists in monitoring the execution behavior over different instances of Grid infrastructures (Figure 6.10). This analysis starts with executing a single algorithm that takes one input satellite image with one month temporal resolution, and ends by assigning 12 input images to 12 different instances of algorithms. Due to the fact that we have independent items, the gProcess platform is able to create a task for each such item and to execute it onto a different Grid worker node.

This aspect explains the chart rendering (Figure 6.10a) for each Grid instance. In other words, the increase of the number of inputs generates an increase in the total execution time, but from Figure 6.10a the chart seems to level around the 10 minutes value (for the testbed001.grid.ici.ro). This can be explained based on the fact that each extra input means an additional task that is executed on a different machine that does not slow down the executions of the other worker nodes.

Even though all the four Grid instances have the same computing resources, the execution time of the interpolation algorithms is different (Figure 6.10a).This is related to the fact that the Grid infrastructure depends on a series of factors (mentioned for the NDVI experiment that influence the workflows execution time) that cannot be controlled by the gProcess platform.

Figure 6.10b highlights the time required by each task in order to process the input with one month temporal resolution. This interpretation identifies another important aspect of the Grid infrastructure: the execution improvement takes place when the main workflow is highly modularize and can be partitioned into multiple tasks that are processed over different working nodes. This is the case of the 12 months related process that completes its execution in about

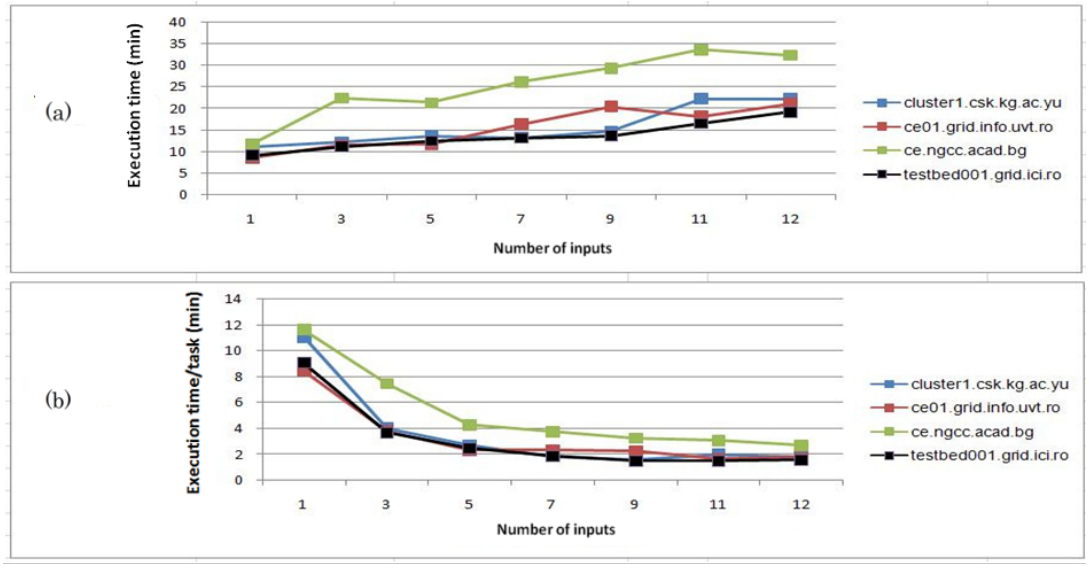


Figura 6.10: Execution time variation over different Grid sites

2 minutes, compared to 8-12 minutes task that corresponds to the case when using a single satellite image input.

4. Identify the execution switching point between the standalone and Grid infrastructures

There are cases when a single algorithm does not require to be executed on the Grid infrastructure because it is not complex enough (e.g. the NDVI experiment). But when multiple instances of the same workflow are needed in order to perform specific measurements, then the previous remark does not apply any more. One possible solution in solving this issue is to develop an automatic decision mechanism that is able to dynamically compute the complexity of the workflows and to decide (before runtime) what is the most appropriate environment for executing these workflows.

This feature was studied in [90], [147] and it is not a direct contribution of this thesis, but it is mentioned only to highlight the benefit of such an algorithm and the flexibility of the GreenLand platform that allows the integration of plug-in modules. It is based on the idea of assigning to each operator a weight value and to compute the overall complexity of the workflow as a sum of all inner weights. The number of pixels (for the raster images) and the number of geometric primitives (for the vector files) were two features that were taken into account when computing the operator's complexity. Currently it provides support only for the processes that are executed over standalone and Grid infrastructures.

It was proven that for the sequential executions (like in the case of the standalone infrastructure) the total complexity of the workflow is the most accurate indicator that predicts the total processing time. On the other hand, in order to approximate the Grid based execution time, a critical path should be computed together with its weight. By comparing the two complexities (for standalone and Grid processing) this algorithm is able to decide where to run the workflow.

The critical path represents a sequence of nodes that has the greatest weighs sum among all the existing paths within the main workflow. This information is important when partition the workflow into tasks, because the gProcess platform aims to create groups of tasks (similar

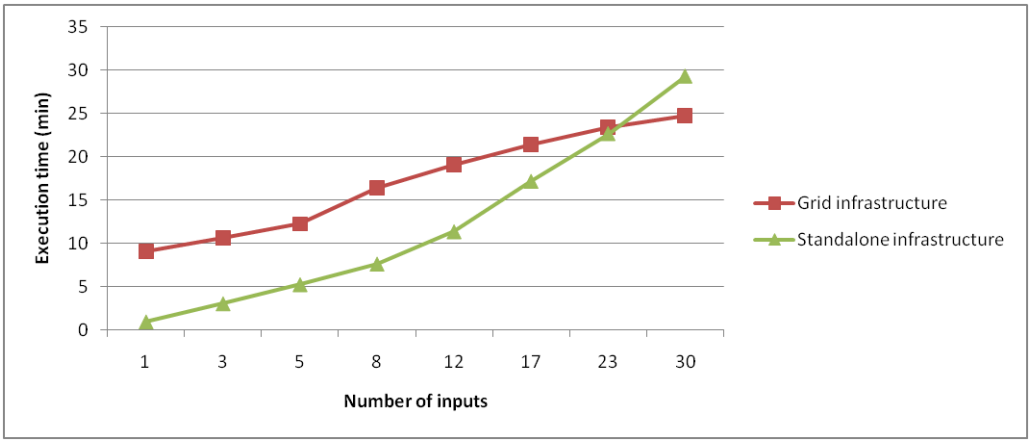


Figura 6.11: Identify the execution switching point between the standalone and Grid infratructures

in complexity) in order to load balance the entire Grid execution. Knowing this value helps the gProcess to create these groups whose complexity should be close to the weight of the critical path.

The interpolation algorithm (from the previous experiment) can be used again in order to highlight the switching point that identifies where the Grid platform performs better than the standalone infrastructure (Figure 6.11). The experiment uses multiple instances (up to 30) of this algorithm, instantiated with specific input data sets. The first study, performed on the Grid infrastructure, revealed that starting from a certain point the execution chart seems to remain around a constant value. On the other hand, the standalone infrastructure has an exponential increase of the total execution time, regardless of the number of instances of the interpolation algorithm.

When distributing the 25 inputs to the 25 instances of the algorithm, the chart from Figure 6.11 highlights that the execution times for the two infrastructures are almost identical, while up to that point the standalone processing was better than the Grid one. This stage is called switching point, because from now on the optimal execution is given by the Grid environment.

The solution proposed in [90] tries to identify this switching point and to locate the execution complexity based on this information. If it is located bellow this point, then the standalone infrastructure can be used to perform the processing. On the other hand, if the complexity of the workflow exceeds this threshold, the Grid infrastructure is recommended to execute the use case.

6.3.3 Platform level interoperability through the WPS standard

The today's research directions promote data sharing among various GIS communities based on a set of rules and guidelines. The OGC consortium [36] is such an organization that aims at providing standards for developers and users in order to produce services for accessing spatial data and to assure the geospatial interoperability.

Among other OGC products the Web Processing Service (WPS) has a major impact in obtaining the interoperability between different platforms, in terms of accessing and executing various remote geospatial algorithms without the need to develop them from scratch or to integrate them within specific applications.

In order to align to the latest GIS tendencies, the GreenLand platform implements the WPS standard and provides an alternative to the traditional execution of the workflows. The WPS standalone execution was already implemented and tested in many other projects and is well defined in terms of development rules and guidelines. On the other hand the WPS over the Grid infrastructure (or distributed platforms in general) was not studied so intensively. This section describes the issues encountered in achieving this goal for the GreenLand platform.

6.3.3.1 WPS general overview

Like any other OGC product, the WPS includes three types of operations: *GetCapabilities* (returns a list of the services together with all their available processes), *DescribeProcess* (for each process it provides a general description of the input and output parameters), and *Execute* (performs the process execution).

The *GetCapabilities* stores the general description of all services and indicates the available operations that can be invoked. The machine that hosts the WPS standard must contain the GeoServer [53] instance that allows the services call through a link similar to: http://<server_host >/wps/wps.py?service=wps&version=1.0.0&request=GetCapabilities. The service name, the version, and the type of the request are mandatory parameters.

Figure 6.12 highlights a fraction of the response return by the *GetCapabilities* operation. The *ServiceIdentification* tag contains a few details about the server that exposes the WPS

```
<wps:Capabilities xml:lang="en" service="WPS" version="1.0.0" >
    <ows:ServiceIdentification>
        <ows:Title>Prototype GeoServer WPS</ows:Title>
        <ows:Abstract/>
        <ows:ServiceType>WPS</ows:ServiceType>
        <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
    </ows:ServiceIdentification>

    <wps:ProcessOfferings>
        <wps:Process wps:processVersion="1.0.0">
            <ows:Identifier>NDVI</ows:Identifier>
            <ows:Title>NDVI index</ows:Title>
            <ows:Abstract>Computes the NDVI index</ows:Abstract>
        </wps:Process>

        <wps:Process wps:processVersion="1.0.0">
            <ows:Identifier>getX</ows:Identifier>
            <ows:Title>Get X Ordinate</ows:Title>
            <ows:Abstract>Returns the X value for point geometries</ows:Abstract>
        </wps:Process>

        <wps:Process wps:processVersion="1.0.0">
            <ows:Identifier>getY</ows:Identifier>
            <ows:Title>Get Y Ordinate</ows:Title>
            <ows:Abstract>Returns the Y value for point geometries</ows:Abstract>
        </wps:Process>
    </wps:ProcessOfferings>
</wps:Capabilities>
```

Figura 6.12: Metadata response of the WPS GetCapabilites request

items, its version, a short description, etc. All the available services are placed within the *ProcessOfferings* tag. As can be seen, this example provides only three services (NDVI, getX, and getY). The *GetCapabilities* metadata file does not contain detailed information about these items, but only their name and a short description.

In order to find more details about one of these services, the *DescribeProcess* operation can be used. It can be invoked in a similar manner to the *GetCapabilities* (http://<server_host >/wps/wps.py?service=wps&version=1.0.0&request=DescribeProcess&identifier=NDVI). The only difference consists in adding the identifier parameter that actually indicates the service for which more details are required. A classical example of the metadata returned by this request is highlighted in Figure 6.13. The name and type of each input is part of this document that also describes the format of the outputs of the service. For the NDVI example, we already know that it has two inputs that correspond to the Red and NIR bands of Landsat satellite image and one output that is computed based on the formula 5.4.

```
<ProcessDescription wps:processVersion="1.0.0" >
    <ows:Identifier>NDVI</ows:Identifier>
    <ows:Title>NDVI index</ows:Title>
    <ows:Abstract>Computes the NDVI index</ows:Abstract>

    <DataInputs>
        <Input maxOccurs="1" minOccurs="1">
            <ows:Identifier>redBand</ows:Identifier>
            <ows:Title>Red band input</ows:Title>
            <ows:Abstract>Stores the path to the Red band</ows:Abstract>
            <LiteralData>
                <ows:DataType>string</ows:DataType>
            </LiteralData>
        </Input>

        <Input maxOccurs="1" minOccurs="1">
            <ows:Identifier>nirBand</ows:Identifier>
            <ows:Title>NIR band input</ows:Title>
            <ows:Abstract>Stores the path to the NIR band</ows:Abstract>
            <LiteralData>
                <ows:DataType>string</ows:DataType>
            </LiteralData>
        </Input>
    </DataInputs>

    <ProcessOutputs>
        <Output>
            <ows:Identifier>result</ows:Identifier>
            <ows:Title>Output result</ows:Title>
            <ows:Abstract>Stores the NIR output result</ows:Abstract>
            <LiteralOutput>
                <ows:DataType>string</ows:DataType>
            </LiteralOutput>
        </Output>
    </ProcessOutputs>
</ProcessDescription>
```

Figura 6.13: Metadata response of the WPS DescribeProcess operation

The third phase of the WPS is the *Execute* operation that actually computes the selected service (algorithm). The following Web address can be used in order to invoke a general *Execute* method: http://<server_host>/wps/wps.py?service=wps&version=1.0.0&identifier=service_name&dataInputs=[input1=value1; input2=value2; . . . ;inputn=valuen]&request=Execute.

A particular example of the standardized execution of the NDVI index is:http://<server_host>/wps/wps.py?service=wps&version=1.0.0&identifier=NDVI&dataInputs=[redBand=resources/Landsat/Red.tif; nirBand=resources/Landsat/Nir.tif]&request=Execute. Because the NDVI has two input parameters of type string, the *value1* and *value2* attributes contain the path to the satellite images, while the *input1* and *input2* must be similar to the identifiers from Figure 6.13.

Because the output of the NDVI service has the string type, the XML response displays the results as an array of characters (e.g. [http://193.226.17.36:80/GreenLandv2/download?globalPath=output&filename=538100cd-4285-4333-b67b-ed32c189f42d/workflow__NDVI__iNDVI.OP1 __final_6_1382.tif]). The right parenthesis specifies an array of results, but in this case only one item is part of this array.

The WPS standard is easy to integrate in applications that contain algorithms whose processing does not exceeds more than a few seconds. On the other hand, the WPS does not provide enough guidelines for how to handle the complex executions that requires the parallel and distributed capabilities of the Grid infrastructure. In order to be able to integrate the WPS in the Grid distribution of the GreenLand, a standard extension was required. This means that a new operation (called *Monitor*) was added to the basic functionality of the OGC WPS service. It provides similar features with the *Execute* method (Figure 6.14), but it does not launches a new execution every time it is invoked. Instead it displays to the user only the status of the processing.

The *Monitor* operation was used for the same NDVI algorithm, executed this time over the Grid infrastructure. The experimental results confirmed that this workflow requires 4-5 minutes to complete its execution. In the previous paragraph we saw that on the standalone infrastructure this algorithm generates a valid result in a few seconds. This is due to the fact that the Grid execution means all the 5 stages that were presented earlier (workflow partitioning, tasks scheduling, execution, monitor, and results composition). Although the actual Grid execution (third step) is similar to the standalone one (in terms of computing time) the rest of the phases are also time consuming. When adding up all these durations, the final sum will be around 4-5 minutes.

We have invoked the *Monitor* operation after 30 seconds the Grid execution was started for the NDVI algorithm. At this stage the processing of the algorithm was not completed, so the result consists in an empty array, like in Figure 6.15 (the *ProcessSucceeded* attribute was set to *processing*, while the *LiteralData* tag displyed the *[]* value). After two minutes we've made a second call of the *Monitor* operation. The response was similar to the first one. Only after 4 and a half minutes the request provided the link with the result ([http://193.226.17.36:80/GreenLandv2/do wnload?globalPath=output&filename=8ec2622c-6a3b-443e-a102-2c043ee8966b/workflow__NDV I__iNDVI.OP1__final_6_1383.tif]) generated by the NDVI execution over the Grid infrastructure and the status of the *ProcessSucceeded* attribute was set to *completed*.

6.3.3.2 System related architecture

Figure 6.16 highlights the main modules that are involved in order to perform the standardized execution of spatial data. It provides two alternatives for spatial data processing. The first one (traditional manner) follows the path identified by the 1, 6, and 7 steps, while the standardized

```
<wps:ExecuteResponse service="WPS" version="1.0.0" >
    <wps:Process wps:processVersion="1" >
        <ows:Identifier>NDVI</ows:Identifier>
        <ows:Title>NDVI index</ows:Title>
        <ows:Abstract>Computes the NDVI index</ows:Abstract>
    </wps:Process>

    <wps:Status creationTime="2013-07-18T12:13:14Z">
        <wps:ProcessSucceeded />
    </wps:Status>

    <wps:ProcessOutputs>
        <wps:Output>
            <ows:Identifier>result</ows:Identifier>
            <ows:Title>Output result</ows:Title>
            <ows:Abstract>Stores the NIR output result</ows:Abstract>
            <wps:Data>
                <wps:LiteralData>[http://193.226.17.36:80/GreenLandv2/download?global
                Path=output&filename=538100cd-4285-4333-b67b-ed32c189f42d/workflow
                __NDVI__iNDVI.OP1__final_6_1382.tif]
                </wps:LiteralData>
            </wps:Data>
        </wps:Output>
    </wps:ProcessOutputs>
</wps:ExecuteResponse>
```

Figura 6.14: The response generated by the WPS Execute method

```
<wps:MonitorResponse service="WPS" version="1.0.0" >
    <wps:Process wps:processVersion="1" >
        <ows:Identifier>NDVI</ows:Identifier>
        <ows:Title>NDVI index</ows:Title>
        <ows:Abstract>Computes the NDVI index</ows:Abstract>
    </wps:Process>

    <wps:Status creationTime="2013-07-19 T09:15:09Z">
        <wps:ProcessSucceeded>processing<wps:ProcessSucceeded/>
    </wps:Status>

    <wps:ProcessOutputs>
        <wps:Output>
            <ows:Identifier>result</ows:Identifier>
            <ows:Title>Output result</ows:Title>
            <ows:Abstract>Stores the NIR output result</ows:Abstract>
            <wps:Data>
                <wps:LiteralData>[]</wps:LiteralData>
            </wps:Data>
        </wps:Output>
    </wps:ProcessOutputs>
</wps:MonitorResponse>
```

Figura 6.15: The execution status interrogation by using the WPS Monitor request

WPS execution regards the 1, 2, 5, 7 phases [148].

The PyWPS [149] is a software application, written in Python programming language that implements the WPS standard. It offers the possibility to access HTTP services that contain different geospatial algorithms. On the other hand, the PyWPS also allows the direct access of GRASS functions.

In cases of executing large scale scenarios, the direct usage of the GRASS features is not enough. The GreenLand platform addresses such complex use cases that need tens of minutes for completing their execution. This is the main reason why a new algorithm development method was implemented that uses the PyWPS for accessing them over the HTTP protocol.

There are several methods for accessing these WPS services: directly from the GreenLand application, API access from external platforms, and browser based access. The first mode assumes that the user is authenticated at the GreenLand level, and has access to all its geospatial workflows. Within this platform, the user has the possibility to select one of these algorithms and to specify its inputs. After that it can establish a bidirectional communication channel with the PyWPS server (step 2 of Figure 6.16):

- PyWPS → GreenLand platform: the user is able to interrogate the list of existing processes (*GetCapabilities*) and to visualize the detailed description for each of them (*DescribeProcess*);
- GreenLand platform → PyWPS: in order to start a new processing over the Grid infrastructure, the user is able to invoke the *Execute* WPS operation. The inputs selected by the user have to be sent as additional parameters to the Python script that further passes them to the *executeChain* service that starts the Grid processing.

In some cases the inputs path to the physical resources involves a complex combination of folders and files (e.g. Landsat/Bands/MOD16A2.A2000001.h00v08.105.2010355153835.tif). In such situations there is a high probability of introducing syntactical errors while specifying these paths manually. The main advantage of accessing the WPS processes from the GreenLand

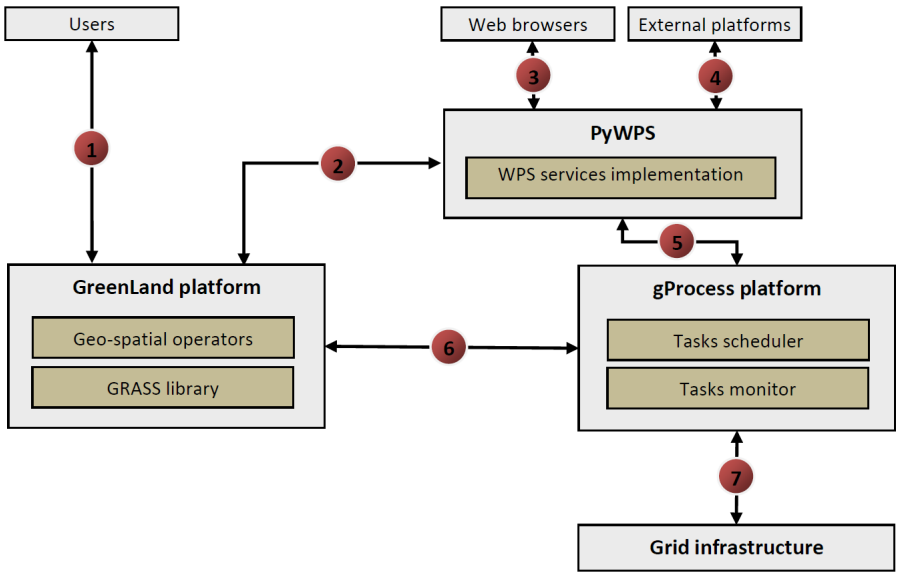


Figura 6.16: System related architecture of spatial data execution

platform is that it leaves no space for such errors. The user is provided with a list of aliases, instead of the real names of the satellite images. After selecting one input, the system generates in the background the entire path to that resource, and sends it to the PyWPS server.

The WPS processes could also be accessed directly from the browser (step 3 of Figure 6.16), by using its URL address. The main disadvantage is that the user has the full control in building the URL, including the paths to the input resources. Errors may occur in these cases, and the system is not able to provide specific support. This is the main reason why we recommend the usage of the first access method of the WPS services.

Using external platforms for invoking the PyWPS processes (step 4 of Figure 6.16) represent the third accessibility mode. It is similar with the GreenLand-WPS mechanism, but it lacks of some important features: the customized version of the metadata retrieved by using the *GetCapabilities* and *DescribeProcess* operations, the access to the GreenLand geospatial data repository, and the automatic generation of the input paths.

The user authentication is the first step highlighted in this system related architecture. This process requires the username and password credentials in order to be able to access the GreenLand resources. The traditional execution mechanism (steps 1, 6, and 7) does not require the PyWPS tool, and instead it passes the workflows directly to the gProcess platform that:

- partitions them into tasks;
- creates groups of tasks of similar complexity in order to load balance the entire execution process;
- allocates Grid machines and assigns them with one such group;
- transfers the input datasets to the corresponding worker nodes;
- starts the monitoring process that provides feedback to the GreenLand graphical user interface;
- combines the intermediate results in order to generate the final output of the workflows.

The second alternative for data execution is through the PyWPS server (the 1, 2, 5, 7 steps sequence) that has to role of exposing the GreenLand geospatial algorithms in a standardized manner that can be accessed by various means (see the previous paragraphs). The main difference between the two approaches is that in this case the PyWPS tool is the one that invokes the gProcess services. Once the GreenLand workflow related information reaches the gProcess platform, the execution process is similar to the steps described for the first mechanism.

The PyWPS tool uses Python scripts in order to describe a standardized method for accessing the GreenLand resources repository. The following paragraphs highlight the process of implementing a WPS standard algorithm that exposes the functionality of the NDVI workflow (Algorithm 6.1).

The header of the algorithm (lines 1-8) provides a general description of the NDVI workflow. The content of this algorithm is closely related to the metadata response generated by the *DescribeProcess* operation (Figure 6.13), regarding the specification of the input and output parameters (lines 10-20). The identifier and the title of each input/output have to be the same in both cases.

The actual connection to the GreenLand workflows repository is performed by the main body section of the algorithm (lines 21-25). It invokes the *executeChain* service (resident on the GreenLand platform) with the proper input/output values. Internally, the *executeChain* process recreates the basic workflow structure and sends it to the gProcess platform. The results

Algorithm 6.1 NDVI implementation as WPS service

```
1: WPSProcess.__init__(self,
2:     identifier = "NDVI",
3:     title="NDVI process",
4:     abstract="Computes the NDVI index.",
5:     version = "1.0",
6:     storeSupported = True,
7:     statusSupported = True,
8:     grassLocation = False)
9: self.NIR = self.addLiteralInput(
10:    identifier="nirBand",
11:    type=type(""),
12:    title="NIR band input;image/tif")
13: self.Red = self.addLiteralInput(
14:    identifier="redBand",
15:    type=type(""),
16:    title="Red band input;image/tif")
17: self.Result = self.addLiteralOutput(
18:    identifier = "result",
19:    title="Output result",
20:    type=type(""))
21: data=urllib.urlopen('http://cgis2ui.mediogrid.utcluj.ro/GreenL
22:    andv2/executeChain&process="NDVI"&inputs=[{
23:    "type":"tif", "value":"'+self.NIR.getValue()+'"},
24:    {"type":"tif","value":+ "'+self.Red.getValue()+'"}]).
25:    read()
26:
27: self.Result.setValue(data)
```

returned by this service is then assigned to the output variable of the Python script (line 27) and displayed in the *Execution* metadata response.

The processing of large volume of data involves the WPS integration within the Grid related execution and requires the extension of this standard with new capabilities. This way the *Monitor* operation was added that has the role of periodically display the information about the status of the execution. Figure 6.15 highlights an example of how the *Monitor* method can be used. Depending on the state of the execution, the *ProcessSucceeded* tag has different values:

- processing: in case the Grid execution is in progress;
- error: signals that the execution failed, without identifying the actual cause;
- completed: means that the spatial data processing was successful and provides a Web link from where the result can be downloaded.

Regardless of the processing type (traditional or standardized) the generated results are available for download. When using the GreenLand graphical interface, this action is implemented as visual buttons that once they are clicked a popup window is displayed from where the users are able to download the content of the results. For external access of the geospatial

workflows (through the WPS standard) the system provides only an array of Web addresses that contain the string representation of the path to these physical data resources. The downloading process is no longer assured by the GreenLand platform, instead the users have to perform it themselves.

6.4 Monitoring phase

During the processing phase the user must be informed all the time about the status of the execution. In order to be able to provide such services, a *monitoring* module (Figure 6.3) was implemented that periodically interrogates the application about its status. Actually, this module can be used for two purposes:

- Provide feedback about the process execution;
- Remote process control, such as canceling the execution. This step is semi-automated, and it is based on the first one. In other words, the user is the one that could cancel the execution or restart it, based on the information contained in the feedback provided by the monitoring module.

6.4.1 System related data model

Monitoring all the applications is hard to accomplish. There are several issues that must be taken into account while implementing this mechanism. One of them is the ability to collect information about each task of the process. It is known that at the execution stage the application is partitioned into multiple tasks, based on a specific algorithm. In some cases, in order to improve the execution time, the scheduler tool creates groups of tasks. In this case, the monitor module should be able to retrieve group level information.

Usually these tasks do not execute on the same physical machine (e.g. in the Grid infrastructure, each task is processed by a different worker node). This feature makes things even harder, because the monitoring module must establish a permanent connection with all these nodes and get information on a regular period of time. In case of a large number of tasks, the data transfer required diminishes the total bandwidth.

When tasks are executed simultaneously by multiple users the mechanism must be able to differentiate between them. This involves a unique identifier for each user, and a specific data model for storing the list of tasks assigned to that user. In other words the monitoring module should filter the requests by combining several parameters, and return relevant information. The monitoring data model parameters are listed in Table 6.1.

Another important aspect of monitoring the workflow execution is related to optimizing the data transfer between the client end-point and the monitor module that generates the feedback. In cases in which there are many tasks, multiple workflows, and a large number of users, the data size increases, and determines a bandwidth bottleneck of the network.

Each task has associated a monitor data model, like the one from Table 6.1. If the monitoring refresh time is low, and the workflow is partitioned in many tasks, transferring all these data models is inefficient. A possible solution to this problem is the verify on the server-side if the feedback generated at request r_i is the same as the one from r_{i-1}. If they are the same the monitoring module should send a simple text message (e.g. "same status"). This way we can avoid sending non-relevant information about the tasks.

One drawback to this solution is the extra computation that takes place at the worker node level. But the time required to perform this operation is incomparable smaller than the total

execution time of the task. On the other hand when the process displays the "running" status, the feedback remains the same almost through all this time. Regarding all these aspects, it could be said that is worth to adopt this compromise and to check the similarity of information on the server-side.

Parameter	Description
executionID	Because we could have simultaneous executions there is the need to differentiate between them by using a unique identifier (*executionID*). This parameter is also useful in situations in which the execution takes a long time (e.g. couple of days). In this case the user will certainly close the application that monitors the execution, and restart it after a while. At this point the monitoring module should be able to provide status information for this process, by querying the database with the *executionID* parameter and to extract all the relevant information.
workflowID	Unique identifier of the workflow. It is provided to the monitoring module by the application layer.
taskID	It is generated by the scheduler component (Figure 6.3) and uniquely identifies the tasks or the group of tasks that were generated during the workflow partitioning process.
userID	There is the need to know the identity of the user that executes the workflow, in order to be able to return him with relevant feedback.
startTime	Represents the time when the execution was started, having a regular date-time format: yyyy-mm-dd hh:mm:ss. In order to avoid the issues generated by the client's timezone, this information is computed based on the server timestamp.
duration	The number of seconds, counted from the *startTime* period. Based on this information the monitoring component could be enriched with a progress bar feature that provides information about the elapsed and the remaining time of the execution. This is possible by comparing the *duration* value with the workflow's processing estimation time value computed by statistic formulas (see previous section).
errorMessage	In order to offer better feedback to the user in case in which the execution fails, an error message will be useful. This message states what was the cause that generates the execution error, and gives the user the possibility to operate accordingly [145].
resultPath	When the entire execution completes successfully, it generates a list of output results. In order to offer user access to these results, the system should store a local path to them.

Parameter	Description
executionStatus	A parameter that identifies the status of the execution. The available list of statuses (submitted, running, completed, and cancelled) has the following significance: • **Submitted**: represents the time required to partition the workflow into tasks, schedule the worker nodes, assign each of them one particular task, and copy all the input resources required by the execution process. In some situations the workflow displays this status for a long time, due to bandwidth limitation that need to transfer large volume of data inputs, or due to the fact that the number of available worker nodes is smaller than the number of tasks. In the last case, the scheduler should wait for other tasks to complete, and then try again; • **Running**: assures that the entire list of tasks was properly submitted to the worker nodes and the actual execution takes place. Depending on the complexity of the process, this status could be maintained for a few seconds, minutes, or even days. One of the recommendations proposed by this methodology is that only workflows that require substantial processing time (e.g. tens of minutes) should be executed over the Grid infrastructure. Otherwise the submission time added with the Grid execution time will be greater than the execution time on multi-core machines. The previous section describes a decision algorithm that was implemented for this purpose. When the execution completes, the next step is to collect the intermediate outputs of all the tasks and to generate the final workflow result. Transferring such data from the Storage Element (SE) to different local machines is also part of this phase; • **Completed**: identifies the stage where all the tasks were executed. At this point the workflow results become available for the end-user for further analysis, interpretation, and download; • **Cancelled**: when one of the tasks generates an error the entire workflow displays the cancel status. This error could be influenced by the hardware infrastructure (e.g. one of the worker nodes becomes unavailable, the network connection is disable, the input resources does not exist, the Storage Element could not be accessed, etc.) or by software implementation of the workflow (e.g. the algorithm generates a null pointer exception). When the execution displays this status, the user has the possibility to re-submit the workflow.

Tabela 6.1: List of parameters of the monitoring data model

6.4.2 Monitoring the execution processes

The theoretical concepts of the monitor process were also implemented as a software module that is able to provide useful information about the execution status and allows the users to interfere (when needed) at runtime to the overall execution process.

This component is customized for each user, meaning that it can display the processes that belong to that user. There are multiple search filters (Figure 6.17) that can be applied in order to display the proper processes:

- Name filter: allows the users to search for particular keywords contained within the name of the processes (projects). The Figure 6.17 displays the results generated by applying the "test" name filter;
- Description filter: has a similar significance as the previous one, excepting the fact that it performs the search within the description of the processes;
- Status filter: has the advantage of presenting to the users only the processes that have a specific status (Submitted, Running, Completed, or Cancelled);
- Creation date filter: identifies all the projects that were created within a specific interval range;
- Additional filters: all the previous filters can be used in combination with the left hand side options. In case of many projects the user is able to reduce the results list by applying these additional filters over the results generated by the previous search algorithms.

In order to improve the processes visualization within the monitor list, specific colors were assigned for each status setup by the gProcess platform: red (cancelled status), blue (submitted processes), orange (processing tasks), and green (for the processes that were successfully executed).

6.5 Results analysis phase

The analysis of the output results is the last stage of the spatial data processing flow (Figure 6.3) and it is based on the fact that the executions of all processes were successfully finished. Usually, a workflow is partitioned at runtime into atomic tasks that are executed on different worker nodes that generate multiple intermediate results.

Most of the times, the workflow has several outputs. An intermediate execution result could be identified as one of the outputs of the workflow. When this happens, the system should

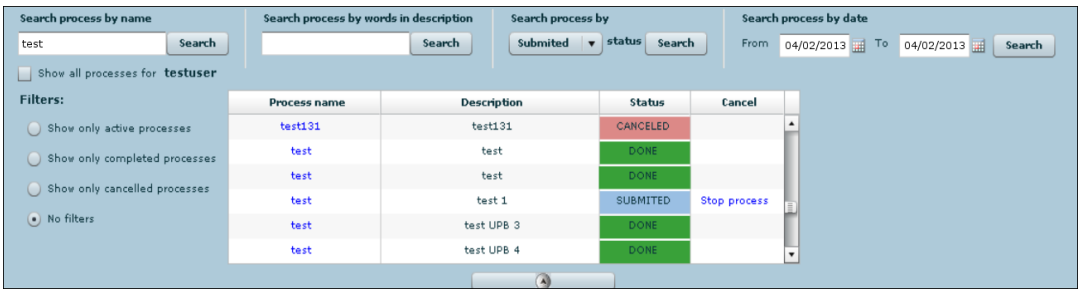


Figura 6.17: Interactive component for monitor the processes execution

mark the result as available and displays it to the user. Meanwhile, the workflow execution continues until all the intermediate tasks are processed.

After the results become available, they are presented to the users in different formats, depending on their needs. The GeoTIFF format is used as a system internal representation of spatial data, regardless of its original type. Keeping the results in GeoTIFF is useful for the analysis performed by the domain field specialists. But in some cases, the results demand to be presented in other shapes, easier to understand by non-technical types of users. The GreenLand results can be accessed by various categories of users for different purposes:

- Domain field specialists need to analyze the outputs in order to verify the correctness of the workflow they have implemented. Usually the result of a workflow that processes spatial data is described as an image that covers a certain geographic area. Extracting data, generating charts, reports, and statistics are the most frequently operations performed by this type of users;
- By analyzing the results, the data providers are able to identify the geographic locations that do not contain valid information, and operate accordingly (e.g. provide more relevant data);
- The results generated by predicting certain natural phenomena could be used by the decision makers in order to adopt the right resolutions for diminish the effects of these phenomena;
- Regular users on the other hand visualize the data for educational or informative purposes.

Figure 6.18 highlights two methods for analyzing the results obtained from spatial data processing. The online method allows the update of information directly from within the platform, while the offline mechanism requires the results to be downloaded on the user's local machine, apply the proper changes, and upload once more the new data into the system database. The online analysis is performed with the tools offered by the system, while the offline solution uses external applications to complete the results interpretation [150].

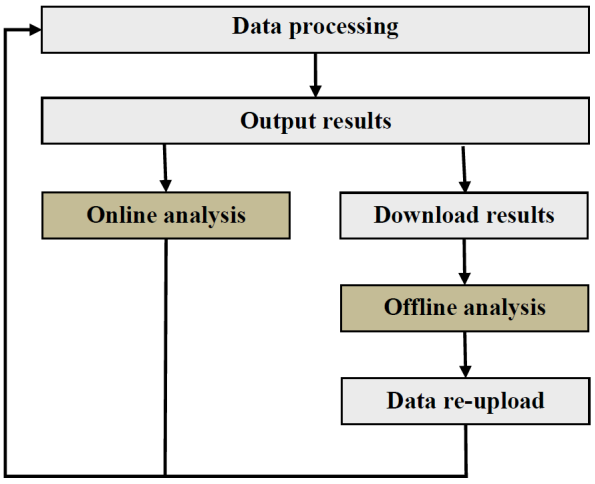


Figura 6.18: Results analysis methods

In some situations the online method provides enough support for the users, but there are cases when additional tools are required. The spatial data processing applications should offer the users the possibility to choose between the two solutions, or to combine them when needed.

The large size of the spatial information prevents data transfer for minor purposes. The main goal is to offer the users the possibility to perform online the basic analysis operations, instead of downloading the results, apply the same analysis algorithms on their local machine, and then re-upload the results based on the conclusions derived from this process.

For example, the calibration of a specific data model requires a large set of simulations, with different data inputs. Based on the results obtained at each simulation, new values should be generated in order to improve the output that will be obtained on the next iteration. In case of large data models, an online text editor for updating the parameters' values is useful, and reduces the time required by the specialist to calibrate the model. In the offline solution, the specialist needs to download the model on its local machine, change the parameters' values, and then re-upload the new files set into the system.

The pre-visualization of the results is another operation that could be provided online. Spatial data (satellite images in particular) have a high resolution (e.g. 7000x7000 pixels). Displaying images of this size is not efficient. Instead the system should generate a preview image that maintains the characteristics of the original data.

Adding certain editing features to the preview image increases the quality of the services provided by the online analysis method. Create supervised classification groups based on the image histogram, attach a specific color to each group, create legends and maps, represent a few possible features that could be integrated within the online data interpretation method.

6.6 Conclusions

The processing of large volume of spatial data is one of the main research directions of this thesis. In order to achieve this feature, an optimized execution data model is required that stores all the processing related data (e.g. input datasets, nodes assignment of the physical machines, the connections between the tasks, etc.). It is also necessary to use a powerful computing infrastructure that provides the desired output in a reasonable amount of time. For this purpose the Grid environment was chosen that offers parallel and distributed processing capabilities.

On the other hand, the GreenLand platform is also capable of executing simple geospatial algorithms (e.g. NDVI) whose processing does not exceed a few seconds. The Grid infrastructure can also be used for this purpose, but the total execution time (meaning: the workflow partition, the Grid nodes assignment, the input data transfer, the actual execution, and the final results generation) is greater than in case of the standalone environments. This way, two instances of the gProcess tool were created that handle the processing requests for the two infrastructures. In other words, there is the possibility to execute the workflows either on the Grid or standalone environments.

This chapter identifies and describes the steps that are involved in the spatial data execution process: pre-processing, actual data execution, monitor phase, and the results visualization and analysis. The theoretical concepts were described for all stages, together with the examples from the GreenLand platform that is the result of putting these concepts into practice.

One important aspect that is worth mentioning is the standardized execution of the GreenLand workflows that allows the achievement of platforms interoperability. This is essential for the applications that activate in the Earth Science domains because they can reuse the geospatial algorithms that were already implemented in other platforms, avoiding this way their development from scratch.

The results analysis and visualization phase was briefly described in this chapter, due to the fact that is going to be presented in more details in the following sections.

When processing large volumes of data, there are two main issues that are encountered in the majority of today's GIS related applications:

- The usage of the WPS standard within distributed environments, like the Grid infrastructure. When used on local machines it is easy to integrate as inner functionality within the applications, but when used on the Grid worker nodes several issues were encountered, like: execution monitor, security, large amount of time required to complete the data processing, etc. The current OGC WPS guidelines do not provide enough specifications of how to deal with these issues, so new ones have to be created;
- The incapacity of automatic extraction of data from remote repository in order to generate the proper input resources for near-real time processing algorithms.

The next paragraphs highlight the main contributions (solutions) that were proposed in order to overcome the issues generated by processing large volumes of spatial data:

1. The identification of the phases required for the spatial data execution process. Each step contains the theoretical background and the practical solutions implemented within the GreenLand platform;
2. The development of an interactive platform (GreenLand) that is able to facilitate the workflows instantiation with different data resources, retrieved from various locations;
3. The implementation of specific modules that are able to process the spatial data in near-real time, by accessing and retrieving the corresponding information from remote repositories, based on the FTP protocol and OGC standard services. This is useful for scenarios that predict certain phenomena and require up to date information;
4. The possibility of achieving platforms interoperability by allowing the execution of the workflows as WPS services. This means that the GreenLand geospatial algorithms can be accessed by other external platforms that implement this standard. The reverse process is also valid, meaning that the GreenLand is able to remotely execute functions developed by other applications;
5. The implementation of an interactive module that is able to monitor the workflows execution over the Grid and standalone infrastructures. This tool allows the users to interfere during the runtime phase, and stop the processing necessary.

Chapter 7

Spatial data visualization

The spatial data visualization and analysis is the last phase of the methodology proposed within this thesis. It assumes that the execution process was performed successfully and that the results are available to the users. The visualization term refers to the spatial data presentation in specific formats that are useful to end-users for the purposes they are visualizing this information. For example:

- the domain field specialists are interested in obtaining accurate data that can be used in different analysis experiments from where they can conclude about a certain phenomena. Statistics, tabular organization of the values of the parameters are only two of the formats needed by this type of users;
- the regular users do not necessary need accurate data, because they are more interested in results that can be easy to interpret and to visualize. They need this information for educational or informative purposes, so the interactive charts or the dynamic images are the most suited presentation styles for the regular users category.

The current chapter describes the theoretical concepts that are used in order to solve the issues related to the process of the spatial data visualization and analysis. This involves: the identification of the techniques and formats that are most suited for the output results presentation to various categories of users, the multi-platforms interoperability achievement through the use of standards, the extension of the GreenLand architecture in order to support standardized features for spatial data access, retrieval, and visualization.

7.1 Presentation techniques

The results generated by the spatial data processing are used for multiple purposes (by different categories of users) and require their presentation is various formats [151]. Figure 7.1 highlights the system related architecture that can be used for displaying these results. The spatial data is retrieved at runtime in its original format and sent for execution to the worker nodes. Once the results become available they are stored within the local or remote data repositories.

Meanwhile the system should be listening for the users requests related to the visualization of these results. At this point, the *Spatial data convertor* module is used in order to provide to the users the needed data. Because most of the times the processing of spatial data generates more than one single value, the adopted solution was to represent all of them as binary files.

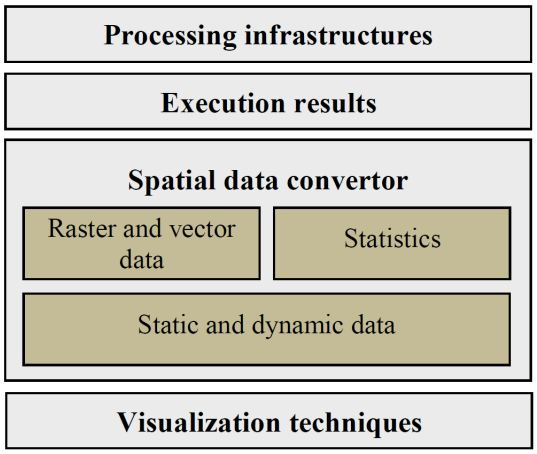


Figura 7.1: Conceptual architecture of the spatial data presentation model

By taken into account the users' requirements the following formats were identified as possible presentation techniques:

1. **Text based outputs**: this category includes the results within text files and delivers them to the end-user with no extra visual improvements. This type of presentation is useful for the domain field specialists that perform certain analysis onto the results generated by the workflows they developed.

   The calibration process of a hydrological model [152] is a perfect example where the text based files are needed. Each calibration step requires the specification of certain values for the input parameters. These values are established by analyzing the expected result and the previous measurements, and demand the repetition of the same step (the update of the input parameters) until the specialists consider that they have reached at the best calibration stage.

   The organization of each dataset of values as text files is one of the best presentation techniques in order for the specialists to complete the model calibration;

2. **Statistics**: useful for users' categories that involve the domain field specialists and the decision makers. An use case example can be about the areas classification process for a specific geographic region. The purpose is to create classes (e.g. vegetation, water, urban buildings) from the content of a satellite image without the user interference. The new generated map should be then compared to an ideal result, and provide through this experiment the statistical performance of the classification process. Based on these results, the domain field specialists are able to determine the best configuration of classes that are close to the ideal model;

3. **Dynamic charts**: represent a more interactive way of displaying the results information. It is most likely dedicated to the regular users, in order to facilitate the process of data interpretation and visualization. If we consider the first scenario (the calibration of the hydrologic model) [152] it will be useful to display the current calibration performance as a chart line, overlapped onto the average performance of the previous ones and the ideal values of the parameters. This way the users can easily identify how good the calibration process was, based on the current set of input data;

4. **Pseudo-colored images**: facilitate the possibility of representing the values within the satellite images as static pictures (e.g. JPEG, PNG, TIF) that are easier to interpret and understand than the text based information. The *Spatial data convertor* from Figure 7.1 is able to convert the output results to different types of images and to display them to the users.

   The results pre-visualization of the satellite images processing can be considered as a use case scenario that involves the representation of the outputs as static images;

5. **Interactive maps**: most of the times the result of processing a satellite image is also a satellite image, in the GeoTIF format. This means that this output can be overlapped onto interactive maps that allow features like: zooming, panning, area selection, distance computation, etc.

   The OGC standard allows this type of functionality through the WMS and WCS services, and improves the entire visualization process by increasing the users' satisfaction degree and the usability of the platforms that integrate these features. The retrieval of the characteristics of a sub-region inside the satellite image is one of the possible use case scenarios, where the interactive maps display technique can be used.

   The pseudo-color images and the interactive maps representation techniques are indicated to be used for the category of regular users, because they display the information in a simple and easy to understand manner, and do not require any special background knowledge;

6. **3D geo-visualization**: it allows the spatial data representation into a highly interactive environment, from where the user can perform certain actions is the same manner as in a real world. Among the processing results, this presentation technique also requires a Digital Elevation Model (DEM) of the geographic area and a set of tools that are able to render the 3D stage (e.g. SVG, X3D, etc.). There are several solutions [153] that provide support for this type of features, but it is worth mentioning that the methodology proposed within this paper does not address this type of visualization of the results, and provide solutions only for the rest of the techniques.

7.2 Platforms level interoperability

The communication based on the Internet services has a major role in propagating the information towards different endpoints. Its success is due to the capabilities of the software platforms to interact one to another, without requiring the user interference. In other words, the interoperability term assumes the existence of at least two applications that are able to expose, exchange, and use information based on a set of common interaction techniques, algorithms, and services.

The interoperability is important because it significantly improves the concept of data sharing among different applications that activate in similar domains. This means that the resources of one platform can be easily reused by other ones, eliminating the need to implement new solutions that generate the same set of resources. The usage of similar interaction techniques (that facilitate the users' actions) for data access is another issue that is addressed by the interoperability process. Instead of having different mechanisms implemented locally for each platform, this process proposes the development of common techniques that can be used within all the applications that interact between them.

Figure 7.2 highlights the non-interoperable communication mechanism between N platforms that provide services for the same activity domains. As can be seen, the *Platform 1* tries to

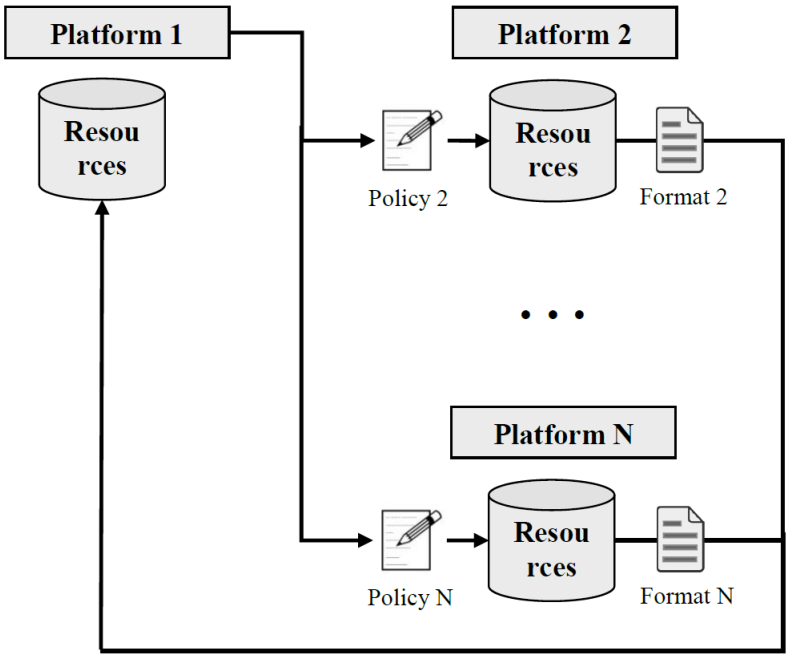


Figura 7.2: Multi-platforms communication mechanism

access data from the repositories of the 2nd and Nth application. In order to do that, it must comply with specific policies required by each of these platforms. It is worth mentioning that *Policy 2*, ..., *Policy N* are not identical and describe different access rules for the data that are stored within the corresponding repositories. This means that *Platform 1* must implement mechanisms that recognize all these rules and the internal organization of all the data repositories.

By following the previous approach there is almost impossible to achieve the interoperability features among all these applications. To complicate thing even more, after gaining access to the storage module, each platform may encode the result in a proprietary format and then sends it as the response of the *Platform 1*'s request. This means additional tools that are able to decode the information. Figure 7.2 highlights the worst case scenario, and of course that in reality it could happen that several platforms require the same policy rules and format the results in the same manner.

The interoperability proposed in this chapter overcomes these communication issues and simplifies the entire architecture to a single set of rules (policies) for accessing the remote data. On the other hand, the final result can be encoded in different formats (all platforms must be able to expose all these formats), but the type should be specified in the initial request.

In order to achieve the multi-platforms interoperability feature, the adopted solution is the usage of the OGC [36] standard, even though there are many other directives (e.g. ISO, [37] W3C [154], OASIS [39], etc.) that could be used. The motivation behind the OGC utilization is the fact that it provides the best guidelines for the spatial data management (see the *2.4.4 Open Geospatial Consortium - OGC section*).

The OGC is a non-profit international organization that provides rules and services for accessing, retrieving, processing, and visualizing the spatial data. The most important services that are used within the GreenLand platform are: Web Map Service (WMS), Web Coverage

Service (WCS), and Web Processing Service (WPS). A detailed description about all of them is going to be presented in the following paragraphs.

The WMS and WCS services provide a common policy for accessing, retrieving, and visualizing the spatial data from remote repositories. This single policy reduces the complex communication architecture (Figure 7.2) to only a few mechanisms that can be shared among all the platforms that are involved in this process. The results encoding protocol supports several formats (e.g. JPEG, TIF, PNG) and do not provide a direct access to the real data. Instead, copies of the original information are packed in separate files that are further passed to the platform that initiate the request.

All the platforms that implement the OGC standard must be able to provide the results in all the previous mentioned formats. Instead of identifying the type only after receiving the result (Figure 7.2), the solution based on the OGC services requires that this format should be transmitted together with other parameters of the original request.

7.3 OGC based standardized visualization

The spatial data are stored within specialized repositories, from where they can be used in different purposes: the modeling of the natural phenomena, the elaboration of decisions, rules, and policies in order to protect the assets from specific activity domains, the informative or educational data visualization, etc.

7.3.1 Theoretical concepts

There are many remote repositories that allow data access based on a proprietary access mode. This approach complicates the data sharing and platforms interoperability processes, due to the fact that for each storage new mechanisms have to be learnt in order to access the needed information. The OGC standard represents one of the possible solutions that can be used in order to overcome these issues that allows the accessing, retrieving, processing, and visualizing the spatial data [36]. From the users' point of view the advantages of integrating these standard services are:

- the improvement of data access by using a common set of interactive techniques, regardless of the platform from where they launch the original request;
- the relevant data search and retrieval based on the metadata information associated for each input;
- the improvement and standardization of the visualization process, by overlapping the spatial data over interactive maps.

The usage of the OGC services within the data visualization process has a major impact over the development of the Earth Science related platforms. Features like data sharing, interoperability between various platforms, optimization of the mechanisms that allow data access and data visualization, are easier to achieve when using the OGC standard.

The Web Map Service (WMS) represents the OGC product that can be used in order to optimize the spatial data visualization process, regardless of the orignal location of data. This service does not allow access to the original information, instead it generates identical copies that are encoded as JPEG, PNG, or TIF images.

The WMS service has the role of reducing the complexity and variety of the interaction techniques that were previously used for data access and visualization. The systems that implement these methods record an increase of the users' satisfaction degree by reducing the time

necessary for performing their tasks, in comparison to the platforms that provide traditional visualization mechanisms.

Depending on the users' requirements, the WMS service can be customized through its parameters that support the specification of: the geographic area of interest, the format of the results, etc. All these actions are possible based on the *GetCapabilities* and *GetMap* operations. Figure 7.3 highlights the conceptual architecture for the systems that implement the OGC related services.

The identification of the information from the remote repositories is the first phase of the WMS process, and it is perform through the *GetCapabilities* operation. The invocation of this method is similar to the WPS service (see section *6.3.3.1 WPS general overview*) and consists in specifying the name and the version parameters: http://<server_ host>/service=wms&version=1.0.0&request=GetCapabilities. The metadata response is similar to the WPS one that contains:

- some basic information about the server that hosts this methods;
- a list of layers (bands) that correspond to individual satellite images;
- for each layer the following metadata are available: geographic location, the supported encoding format, a list of keywords that facilitate the search actions, the type of the projection, etc.

The next step is represented by the extraction of the relevant data, based on the *GetMap* operation. The original layer has a well defined geographic location, but this service allows the users to specify inner areas and to visualize only fractions of the initial data. This is possible through the list of the parameters that are required by the *GetMap* operation, such as: the geographic coordinates of the area of interest (*bbox* parameter), the list of bands that should be retrieved from the repository (*layers*), the type of the output result (*format*).

A customized example of using the *GetMap* method is given bellow: http://<server_ host>/service=wms&version=1.0.0&request=GetMap&bbox=20,48,29,43&layers=snow_classification&format=image/jpg. The obtained result is of JPEG type and represent an inner area of the layer that has the *snow_ classification* identifier.

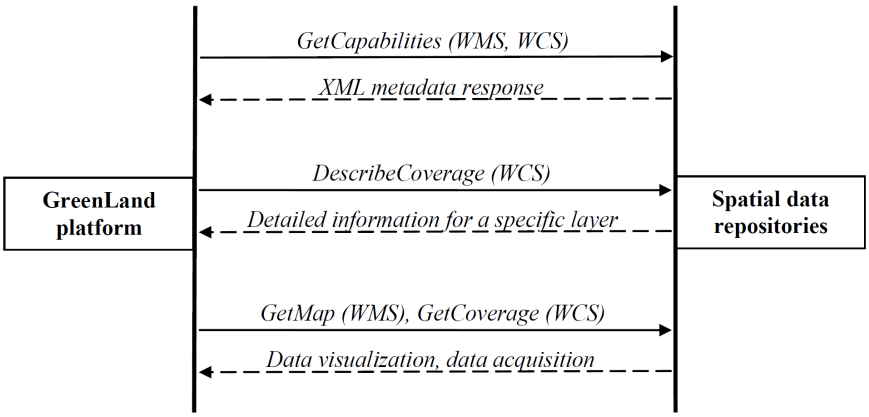


Figura 7.3: The conceptual architecture for spatial data visualization and retrieval through the WMS and WCS services

In its original implementation, the WMS service provides the outputs as static images. In order to improve the visualization process, the solution was to overlap these results over interactive maps and to make use of the features they provide. The advantages of using this approach are:

- the extension of the visualization process to other categories of users that do not have special background knowledge about the process that generated that result;
- a better identification of the geographical areas contained within the output image;
- the visual selection of sub-areas that the user is interested in;
- data sharing achievement between platforms that provide services for the same activity domains;
- the interactive display of prediction results and the possibility to visually compare different sets of data;
- the increase of the users' satisfaction degree.

7.3.2 The GreenLand visualization process improvement

Figure 7.3 highlights the GreenLand system related architecture that allows the standardized visualization of spatial data from local and remote repositories. It is based on the request-response model and establishes a direct connection between the GreenLand platform and the data repositories that expose the information in an OGC compliant mode [155].

The spatial data visualization within the GreenLand platform is a 4-step process that starts with implementing the required OGC WMS policies for data access, and goes up to the satellite image display over the interactive maps:

1. **Data access**: It is the first phase of this process that establishes a direct communication channel to the remote repository that stores the spatial data. The GreenLand platform provides a large number of algorithms for the spatial data related domains that can be instantiated with entities from the default GreenLand storage. The access to this information is done through the user authentication mechanisms.

   In some cases, the execution of such an algorithm requires data from remote repositories that are not part of the GreenLand platform and whose information are exposed based on the metadata attributes. The main advantage of using these resources is that their maintenance is not one of the duties of the GreenLand users.

   Figure 7.4 highlights the GreenLand graphical user interface that was designed for the management of the OGC services. The Web address of the repository is the only information needed by the GreenLand platform in order to establish the communication channel between the two entities. The *Connection* button can be used to achieve this and to allow the users to access the required data.

   Throughout the experiments, the WMS service was tested on several remote storages. Because of the requirements of the Black Sea catchment case study [77], the research was focused especially on the URM data repository. One of the goals of this scenario was to validate the OGC services integration within the GreenLand platform, and to develop new interaction techniques that improve the overall spatial data visualization process;

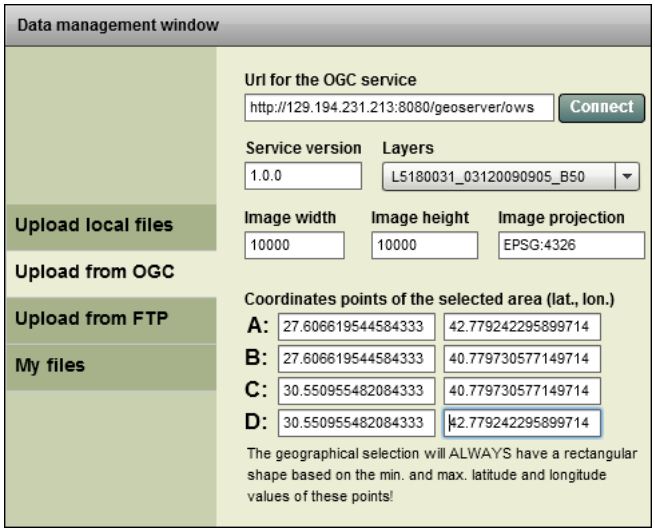


Figura 7.4: The GreenLand graphical interface for OGC products management

2. **The relevant information search**: Once the connection to the remote repository was successfully established, the users have the possibility to query the entire database in order to obtain the needed information. The *GetCapabilities* operation can be used for searching the relevant data. The request of this method is done automatically and includes the Web address of the server and a list of mandatory parameters. As a response, the remote server sends a XML file that lists all the data (layers) that are stored within the repository.

   Each XML entry is similar to the ones presented in Figure 6.12. The manual analysis of the XML files is not that complicated, as long as the inner information has a less complex structure. It is worth mentioning that in most of the cases, the quantity and the structure of the spatial data generates a large XML document that is hard to analyzed, even by the domain field specialists.

   This way a new graphical interface was implemented that optimizes the display of the XML information into a format that is easy to interpret by any type of users. The proposed solution reduces the number of user-actions required for relevant data search, and simplifies the procedure of identifying and selecting the needed resources (Figure 7.4).

   A new mechanism was implemented within the GreenLand platform that is able to automatically interpret the *GetCapabilities* response and to display it as an interactive list of layers. When selecting such an item, the system provides extra details about: the geographic location (as latitude and longitude coordinates), the projection type, pixels dimension of the map surface represented by this input, etc. All these information have a corresponding GreenLand graphical module (see Figure 7.4) that can be customized with specific values, before the actual data visualization. The optimal XML representation increases the GreenLand usability, by organizing the information into graphical modules and by removing un-necessary metadata.

   In the example from Figure 7.4, the layers list marks as selected the item L5180031_031120 090905_B50. The *GetCapabilities* metadata response identifies for the selected layer the following information: width in pixels (10000), height in pixels (10000), and its original

projection (EPSG: 4326). All these data are part of the WMS implementation within the GreenLand platform, while the coordinates related graphical elements represent the WCS module.

When dealing with experiments (scenarios) that require a large number of input data, the above solution is inefficient, due to the fact that the user has to navigate through all the layers that are available in the remote data repository. A second approach can be used, in order to overcome this issue, and consists in filtering these items based on: a list of keywords or by reducing the geographic area of interest to a certain region.

A set of experiments were performed that specify different values for these filters that are interpreted by the system and applied directly on the content of the remote repository. After this process, the *GetCapabilities* response stores only the results that are relevant to the user's search;

3. **Results generation**: Figure 7.3 highlights the working mode of the WMS service. Until now only the *GetCapabilities* operation was describe that has the role of presenting the available resources as XML entries. Once the user specifies all the required information (image width, image height, and projection type) the next step is to retrieve the associated data that fulfills this configuration. The *GetMap* operation can be used in order generate the final result in the JPEG format (even thought the WMS is able to provide the same data in multiple formats, the adopted solution was to use the JPEG one because of the fact that it is the best alternative regarding the quality/transfer rate ratio).

   It is worth mentioning that the *GetCapabilities* and the *GetMap* operations are automatically invoked by the GreenLand system, based on the user-actions that take place at graphical interface. In the simplest form, the *GetMap* result is presented to the user as a static JPEG image that can be downloaded from the GreenLand platform. The next step describes the approach used for overlapping this image onto interactive maps, increasing this way the system usability and the users satisfaction;

4. **Interactive visualization**: The geospatial algorithms require as input georeferenced data. It is worth mentioning that the *GetMap* operation does not return the real spatial information, and instead it generates a copy of it as JPEG, PNG, or TIFF image without the georeferenced metadata. The solution to this problem is to combine the static image result with the initial WMS request and to provide the correct input dataset for the processing of the geospatial algorithms.

   There are two techniques in visualizing the WMS outputs: the direct view of the static images (by using the online or offline editors) or the dynamic display over interactive maps that provide more powerful tools in the domain of the spatial data analysis and visualization. The GreenLand solution uses the second approach in order to simplify the visualization process (from the user point of view), by avoiding the action of downloading the result onto the user's local machine and to analyze it by using one of the existing image editors.

   The visualization of spatial data is closely related to the geographical area that it represents. This way, the interactive map based technique is adequate and facilitate users to identify the needed information by the use of the features provided by this dynamic map.

   The Esri map [156] is one of the possible products that can be used in the interactive visualization process of the spatial data. It provides the common features (e.g. pan, zoom, locations search) and also some additional mechanisms that facilitate the management of the satellite images.

In order to exemplify these concepts, the NDVI classification index was apply for the Istanbul geographic area. This algorithm is used for identifying the vegetation areas and the regions that presents other features. The output result was transferred to the GreenLand repository and accessed by the OGC WMS service. The generated JPEG image was overlapped onto the Esri map, like highlighted in Figure 7.5. The color significance is the following: green (the vegetation area within the Istanbul range), blue (the water resources), and red (the areas covered with urban buildings). The black contour that is presented on the border of the image represents the geographic locations that cannot be correctly classified.

7.4 Relevant data retrieval

The visualization process (described above) provides a simple and interactive method for presenting the results of the geospatial algorithms. For the majority of the users these features are enough, because they want to view the results and to easily interpret it. An important characteristic for the Earth Science domain field specialists is the possibility to analyze fractions of the original result, by using powerful mechanisms that allows the extraction of relevant data

7.4.1 Theoretical concepts

This is done by the Web Coverage Service (WCS) method that initially looks similar to the WMS one, meaning that both can be used for data search through the *GetCapabilities* request. Figure 7.3 highlights the communication process, based on the WCS service between the spatial data related platform and remote repositories that store this information. For the WCS method this process contains three phases (*GetCapabilities*, *DescribeCoverage*, and *GetCoverage*) and represents the first difference in comparison to the WMS that requires only two operations.

The *GetCapabilities* method is similar to the ones presented for the WMS and WPS products (Figure 6.12), and has the role of providing relevant metadata about the content of the remote repository. It is the first phase of the communication process that also grants the access to the storage resources.

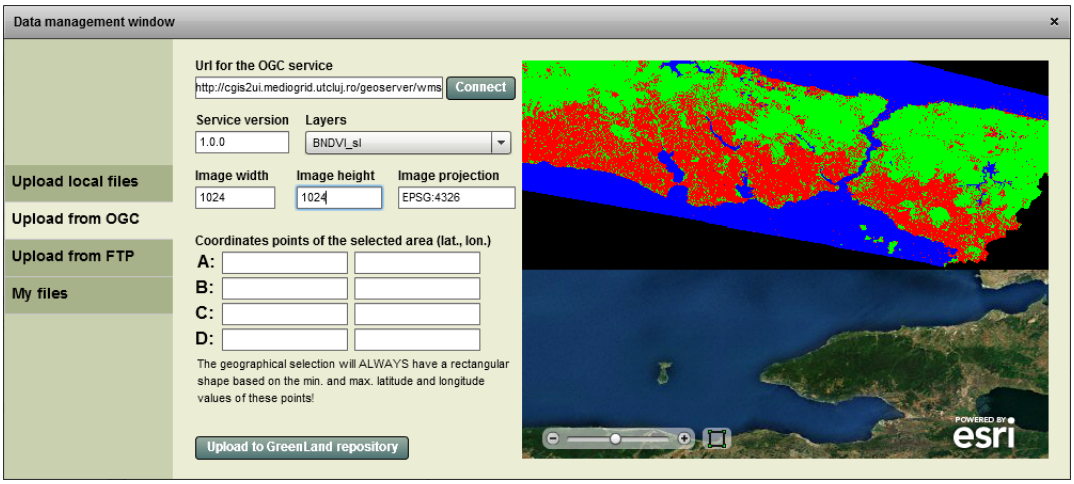


Figura 7.5: Standardized visualization of the NDVI index based on the WMS service

The second operation (*DescribeCoverage*) is used in order to find out more details about a specific entry from the remote repository. The version, service name, request, and the identifier of the coverage are the mandatory parameters when invoking this service: http://<server_ host>/service=wcs&version=1.0.0&request=DescribeCoverage&coverage=NDVI. Figure 7.6 exemplifies the response for the NDVI layer that contains: the resource description, the supported formats in case of retrieving the data, a list of interpolation algorithms that can be used for further processing, etc.

The *GetCoverage* operation is the third step of the WCS based data retrieval process and has the effect of generating the required output, based on the parameters specified when invoking this service. The results are packed as one of the formats described in the *supportedFormats* tag (Figure 7.6). In the simplest version, the *GetCoverage* request can be used like this: http://<server_ host>/service=wcs&version=1.0.0&request=GetCoverage&coverage=NDVI& crs=EPSG:4326& bbox=40.5, 28.3, 41.05, 29.8&width=1024&height=1024. There are several additional parameters that do not appear for the previous operations:

- Coordinate Reference System(CRS): represents the projection type of the result. Depending on the further usage of the image, the final output can be generated with different projections in order to ease its analysis for specific activities;
- bbox: a virtual rectangle that indicates the area inside the original image that should be retrieved by the *GetCoverage* request;
- width and height: the result's dimension (in pixels).

```
<wcs:CoverageDescription version="1.0.0">
    <wcs:CoverageOffering>
        <wcs:description>The NDVI result for the Istanbul geographic area</wcs:description>
        <wcs:label>NDVI_result</wcs:label>

        <wcs:lonLatEnvelope>
            <gml:pos>27.967493192870162 40.78920896353799</gml:pos>
            <gml:pos>29.980150494419515 41.58791671487258</gml:pos>
        </wcs:lonLatEnvelope>

        <wcs:supportedCRSs>
            <wcs:requestResponseCRSs>EPSG:4326</wcs:requestResponseCRSs>
        </wcs:supportedCRSs>

        <wcs:supportedFormats nativeFormat="GeoTIFF">
            <wcs:formats>TIFF</wcs:formats>
            <wcs:formats>GeoTIFF</wcs:formats>
            <wcs:formats>JPEG</wcs:formats>
            <wcs:formats>PNG</wcs:formats>
        </wcs:supportedFormats>

        <wcs:supportedInterpolations default="nearest neighbor">
            <wcs:interpolationMethod>nearest neighbor</wcs:interpolationMethod>
            <wcs:interpolationMethod>bilinear</wcs:interpolationMethod>
            <wcs:interpolationMethod>bicubic</wcs:interpolationMethod>
        </wcs:supportedInterpolations>
    </wcs:CoverageOffering>
</wcs:CoverageDescription>
```

Figura 7.6: Metadata response of the DescribeCoverage request

The *GetCoverage* response is no longer a XML file. Instead the final output is a static image, packed in one of the types available in the *supportedFormats* tag. If the format is not specified as an additional request parameter, the default one will be used (see Figure 7.6). The main advantages of using the WCS service for data retrieval are:

- a simplified communication architecture between multiple platforms that store and process spatial data (Figure 7.2);
- Data sharing and platforms interoperability improvement;
- The implementation of a set of common interaction techniques that can be used by various categories of users, without the need to have domain related background knowledge.

7.4.2 Interactive data retrieval through the GreenLand platform

The WCS implementation within the GreenLand platform is a little bit extended than the original version. Instead of manual specification of all operations (*GetCapabilities*, *DescribeCoverage*, and *GetCoverage*) the GreenLand platform provides an interactive module that automatically performs these requests, based on the user-actions from the graphical interface level [157].

Figure 7.7 highlights the data retrieval process for the NDVI result generated for the Istanbul geographic area. This module is identical with the one used for the interactive visualization process (Figure 7.5) and includes features that cover both WMS and WCS services. These two methods are handled by the same component because usually the first step is to select and visualize the original data and only after that the retrieval of certain areas.

After the identification of the data resource and its display onto the dynamic map (WMS and WCS *DescribeCoverage* sections) the user is able to interactively specify the coordinates of the geographic region that he is interested in (WCS section). This action can be achieved through the following steps:

- Enable the drawing mode feature, by using the bottom-right button, near the zoom functionality;
- Click on the map in order to establish the coordinates of the point A that represents the starting corner of the selection;
- During the mouse down event, the point D position updates according with the movement of the mouse device. At the same time the values for the B and C items are setup automatically. As can be seen there is a perfect coordination between the user-action performed at the map level and the graphical elements from the left corner of Figure 7.7;
- The mouse button release corresponds to the final shape of the rectangle that graphically represents the geographic region that is used in the data retrieval process. At this stage, the horizontal and vertical resize and the translation of the rectangle are valid actions;
- If parts of the rectangle are outside of the original image, the GreenLand system automatically performs the proper adjustments in order to assure a correct identification of the geographical area of interest.

The previous approach is useful for cases that do not require an accurate identification of the geographic region. For domain field specialists that perform complex studies an error of a few seconds-degree in latitude or longitude can introduce wrong input datasets. In these

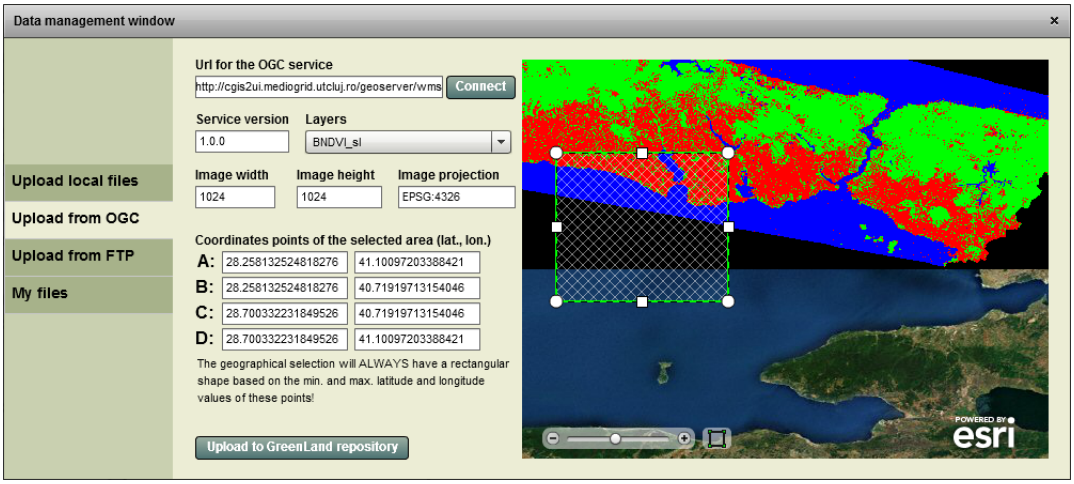


Figura 7.7: The GreenLand interactive data retrieval process

situations the GreenLand platform provides an alternative technique for specifying the area of interest, based on the graphical text inputs that support float numbers with multiple decimal places. The four attributes (A, B, C, and D) represent the same points as in the case of the mouse-based selection and identify the four corners of the rectangle.

Once the selection process completes, the GreenLand system automatically collects these metadata (image width and height, projection type, and the bounding box coordinates) and use them in order to retrieve the specified geographic area (through the *GetCoverage* operation). The result of this method is stored within the GreenLand data repository and become available to the users for further processing.

7.5 Conclusions

The GreenLand platform provides several features that intend to improve the visualization process of spatial data. First of all an identification of the best data presentation techniques is performed at the beginning of this chapter that has to role of specifying what methods should be used for the: domain field specialists, data providers, decision makers, and regular users. The data accuracy, the simplicity in using the corresponding interaction techniques, the local or remote data analysis are the main factors that interfere with this decision.

One of the best data visualization techniques it to extend the services provided by the OGC standard and to integrate them with different dynamic maps in order to increase the overall system usability. The GreenLand approach is to display the WMS results over the Esri [156] interactive map and to allow users to manage the spatial data by visual means.

The OGC implementation also facilitates the data sharing and platforms interoperability processes. The first one is achieved through the WCS service, while the WPS exposes the GreenLand workflows in a standardized manner with the purpose of being reused by external platforms.

When providing solutions that are related to the spatial data visualization and analysis there are several problems that have to be taken into account:

1. Facilitate the data sharing among different communities of users that activate in the same research domains;

2. Improve the communication between applications that provide similar services. The resources re-usage can be a benefic solution in most of the cases;

3. Facilitate the access of non-technical users to the features provided by the online spatial data visualization applications. Most of them are dedicated only to domain field specialists that are required to learn new functionality every time updates arose.

The main contributions that can be defined based on the features presented in this chapter, are listed in the following paragraphs:

1. The implementation of an automatic mechanism that translates the XML based response of the OGC services into visual components that facilitate their utilization by various types of users;

2. The development of a communication architecture (based on OGC services) between the GreenLand platform and remote spatial data repositories. This module is able to improve the data sharing and applications interoperability that provide services for the same activity domains;

3. An identification of the most important techniques that can be performed for various types of users, regardless of their background domain knowledge.

Chapter 8

GreenLand platform validation on real use case studies

The theoretical concepts proposed within this thesis define a methodology for developing, processing, and visualizing various use cases from the Earth Science activity domains. Even though these concepts were partially validated in the sections they were described, this chapter proposes two complex experiments in order to test all the functionalities implemented through this methodology.

These two scenarios represent real use case studies that were formulated within the enviroGRIDS FP7 project [80], by domain field specialists and reflect the current problems they are confronting with, in terms of water quality/quantity and land cover/land use aspects.

These experiments focus on validating the concepts (proposed by this methodology) implementation within the GreenLand platform. A detailed description of the performance evaluation was given in section *6.3.2 Execution management over different infrastructures.*

8.1 The main features of the GreenLand platform

The GreenLand platform activates in the Geographic Information System (GIS) domain and provides solutions for spatial data management, modeling, processing, and interactive visualization. Even though it handles geospatial algorithms, it can be easily extended to other Earth Science activity domains, because of its flexible structure of representing various types of experiments and use cases.

The GreenLand system is based on the methodology proposed within this thesis, and implements all the theoretical concepts presented throughout this paper. It does not intend to re-implement from scratch the basic GIS functionalities (e.g. the NDVI algorithm) and for this reason it uses the GRASS library [59]. Instead, it is focused on improving the existing interaction techniques in order to be used by non-technical persons that do not have an impressive background knowledge in the related Earth Science domains. On the other hand, the GreenLand platform tries to fill the gaps between various functionalities, and provides optimized solutions for processing large volume of spatial data over different computing infrastructures. The main features of the GreenLand platform are going to be described in the following paragraphs.

8.1.1 Interactive use cases description

The proposed methodology describes the operator and workflow concepts as possible entities for modeling various use cases and experiments. The operators can be used for representing unitary modules (formulas, algorithms) that cannot be partitioned into other smaller entities. The implementation of these operators is facilitated through the OperatorEditor tool that provides a specific API with the default behavior of a generic geospatial algorithm. The users are then assisted in every phase of the development process: implementation, integration within the GreenLand platform, solving any possible errors and issues, etc.

A classical example of geospatial operator is the NDVI algorithm that is used for automatic classification of the vegetation from specific geographic regions. It is based on formula 5.4 that is applied on each pixel from the Red and NIR satellite image inputs. In the end it generates a monochromatic map with values within [-1; 1] interval. The NDVI algorithm cannot be partitioned into independent sub-algorithms and that is why it fits the exemplification of the operator concept. Figure 5.6 highlights the NDVI implementation as a GreenLand operator and how it can be used in further processing.

In most cases, the experiments consist in a large number of modules that interact based on a specific set of rules. Once the requirements of the use case are described in natural language, the WorkflowEditor application provides mechanisms that facilitate their definition within the GreenLand system. The functionality of each module should be identified as one of the existing operators/workflows. Otherwise, the users have to create a new structure (with existing resources) that encapsulates the desired function.

When the entire use case is well defined, the domain field specialists are able to use the WorkflowEditor features in order to translate it from the natural language description into a visual representation and then into an internal XML format (called workflow), recognized by the GreenLand platform.

Another main concept that was defined at the GreenLand level is called project that can be described as a virtual container that stores multiple workflows. Once the use case (represented as a workflow) is available within the GreenLand platform, the users are able to pack it as one of the entries of the project that will be further used at runtime.

It is worth mentioning that one project can store multiple workflows, even instances of the same use case (Figure 8.1). At execution time each workflow is consider being independent and generates its own result that cannot be linked as input to other entities. This approach is useful when the users need to predict certain phenomena and need multiple instances of the same workflow, with different input data types.

Figure 8.1 highlights a project example (called *Demo project*) that contains 3 different workflows (*BlackSeaMosaicPDG*, *DensitySlicingPDG*, *LandsatAtmosphericCorrectionPDG*) and 3 instances of the *SubsetPDG* item. An automatic grouping mechanism is applied to these instances that allow expansion, retraction, adding, and removing specific features. The execution process cannot start until the GreenLand project contains at least one workflow.

The project concept contains only the most important information regarding its content and the execution process description. In Figure 8.1, the selected project has the *Content provided* status which means that it has at least one workflow and that it is ready for execution. Other possible statuses are: *Empty* (setup just after the project creation), *Executing project* (during the execution process), *Error* (the processing failed), and *Execution completed* (the processing results were properly generated);

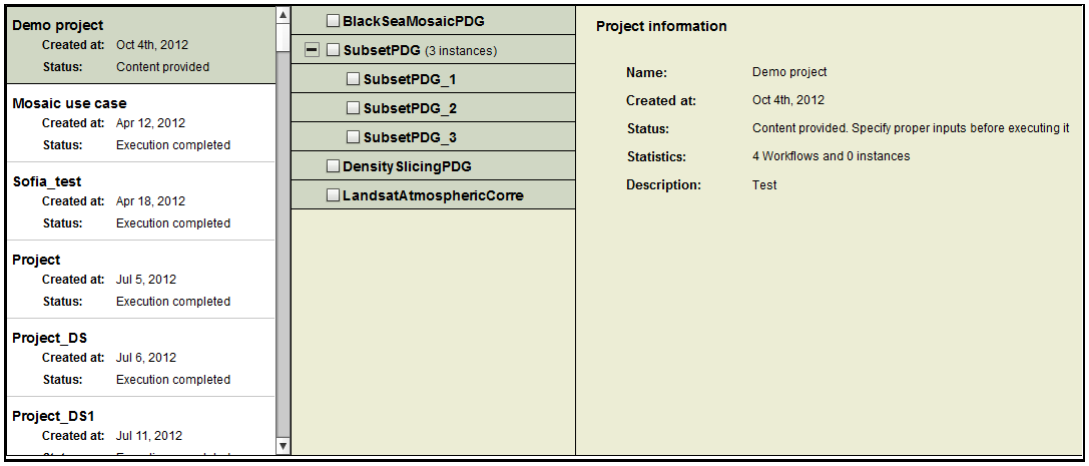


Figura 8.1: Workflows organization inside GreenLand projects

8.1.2 Data retrieval mechanisms

After identifying the operators and describe the general use case through the GreenLand workflow concept, the next step is to assign to each input the proper spatial data. Depending on the use case requirements, there are three types of data retrieval that could be used: local upload of information, FTP resources retrieval from remote repositories, and OGC WCS based technique that was described in the previous chapter. It is worth mentioning that in all these cases, besides the actual data transfer, the GreenLand performs an automatic conversion of the original information into the GeoTIFF format. This way it is easier to unitary handle various types of inputs.

The local upload of data allows the users to transfer the spatial information from their own computers to the GreenLand repository. If multi-band satellite images are uploaded, the system automatically partitions them into individual bands. Figure 8.2 highlights the case of

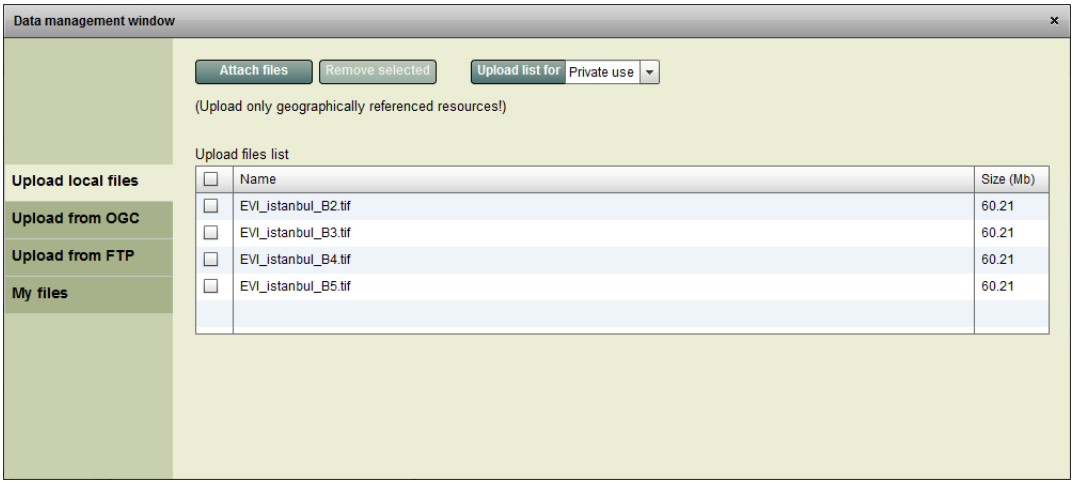


Figura 8.2: Local upload mechanism

transferring four bands of the *EVI_istanbul* satellite image. A short summary regarding the name and the size of each input is presented to the user.

After importing the image, the GreenLand platform automatically converts it to the GeoTIFF format. The conversion result, together with other useful features (e.g. search, download, change privacy) are found in the *My files section* (Figure 8.3). It is worth mentioning that multiple resources can be downloaded at the same time, encoded as an archive.

In case of workflows that require large amount of data, the local files upload is not an efficient solution. Instead, the FTP based approach can be used that provides support for downloading specific resources from remote repositories by using the FTP protocol. For example, the Black Sea catchment modeling workflow (called Mosaic12) uses this solution in order to retrieve more than 500GB of information. Another advantage of this mechanism is that it can extract relevant data at runtime, avoiding this way the process of manually specify the input datasets. This feature is also important for processing near real time algorithms that require fresh data, updated periodically into specialized repositories.

The third method is more interactive then the previous ones and has the advantage of allowing the retrieval of sub-sets of the original image. Figure 7.5 describes the entire process of using the WCS service in order to achieve this functionality.

After all the required data is available into the GreenLand platform, the next step is to assign it to the corresponding inputs of the workflow that is going to be executed. Figure 6.7 highlights how the input specification process is performed within the GreenLand system, by taking into account the NVDI algorithm. In this case, each input is displayed on a new line and it is represented by three visual components that define: its name, its description, and its value. For the satellite image type, the users are able to select one item from a list of multiple possible GeoTIFF bands. For numerical based inputs, the system applies a specific number validator.

There are cases (e.g. Mosaic12, Density slicing workflows) whose input specification process is slightly different, and requires a visual representation for a palette color, a numeric interval, a list of items, etc. For all these situations, the GreenLand proposes different dynamic elements that correspond to these types of resources. The color picker could be used to select the appropriate nuance, two text inputs can simulate the role of a mathematical interval, multiple check boxes are adequate for representing the list of parameters that have to be sent at runtime

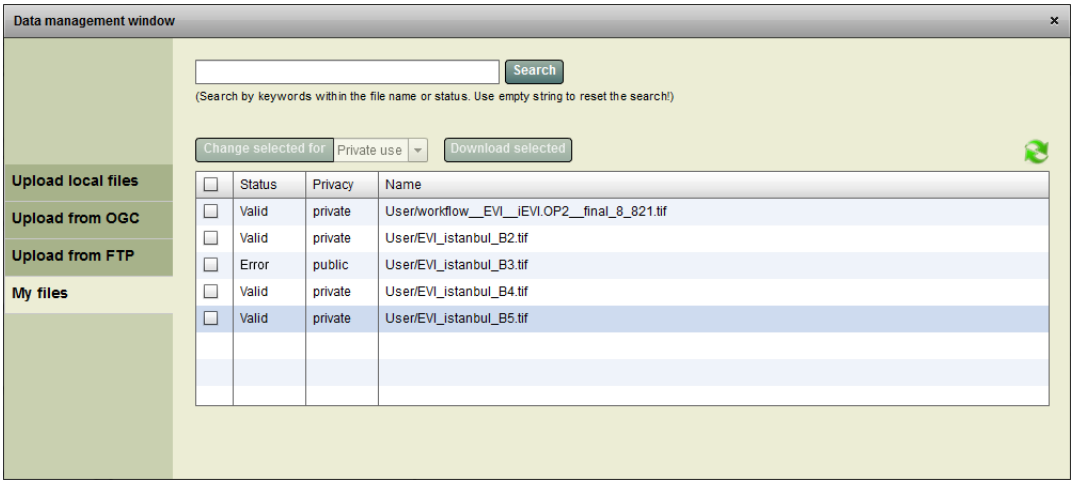


Figura 8.3: GreenLand spatial data management system

(Figure 6.7, Figure 6.8);

8.1.3 Optimized spatial data processing

The methodology proposed within this thesis provides flexible solutions that allow the spatial data processing over different computing infrastructures, such as: Grid, Cloud, clusters, multi-core machines, etc. This situation is possible due to the fact that the entire use case is developed as a mathematical graph that can be partitioned into tasks that can be executed in parallel over these infrastructures.

The GreenLand platform experimented only the Grid and standalone environments and provides manual and automatic mechanisms that are able to compute the workflow's complexity and to redirect the execution over the proper infrastructure. The previous section highlights these two situations and how the GreenLand manages them: the workflows with higher weights (that can be partitioned into multiple independent tasks) are recommended to be executed over the Grid infrastructure, while the less complex use cases should be computed on the standalone environments.

The GreenLand platform addresses various categories of users when it comes to processing large volume of spatial data, due to the fact that it hides from these users the complex mechanisms that are involved in the overall execution process. That is why even non-technical persons are able to launch new processes in order to visualize and analyze the behavior of a specific workflow.

The gProcess platform is used as a middleware layer between the GreenLand system and the Grid/standalone infrastructures. It has the role of receiving the workflow's structure and to partition it into smaller tasks and to execute them over different physical machines. When needed, the gProcess platform is able to manage parallel and distributed executions and to recompose the final results based on all intermediate outputs that are connected through the edges within the workflow. During the entire execution process the user is able to see the status for all the active processing and to interfere in the overall execution process.

An estimation of the remaining execution time is not possible when processing the spatial data over the Grid infrastructure, due to the fact that multiple factors can interfere and unbalance the estimation: network traffic, the availability of the worker nodes, the complexity of the workflow and the number of tasks generated by the gProcess platform, etc. Instead, the GreenLand platform measures the actual execution time (Figure 8.4) that is also part of the monitoring module.

The table from Figure 8.4 highlights all the workflows that were contained within the project that was launched for execution. When one of the results become available, three options are allowed to be performed: reuse it as input for further processing, export the output to external platforms by using the GeoServer tool [53], and download the result onto the user's local

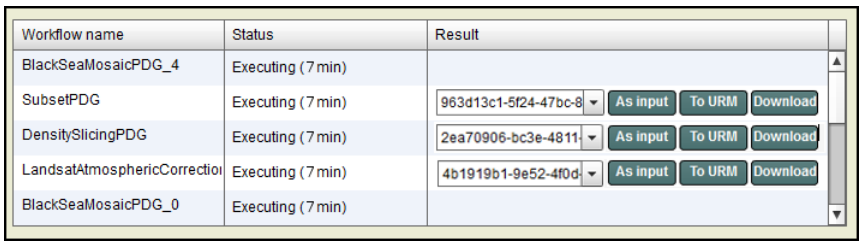


Figura 8.4: GreenLand execution monitor

machine. If the workflow generates multiple outputs, they are all displayed into the combo box located in the last column of Figure 8.4;

8.1.4 Data sharing and platforms interoperability

Due to the fact that more and more application activate in the Earth Science domains, it is necessary to provide compatible tools, in order to be used when needed and to avoid the re-implementation of the already existing mechanisms.

The data sharing process is achieved based on the OGC services that allow data to be exposed in a standardized manner to other platforms that are able to understand these concepts. The *GetCapabilities* operation can be used in order to obtain a list with all the available resources from local or remote repositories, while the WMS and WCS services provide specialized tools for data visualization and data retrieval. More information about these features were presented in the previous chapter *Spatial Data Visualization*.

Besides the standard data sharing, the OGC services, implemented within the GreenLand platform, facilitate the user-actions required for data search, retrieval, analysis, and visualization. This is possible by developing a set of common interaction techniques that can be used throughout all the applications, instead of learning the proprietary mechanisms implemented by each of these platforms.

In order to exemplify the data sharing concept between multiple users' communities and platforms, Figure 8.5 presents the case of using the URM [77] framework for querying the GreenLand spatial data repository. This example displays the NDVI result within the URM system, and it is similar to the one presented into the GreenLand platform (Figure 7.7).

The WPS implementation has the role of achieving the interoperability between the GreenLand system and other platforms by exposing the GreenLand workflows in a standardized manner and allowing their remote invocation from external applications. The NDVI workflow (described in Algorithm 6.1) is an example of how the GreenLand platform is able to provide this type of functionality.

Due to the fact that the Grid based execution requires a longer period of time to complete, the WPS service was extended with another operation (called *Monitor*). In order to follow the same monitoring guidelines as in the case of the classic data processing (Figure 8.4), the *Monitor* method produces similar feedback in the GreenLand graphical interface. Figure 8.6 highlights how the NDVI result is displayed to the user as a Web address that can be used in any browser. When the WPS geospatial algorithms are not monitored through the GreenLand platform, the

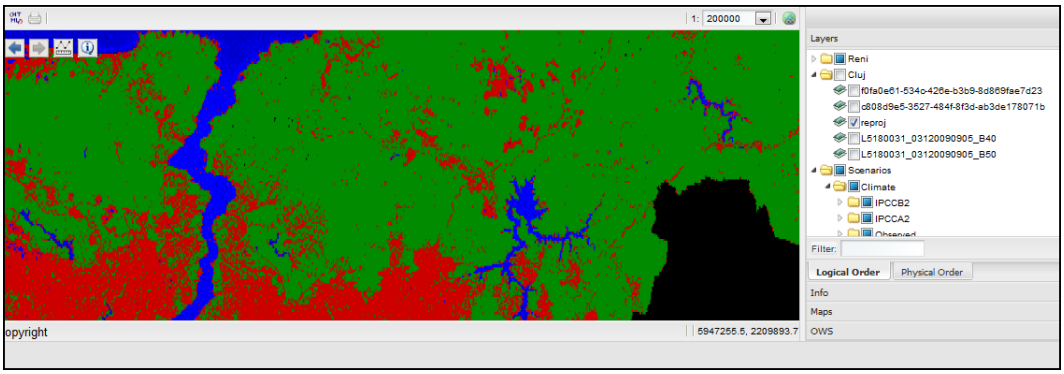


Figura 8.5: The NDVI visualization onto external platforms

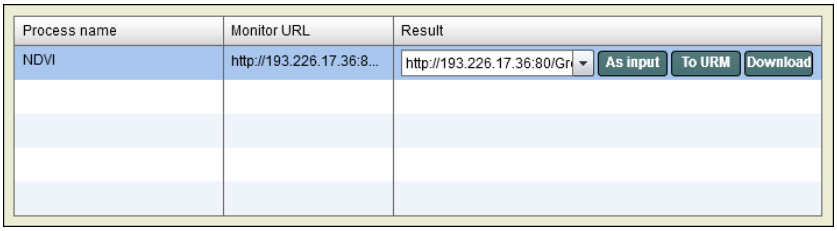


Figura 8.6: Monitor the standardized WPS execution

same Web address is displayed directly into the browser at the end of the execution process;

8.1.5 Interactive visualization of the results

Due to the fact that the GreenLand features are not dedicated only to the domain field specialists (but also to other categories of users that do not have solid background knowledge about the spatial data related domains), the visualization techniques intend to simplify as much as possible the classical pre-visualization of the outputs as static images. In order to achieve this functionality, the WMS and WCS services were implemented within the GreenLand platform, and integrated to the Esri dynamic map product. It is worth mentioning that the visualization process is also possible with spatial data resident on remote repositories.

Two use cases (land cover/land use analysis of the Istanbul geographic region and the Black Sea catchment modeling) were proposed in order to validate the concepts implemented within the GreenLand platform. These scenarios are part of the enviroGRIDS project [80] and were described by domain field specialists with intensive research in the meteorology and hydrology domains.

The validation process is present in every step of this methodology and has the role of verifying if the initial requirements can be solved by the GreenLand solutions. The whole process starts by modeling these use cases through the OperatorEditor and WorkflowEditor tools, and continues with the optimal execution and interactive visualization steps.

8.2 Use case 1 - Land cover/land use analysis of the Istanbul geographic area

The aim of this scenario is to generally describe the algorithms and the data flow for classifying the areas around the Istanbul region, based on Landsat 5 TM data [77]. Once this description is finished, the next step is to implement it into the GreenLand application and to provide automatic and interactive features in order to perform this experiment as easy as possible. Different vegetation indices have been developed in order to classify the land cover areas. The NDVI is one of them that is based on a simple formula (5.4) that it applied on each pixel within the Red and NIR inputs [158].

8.2.1 Study area

The Istanbul city has experienced a rapid land use/land cover change due to the population growth and the powerful industrialization from the last decades. As a result to all these human actions, in the near future the Istanbul will face serious problems regarding the water quality.

This experiment aims to provide a real situation about the current state of the vegetation growth compared to the previous time periods, by using Landsat images [17] with 30 meter spatial resolution and 6 spectral bands. For this scenario, only the 1987 and 2009 spatial data are used from the Warehouse Inventory Search Tool (WIST) repository [159]. The idea of processing data from the past is to progressively measure the vegetation growth and to provide statistics related to this aspect. The flowchart of this use case involves the following steps:

1. The Landsat data retrieval (from the WIST remote repository) that are relevant for the Istanbul scenario. During this experiment the users have manually downloaded these images onto their local machines, and uploaded them by hand into the GreenLand repository. The other two download features were not needed in this case, due to the fact that there are a relatively small number of primary inputs;

2. Once the data transfer completes, the next step is to pre-process these images by extracting relevant information from the original image (Subset algorithm) or by combining multiple bands in order to generate the representation for the entire Istanbul geographic area (Mosaic algorithm).

   It is worth mentioning that the initial Landsat data (from the WIST repository) do not entirely cover this area. Instead it requires two adjacent bands that have to be combined by the Mosaic function;

3. Once the users assure that they have the proper satellite images, it is time to use the NDVI index in order to classify the Istanbul geographic areas. The output result is a black and white image, with pixel values between [-1; 1]. The negative values correspond to the water class, the ones that are near 0 are identified as urban buildings, while the rest of the interval is marked as vegetative areas;

4. For the domain field specialists, the [-1;1] pixel values have a powerful significance, but this use case also addresses other categories of users. In these situations a thematic map (by using the Density slicing algorithm) is the proper manner of displaying the NDVI results. This map is similar to the one presented in Figure 7.5 and assigns a color to each class identified by the NDVI algorithm;

5. The pseudo-coloring process should be repeated multiple times, until the best configuration is achieved, by comparing it to an already known result. Besides the Landsat images, the Istanbul scenario also uses some ground based measurements for the 1987 and 2009 time periods. These data represent the ideal model that should be achieved through automatic NDVI classification process.

   In order to find out how good the current classification is, the user has to be provided with statistical data about the differences between the ideal model and the results obtained through the Landsat satellite images.

8.2.2 Mapping the use case features onto the GreenLand concepts

Once we have the natural language description of the Istanbul case study, the next steps are related to its implementation within the GreenLand platform. This requires concepts identification for each step of the proposed methodology (GreenLand is based on this methodology) and the concepts mapping onto the features provided by the GreenLand system.

Context definition and requirements analysis

By following the previously described phases, the context definition and requirements analysis steps were already presented in the previous paragraphs.

Data retrieval

For this use case, the domain field specialists were using data from the WIST repository. It was not necessary to use the GreenLand FTP or OGC data extraction features, due to the fact that there was a relatively small number of a primary inputs. Instead, these inputs were manually downloaded from the repository as Landsat images and uploaded by hand into the GreenLand storage.

Modeling the spatial data related processes

The modeling phase of the Istanbul use case study has the role of identifying the operators and the workflows that can be used in order to develop the main scenario. The abstract description of each operator plays an important role in implementing the algorithm's functionality. From the previous description, several atomic entities could be identified: atmospheric correction, mosaic, subset, NDVI, density slicing, and accuracy assessment. These operators provide the basic functions for the development of the main Istanbul workflow, while the connections between them are going to be described in another methodological stage.

The atmospheric correction is required in removing potential distortions that might occur in the original Landsat images. It assumes that "dark objects exist within an image and these objects should have values very close to zero (such as water bodies), and that radiance values greater than zero over these areas can be attributed to atmospheric scattering and thereby subtracted from all pixel values in an image" [160]. The atmospheric correction operator (also called *AtmosphericCorrection*) requires two inputs: the Landsat band that needs to be corrected and a numerical value that is used as a reference point for removing the maximal groups whose pixels sum is less then this value.

By default, one Landsat image does not cover the entire Istanbul geographic region, and that is the reason why two such images are needed. The *Mosaic* operator was chosen for solving this issue. It receives two independent satellite images and merges them in order to generate an extended area that contains both inputs. The next GreenLand operator (*Subset*) is used to crop the result, generated by the *Mosaic* algorithm, to fit the geographical coordinates of the Istanbul area.

The NDVI algorithm represents one of the methods for classifying the greenness that cover certain geographic regions. This operator requires two inputs that represent the Red and NIR bands of the original Landsat satellite image, and applies the formula 5.4 for generating black and white images with pixel values between [-1; 1]. Usually, the common range for the vegetation is [0.2; 0.8].

The digital image classification attempts to classify each pixel from the NDVI image, based on the information contained within. The goal of the *DensitySlicing* operator is to find a corresponding class (e.g. water, forest, soil, etc.) for each pixel in the original image and to provide a pseudo-colored map for all these clusters. By default, the GreenLand application tries to create three types of classes (water, vegetation, and urban buildings) by dividing the [-1; 1] interval into [-1; -0.33], [-0.33; 0.33], and [0.33; 1]. By statistic measurements, these values produce the best classification for the Istanbul area, but the users are able to change these settings and to add other groups to the classification process (Figure 6.7).

The *AccuracyAssessment* operator is used to evaluate the performance of the classification process, by comparing the *DensitySlicing* result to the ones obtained by using the ground truth data (that represent an ideal model, due to the fact that these data are collected manually

and have a high measurement accuracy) [161]. The *AccuracyAssessment* operator requires two inputs: the classified image and an archive with three vector files that store the ground truth data.

It is based on a confusion matrix, where each row represents one of the information clusters used by the classification algorithm. Each column displays the corresponding ground truth classes in an identical order, while the diagonal items highlight the number of pixels that were identified correctly. It is based on the K coefficient from formula 8.1:

$$K = \frac{n \cdot \sum_{i=1}^{q} n_{kk} - \sum_{i=1}^{q} n_{k+} \cdot n_{k+i}}{n^2 - \sum_{i=1}^{q} n_{k+} \cdot n_{+k}} \tag{8.1}$$

, where:

- n: the number of pixels from the classified image;
- q: the number of classes;
- n_{kk}: the diagonal item from the confusion matrix;
- n_{k+}: the total number of row pixels for the class k;
- n_{+k}: the total number of column pixels for the class k.

It is worth mentioning that the development of these operators is based on the GRASS library whose functionality is encapsulated within the GreenLand platform. It provides geospatial algorithms for processing various types of raster and vector data that can also be applied for the Istanbul use case.

The implementation of these operators as executable files (by using high level programming languages) is similar to the one presented in *5.2.2.2 Extension of the default operator*. Most of the algorithms are based on the GRASS functionality, and in conclusion they have to extend the *DefaultOperator* class. Without going into further details, the next step in modeling the Istanbul use case is to create a GreenLand internal representation of this scenario by using the WorkflowEditor tool.

As already known, the WorkflowEditor application allows the interactive development of complex scenarios (such as the Istanbul one) through the use of graphs. The visual representation of this study if presented in Figure 8.7, where each of the previous identified operators are connected based on the initial set of requirements. At the end, the graph developed within the WorkflowEditor tool has identical functionalities with the original use case description performed in natural language.

Spatial data processing

Once the Istanbul workflow is available within the GreenLand platform, the users are able to process it in order to obtain the classification results. There are a few primary inputs that need to be specified before runtime:

- 4 Landsat bands: the NDVI algorithm requires the Red and NIR bands that correspond to the items number 3 and 4 from the Landsat satellite images. For this reason the domain field specialists were using the following resources: *L5180031_03119870925_B30.tif*, *L5180031_03119870925_B40.tif* (for the upper part of the Istanbul area) and *L5180032_03219870925_B30.tif*, *L5180032_03219870925_B40.tif* (for the lower region). As can be

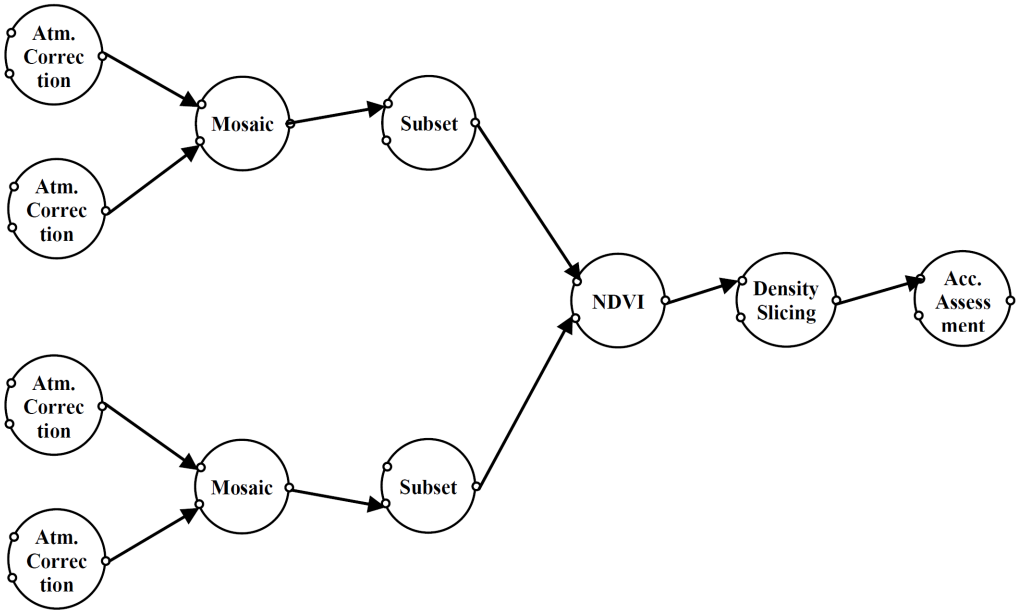


Figura 8.7: Istanbul use case development through the WorkflowEditor tool

seen these four inputs correspond to the 1987 year, but the same approach can be apply to the 2009 processing;

- A text based representation (e.g. metadata files) of the Istanbul geographic coordinates;
- 2 *DensitySlicing* based intervals: the first one specifies the range of pixel values and the number of classes that the user wants to obtain, while the second interval must have the same number of sub-domains that assign a certain color for each such cluster;
- A set of three vector files that contain the ideal classification model. Based on this information, the *AccuracyAssessment* operator generates the statistical results. The gProcess platform is responsible for partitioning the Istanbul workflow into tasks and for scheduling the worker nodes. Due to the fact that the specified operators do not interleave in terms of functionality, the gProcees tool creates a task for each operator. This means that in the end we'll have 6 tasks, executed onto 6 different Grid nodes.

The connection between these tasks is then achieved based on the workflow inner structure, described through the use of edges (Figure 8.7). As can be seen there are several inputs that are not connected to other visual elements. These are called primary inputs of the workflow that have the be specified by the user before runtime, and represent the previously described resources: the inputs for the AtmosphericCorrection operators, the geographic coordinate of the Subset algorithm, the DensitySlicing value and color ranges, and the ground based measurements for the AccuracyAssessment item.

Each operator generates a specific result that is used as input in the overall communication network. Figure 8.8 highlights the outputs obtained at each execution phase, including the *NDVI*, *DesnsitySlicing*, and the *AccuracyAssessment* ones that also represent the final results of the workflow.

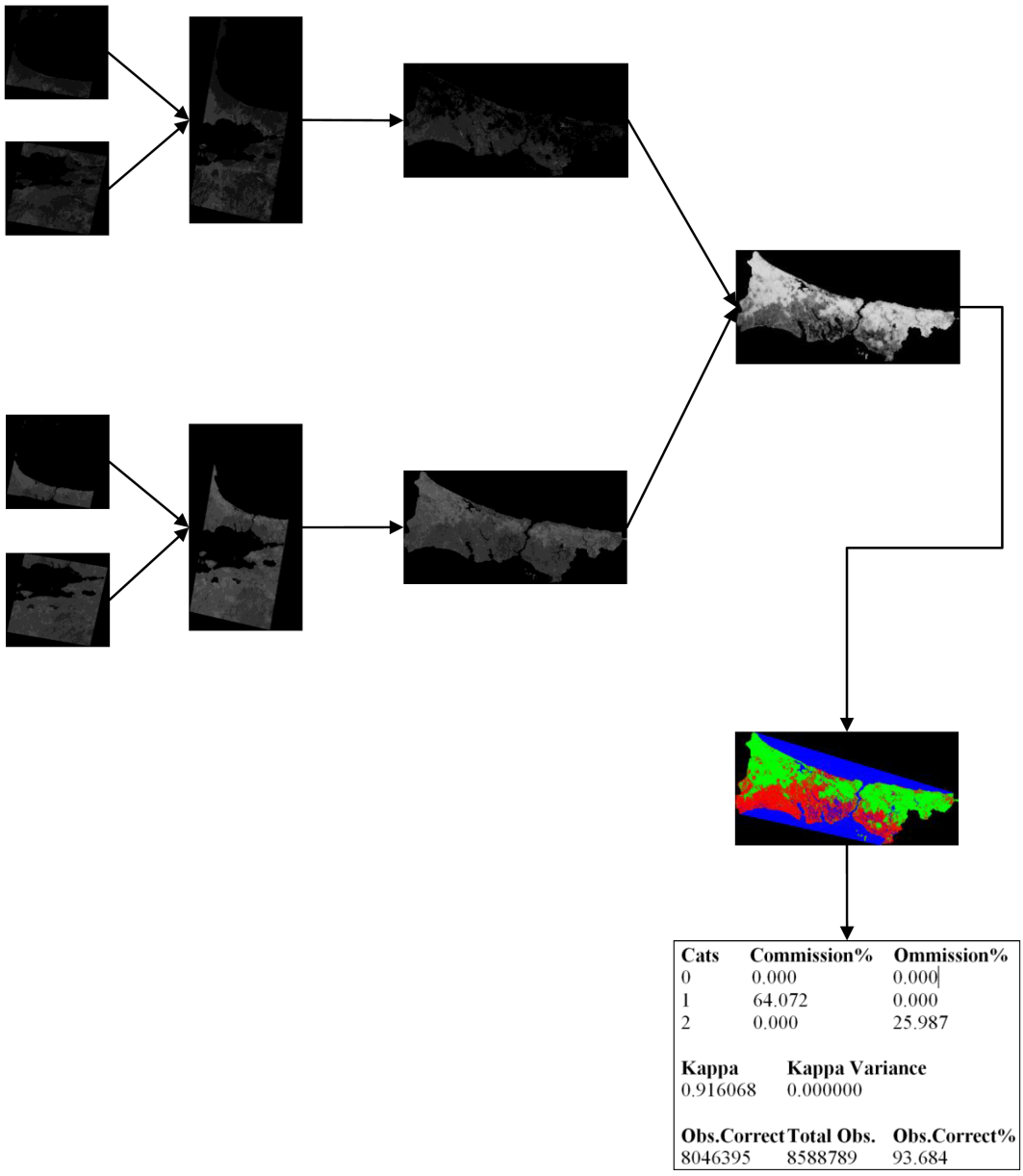


Figura 8.8: Results exemplification for the Istanbul use case study

Spatial data visualization

The Istanbul workflow execution generates three outputs: the *NDVI* result, the *DensitySlicing* thematic map, and the statistical data of the *AccuracyAssessment*. For the first two, the user has the possibility of performing an analysis based on the OGC standard techniques, implemented at the GreenLand level. A suggestive exemplification is presented in Figure 7.5 that highlights the pseudo-color image display over the interactive map. Exporting the GeoTIFF images to external platforms, and apply the analysis process from there is another possible

feature that will be described for the next use case.

Not all results can be visualized in a standardized manner. For example the *AccuracyAssessment* generates a text-based file that contains the statistical report for the current classification process, compared to an ideal model. For this types of results, the online text visualization or its download onto the user's local machine are the only two possible solutions.

The Istanbul use case validates the methodology for developing applications that process and visualize spatial data in various formats. It is not intended to measure the execution performance among different processing infrastructures due to the fact that is not that computing intensive. Based on the fact that the methodological steps of this use case fit properly the general ones (used for developing the GreenLand application) we can conclude that the proposed methodology is correct and allows users to interactively describe complex scenarios (experiments) in a simple and easy to understand manner through the use of the GreenLand platform and all its related applications.

8.3 Use case 2 - Hydrologic modeling of the Black Sea catchment

Because the methodology was validated through the Istanbul scenario, this second use case intends to describe in more detail certain GreenLand features, such as: optimal data retrieval, near-real time processing of large volume of spatial data, and data sharing among different scientific communities. It is called the Black Sea catchment use case [77] due to the fact that it aims to provide a study regarding the water quality and quantity for the geographic area that contains all the rivers that contribute to the Black Sea watershed.

8.3.1 Study area

This experiment uses hydrological data from the NTSG and USGS repositories that provide periodic updates, starting from 2000 until the present time. At this stage of development only two MODIS products (MOD15 and MOD16) are required that are generally described in the EOS data products Handbook [162]: "MOD16 consists of surface resistance and evapotranspiration which are essential parameters to global modeling of climate, water balance, and trace gases. Practical applications of this product are for monitoring wildfire danger and crop/range drought.

MOD15 Leaf Area Index (LAI) and Fraction of Photosynthetically Active Radiation (FPAR) absorbed by vegetation are biophysical variables that describe canopy structure and are essential in calculating terrestrial energy, carbon, water-cycle processes, and biogeochemistry of vegetation. LAI defines the one-sided leaf area per unit ground area (value between 0 and 8), when FPAR measures the proportion of available radiation (400 to 700 nm) that a canopy can absorbs (value between 0 and 1)".

An internal organization of the NTSG repository was already described in *2.4.3 Global Spatial Data Infrastructure - GSDI*. This *Mosaic12* experiment requires 8-days temporal resolution MODIS images starting from 2000 and up to 2010, and a geographic extent of 12 MODIS tiles that covers the entire Black Sea catchment area (Figure 8.9). Each of the two MODIS products has multiple bands (as any satellite image) but for this case study only the Evapotranspiration and Fraction of Photosynthetically Active Radiation are required.

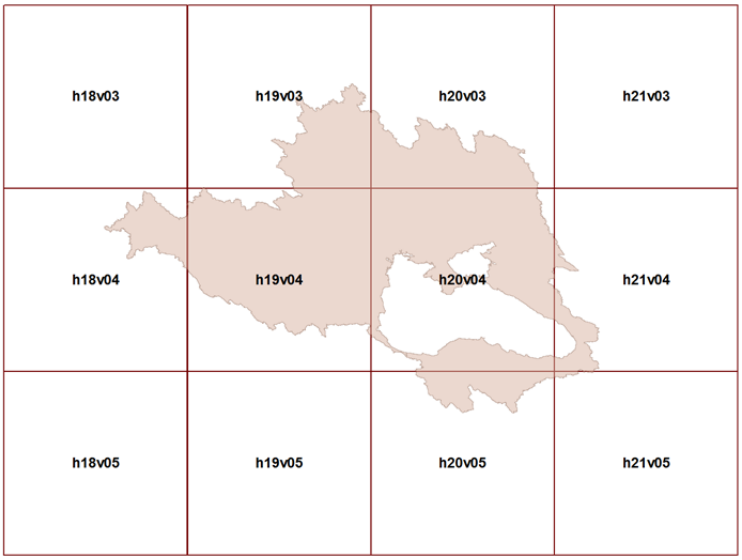


Figura 8.9: Black Sea catchment area partition into adjacent tiles

8.3.2 The Black Sea catchment workflow

The GreenLand implementation of this use case is called *Black Sea Mosaic* (or *Mosaic12*) due to the fact that it requires the merge of 12 adjacent tiles in order to cover the entire watershed. The difficulty is to map the use case requirements to the GreenLand concepts and to provide an optimal execution for a large volume of data.

The goal of this experiment is to provide ten years results (between 2000 and 2010) for the MOD15 and MOD16 products. Due to the fact that it is a time consuming algorithm, the entire use case implementation was divided into single processing years that contain 45 MOD15 and 45 MOD16 datasets. In conclusion, we have to process 90 MODIS products and download more than 50GB of data (approximately 10000 files) for a single year.

The manual specification of the input datasets is almost impossible. Instead, the GreenLand platform provides an automatic FTP data retrieval mechanism (Figure 6.8) that queries the entire remote repository's content and extracts only the relevant information. From the graphical interface, the user is required only to specify the processing period, while the system automatically starts the data transfer in the background. It is worth mentioning that the data retrieval process takes place at runtime and assigns each input to the corresponding Grid task.

The *Mosaic12* workflow takes 12 inputs of MODIS type and merges them together in order to generate a single satellite image that contains them all. It is based on the classical *Mosaic* workflow that handles only 2 layers. Progressively, the inner levels of the workflow are linked to the outputs generated earlier, and so on until the final result is computed. Figure 8.10 highlights this process, where the primary inputs correspond to the tiles h18v03, h19v03, ..., h21v05 (taken in this order) from Figure 8.9. It is worth mentioning that the *Mosaic* output from level 3 is connected directly to the input of the level 5 operator.

In order to speed up the execution, the one year processing was partitioned into 18 groups of 5 *Mosaic12* instances. Due to the fact that this is a complex use case scenario, the Grid infrastructure was taken into account, where each group was executed on a different worker node. The 5 items within a group condition is also the result of the fact that the NTSG repository limits to 5 the number of simultaneous downloads. The total execution time (required to process

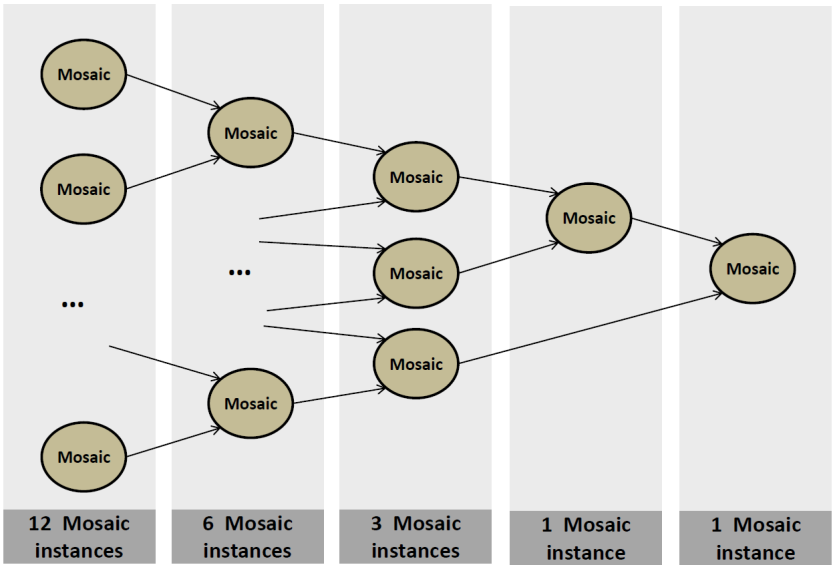


Figura 8.10: Inner functionality of the Mosaic12 workflow

one year data) was 2 hours and 5 minutes, based on the parallel and distributed capabilities provided by the Grid infrastructure.

Another use case requirement was to improve the results data sharing among different communities of users. In order to achieve this purpose, the URM repository [77] was chosen to provide the support for storing all these results. The GreenLand manages this aspect through the GeoServer [53] tool that is able to expose OGC compliant data. On the other hand, the URM platform is able to interpret this format and to correctly process it. Figure 8.11 highlights the *Mosaic12* result visualization in the URM framework.

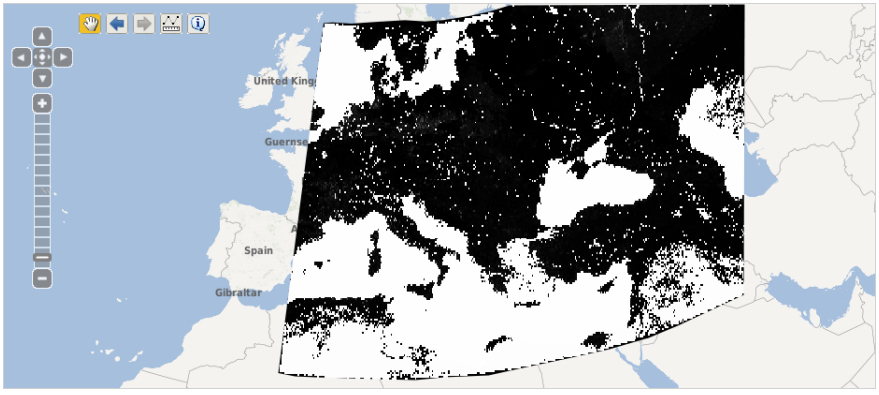


Figura 8.11: The Black Sea Mosaic workflow's result visualization on external platforms

8.4 Conclusions

Each new methodology requires a validation stage where all the theoretical concepts are verified with real or simulated data, in order to find out if their implementation is correctly performed. Two scenarios were used for the GreenLand validation: the first one aims to create a complex workflow that allows the analysis of the land cover/land use of the Istanbul geographic area, while the second one intends to model the Black Sea catchment in order to provide useful hydrological data for further processing.

The first experiment validates all the methodological steps proposed within this thesis by progressively creating the Istanbul workflow. It starts with a short description of the use case, followed by the identification and implementation of the operators, and ends with the workflow processing and results visualization and analysis. Even though it does not provide information about the performance of the execution, the Istanbul use case proved that the GreenLand implementation (based on the proposed methodology) was correct and provides useful features in modeling, executing, and visualizing the spatial data related processes.

The modeling of the Black Sea catchment aims to validate GreenLand features that were not taken into account in the first use case: automatic data retrieval from remote repositories, the processing of a large volume of data in a reasonable amount of time, assure data sharing among different scientific communities, improve the interaction techniques in order to allow their usage by different categories of users, etc. Details about the inner functionality of the *Mosaic12* workflow were given, together with specification about optimizing the overall Grid execution by creating groups of tasks and processing them over different computing nodes.

The GreenLand features that were not covered by the development of the two use cases are highlighted in order to provide an overall description about the functionalities that can be achieved in terms of spatial data modeling, processing, analysis, and visualization.

The following paragraphs describe the most important contributions that were achieved throughout this chapter:

- The validation of the theoretical concepts implemented within the GreenLand platform, by developing the two use cases;
- The validation of the interaction techniques that address non-expert users that manage complex spatial data related processes;
- The definition of experimental tests for improving the processing of large volume of spatial data. The obtained results demonstrate that the proposed solutions were good and allow the achievement of such goals;
- The development of the GreenLand platform that allows the flexible description of various use cases from different activity domains, such as: land cover/land use, hydrology, meteorology, etc.

Chapter 9

Conclusions

The current thesis outlines the methodological steps that are required in order to develop interactive applications that provide optimal solutions for modeling, processing, visualizing, and analyzing complex use case scenarios from the Earth Science related domains. A number of 7 methodological steps were identified that start by defining the problem that needs to be solved and specifying the spatial data sets that are required for implementing the real and simulated use case scenarios. One important aspect is the modeling of the algorithms (methods, formulas) as basic operators and complex workflows, and the establishment of a general data flow that corresponds to the requirements of the original use case. This step has a major influence in the processing performance, due to the fact that the GreenLand application partitions the entire workflow into groups of tasks that intend to improve the execution time.

The GreenLand representation of the natural phenomena and experiments is possible through the use of the concepts inspired from the mathematical graphs theory. Each node contains a basic functionality (called operator) or a complex structure (called workflow) that is connected to other graph's elements by the use of uni-directional edges. This means that such a node may contain other workflows inside, who in their turn are able to address other complex elements, and so on without imposing a limit to this recurrent process.

This flexible description of the processes allows the GreenLand application to be configured for other domains (besides the Earth Science related ones) as long as the issues that need to be solved can be defined as graph based interconnected modules.

In order to facilitate the interactive description of the GreenLand workflows, a specialized tool was implemented (called WorkflowEditor) that allows the users to visually create the graph whose structure corresponds to the original flow of the use case scenario. Syntax errors prevention, automatic adjustments of placing and interconnecting the items, collaborative development are only a few of the features provided by this application that have the effect of reducing the time required to develop the complex experiments.

The processing of a large volume of spatial data is another issue that is addressed within this thesis. The proposed solution is based on the usage of the parallel and distributed capabilities of the Grid infrastructure in order to achieve an improved execution time. The gProcess platform is part of the GreenLand system, and has the role of partitioning the entire experiment (workflow) into tasks and executes them over different physical machines. The Black Sea catchment modeling is a good example of using this approach, meaning that it provides a significant execution speed-up (compared with standalone execution) when processing more than 500 GB of spatial data.

The WPS service has a major impact in obtaining the interoperability between different

platforms, in terms of accessing and executing various remote geospatial algorithms without the need to develop them from scratch or to integrate them within specific applications. In order to align to the latest Earth Science tendencies, the GreenLand platform implements the WPS standard and provides an alternative to the traditional execution of the workflows.

The WPS standalone execution was already implemented and tested in many other projects and is well defined in terms of development rules and guidelines. On the other hand the WPS over the Grid infrastructure (or distributed platforms in general) was not studied so intensively. The GreenLand solution extends the WPS service with the *Monitor* operation in order to be able to adapt it to the requirements of the Grid environment.

There are many remote repositories that allow data access based on proprietary mechanisms. This approach complicates the data sharing and platforms interoperability processes, due to the fact that for each storage new mechanisms have to be learnt in order to access the needed information. The OGC standard represents one of the possible solutions that can be used in order to overcome these issues that allow the accessing, retrieving, processing, and visualizing the spatial data. The WMS represents the OGC product that can be used in order to optimize the spatial data sharing and visualization processes, regardless of the original location of data. The GreenLand implementation integrates this service with the features provided by the Esri's interactive map, having the effect of providing more dynamicity in visualizing and analyzing the output results.

An important characteristic for the Earth Science domain field specialists is the possibility to analyze fractions of the original result, by using powerful mechanisms that allows the extraction of relevant data. This is accomplished by the WCS service that queries through the spatial data repository for information that matches the filters specified by the users at the GreenLand graphical interface. The response is then displayed in a similar manner to the WMS one.

The GreenLand implementation of the theoretical concepts proposed within the thesis is validated based on two real use case studies: land use/land cover of the Istanbul geographic area and the hydrologic modeling of the Black Sea catchment. The first one requires the creation of a complex workflow that has the effect of classifying the greenness areas from the Istanbul geographic region, in order to predict the vegetation growth. This development process involved all the steps within the proposed methodology and was a good example of validating it. The second example is more computing intensive and requires runtime data retrieval from remote repositories. The Grid based execution proved to be more efficient that the standalone one, and completed the entire processing is approximately 2 hours.

The validation of the implemented solutions is recommended to be performed regularly (on each stage of development) and also at the end of the implementation process, with real end-users that will utilize the platform in various experiments. These validations intend to determine: potential errors in the system related architecture, execution issues when dealing with a large volume of data that need an optimal organization of tasks, problems in the data retrieval process, the proper results presentation techniques that address a particular category of users, etc.

9.1 Contributions of the thesis

The main contribution of this thesis is the creation of a new methodology for developing interactive applications that model, process, visualize, and analyze a large volume of spatial data. Other contributions that have a direct impact in achieving this feature are listed in the following paragraphs [73]:

1. The definition of the basic operator and workflow concepts that can be used for modeling

the scenarios from various activity domains. This contribution (in combination with the GreenLand services) can also be used in order to solve the issue of extending the platform to other disciplines, besides the Earth Science related ones [78];

2. The abstract description of the basic operators and complex workflows [78];
3. The identification of a predefined set of types, rules, and constraints that can be used for specifying the inputs and outputs of the operators and workflows [79];
4. The development of two applications (OperatorEditor and WorkowEditor) that implement these theoretical concepts and provide interactive techniques for developing the complex use case scenarios [78];
5. The identification of the phases required for the spatial data execution process. Each step contains the theoretical background and the practical solutions implemented within the GreenLand platform [73];
6. The development of an interactive platform (GreenLand) that is able model, process, visualize, and analyze various types of spatial data. It is the direct result of the development process performed through the methodology proposed within this thesis [79];
7. The implementation of specific modules that are able to process the spatial data in near-real time, by accessing and retrieving the corresponding information from remote repositories, based on the FTP protocol and OGC standard services. This is useful for scenarios that predict certain phenomena and require up to date information [77];
8. The possibility of achieving platforms interoperability by allowing the execution of the workflows as WPS services. This means that the GreenLand geospatial algorithms can be accessed by other external platforms that implement this standard. The reverse process is also valid, meaning that the GreenLand is able to remotely execute functions developed by other applications [148];
9. The extension of the WPS service, with the *Monitor* operation that can be used in Grid based processing of data [148];
10. The implementation of an interactive module that is able to monitor the workflows execution over the Grid and standalone infrastructures. This tool allows the users to interfere during the runtime phase, and stop the processing if necessary [104];
11. The definition of an automatic mechanism that translates the XML based response of the OGC services into visual components that facilitate their utilization by various types of users [157];
12. The development of a communication architecture (based on OGC services) between the GreenLand platform and remote spatial data repositories. This module is able to improve the data sharing and applications interoperability that provide services for the same activity domains [155];
13. An identification of the most important visualization techniques that can be performed for various types of users, regardless of their background domain knowledge [155];
14. The definition of experimental tests for improving the processing of large volume of spatial data. The obtained results demonstrate that the proposed solutions were good and allow the achievement of such goals [158].

9.2 Future work

The theoretical concepts and the features described within this thesis can be extended based on the following furture research directions:

- the extension of the GreenLand functionalities over the Cloud and clusters infrastructures. The theoretical concepts are flexible enough to support this feature, but it requires an update of the gProcess platform;
- the integration of the conditional (e.g. if, switch) and looping (e.g. for, while) structures in the internal language that performs the modeling of the use case scenarios;
- the concepts extension to other activity domains and their validation based on real or simulated use case scenarios.

9.3 List of publications

9.3.1 Journal articles

1. **D. Mihon**, V. Colceriu, V. Bacu, and D. Gorgan, "Tehnici de vizualizare a imaginilor satelitare prin servicii geospatiale," Revista Romana de Interactiune Om-Calculator, vol. 6, no. 1, pp. 1-22, 2013.
2. **D. Mihon**, V. Colceriu, V. Bacu, K. Allenbach, D. Rodila, G. Giuliani, and D. Gorgan, "OGC compliant services for remote sensing processing over the Grid infrastructure," International Journal of Advanced Computer Science and Applications(IJACSA), in press.
3. **D. Mihon**, V. Colceriu, V. Bacu, and D. Gorgan, "Grid based processing of satellite images in GreenLand platform," International Journal of Advanced Computer Science and Applications(IJACSA), in press.
4. V. Bacu, **D. Mihon**, T. Stefanut, D. Rodila, K. Abbaspour, E. Rouholahnejad, and D. Gorgan, "Calibration of SWAT hydrological models in a distributed environment using the gSWAT application," International Journal of Advanced Computer Science and Applications(IJACSA), in press.
5. V. Colceriu, **D. Mihon**, A. Minculescu, V. Bacu, D. Rodila, and D. Gorgan, "Workflow based description and distributed processing of satellite images," International Journal of Advanced Computer Science and Applications(IJACSA), in press.
6. F. B. Balcik, **D. Mihon**, V. Colceriu, K. Allenbach, C. Goksel, A. O. Dogru, M. Gvilava, G. Giuliani, and D. Gorgan, "Remotely sensed data processing on Grids by using GreenLand Web-based platform," International Journal of Advanced Computer Science and Applications(IJACSA), in press.
7. **D. Mihon**, A. Minculescu, V. Colceriu, and D. Gorgan, "Descrierea diagramatica a prelucrarii distribuite a datelor spatiale," Revista Romana de Interactiune Om-Calculator, vol. 5, no. 3, pp. 59-80, 2012.
8. D. Gorgan, V. Bacu, **D. Mihon**, D. Rodila, K. Abbaspour, and E. Rouholahnejad, "Grid based calibration of SWAT hydrological models," Natural Hazards and Earth System Science, vol. 12, no. 7, pp. 2411-2423, 2012.

9. D. Gorgan, V. Bacu, T. Stefanut, D. Rodila, and **D. Mihon**, "Earth observation application development based on the Grid oriented ESIP satellite image processing platform," Comput. Stand. Interfaces, vol. 34, no. 6, pp. 541-548, 2012.

10. D. Gorgan, V. Bacu, **D. Mihon**, T. Stefanut, D. Rodila, P. Cau, K. Abbaspour, G. Giuliani, N. Ray, and A. Lehmann, "Software platform interoperability throughout enviroGRIDS portal," International Journal of Selected Topics in Applied Earth Observations and Remote Sensing - JSTARS, vol. 5, no. 6, pp. 1617-1627, 2012.

11. **D. Mihon**, V. Bacu, D. Gorgan, R. Meszaros, and G. Gelybo, "Practical considerations on the GreenView application development and execution over SEE-GRID," Earth Science Informatics, vol. 3, no. 4, pp. 247-258, 2010.

12. D. Gorgan, T. Stefanut, V. Bacu, **D. Mihon**, and D. Rodila, "Grid based environment application development methodology," Springer Journal Lecture Notes in Computer Science, vol. 5910, pp. 499-506, 2010.

9.3.2 Conference articles

1. **D. Mihon**, V. Colceriu, V. Bacu, T. Stefanut, and D. Gorgan, "Optimizarea interfetelor grafice pentru aplicatiile de prelucrare a datelor spatiale," Revista Romana de Interactiune Om-Calculator, vol. 6, no. Special issue, pp. 79-85, 2013.

2. V. Colceriu, **D. Mihon**, and D. Gorgan, "Optimizing the satellite images execution process through a decision mediator approach," Proceedings of the 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing, pp. 343-346, IEEE Computer Society, 2013.

3. **D. Mihon**, V. Colceriu, A. Minculescu, V. Bacu, and D. Gorgan, "Descrierea diagramatica a prelucrarii imaginilor satelitare in aplicatia GreenLand," Revista Romana de Interactiune Om-Calculator, vol. 5, no. 2, pp. 129-134, 2012.

4. **D. Mihon**, V. Bacu, T. Stefanut, and D. Gorgan, "Tehnici de interactiune pentru executia modelelor hidrologice pe Grid - aplicatia gSWAT," Revista Romana de Interactiune Om-Calculator, vol. 4, no. Special issue, pp. 103-106, 2011.

5. V. Bacu, **D. Mihon**, T. Stefanut, D. Rodila, D. Gorgan, P. Cau, and S. Manca, "Grid based services and tools for hydrological model processing and visualization," in Proceedings of the 2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC '11, pp. 291-298, IEEE Computer Society, 2011.

6. D. Gorgan, K. Abbaspour, P. Cau, V. Bacu, **D. Mihon**, G. Giuliani, N. Ray, and A. Lehmann, "Grid based data processing tools and applications for Black Sea catchment basin.," in Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference, vol. 1, pp. 223-228, 2011.

7. V. Bacu, **D. Mihon**, D. Rodila, T. Stefanut, and D. Gorgan, "Grid based architectural components for SWAT model calibration," HPCS 2011 - International Conference on High Performance Computing and Simulation, 4-8 July, Istanbul, Turkey, ISBN 978-1-61284-381-0, pp. 193-199, IEEE, 2011.

8. **D. Mihon**, V. Bacu, T. Stefanut, and D. Gorgan, "Tehnici de interactiune utilizator in aplicatiile de prelucrare a imaginilor satelitare - exemplificare pe baza aplicatiilor GreenView si GreenLand," Revista Romana de Interactiune Om-Calculator, vol. 3, no. Special issue, pp. 37-44, 2010.

9. **D. Mihon**, V. Bacu, T. Stefanut, and D. Gorgan, "Considerations on the Grid oriented environmental application development framework," Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing, ICCP '10, pp. 419-426, IEEE Computer Society, 2010.

10. V. Bacu, D. Rodila, **D. Mihon**, T. Stefanut, and D. Gorgan, "Error prevention and recovery mechanisms in the ESIP platform," Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing, ICCP '10, pages 411–417, IEEE Computer Society, 2010.

11. **D. Mihon**, V. Bacu, R. Meszaros, G. Gelybo, and D. Gorgan, "Satellite Image Interpolation and Analysis through GreenView Application," CISIS series, Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems, pages 514-519, IEEE Computer Society, 2010. ISBN: 978-0-7695-3967-6.

12. **D. Mihon**, V. Bacu, T. Stefanut, and D. Gorgan, "Grid based environment application development - GreenView application," Proceedings of the 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing, ICCP '09, pp. 275-282, IEEE Computer Society, 2009.

9.3.3 Book chapters

1. **D. Mihon**, V. Bacu, D. Rodila, T. Stefanut, K. Abbaspour, E. Rouholahnejad, and D. Gorgan, Grid Based Hydrologic Model Calibration and Execution. Advanced in Intelligent Control Systems and Computer Science, Springer-Verlag, 2013. ISBN: 978-3-642-32547-2.

9.3.4 Workshops and trainings

1. **D. Mihon**, V. Colceriu, V. Bacu, T. Stefanut, D. Rodila, and D. Gorgan, "The enviroGRIDS Tools - GreenLand platform", Novi Sad GEO Workshop, Novi Sad, Serbia, 19 - 21 September 2013.

2. **D. Mihon**, V. Colceriu, V. Bacu, T. Stefanut, D. Rodila, and D. Gorgan, "Satellite image processing through the GreenLand platform", EnviroGRIDS Final meeting, Batumi, Georgia, 30 October - 1 November 2012.

3. **D. Mihon**, V. Colceriu, V. Bacu, T. Stefanut, D. Rodila, and D. Gorgan, "GreenLand application supporting satellite image processing", EnviroGRIDS General Assembly Meeting Workshops, Sofia, Bulgaria, 18-20 April 2012.

4. **D. Mihon**, and D. Gorgan, "Spatial Data Processing and Visualization", Computer Science Student Conference, Cluj-Napoca, 2012.

5. **D. Mihon**, V. Colceriu, F. B. Balcik, K. Allenbach, M. Gvilava, and D. Gorgan , "Spatial Data Exploring by Satellite Image Distributed Processing", Geophysical Research Abstracts, Vol. 14, EGU2012-13278, EGU2012 - European Geosciences Union General Assembly, Vienna, April, 2012.

6. V. Bacu, **D. Mihon**, T. Stefanut, D. Rodila, P. Cau, S. Manca, C. Soru, and D. Gorgan, "Hydrological Scenario Using Tools and Applications Available in enviroGRIDS Portal", Geophysical Research Abstracts, Vol. 14, EGU2012 - European Geosciences Union General Assembly, Viena, April, 2012.

7. **D. Mihon**, D. Gorgan, V. Bacu, and D. Rodila, "SWAT Model Calibration and Execution on GRID Infrastructure by gSWAT Application", Sustainable Watershed Management (SuWaMa), September, Istanbul, 2011.

8. **D. Mihon**, T. Stefanut, V. Bacu, D. Rodila, and D. Gorgan, "Grid Based Environment Oriented Tools and Applications", Geophysical Research Abstracts, Vol. 13, EGU2011 - European Geosciences Union General Assembly, Viena, April, 2011.

9. D. Gorgan, V. Bacu, S. Manca, G. Giuliani, D. Rodila, T. Stefanut, **D. Mihon**, K. Abbaspour, E. Rouholahnejad, A. van Griensven, L. Kokoszkiewicz, and P. Cau, "Tools and applications in enviroGRIDS project for spatial data processing of Black Sea catchment basin." EGI User Forum, Vilnius, 11-15 April, 2011.

10. **D. Mihon**, V. Bacu, D. Gorgan, R. Meszaros, G. Gelybo, and T. Stefanut, "GreenView and GreenLand applications development on SEE-GRID infrastructure", Geophysical Research Abstracts, Vol. 12, EGU2010 - European Geosciences Union General Assembly, Vienna, Austria, EGU2010-14455, eISSN: 1607-7962, 2-7 May, 2010.

11. D. Gorgan, V. Bacu, T. Stefanut, D. Rodila, and **D. Mihon**, "Grid based satellite image processing platform for Earth observation application development," Workshop on Intelligent Data Acquisition and Advanced Computing Systems, pp. 247-252, 2009.

12. D. Gorgan, **D. Mihon**, and V. Bacu, "GreenView - Refinement of surface and vegetation parameters in SEE region based on satellite images", SEE-GRID-SCI PSC-05, Dubrovnik, September, 2009.

Bibliografie

[1] S. Steiniger and A. Hunter, "Free and open source GIS software for building a spatial data infrastructure," *Geospatial Free and Open Source Software in the 21st Century*, pp. 247–261, 2012. 7

[2] L. Anselin, "Spatial data analysis with GIS: An introduction to application in the social sciences," *Technical report, National Center for Geographic Information and Analysis, University of California*, 1992. 7

[3] J. R. Jensen, *Remote Sensing of the Environment: An Earth Resource Perspective*. Prentice Hall series in geographic information science, Prentice Hall, 2nd ed., 2006. 8

[4] T. M. Lillesand, *Remote Sensing and Image Interpretation*. John Wiley & Sons, 5th ed., 2006. ISBN: 0470088273. 8, 10, 12

[5] I. Thomas and M. Elliot, *Environmental Impact Assessment in Australia: Theory and practice*. The Federation Press, 4th ed., 2008. ISBN: 1862875383. 8

[6] R. Hirji and R. Davis, *Environmental Flows in Water Resources Policies, Plans, and Projects : Findings and Recommendations*. Water Resources Management, Planning, and Economics series, The World Bank, 2009. 8

[7] M. Varming, C. Seeberg-Elverfeldt, and M. Tapio-Biström, *Agriculture, Forestry and Other Land Use Mitigation Project Database: An Assessment of the Current Status of Land-based Sectors in the Carbon Markets*. Mitigation of climate change in agriculture series, Food and Agriculture Organization of the United Nations, 2010. 8

[8] H. Rua, "Geographic information systems in archeological analysis: a predictive model in the detaection of rural roman villae," *Journal of Archeological Science*, vol. 36, pp. 224–235, 2009. 9

[9] L. Loubert, *GIS for Urban and Regional Planning. Mapping Urban Inequalities with GIS*. ESRI GIS Best Practices series, ESRI, 2011. 9

[10] T. Sutton, O. Dassau, and M. Sutton, *A Gentle Introduction to GIS*. Province of the Eastern Cape, Office of the Premier, 2009. 9, 17, 18, 36

[11] K. Chang, *Introduction to Geographic Information Systems*. McGraw-Hill Higher Education, 3rd ed., 2006. ISBN: 9780073312798. 9

[12] M. DeMers, *Fundamentals of Geographic Information Systems*. John Wiley and Sons, 2000. ISBN: 9780471314233. 10

[13] J. B. Campbell and R. H. Wynne, *Introduction to remote sensing.* Guilford Press New York, 5th ed., 2011. ISBN: 1572306408. 10

[14] W. D. Philpot and W. R. Philipson, *Remote Sensing Fundamentals.* Cornell University, 2009. 10

[15] R. B. Smith, *Introduction to Remote Sensing of Environment (RSE).* Image Rochester, 2012. 11

[16] G. A. Shaw and H.-h. K. Burke, "Spectral imaging for remote sensing," *Lincoln Laboratory Journal*, vol. 14, pp. 3–28, 2003. 11

[17] Landsat remote sensing platform. `http://landsat.gsfc.nasa.gov/`. 13, 157

[18] M. R. N. Ritter, *GeoTIFF Format Specification.* GeoTIFF Revision 1.0, GeoTIFF Revision, 2000. 13, 107

[19] M. Reeves, M. Zhao, and S. Running, "Applying improved estimates of modis productivity of characterize grassland vegetation dynamics," *Rangeland Ecology and Management*, vol. 59, pp. 1–10, 2006. 14

[20] R. McGrath, "Xml and scientific file formats," *National Center for Supercomputing Applications, University of Illinois*, pp. 1–25, 2003. 14

[21] ASTER remote sensing platform. `http://www.satimagingcorp.com/satellitesensors/aster.html`. 15

[22] SPOT platform. `http://www.satimagingcorp.com/satellite-sensors/SPOT-5ResolutionSpectralModes.pdf`. 15

[23] S. G. Ungar, J. S. Pearlman, J. A. Mendenhall, and D. Reuter, "Overview of the earth observing one (eo-1) mission," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, pp. 1149–1159, 2003. 15

[24] L. Wang, W. P. Sousa, P. Gong, and G. S. Biging, "Comparison of IKONOS and QuickBird images for mapping mangrove species on the caribbean coast of panama," *Remote Sensing of Environment*, vol. 91, pp. 432–440, 2004. 15

[25] IKONOS Imagery Products Guide. `http://www.glcf.umd.edu/library/guide/IKONOS_Product_Guide_jan06.pdf`. 15

[26] G. Ottavianelli, *Synthetic Aperture Radar Remote Sensing for Landfill Monitoring.* Ph.D.Thesis, Cranfield University, 2007. 16

[27] M. Sharma, G. B. Paige, and S. N. Miller, "Dem development from ground-based lidar data: A method to remove non-surface objects," *Remote Sensing*, vol. 2, no. 11, pp. 2629–2642, 2010. 16

[28] T. G. Wade, J. D. Wickham, M. S. Nash, A. C. Neale, K. H. Riitters, and B. K. Jones, "A comparison of vector and raster GIS methods for calculating landscape metrics used in environmental assessments," *Photogrammetric Engineering and Remote Sensing*, vol. 69, no. 12, pp. 1399–1405, 2003. 17

[29] A. Butler, *Information Systems Security Requirements for Federal GIS Initiatives.* ESRI Federal User Conference, 2006. 18

[30] O. Ostensen, "Global geomatics standards supporting sustainable geospatial data infrastructures," *GSDI 5 Conference Proceedings*, pp. 1–6, 2001. 18

[31] G. Almirall, M. Bergada, P. Ros, and M. Craglia, *The Socio-Economic Impact of the Spatial Data Infrastructure of Catalonia.* JRC Scientific and Technical Reports, 2007. 18

[32] B. Battrick, *Global Earth Observation System of Systems GEOSS: 10-year Implementation Plan Reference Document.* ESA SP series, ESA Publication Division, 2005. 19

[33] The European Parliament and Council, "Directive 2007/2/EC of the european parliament and of the council of 14 march 2007 establishing an infrastructure for spatial information in the european community (INSPIRE)," *Official Journal of the European Union*, vol. 50, no. L 108, 2007. 19

[34] The European Parliament and Council, *INSPIRE Generic Conceptual Model.* 2008. 19

[35] A. Rajabifard and I. P. Williamson, "SDI development and capacity building," *Paper presented at the GSDI-7*, 2004. 20

[36] "OGC Web Services Common Specification," tech. rep., OGC, 2007. 22, 35, 46, 55, 121, 139, 140

[37] D. Wesloh and J. Sturley, "Implementing ISO data quality standards using ESRI's GIS data reviewer," *National Geospatial-Intelligence Agency*, 2004. 22, 25, 139

[38] "Spatial Data Transfer Standard (SDTS). Part 7: CADD Profile," tech. rep., FGDC Facilities Working Group, 2000. 22

[39] OASIS standard. `http://www.oasisopen.org/`. 22, 25, 139

[40] G. Xiaoli, Z. Shuliang, S. Yehua, and L. Guonian, "Design and implementation of GML paper based on grammatical and semantic database," *Information Technology and Artificial Intelligence Conference (ITAIC)*, pp. 316–320, 2001. 24, 25

[41] R. Herring, "OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option," 2010. 24

[42] D. Jens, "KML center introduction to col gullet," *Collaborative Agent Design Research Center Presentation*, 2011. 25

[43] "ISO/TC 211 geographic information/geomatics," tech. rep., H. Tom and C. Roswell, 2009. 25

[44] V. Olaya, *SEXTANTE User's Manual (v1.0).* 2011. 26, 44

[45] V. Olaya and J. C. Gimenez, "SEXTANTE, a versatile open-source library for spatial data analysis," 2011. 27

[46] M. Johansson and L. Harrie, "Using Java topology suite for real-time data generalisation and integration," 2002. 27, 31

[47] C. Granell, L. Díaz, and M. Gould, "Service-oriented applications for environmental models: Reusable geospatial services," *Environmental Modelling & Software*, vol. 25, no. 2, pp. 182–198, 2010. 27, 28

[48] B. Beyhan, "Developing graph theoretic analysis tools in FOSS4GIS: An experiment in OpenJUMP with a specific focus on space syntax," *Geoinformatics Conference*, 2012. 28, 31

[49] A. Anguix and L. Diaz, "gvSIG: a GIS desktop solution for an open SDI," *Geoinformatics Conference*, vol. 1, pp. 41–48, 2008. 28, 32

[50] A. Beneventi, A. Poggi, M. Tomaiuolo, and P. Turci, "Integrating rule and agent-based programming to realize complex systems," *WSEAS Transactions on Information Science and Applications*, pp. 487–493, 2004. 30

[51] "User friendly desktop internet GIS (uDIG) for OpenGIS spatial data infrastructures," tech. rep., Refractions Research Inc., 2004. 30, 44

[52] MapServer documentation (2010). `http://mapserver.gis.umn.edu/`. 30

[53] GeoServer user manual (2010). `http://docs.geoserver.org/stable/en/user/index.html`. 30, 105, 122, 154, 164

[54] S. Godilano, "Migration to ArcGIS, training manual V2.0," *Bureau of Agricultural Research*, pp. 1–33, 2005. 30

[55] "User-friendly desktop internet GIS," tech. rep., Refractions Research Inc., 2005. 31

[56] "GeoTools user guide," tech. rep., Open Source Geospatial Foundation, 2011. 31

[57] T. Mitchell, *Web Mapping Illustrated: Using Open Source GIS Toolkits*. O'Reilly Media, 1st ed., 2005. 32, 33, 35, 36

[58] A. Sorokine, "Implementation of a parallel high-performance visualization technique in GRASS GIS," *Computers and Geosciences*, vol. 33, no. 5, pp. 685–695, 2007. 32, 107

[59] J. Stankovic, M. Neteler, and R. Flor, "Mobile wireless GRASS GIS for handheld computers running GNU/linux," *Transactions in GIS*, vol. 8, no. 2, pp. 225–233, 2004. 32, 44, 150

[60] P. Löwe, "A spatial decision support system for radar-meteorology data in South Africa," *Transactions in GIS*, vol. 8, no. 2, pp. 235–244, 2004. 32

[61] J. J. Carrera-Hernandez and S. J. Gaskin, "The groundwater modeling tool for GRASS (GMTG): Open source groundwater flow modeling," *Comput. Geosci.*, vol. 32, no. 3, pp. 339–351, 2006. 32

[62] "New GUI for GRASS GIS based on wxPython," tech. rep., M. Landa, Departament of Geodesy and Cartography, 2008. 33

[63] "Cartographic projecton procedures for the UNIX environment: A users manual," tech. rep., G. Evenden, United States Departament of the Interior Geological Survery, 2003. 33, 35

[64] SAGA GIS. `http://www.saga-gis.org/en/about/software.html`. 33

[65] V. Olaya, *A gentle introduction to SAGA GIS*. University of Göttingen, Germany, 2004. 33

[66] QGIS software review. http://spatialnews.geocomm.com/articles/qgisreview/. 34

[67] "Quantum GIS 1.7.0 geographic information system user guide," tech. rep., T. Athan, O. Dassau and A. Ghisla, Open Source Geospatial Foundation, 2011. 34, 35, 36

[68] PostgreSQL global development group. http://www.postgresql.org/docs/9.1/static/index.html. 34

[69] ERDAS and Intergraph enhancement. http://geospatial.intergraph.com/Company/AboutUs.aspx. 36, 37, 107

[70] H. Rosengarten, *Intergraph's World of Earth Imaging*. Dieter Frisch, 2005. 36

[71] W. Sanders and D. McAllister, "Producing anaglyphs from synthetic images," *Proceeding SPIE*, pp. 348–358, 2003. 37

[72] D. Mihon, V. Bacu, T. Stefanut, and D. Gorgan, "Grid based environment application development - GreenView application," in *Proceedings of the Proceedings of the 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing*, ICCP '09, pp. 275–282, IEEE Computer Society, 2009. 38

[73] D. Gorgan, T. Stefanut, V. Bacu, D. Mihon, and D. Rodila, "Grid based environment application development methodology," *Springer Journal Lecture Notes in Computer Science*, vol. 5910, pp. 499–506, 2010. 38, 66, 167, 168

[74] D. Mihon, V. Bacu, R. Mészáros, G. Gelybó, and D. Gorgan, *Satellite Image Interpolation and Analysis through GreenView Application*. CISIS series, Proceedings of the 2010 International Conference on Complex, Intelligent and Software Intensive Systems, IEEE Computer Society, 2010. ISBN: 978-0-7695-3967-6. 39, 75

[75] V. Bacu, T. Stefanut, D. Rodila, and D. Gorgan, "Process description graph composition by gProcess platform," *3rd International Workshop on High Performance Grid Middleware, CSCS-17 Conference*, vol. 2, pp. 423–430, 2009. 39, 112, 113

[76] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and Grid computing 360-degree compared," *Grid Computing Environments Workshop*, pp. 1–10, 2008. 39, 99

[77] "Remote sensing services: ESIP platform and hot spot inventory case studies," tech. rep., B.F. Bektas, C. Goksel, S. Sozen, K. Allenbach, M. Gvilava, K. Rahman, D. Gorgan, and D. Mihon, enviroGRIDS deliverable D2.11, 2012. 40, 117, 142, 155, 156, 162, 164, 168

[78] D. Mihon, A. Minculescu, C. V., and D. Gorgan, "Descrierea diagramatica a prelucrarii distribuite a datelor spatiale," *Revista Romana de Interactiune Om-Calculator*, vol. 5, no. 3, pp. 59–80, 2012. 40, 75, 77, 84, 85, 168

[79] D. Mihon, V. Colceriu, V. Bacu, and D. Gorgan, "Grid based processing of satellite images in GreenLand platform," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. Special issue, in press. 40, 168

[80] enviroGRIDS project (2013). http://envirogrids.net/. 40, 150, 156

[81] D. Gorgan, V. Bacu, T. Stefanut, D. Rodila, and D. Mihon, "Earth observation application development based on the Grid oriented ESIP satellite image processing platform," *Comput. Stand. Interfaces*, vol. 34, no. 6, pp. 541–548, 2012. 45, 80

[82] G. Aloisio and M. Cafaro, "A dynamic Earth observation system," *Parallel Comput.*, vol. 29, no. 10, pp. 1357–1362, 2003. 45

[83] D. Gorgan, V. Bacu, T. Stefanut, D. Rodila, and D. Mihon, "Grid based satellite image processing platform for Earth observation application development," *Workshop on Intelligent Data Acquisition and Advanced Computing Systems*, pp. 247–252, 2009. 45

[84] Y. Wang, S. Wang, and D. Zhou, "Retrieving and indexing spatial data in the Cloud computing environment," in *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pp. 322–331, Springer-Verlag, 2009. 45

[85] G. Giuliani, S. Nativi, A. Lehmann, and N. Ray, "WPS mediation: An approach to process geospatial data on different computing backends," *Comput. Geosci.*, vol. 47, pp. 20–33, 2012. 45

[86] Z. Chen, N. Chen, C. Yang, and L. Di, "Cloud computing enabled Web Processing Service for Earth observation data processing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 5, pp. 1637–1649, 2012. 45

[87] D. Mihon, V. Bacu, T. Stefanut, and D. Gorgan, "Tehnici de interactiune pentru executia modelelor hidrologice pe Grid - aplicatia gSWAT," *Revista Romana de Interactiune Om-Calculator*, vol. 4, no. Special issue, pp. 103–106, 2011. 45

[88] C. Larman and V. Basili, "Iterative and incremental development: A brief history," *IEEE Computer*, vol. 36, no. 6, pp. 47–56, 2003. 47

[89] J. Kalermo and J. Rissanen, "Agile software development in theory and practice." Master Thesis, University of Jyvaskyla, Department of Computer Science and Information Systems, 2002. 47

[90] V. Colceriu and D. Gorgan, "Optimizarea prelucrarii datelor geospatila pe infrastructuri distribuite." Master Thesis in Computer Science, Cluj-Napoca, 2013. 57, 120, 121

[91] D. Groeneveld and W. Baugh, "Correcting satellite data to detect vegetation signal for eco-hydrologic analyses," *Journal of Hydrology*, vol. 344, pp. 135–145, 2007. 61

[92] H. Haifeng, K. Jianrog, Z. Xiaoke, and D. Kaiyuan, "Atmospheric correction of SPOT satellite images based on radiation transfer model," *Computer Application and System Modeling(ICCASM)*, vol. 5, pp. 188–191, 2010. 61

[93] C. Junqing, J. Tao, L. Wenhu, and H. Mingqin, "The geometric correction and accuracy assessment based on Cartosat-1 satellite image," *Image and Signal Processing (CISP)*, vol. 3, pp. 1253–1257, 2010. 61

[94] P. Sellers, "Canopy reflectance, photosynthesis and transpiration," *International Journal of Remote Sensing*, vol. 6, pp. 1335–1372, 1985. 63

[95] B. D. Wardlow and S. L. Egbert, "A comparison of MODIS 250-m EVI and NDVI data for crop mapping: a case study for southwest kansas," *Int. J. Remote Sens.*, vol. 31, no. 3, pp. 805–830, 2010. 63

[96] B. Pinty and M. M. Verstraete, "GEMI: a non-linear index to monitor global vegetation from satellites," *Vegetatio*, vol. 100, pp. 15 – 20, 1992. 63

[97] J. Gamon and J. Surfus, "Assessing leaf pigment content and activity with a reectometer," *New Phytologist Journal*, vol. 143, pp. 105–117, 1999. 63

[98] E. Hunt and B. Rock, "Detection of changes in leaf water content using near-and middle-infrared reflectances," *Remote Sensing of Environment*, vol. 30, pp. 43–54, 1989. 63

[99] L. Samaniego and K. Schulz, "Supervised classification of agricultural land cover using a modified k-NN technique (MNN) and Landsat remote sensing imagery," *Remote Sensing*, vol. 1, pp. 875–895, 2009. 63

[100] J. Karl and B. Maurer, "Multivariate correlations between imagery and field measurements across scales: comparing pixel aggregation and image segmentation," *Landscape Ecology*, vol. 25, no. 4, pp. 591–605, 2010. 63

[101] D. Mihon, V. Bacu, D. Gorgan, R. Mészáros, and G. Gelybó, "Practical considerations on the GreenView application development and execution over SEE-GRID," *Earth Science Informatics*, pp. 1–12, 2010. 64, 108, 119

[102] D. Mihon, V. Bacu, D. Gorgan, R. Meszaros, G. Gelybo, and T. Stefanut, "GreenView and GreenLand applications development on SEE-GRID infrastructure." EGU2010 - European Geosciences Union General Assembly, Vienna, Austria, Geophysical Research Abstracts, Vol. 12, EGU2010-14455, eISSN: 1607-7962, 2010. 64, 80

[103] PROJ4 user manual. `ftp://ftp.remotesensing.org/proj/OF90-284.pdf`. 65

[104] D. Mihon, V. Bacu, T. Stefanut, and D. Gorgan, "Considerations on the Grid oriented environmental application development framework," in *Proceedings of the Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, ICCP '10, pp. 419–426, IEEE Computer Society, 2010. 66, 108, 168

[105] D. Gorgan, V. Bacu, S. Manca, G. Giuliani, D. Rodila, T. Stefanut, D. Mihon, K. Abbaspour, E. Rouholahnejad, A. van Griensven, L. Kokoszkiewicz, and P. Cau, "Tools and applications in enviroGRIDS project for spatial data processing of Black Sea catchment basin." EGI User Forum, Vilnius, 11-15 April, 2011. 75

[106] V. Colceriu, D. Mihon, A. Minculescu, V. Bacu, D. Rodila, and D. Gorgan, "Workflow based description and distributed processing of satellite images," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. Special issue, in press. 75

[107] D. Mihon, V. Colceriu, A. Minculescu, V. Bacu, and D. Gorgan, "Descrierea diagramatica a prelucrarii imaginilor satelitare in aplicatia GreenLand," *Revista Romana de Interactiune Om-Calculator*, vol. 5, no. 2, pp. 129–134, 2012. 77

[108] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, *Object-oriented modeling and design*. Prentice-Hall, Inc., 1991. 78

[109] D. Mihon, V. Bacu, D. Rodila, T. Stefanut, K. Abbaspour, E. Rouholahnejad, and D. Gorgan, *Grid Based Hydrologic Model Calibration and Execution*. Advanced in Intelligent Control Systems and Computer Science, Springer-Verlag, 2013. ISBN: 978-3-642-32547-2. 80

[110] H. C. Purchase, E. E. Hoggan, and C. Görg, "How important is the "mental map"? - an empirical investigation of a dynamic graph layout algorithm.," in *Graph Drawing*, vol. 4372 of *Lecture Notes in Computer Science*, pp. 184–195, Springer, 2006. 85

[111] M. R. Garey and D. S. Johnson, "Crossing number is NP-complete," *SIAM Journal on Algebraic and Discrete Methods*, vol. 4, no. 3, pp. 312–316, 1983. 86

[112] M. Baur and U. Brandes, "Crossing reduction in circular layouts," in *Proc. Workshop on Graph-theoretic Concepts in Computer Science*, vol. 3353, pp. 332–343, Springer, 2004. 86

[113] M. Eiglsperger, S. P. Fekete, and G. W. Klau, "Orthogonal graph drawing.," in *Drawing Graphs*, vol. 2025 of *Lecture Notes in Computer Science*, pp. 121–171, Springer, 1999. 86

[114] W. Didimo, G. Liotta, and S. A. Romeo, "Topology-driven force-directed algorithms," in *Proceedings of the 18th international conference on Graph drawing*, GD'10, pp. 165–176, Springer-Verlag, 2011. 86

[115] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1st ed., 1998. 86

[116] D. Mihon, V. Colceriu, and D. Gorgan, "Distributed processing and analysis of medical images based on Grid infrastructure." MEI 2012 - International Conference on Medical Education Informatics, 6-7 April, Thessaloniki, 2012. 93

[117] V. Colceriu, D. Mihon, and D. Gorgan, "Distributed computer aided detection algorithms." MEI 2012 - International Conference on Medical Education Informatics, 6-7 April, Thessaloniki, 2012. 93

[118] V. Bacu, D. Mihon, D. Rodila, T. Stefanut, and D. Gorgan, "gSWAT platform for Grid based hydrological model calibration and execution," in *Proceedings of the 2011 10th International Symposium on Parallel and Distributed Computing*, ISPDC '11, pp. 288–291, IEEE Computer Society, 2011. 93

[119] D. Gorgan, K. Abbaspour, P. Cau, V. Bacu, D. Mihon, G. Giuliani, N. Ray, and A. Lehmann, "Grid based data processing tools and applications for Black Sea catchment basin.," pp. 223–228, IEEE, 2011. 94

[120] Intel processors specifications. `http://www.intel.com/design/corei7/documentation.htm`. 94

[121] AMD processors specifications. `http://support.amd.com/us/Pages/techdocs.aspx`. 94

[122] "Multi-core computing in embedded applications: Global market opportunities and requirements analysis," tech. rep., E. Heikkila, and J.E.Gulliksen, Embedded Hardware and Systems Practice, VDC Research Group, 2007. 94

[123] H. Zhong, S. A. Lieberman, and S. A. Mahlke, "Extending multicore architectures to exploit hybrid parallelism in single-thread applications," in *Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture*, HPCA '07, pp. 25–36, IEEE Computer Society, 2007. 95

[124] G. Bronevetsky, D. J. Marques, K. K. Pingali, R. Rugina, and S. A. McKee, "Compiler-enhanced incremental checkpointing for openmp applications," in *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, PPoPP '08, pp. 275–276, ACM, 2008. 95

[125] X. Sui, D. Nguyen, M. Burtscher, and K. Pingali, "Parallel graph partitioning on multicore architectures.," in *LCPC*, vol. 6548 of *Lecture Notes in Computer Science*, pp. 246–260, Springer, 2010. 95

[126] M. Baker, "Cluster computing white paper," tech. rep., 2000. 96

[127] Titan cluster. `http://www.top500.org/blog/lists/2012/11/press-release/`. 96

[128] A. Plastino, C. C. Ribeiro, and N. Rodriguez, "Developing SPMD applications with load balancing," *Parallel Comput.*, vol. 29, no. 6, pp. 743–766, 2003. 96

[129] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach, 3rd Edition.* Morgan Kaufmann Publishers Inc., 2003. 96

[130] M. Lee, E. J. Kim, K. H. Yum, and M. Yousif, "An overview of security issues in cluster interconnects.," in *CCGRID*, p. 25, IEEE Computer Society, 2005. 97

[131] J. M. Francioni and C. M. Pancake, "A debugging standard for high-performance computing," *Journal of Scientific Programming*, vol. 8, pp. 95–108, 2000. 97

[132] TotalView debugger documentation. `http://www.roguewave.com/support/product-documentation/totalview.aspx`. 97

[133] Amazon EC2. `http://aws.amazon.com/whitepapers/`. 98

[134] Windows Azure Virtual Machines. `http://www.windowsazure.com`. 98

[135] Google App Engine. `https://developers.google.com/appengine/docs/whatisgoogleappengine`. 98

[136] "Cloud computing use cases whitepaper," tech. rep., Cloud Computing Use Case Group, 2009. 98

[137] S. Lim, G. Fox, S. Pallickara, and M. Pierce, "Web service robust GridFTP," in *The 2004 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA04)*, 2004. 99

[138] V. Bacu, D. Mihon, T. Stefanut, D. Rodila, D. Gorgan, P. Cau, and S. Manca, "Grid based services and tools for hydrological model processing and visualization," in *Proceedings of the 2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, SYNASC '11, pp. 291–298, IEEE Computer Society, 2011. 99

[139] Grid security mechanisms. `http://gdp.globus.org/progtutorial/gt3-tutorial-versions/0.4.3/multiplehtml/`. 100

[140] V. Bacu, D. Mihon, D. Rodila, T. Stefanut, and D. Gorgan, "Grid based architectural components for SWAT model calibration," in *HPCS*, pp. 193–199, IEEE, 2011. 100

[141] gLite documentation. `http://glite.cern.ch/admindocumentation`. 101, 116

[142] Globus Toolkit documentation. `https://www.globusonline.org/docs/`. 101, 116

[143] I. Masser, *GIS worlds: creating spatial data infrastructures.* ESRI Press, 2005. 107

[144] H. Zhu and C. P. Yang, "Data compression for network GIS," in *Encyclopedia of GIS*, pp. 209–213, Springer, 2008. 107

[145] D. Gorgan, V. Bacu, D. Mihon, T. Stefanut, D. Rodila, P. Cau, K. Abbaspour, G. Giuliani, N. Ray, and A. Lehmann, "Software platform interoperability throughout enviroGRIDS portal," *International Journal of Selected Topics in Applied Earth Observations and Remote Sensing - JSTARS*, vol. 5, no. 6, pp. 1617–1627, 2012. 112, 130

[146] D. Gorgan, D. Mihon, and V. Bacu, "Satellite image processing based on the GreenLand application." OpenWater symposium and workshops, UNESCO-IHE, 2011. 118

[147] V. Colceriu, D. Mihon, and D. Gorgan, "Optimizing the satellite images execution process through a decision mediator approach," in *Proceedings of the Proceedings of the 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing*, ICCP '13, pp. 343–346, IEEE Computer Society, 2013. 120

[148] D. Mihon, V. Colceriu, V. Bacu, K. Allenbach, D. Rodila, G. Giuliani, and D. Gorgan, "OGC compliant services for remote sensing processing over the Grid infrastructure," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. Special issue, in press. 126, 168

[149] "PyWPS 2.0.0: the presence and the future," tech. rep., J. Cepicky, Geoinformatics, 2012. 126

[150] D. Gorgan, V. Bacu, D. Mihon, D. Rodila, K. Abbaspour, and E. Rouholahnejad, "Grid based calibration of SWAT hydrological models," *Natural Hazards and Earth System Science*, vol. 12, no. 7, pp. 2411–2423, 2012. 133

[151] D. Mihon, V. Bacu, T. Stefanut, and D. Gorgan, "Tehnici de interactiune utilizator in aplicatiile de prelucrare a imaginilor satelitare - exemplificare pe baza aplicatiilor GreenView si GreenLand," *Revista Romana de Interactiune Om-Calculator*, vol. 3, no. Special issue, pp. 37–44, 2010. 136

[152] V. Bacu, D. Mihon, T. Stefanut, D. Rodila, K. Abbaspour, E. Rouholahnejad, and D. Gorgan, "Calibration of SWAT hydrological models in a distributed environment using the gSWAT application," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. Special issue, in press. 137

[153] T. Heinen, M. May, and B. Schmidt, "3d visualisation in spatial data infrastructures," in *Smart Graphics*, vol. 3638 of *Lecture Notes in Computer Science*, pp. 222–229, Springer Berlin Heidelberg, 2005. 138

[154] W3C standard. `http://www.w3.org/2005/10/Process-20051014/tr`. 139

[155] D. Mihon, V. Colceriu, V. Bacu, and D. Gorgan, "Tehnici de vizualizare a imaginilor satelitare prin servicii geospatiale," *Revista Romana de Interactiune Om-Calculator*, vol. 6, no. 1, pp. 1–22, 2013. 142, 168

[156] Esri Maps, An Esri white paper. `http://www.esri.com/library/whitepapers/pdfs/esri-maps-for-ibm-cognos.pdf`. 144, 148

[157] D. Mihon, V. Colceriu, V. Bacu, T. Stefanut, and D. Gorgan, "Optimizarea interfetelor grafice pentru aplicatiile de prelucrare a datelor spatiale," *Revista Romana de Interactiune Om-Calculator*, vol. 6, no. Special issue, pp. 79–85, 2013. 147, 168

[158] F. B. Balcik, D. Mihon, V. Colceriu, K. Allenbach, C. Goksel, A. O. Dogru, M. Gvilava, G. Giuliani, and D. Gorgan, "Remotely sensed data processing on Grids by using GreenLand Web-based platform," *International Journal of Advanced Computer Science and Applications(IJACSA)*, vol. Special issue, in press. 156, 168

[159] Landsat remote sensing platform. `http://wist-ops.echo.nasa.gov/api/`. 157

[160] M. Moran, R. D. Jackson, P. N. Slater, and P. M. Teillet, "Evaluation of simplified procedures for retrieval of land surface reflectance factors from satellite sensor output," *Remote Sensing of Environment*, vol. 41, no. 2-3, pp. 169 – 184, 1992. 158

[161] G. M. Foody, "Status of land cover classification accuracy assessment," *Remote Sensing of Environment*, vol. 80, no. 1, pp. 185–201, 2002. 159

[162] *EOS Data Products Handbook*. No. v. 2, NASA, 2000. 162

Appendix A

Copies of some published articles

Revista Română de Interacţiune Om-Calculator

Revistă editată de RoCHI (ACM SIGCHI Romania)

ISSN 1843-4460

Revistă publicată de © MATRIX ROM

C.P. 16-162, 062510 – BUCURESTI, Tel. 021 4113617 Fax. 021 4114280

E-mail: office@matrixrom.ro

Revista Română de Interacțiune Om - Calculator

Volumul 6 Numărul 1 (2013)

ISSN 1843-4460

Tehnici de vizualizare a imaginilor satelitare prin servicii geospaţiale

Vasile Dănuţ Mihon, Vlad Colceriu, Victor Bâcu, Dorian Gorgan

Universitatea Tehnică din Cluj-Napoca
Str. Memorandumului, Nr. 28, 400114, Cluj-Napoca
E-mail: {vasile.mihon, vlad.colceriu, victor.bacu, dorian.gorgan}@cs.utcluj.ro

Rezumat. Analiza unui volum mare de date, extragerea informaţiilor de interes, expunerea şi vizualizarea interactivă a rezultatelor reprezintă procesele de bază în modelarea fenomenelor naturale, bazate pe prelucrarea imaginilor satelitare. Datorită dimensiunii mari şi a frecvenţei ridicate de colectare, imaginile satelitare sunt păstrate în depozite de date specializate, care încurajează accesul la informaţii prin intermediul standardelor existente. Lucrarea prezintă experimentele din platforma GreenLand pentru realizarea unor tehnici de interacţiune cu o înaltă utilizabiltate pentru accesul, căutarea, extragerea şi vizualizarea imaginilor satelitare. Soluţiile de implementare folosesc serviciile geospaţiale oferite de standardul OGC. Beneficiile rezultate în urma integrării acestui standard cu platforma GreenLand şi cu serviciile oferite de infrastructura Grid reprezintă un alt subiect esenţial al acestei lucrări.

Cuvinte cheie: standarde OGC, tehnici de interacţiune, vizualizare interactivă, imagini satelitare, servicii geospaţiale.

1. Introducere

Performanţa execuţiei şi acurateţea rezultatelor obţinute în urma modelării fenomenelor naturale, a experimentelor şi a scenariilor complexe, necesită procesarea unui volum mare de date spaţiale. Acestea reprezintă caracteristici (temperatură, umiditate etc.) ale suprafeţei terestre în formate digitale, cunoscute sub denumirea de imagini satelitare. Datorită frecvenţei ridicate de colectare (ex.: zilnic, săptămânal) şi a gradului mare de acoperire a regiunilor geografice, imaginile satelitare sunt folosite intens în studiul fenomenelor din domeniul Ştiinţa Pământului.

Datele spaţiale sunt stocate în depozite specializate, de unde pot fi folosite în diverse scopuri, de către diferite categorii de utilizatori: modelarea fenomenelor naturale (de către specialişti în domeniul Ştiinţa Pământului), elaborarea de decizii, reguli şi politici (de către persoane din

domeniul administrativ), vizualizarea datelor în scop educativ sau informaţional (de către marea majoritate a utilizatorilor) etc.

Există, la nivel internațional, o gamă variată de depozite de imagini satelitare, organizate pe baza unor reguli proprietare. Accesul la informaţii este astfel îngreunat, deoarece utilizatorii trebuie să respecte politicile de acces impuse de aceste organizaţii. Drept urmare, au fost adoptate noi standarde de expunere a datelor, care maschează dificultăţile de acces descrise anterior. Unul dintre acestea este Open Geospatial Consortium (OGC), care permite accesarea, căutarea, prelucrarea şi vizualizarea imaginilor satelitare prin intermediul serviciilor geospaţiale (OGC, 2007).

Principalele avantaje ale folosirii de către utilizatori a standardului OGC sunt: simplificarea procedurii de acces la date prin implementarea unor tehnici de interacţiune comune, facilitarea căutării şi extragerii de informaţii relevante pe baza metadatelor asociate, standardizarea vizualizării prin folosirea hărţilor interactive şi afişarea dinamică a datelor.

Lucrarea de faţă are ca obiective principale integrarea serviciilor geospaţiale OGC în cadrul platformei GreenLand, precum şi implementarea tehnicilor de interacţiune utilizator pentru exploatarea beneficiilor descrise anterior. Aplicaţia GreenLand (Mihon et al., 2012) oferă suport pentru prelucrarea algoritmilor din domeniul Geographic Information System (GIS), fiind folosită pentru gestionarea, modelarea, procesarea, analiza şi vizualizarea datelor spaţiale.

În particular, platforma GreenLand este utilizată pentru implementarea a trei studii de caz: modelarea hidrologică a bazinului Mării Negre, clasificarea zonelor de vegetaţie, apă şi locaţii urbane din cadrul regiunii Istanbul şi, modelarea hidrologică a râului Rioni. Puterea de calcul şi capacitatea de stocare a infrastructurii Grid au fost necesare pentru implementarea cerinţelor impuse în cadrul acestor scenarii (Bektaş et al., 2012).

Lucrarea de faţă descrie conceptele teoretice şi soluţiile practice adoptate pentru integrarea serviciilor OGC în cadrul platformei GreenLand. Tehnicile de interacţiune utilizator implementate şi beneficiile aduse de acestea, sunt de asemenea detaliate în paragrafele următoare.

2. Alte realizări în domeniu

Există în literatura de specialitate un număr mare de aplicaţii de prelucrare a datelor spaţiale, care folosesc metode proprietare pentru analiza şi vizualizarea rezultatelor, existând în acest sens diferite experimente pe

infrastructura Grid (Aloisio&Cafaro, 2003; Gorgan et al., 2009) şi Cloud (Wang et al., 2009). Soluţiile abordate sunt fezabile în cazul folosirii independente a acestor platforme semi-deschise, în care datele de intrare sunt specificate manual, de către fiecare utilizator în parte.

În prezent, cercetările din domeniul GIS recomandă partajarea datelor şi folosirea unor tehnici de interacţiune comune pentru accesul la aceste informaţii distribuite. Se impune aşadar folosirea unor standarde care să permită interoperabilitatea atât la nivel de date cât şi la nivel de aplicaţii.

Deşi următoarele generaţii de platforme GIS folosesc standarde (ex.: OGC) pentru eliminarea acestor neajunsuri, ele se adresează studiilor de caz de complexitate medie, a căror execuţie nu necesită resurse puternice de calcul. În această categorie pot fi incluse sistemele: Sextante (Sextante, 2011), uDig (Ramsey, 2004), Quantum GIS (Athan et al., 2013) etc.

Toate aceste aplicaţii oferă un număr mare de algoritmi de prelucrare, analiză şi vizualizare a datelor spaţiale, inclusiv suport pentru serviciile OGC. Soluţia GreenLand propusă în cadrul acestei lucrări se diferenţiează de aceste sisteme prin:

- execuţia paralelă şi distribuită a datelor spaţiale, folosind infrastructura Grid;
- dezvoltarea unui mecanism de monitorizare a prelucrărilor descrise anterior;
- o mai bună securitate a datelor, prin introducerea certificatelor Grid pentru accesul, prelucrarea şi vizualizarea imaginilor satelitare (Gorgan et al., 2012).

O a treia categorie de aplicaţii se adresează studiilor de caz complexe, care necesită resurse puternice de calcul, precum cele oferite de infrastructura Grid sau Cloud. O altă caracteristică importantă a acestor sisteme (care le diferenţiează de cele prezentate anterior) constă în capacitatea de folosire a standardelor pentru accesul, prelucrarea şi vizualizarea datelor spaţiale.

Platforma propusă în Giuliani et al. (2011) descrie conceptele teoretice referitoare la implementarea unui sistem capabil să redirecţioneze execuţia pe diferite platforme distribuite, în funcţie de complexitatea algoritmilor. Prelucrarea şi vizualizarea standardizată a imaginilor satelitare, folosite pentru implementarea indicelui de vegetaţie Normalized Difference Vegetation Index – NDVI, fac parte din studiul experimental propus de autori, în vederea validării conceptelor descrise.

Un alt sistem (Chen et al., 2012), care poate fi inclus în această ultimă categorie, integrează în cadrul prelucrăriilor Cloud doar serviciul de

execuţie oferit de standardul OGC, în timp ce vizualizarea rezultatelor se face în manieră tradiţională.

La rândul ei, platforma GreenLand poate fi catalogată ca făcând parte din aceeaşi categorie cu sistemele propuse de Giuliani şi Chen. Platforma foloseşte infrastructura Grid pentru execuţia paralelă şi distribuită a algoritmilor de prelucrare a imaginilor satelitare, integrând marea majoritate a serviciilor OGC. Diferenţele faţă de aplicaţiile descrise anterior constă în:

- tehnicile de interacţiune utilizator folosite, care permit folosirea aplicaţiei de către o gamă variată de utilizatori, fără a ţine seama de gradul de pregătire a acestora în domeniul calculatoarelor;
- interoperabilitatea cu alte sisteme, utilizatorii putând accesa, extrage, prelucra şi vizualiza imagini satelitare din depozite de date localizate în diferite regiuni geografice;
- disponibilitatea serviciilor GreenLand, acestea trecând din faza conceptuală în cea de implementare, optimizare şi mentenanţă.

3. Concepte teoretice

Conceptul de vizualizare a imaginilor satelitare, presupune existenţa unei platforme care oferă suport pentru astfel de operaţii, a unui set de date spaţiale care urmează a fi analizate, precum şi un set de reguli de acces la aceste informaţii. Particularizarea noţiunilor curente în contextul lucrării, este următoarea:

- algoritmii şi complexitatea soluţiilor de implementare a acţiunilor-utilizator sunt mascate prin intermediul tehnicilor de interacţiune utilizator dezvoltate în cadrul platformei GreenLand (Mihon et al., 2012).
- Din cerinţele celor trei studii de caz (descrise în capitolul *Introducere*) s-a ajuns la concluzia necesităţii de interoperabilitate la nivel de date cu alte platforme existente. Avantajele obţinute în urma implementării acestei funcţionalităţi, oferă posibilitatea de accesare şi expunere standardizată a datelor spaţiale în multiple depozite locale sau la distanţă;
- imaginile satelitare reprezintă setul de date de intrare folosit pentru algoritmii dezvoltaţi în cadrul platformei GreenLand. Datorită diversităţii şi răspândirii teritoriale a depozitelor care înmagazinează aceste informaţii, colectarea imaginilor satelitare devine un proces dificil de realizat de către utilizatori;

- integrarea serviciilor oferite de standardul OGC în cadrul platformei GreenLand, reprezintă o posibilă abordare în soluţionarea problemelor de acces, colectare şi vizualizare a acestor tipuri de date. Se folosesc, în special, serviciile WCS (pentru colectarea şi analiza datelor) şi WMS (pentru vizualizarea acestora).

3.1 Standarde şi interoperabilitatea datelor spaţiale

Comunicarea prin intermediul serviciilor Internet are un rol major în propagarea informaţiei, ocupând un loc tot mai important în cadrul societăţii moderne. Succesul este datorat în mare parte ”capacităţii sistemelor şi a produselor de a interacţiona între ele, fără a necesita un efort special din partea utilizatorilor” (OGC, 2004), definiţie care poate fi atribuită conceptului de interoperabilitate.

Pentru a particulariza această descriere pentru contextul lucrării de faţă, se poate spune că interoperabilitatea datelor presupune existenţa a două sau mai multe platforme capabile de a expune, schimba şi folosi informaţii prin intermediul unui set comun de tehnici de interacţiune, algoritmi şi servicii.

Standardele reprezintă acorduri documentate, care conţin specificaţii tehnice, definiţii, caracteristici şi reguli folosite în scopul îndeplinirii unor obiective comune. Se poate vedea, din cele două definiţii, faptul că standardele asigură interoperabilitea atât la nivel de date, cât şi la nivel de sisteme şi platforme.

Una dintre problemele majore în folosirea imaginilor satelitare o reprezintă faptul că acestea sunt înmagazinate în depozite de date care îşi expun informaţiile într-un mod proprietar. Pe de altă parte, fiecare aplicaţie prelucrează datele spaţiale în mod independent. Toate acestea contribuie la necesitatea utilizatorilor de a învăţa diferitele tehnici de acces şi vizualizare a imaginilor satelitare, aşa cum sunt impuse de fiecare dintre aceste sisteme.

Din punctul de vedere al utilizatorilor, beneficiile folosirii standardelor în domeniul Ştiinţa Pământului, sunt reflectate în vizualizarea datelor spaţiale în acelaşi stil grafic, prin intemediul unor tehnici de interacţiune comune. Se obţine astfel o creştere a gradului de satisfacţie al utilizatorilor, precum şi reducerea timpului necesar pentru realizarea anumitor sarcini de lucru.

Pentru implementarea obiectivelor menţionate, soluţia propusă în cadrul lucrării foloseşte standardul OGC. Cu toate acestea, există un număr mare de standarde şi directive care puteau fi utilizate, precum: International Organization for Standardization (ISO), World Wide Web Consortium (W3C), Organization for the Advancement of Structured Information

Standards (OASIS). Motivaţia folosirii serviciilor OGC poate fi exprimată prin faptul că acestea oferă cele mai bune principii pentru gestionarea datelor spaţiale.

3.2 Clasificarea datelor spaţiale

Datele spaţiale pot fi clasificate în funcţie de metodele de colectare utilizate şi de structura internă de organizare a informaţiilor. Din prima categorie se pot desprinde următoarele tipuri: imagini satelitare, date aeriene şi date colectate manual prin măsurători efectuate la nivelul suprafeţei terestre.

Imaginile satelitare se obţin prin intermediul senzorilor montaţi la bordul sateliţilor artificiali care orbitează în jurul Pământului, colectând periodic caracteristicile (ex.: umiditate, temperatură) suprafeţei terestre prin scanarea acesteia în diferite benzi de frecvenţă.

Datele colectate sunt organizate pe niveluri, fiecare bandă conţinând o singură caracteristică analizată. În funcţie de tipul sateliţilor, se poate ajunge la imagini satelitare formate din 7 benzi Landsat (Landsat, 2013), 15 benzi ASTER (ASTER, 2013), 36 de benzi MODIS (Reeves et al., 2006) etc.

Imaginile aeriene sunt preluate prin senzorii montaţi la bordul aeronavelor, care permit scanarea în detaliu a caracteristicilor suprafeţei terestre, în această categorie putând fi incluse produsele SPOT (SPOT, 2013) şi QuickBird (Wang et al., 2004). În funcţie de cerinţele funcţionale şi non-funcţionale ale aplicaţiilor, se pot folosi ca date de intrare fie imaginile satelitare, fie imaginile aeriene, fie o combinaţie între ele. Diferenţele între cele două tipuri de date constă în faptul că:

- imaginile satelitare au o rezoluţie spaţială superioară, acoperind o regiune geografică mai întinsă;
- imaginile satelitare au o rezoluţie temporală mai bună, durata de revizitare a aceleiaşi regiuni geografice fiind de la una la opt zile;
- senzorii imaginilor aeriene scanează suprafeţele mai în detaliu decât sateliţii artificiali, aceştia fiind folosiţi în cadrul scenariilor de utilizare care necesită o mai bună acurateţe a rezultatelor.

De obicei, datele achiziţionate prin măsurători efectuate la nivelul suprafeţei terestre sunt folosite în procesele de calibrare a diferitelor modele experimentale. Procedura de colectare este mai dificilă decât în cazurile precedente, iar suprafaţa de acoperire este limitată din punct de vedere geografic.

În prezent, datele spaţiale sunt folosite în diferite studii, precum: clasificarea vegetaţiei din regiunile geografice de interes (activitate

importantă din punct de vedere economic şi al dezvoltării durabile), măsurarea cantităţii apei şi monitorizarea calităţii acesteia, clasificarea regiunilor agricole şi forestiere (oferind soluţii practice pentru îmbunătăţirea producţiei), planificarea urbană etc.

Datele spaţiale pot fi clasificate şi în funcţie de modul de organizare internă. Astfel, se pot distinge imaginile raster şi vectoriale. Primele folosesc noţiunea de pixel pentru reprezentarea caracteristicilor măsurate, în timp ce imaginile vectoriale se bazează pe conceptul de primitivă vectorială: punct, linie şi poligon.

Principalul dezavantaj al datelor raster este reprezentat de imposibilitatea de a reţine, în cadrul unei singure benzi de frecvenţă, mai multe atribute caracteristice regiunii geografice de studiu, precum şi erorile care pot interveni pentru imaginile cu rezoluţii mari. Din cauza acestor impedimente, datele raster au dimensiuni mari, iar informaţiile sunt împachetate în benzi multispectrale. Soluţia alternativă în aceste situaţii este reprezentată de informaţiile vectoriale, care permit scanarea unor suprafeţe de dimensiuni mari, fiind capabile de a reţine o multitudine de atribute geografice.

Soluţiile dezvoltate în cadrul platformei GreenLand oferă suport atât pentru datele vectoriale cât şi pentru datele raster, colectate prin intermediul sateliţilor artificiali sau prin măsurătorile manuale, efectuate la nivelul suprafeţei terestre. Ca urmare, în cadrul experimentelelor care urmează a fi descrise, vor fi utilizate doar aceste tipuri de date spaţiale.

3.3 Servicii geospaţiale OGC

Open Geospatial Consortium (OGC) reprezintă o organizaţie internaţională non-profit, care oferă practici pentru dezvoltarea serviciilor de prelucrare şi vizualizare a datelor spaţiale. Cele mai cunoscute produse sunt: Web Map Service (WMS), Web Coverage Service (WCS), Web Feature Service (WFS) şi Web Processing Service (WPS). Toate acestea pot fi accesate sub formă de adrese Web, folosind protocolul HTTP (OGC, 2007).

Datorită faptului că WFS face referire la tipuri de date care nu sunt acceptate momentan în cadrul platformei GreenLand, descrierea acestui produs nu este detaliată în cadrul secţiunilor următoare. Serviciile WPS sunt folosite pentru prelucrarea standardizată a diferiţilor algoritmi geospaţiali. Ca urmare a faptului că lucrarea de faţă nu acoperă partea de procesare a datelor, serviciile WMS şi WCS sunt singurele produse care urmează a fi descrise în contextul vizualizării datelor spaţiale.

Serviciul WMS

Serviciul WMS (Web Map Service) reprezintă o modalitate de accesare și vizualizare a datelor spațiale, indiferent de locaţia depozitului folosit pentru stocarea lor. Acest produs nu oferă acces la informaţiile originale, ci are rolul de a genera automat reprezentări grafice (ex.: imagini JPEG, TIF, PNG) ale acestora. O astfel de reprezentare poartă denumirea de bandă (layer) şi poate fi identificată ca şi un nivel de frecvenţă din cadrul imaginilor satelitare.

Odată cu impunerea unor cerinţe calitative din ce în ce mai complexe, pentru aplicaţiile din domeniul Ştiinţa Pământului, au fost dezvoltate tehnici de interacţiune tot mai avansate. Folosirea acestora era posibilă doar de către specialiştii în domeniu, chiar şi pentru aceştia fiind necesară implementarea unor cursuri de instruire.

Serviciile WMS au rolul de a reduce diversitatea şi complexitatea tehnicilor de interacţiune corespunzătoare vizualizării datelor spaţiale. Se realizează astfel o creştere a gradului de satisfacţie a utilizatorilor şi o reducere în ceea ce priveşte timpul necesar pentru efectuarea sarcinilor de lucru.În funcţie de cerinţele utilizatorilor, servicul WMS poate fi particularizat prin intermediul parametrilor săi. Se poate specifica astfel regiunea geografică de interes, tipul proiecţiei, formatul rezultatului care urmează a fi generat etc. Toate aceste operaţii se desfăşoară pe parcursul a două faze: *GetCapabilities* şi *GetMap*.De cele mai multe ori identificarea datelor de intrare pentru un algoritm geospaţial, presupune în prealabil o căutare a informaţiilor relevante. Datorită numărului mare de depozite de date, această operaţie este costisitoare în cazul utilizării manuale. Facilităţile oferite de serviciile WMS reprezintă o posibilă soluţie la această problemă, permiţând automatizarea căutării în cadrul metadatelor asociate.

Căutarea automată, pe bază de cuvinte cheie, s-a dovedit a fi eficientă pentru prelucrarea multiplă a aceloraşi algoritmi, instanţiaţi cu date de intrare diferite. Folosirea acestei metode reduce semnificativ, de la ordinul orelor la câteva minute, timpul necesar utilizatorilor pentru realizarea sarcinilor de lucru corespunzătoare (Bektaş et al., 2012).

Identificarea informaţiilor existente în depozitele de date reprezintă prima fază a procesului WMS, fiind realizată prin intermediul operaţiei *GetCapabilities*. În forma cea mai simplă, un exemplu de utilizare a serviciului WMS conţine ca parametru doar versiunea acestui standard:
http://adresaServer/service=WMS&request=GetCapabilities&version=1.1.1.

```
<Capability>
    <Layer queryable="1">
        <Name>preview:snow_classification</Name>
        <Title>Snow classification</Title>
        <SRS>EPSG:4326</SRS>
        <KeywordList>
            <Keyword>snow</Keyword>
            <Keyword>classification</Keyword>
        </KeywordList>
        <LatLonBoundingBox minx="-180.0" miny="-56.0" maxx="179.989"
                                maxy="84.011"/>
        <Style>
            <Name>snow_classification_style</Name>
            <Title>Raster</Title>
            <Format>image/jpg</Format>
        </Style>
    </Layer>
</Capability>
```

Figura 1. Rezultatul operaţiei GetCapabilities

Rezultatul obţinut de utilizatori, descrie în formatul XML metadatele asociate informaţiilor spaţiale. O versiune simplificată a acestora, ca urmare a folosirii exemplului anterior, este redată în Figura 1.

Fiecare bandă (layer) este reprezentată prin metadatele descrise în Figura 1. Se poate observa existenţa unui nume, a proiecţiei folosite (atributul *SRS*), a regiunii geografice reprezentate de această bandă (atributul *LatLonBoundingBox*) şi a formatelor disponibile (în cazul acestui exemplu se poate folosi doar formatul JPEG). Căutarea automată este posibilă datorită cuvintelor cheie, existente în cadrul etichetei *KeywordList*.

După analiza acestui document de metadate, utilizatorul are posibilitatea de a obţine efectiv datele de interes. Aceasta reprezintă a două fază a procesului WMS, fiind realizată prin intermediul operaţiei *GetMap*. Cei mai importanţi parametrii oferă utilizatorului posibilitatea de a specifica regiunea de interes prin coordonate geografice (*bbox*), setul de benzi pe care doreşte să le vizualizeze (*layers*), proiecţia (*SRS*), formatul imaginii rezultat (*format*) etc. Un posibil exemplu de utilizare este http://adresa_server/service=WMS&request=GetMap&version=1.1.1&bbox =20, 48,29,43&layers=preview:snow_classification&SRS=EPSG:4326& format=image /jpg.

Rezultatul obţinut este o imagine în formatul JPEG. Soluţia propusă în cadrul acestei lucrări pune accent pe îmbunătăţirea calităţii procesului de vizualizare a informaţiilor generate de serviciul WMS. Astfel, s-a ajuns la concluzia implementării unei funcţionalităţi care să permită suprapunerea directă a rezultatului *GetMap* peste diferite tipuri de hărţi interactive.

În urma acestei abordări s-a constatat o creştere a utilizabilităţii platformei GreenLand şi a gradului de satisfacere a utilizatorilor acestui sistem. Procesul de vizualizare este astfel mult îmbunătăţit, în comparaţie cu abordarea tradiţională, oferind posibilitatea de identificare directă a corespondenţei rezultatului cu regiunea geografică de interes. De asemenea, utilizatorul are posibilitatea de a realiza şi o analiză detaliată a rezultatelor, prin aplicarea operaţiilor de redimensionare şi repoziţionare interactivă a imaginii finale.

Serviciul WCS

La o primă vedere serviciul WCS (Web Coverage Service) este identic cu cel prezentat anterior (WMS), întrucât ambele sunt folosite pentru căutarea datelor spaţiale, analiza setului de metadate, obţinerea unor rezultate şi vizualizarea acestora în diferite moduri. Există totuşi şi diferenţe între cele două produse.

Prima deosebire constă în faptul că întregul proces de comunicare utilizator- WCS se desfăşoară în trei faze (WMS necesită numai două etape): *GetCapabilities*, *DescribeCoverage* şi *GetCoverage*. Prima şi ultima sunt similare cu operaţiile *GetCapabilities* şi *GetMap* descrise anterior. În schimb, cea de-a doua operaţie (*GetCoverage*) identifică banda de interes specificată de utilizator şi oferă detalii suplimentare despre aceasta, incluzând: algoritmii de interpolare ce pot fi folosiţi pentru prelucrări ulterioare, o listă de etichete obţinute în urma unor clasificări a regiunilor geografice etc.

Pentru fiecare dintre aceste operaţii, există o listă de parametri care se recomandă a fi folosiţi, dintre care se pot enumera: versiunea serviciului, coordonatele regiunii geografice, tipul proiecţiei, formatul rezultatului etc.

O a doua diferenţă dintre cele două produse, constă în faptul că primul serviciu poate fi folosit numai pentru vizualizarea datelor, întrucât rezultatele generate nu sunt georeferenţiate, ci sunt imagini în formatul JPEG, TIF sau PNG.

Există însă situaţii în care aceste date trebuie analizate din diferite puncte de vedere: clasificarea regiunilor de vegetaţie, generearea hărţilor tematice, identificarea zonelor de interes, extragerea informaţiilor relevante etc. În

astfel de cazuri se folosesc rezultatele generate de serviciile WCS, întrucât acestea conţin date georeferenţiate care pot fi prelucrate în vederea obţinerii efectelor dorite.

4. Implementare şi experimente

Soluţiile implementate în cadrul platformei GreenLand vizează un grup larg de utilizatori, din diferite domenii de activitate. Cele mai importante categorii includ specialiştii în domeniul calculatoarelor şi a Ştiinţei Pământului (cu rolul de a modela şi simula fenomenele naturale), persoane cu funcţii administrative (deţin puterea de elaborare de decizii, reguli şi politici, în funcţie de rezultatele obţinute de către specialişti), utilizatorii neiniţiaţi care folosesc serviciile GreenLand în scop educativ sau informaţional.

Este important de menţionat faptul că prin tehnicile de interacţiune oferite de această platformă s-a acoperit marea majoritate a necesităţilor acestor grupuri de utilizatori. Cu alte cuvinte, acelaşi tip de interacţiune poate fi folosit pentru deservirea operaţiilor uzuale, dar în acelaşi timp facilitează şi acţiunile-utilizator complexe. Drept dovadă stau cele trei studii de caz (Bektaş et al., 2012) la dezvoltarea cărora au participat activ persoane care fac parte din categoriile menţionate anterior.

Complexitatea şi diversitatea imaginilor satelitare reprezintă o problemă importantă în vederea colectării unui volum mare de date. Achiziţionarea manuală este fezabilă doar în cazul scenariilor de utilizare care necesită un număr redus de date de intrare, în restul situaţiilor fiind nevoie de automatizarea acestui proces. În acest sens standardul OGC propune o metodă comună de organizare a datelor, indiferent de tipul acestora şi de locaţia fizică a lor.

Platforma GreenLand oferă interoperabilitate la nivel de informaţii cu diferite sisteme externe, prin implementarea serviciilor WMS şi WCS de căutare, vizualizare şi analiză a datelor spaţiale. Complexitatea schimbului de date, este mascată prin intermediul tehnicilor de interacţiune, care au scopul de a simplifica acţiunile utilizator efectuate la nivelul interfeţei grafice. Printre aceste tehnici se pot enumera:

- inter-conectarea serviciilor WMS şi WCS cu hărţi interactive, fiind posibilă afişarea şi analiza dinamică a informaţiilor;

- căutarea automată a informaţiilor pe baza criteriilor specificate de utilizatori. Are loc astfel automatizarea procesului de descoperire a datelor spaţiale;
- extragerea informaţiilor relevante folosind metode interactive de specificare a regiunii de interes. Sistemul foloseşte serviciile WCS pentru a realiza cererea descrisă de utilizatori la nivelul interfeţei grafice;
- reutilizarea datelor, întrucât din aceeiaşi imagine satelitară pot fi extrase diferite informaţii.

4.1 Arhitectura de vizualizare şi colectare a datelor spaţiale

Integrarea serviciilor OGC în cadrul platformei GreenLand, presupune existenţa unei arhitecturi de comunicare, bazată pe protocolul cerere-răspuns. Figura 2 prezintă o astfel de comunicare realizată între sistemul GreenLand şi diferite depozite de date spaţiale, a căror informaţii sunt expuse conform standardelor OGC. Din arhitectura propusă în Figura 2, se pot distinge patru faze ale acestui proces: stabilirea unui canal de comunicare între cele două entităţi, căutarea şi vizualizarea informaţiilor de interes şi posibilitatea de colectare a datelor relevante.

4.2 Accesul la date

Platforma GreenLand oferă utilizatorilor un număr mare de algoritmi folosiţi în scopul prelucrării datelor spaţiale. Există implicit o bază de date comună, de unde se pot specifica intrări pentru diferite instanţe ale algoritmilor. Accesul la aceste informaţii se realizează prin autentificarea utilizatorilor la nivel de aplicaţie.

Există situaţii în care execuţia unui astfel de algoritm presupune folosirea altor tipuri de date, care fac parte din depozite localizate în afara sistemului, a căror conţinut este expus prin intermediul metadatelor. Întreţinerea şi actualizarea periodică a acestor depozite nu intră în atribuţiile utilizatorilor platformei GreenLand. Acesta este un prim avantaj al folosirii unor astfel de date, în raport cu utilizarea resurselor de pe calculatorul propriu.

Simplificarea acţiunilor-utilizator de la nivelul interfeţei grafice, prin mascarea complexităţii conexiunilor cu depozitele de date, reprezintă un al doilea avantaj al folosirii serviciilor OGC.

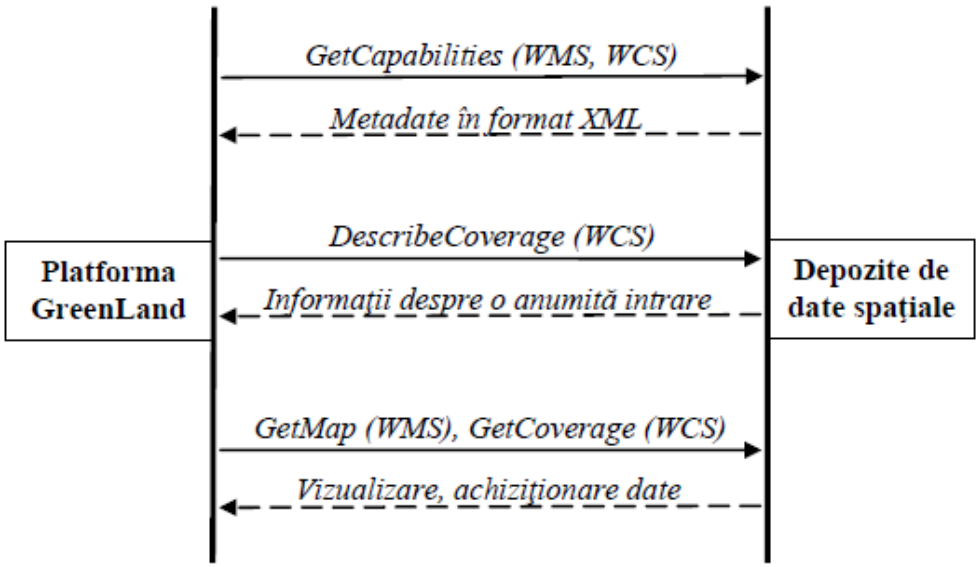


Figura 2. Arhitectura de vizualizare şi achiziţionare a datelor spaţiale, folosind serviciile WMS şi WCS

În acest sens, adresa Web a depozitului şi folosirea mecanismului WMS de conectare oferit de platforma GreenLand sunt suficiente pentru a stabili un canal de comunicare între aceste două entităţi.

Deşi în cadrul experimentelor au fost verificate conexiunile cu mai multe depozite de date, pentru lucrarea de faţă a fost ales depozitul URM (Bektaş et al., 2012). Scopul acestui experiment este de a valida implementarea serviciilor OGC, folosind tehnici de interacţiune utilizator specifice pentru vizualizarea dinamică a datelor spaţiale şi colectarea interactivă a informaţiilor de interes pentru utilizatori.

4.3 Căutarea datelor de interes

Să presupunem că un utilizator al platformei GreenLand doreşte să execute un algoritm de clasificare a zonelor de vegetaţie, apă şi locaţii urbane din cadrul regiunii Istanbul. Etapa de specificare a datelor de intrare constă în căutarea imaginilor satelitare care corespund cerinţelor non-funcţionale, urmând apoi o analiză amănunţită a acestora.

După stabilirea conexiunii GreenLand-URM, utilizatorul are posibilitatea de a interoga conţinutul acestui depozit de date, în vederea descoperirii

informaţiilor de interes pentru realizarea experimentului. În acest scop se foloseşte operaţia *GetCapabilities* (Figura 2), existentă în cadrul serviciului WMS. Ca şi răspuns, serverul URM transmite un fişier XML, care descrie în termeni generali informaţiile pe care le conţine.

Fiecare intrare indexată, din depozitul de date, este descrisă ca in Figura 1. Analiza manuală a unui document XML este un proces relativ simplu, atâta timp cât şi informaţiile prezentate au o structură mai puţin complexă. Însă în majoritatea cazurilor, volumul şi structura datelor spaţiale contribuie la generarea unui document XML de dimensiuni mari, greu de analizat de către utilizatori.

Astfel, este necesară optimizarea procesului de afişare a informaţiilor, folosind o metodă similară cu cea propusă de Grigoriu şi Buraga (2011). Soluţia abordată reduce numărul de acţiuni-utilizator necesare pentru realizarea operaţiei de căutare, simplificând în acelaşi timp şi procedura de identificare a datelor de interes.

Drept urmare, în cadrul platformei GreenLand a fost implementat un mecanism de interpretare automată a acestor metadate. Rezultatul produs constă în afişarea sub formă de listă interactivă, la nivelul interfeţei grafice a aplicaţiei, a denumirii intrărilor descrise în fişierul XML. Prin selectarea unei astfel de intrări, sistemul oferă detalii suplimentare despre: locaţia

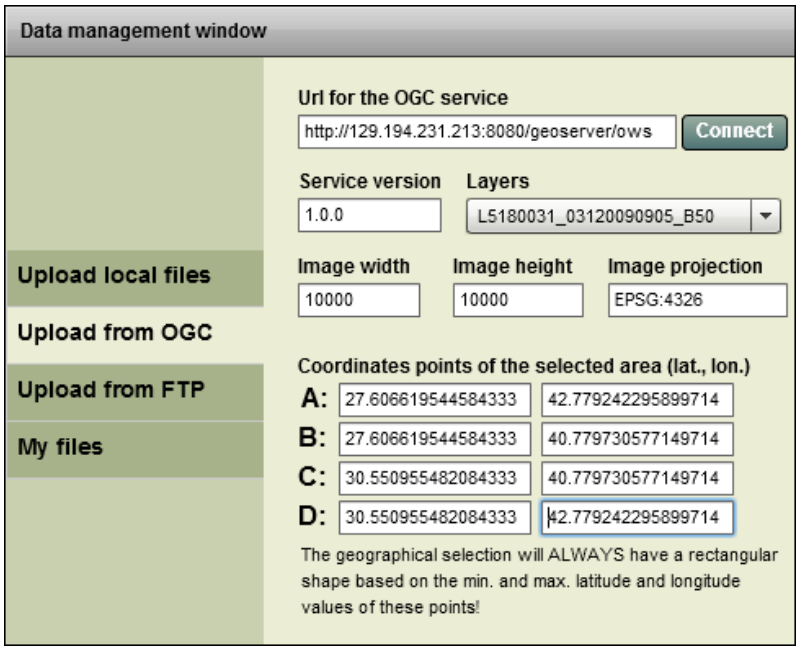


Figura 3. Interfaţa grafică pentru afişarea metadatelor obţinute în urma invocării operaţiei GetCapabilities

geografică (exprimată prin coordonate latitudinale şi longitudinale), tipul proiecţiei folosite, dimensiunea în pixeli a întregii suprafeţe reprezentate de aceea intrare etc.

Figura 3 prezintă modul de afişare a caracteristicilor unei imagini satelitare din regiunea Istanbul. S-a avut în vedere creşterea gradului de utilizabilitate a sistemului, prin organizarea pe categorii a componentelor grafice şi reducerea afişării informaţiilor irelevante. Din această manieră de reprezentare a informaţiilor, utilizatorul are posiblitatea de selectare a datelor de interes.

În cazul experimentelor (algoritmilor) care necesită un volum mare de date de intrare, această metodă de căutare este ineficientă, datorită faptului că necesită ca utilizatorul să parcurgă manual fiecare intrare. Soluţia propusă, în cadrul platformei GreenLand, constă în implementarea unui mecanism de automatizare a procesului de căutare, pe baza listei de cuvinte cheie, a descrierii şi a coordonatelor geografice ataşate fiecărei intrări.

Au fost efectuate în acest sens o serie de experimente, în care utilizatorul specifică informaţii referitoare la regiunea geografică de interes, cuvinte conţinute în câmpurile de denumire sau descriere etc. Acestea sunt apoi interpretate de către sistem şi aplicate ca şi criterii de căutare asupra metadatelor XML generate în urma invocării operaţiei *GetCapabilities*.

Rezultatul obţinut este prezentat utilizatorilor într-o manieră similară cu cea din Figura 3. Diferenţa constă în faptul că lista *Layers* va conţine în primul caz toate intrările din fişierul de metadate XML, în timp ce în cazul căutării automate, lista va fi populată doar cu rezultatele care corespund criteriilor specificate de utilizatori.

Informaţiile prezentate în Figura 3 sunt obţinute prin interpretarea rezultatului generat de operaţia *GetCapabilities*, existentă în cadrul serviciului WMS. Pentru majoritatea utilizatorilor platformei GreenLand, aceste informaţii sunt suficiente pentru identificarea datelor de interes.

Există situaţii în care utilizatori specializşti în domeniul Ştiinţa Pământului, au nevoie de detalii suplimentare despre o anumită intrare. Pentru aceste cazuri se poate folosi serviciul WCS, împreună cu operaţia *DescribeCoverage* (Figura 2), care pe baza identificatorului de nume, oferă informaţii precum: algoritmii de interpolare ce pot fi folosiţi pentru o analiză mai detaliată a intrării, o listă de etichete obţinute în urma unor clasificări a regiunilor geografice etc.

Atât WMS cât şi WCS pot fi folosite pentru căutarea datelor de interes. Odată identificate, acestea pot fi vizualizate prin utilizarea primului serviciu,

sau pot fi supuse unor prelucrări suplimentare prin folosirea serviciilor WCS, eventual în combinaţie cu WMS.

4.4 Vizualizarea datelor

Se realizează prin intermediul operaţiei *GetMap* (Figura 2), existentă în cadrul serviciilor WMS. Deşi rezultatul generat este disponibil în mai multe formate (ex.: JPEG, TIF sau PNG), soluţia adoptată în cadrul platformei GreenLand face referire doar la formatul JPEG, datorită faptului că raportul calitate/rată de transfer este superior în cazul imaginilor de asemenea dimensiuni.

Algoritmii din domeniul Ştiinţa Pământului, existenţi în cadrul sistemului, necesită date de intrare referenţiate geografic. Trebuie remarcat faptul că în urma utilizării operaţiei *GetMap,* utilizatorul nu are acces la datele spaţiale reale, întrucât rezultatul generat nu conţine informaţii georeferenţiate. Acesta este principalul motiv pentru care serviciile WMS sunt folosite numai pentru vizualizarea datelor, nu şi pentru utilizarea lor în prelucrări ulterioare.

Există două modalităţi de vizualizare a datelor WMS: afişarea directă a rezultatului JPEG prin intemediul editoarelor de imagini externe sau folosirea hărţilor interactive peste care se suprapune rezultatul obţinut anterior. Soluţia implementată în cadrul platformei, foloseşte a doua metodă, simplificând astfel întregul proces de vizualizare, deoarece utilizatorul nu mai este nevoit să descarce imaginea pe calculatorul personal şi să o vizualizeze prin intermediul unui editor de imagini extern platformei GreenLand.

Vizualizarea imaginilor satelitare este strâns legată de regiunea geografică reprezentată prin aceste date. Drept urmare, tehnica de interacţiune bazată pe folosirea hărţilor interactive, constituie un punct esenţial în ceea ce priveşte creşterea utilizabilităţii platformei GreenLand şi a gradului de satisfacere a utilizatorilor acestui sistem.

Procesul de vizualizare este astfel mult îmbunătăţit, în comparaţie cu abordarea tradiţională, facilitând identificarea directă a corespondenţei rezultatului cu regiunea geografică de interes.

Utilizatorii au posibilitatea de a realiza şi o analiză detaliată a rezultatelor, prin aplicarea operaţiilor de redimensionare şi repoziţionare interactivă a imaginii finale. Redimensionarea câmpului de vizualizare depinde în mare măsură de disponibilitatea datelor, la nivel de hartă interactivă, pentru zona geografică reprezentată de imaginea satelitară.

A fost folosită harta interactivă pusă la dispoziţie de Esri (Esri Maps, 2012), care oferă diferite facilităţi pentru afişarea imaginilor obţinute în urma prelucrării datelor spaţiale. Folosirea indicelui de vegetaţie Normalized Difference Vegetation Index (NDVI) pentru clasificarea regiunilor terestre din jurul oraşului Istanbul, reprezintă un exemplu concludent în ceea ce priveşte vizualizarea interactivă a datelor spaţiale (Figura 4).

Semnificaţia culorilor folosite este următoarea: verde (pentru zonele de vegetaţie), albastru (reprezintă suprafeţele de apă) şi roşu (identifică zonele urbane, ocupate de clădiri). Conturul negru care apare în jurul imaginii rezultat, reprezintă regiunile care nu au putut fi clasificate, datorită faptului că setul original de date nu conţine valori valide pentru acestea.

Se poate observa că metoda de vizualizare este strâns legată de lista de intrări (layere) generate în urma folosirii operaţiei *GetCapabilities*. Pentru afişarea informaţiilor pe harta interactivă, este nevoie de un număr redus de acţiuni-utilizator. În primul pas se realizează conexiunea cu depozitul de date (URM în cazul de faţă) prin specificarea adresei Web, urmând ca apoi sistemul să invoce operaţiile OGC corespunzătoare, să interpreteze fişierul de metadate XML şi să îl afişeze sub formă de listă (Figura 4).

A doua acţiune-utilizator necesită selectarea unei intrări din lista *Layers*, în urma căreia platforma GreenLand suprapune imaginea (obţinută prin operaţia *GetMap*) peste harta interactivă. Manipularea prin folosirea dispozitivului mouse, redimensionarea şi deplasarea imaginii sunt operaţii posibile în acest context.

Un studiu de caz interesant îl constituie folosirea realităţii îmbogăţite pentru afişarea şi vizualizarea acestor informaţii. Se pot folosi, ca punct de referinţă, soluţiile descrise de Dârdală et al. (2011).

4.5 Colectarea datelor

O altă funcţionalitate, dezvoltată în urma integrării serviciilor OGC în cadrul platformei GreenLand, este reprezentată de posibilitatea de extragere (colectare) a datelor spaţiale. În acest sens se foloseşte operaţia *GetCoverage* (Figura 4), existentă ca metodă WCS, care generează ca şi rezultat o imagine georeferenţiată, asupra căreia pot fi aplicaţi diferiţi algoritmi de clasificare a vegetaţiei dintr-o anumită regiune geografică, de generare a hărţilor tematice, de modelare hidrologică etc.

Extragerea unui subset din datele originale, poate fi descris ca un proces interactiv, din punct de vedere al acţiunilor-utilizator. La fel ca şi în cazul

manipulării rezultatelor WMS, imaginea georeferenţiată poate fi vizualizată dinamic, prin suprapunerea acesteia peste harta interactivă.

Specificarea regiunii de interes are tot timpul o formă dreptunghiulară (ABCD) şi se realizează prin acţionarea dispozitivului mouse, în acest fel:

- se face clic pe hartă pentru stabilirea coordonatelor geografice ale punctului A;
- ţinând apăsat butonul stânga al dispozitivului mouse, poziţia punctului D urmează traiectoria descrisă prin mişcarea dispozitivului. La fiecare pas sistemul determină şi afişează automat şi valorile punctelor B şi C (Figura 4);
- eliberarea butonului mouse coincide cu specificarea regiunii din imagine pentru care utilizatorul doreşte extragerea de informaţii;
- după parcurgerea acestor trei paşi, zona de interes este marcată de un dreptunghi de culoare verde (Figura 4), care poate fi repoziţionat sau redimensionat.

  Extragerea datelor relevante presupune selecţia unei regiuni interne imaginii afişate. În cazul în care anumite părţi din selecţia dreptunghiulară specificată de utilizator sunt în afara imaginii georeferenţiate, sistemul face ajustările corespunzătoare, astfel încât selecţia rezultată să fie validă.

Abordarea anterioară este utilă în special în situaţiile în care nu se doreşte marcarea precisă a regiunii geografice de interes. Pentru utilizatorii specialişti, care efectuează studii cu grad înalt de acurateţe, platforma GreenLand oferă alternativa introducerii manuale a acestor informaţii (Figura 4) prin intermediul căsuţelor text. Acestea suportă valori reale, care corespund celor patru puncte ale suprafeţei marcate pe hartă.

Trebuie remarcat faptul că există o corespondenţă între cele două metode. Astfel, la acţiunea de selectare a regiunii folosind dispozitivul maus, sistemul reproduce valorile, indicate de utilizator, în căsuţele text corespunzătoare. Mecanismul invers este de asemenea valabil, în final suprafaţa marcată de utilizator având forma dreptunghiulară.

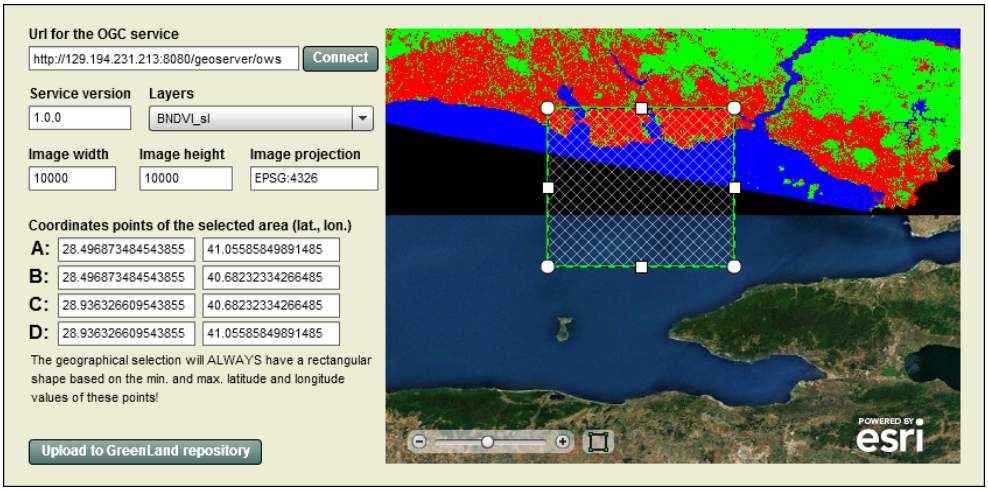


Figura 4. Vizualizarea interactivă a indicelului de vegetaţie NDVI

Odată ce acţiunea de selecţie a regiunii geografice a fost finalizată de către utilizator, sistemul preia automat informaţiile introduse, realizând achiziţia datelor de interes. Pe lângă transferul propriu-zis al datelor, are loc şi reproiecţia imaginii originale, conform specificaţiilor din interfaţa grafică. Existenţa aceluiaşi rezultat în diferite proiecţii, facilitează studiul aceluiaşi fenomen, aplicat în regiuni geografice diferite. Acesta este motivul principal pentru care platforma GreenLand pune la dispoziţie opţiunea de reproiecţie, având în vedere faptul că sistemul se adresează mai multor categorii de utilizatori.

Coordonatele geografice specificate anterior sunt transmise ca şi parametrii operaţiei *GetCoverage*, care are ca efect extragerea informaţiilor de interes din setul original de date. Odată ce rezultatul este disponibil în cadrul platformei GreenLand, poate fi folosit ca intrare pentru algoritmii de prelucrare a datelor spaţiale.

5. Mulţumiri

Această lucrare a beneficiat de suport financiar prin proiectul ”Creşterea calităţii studiilor doctorale în ştiinţe inginereşti pentru sprijinirea dezvoltării societăţii bazate pe cunoaştere”, contract: POSDRU/107/1.5/S/78534, proiect cofinanţat din Fondul Social European prin Programul Operaţional Sectorial Dezvoltarea Resurselor Umane 2007-2013, precum şi prin

proiectul FP7 enviroGRIDS, cofinanţat de Comisia Europeană, prin contractul 226740.

6. Concluzii

Accesul, prelucrarea si vizualizarea datelor spaţiale masive necesită gestionarea unor resurse şi infrastructuri complexe. De asemenea, coordonarea prelucrarilor paralele şi distribuite sau vizualizarea stării proceselor şi a rezultatelor sunt sarcini greu abordabile majorităţii utilizatorilor.

Lucrarea evidenţiază experimentele bazate pe servicii OGC, prin care se dezvoltă şi evaluează în interfaţa utilizator grafică a aplicaţiei GreenLand, tehnici de interacţiune utilizator cu o înaltă utilizabilitate. Aceste experimente au dus la implementarea unor tehnici de interacţiune care au ca principal obiectiv simplificarea acţiunilor-utilizator şi reducerea timpului necesar pentru efectuarea sarcinilor de lucru.

În lucrare se prezintă conceptele teoretice şi practice care au stat la baza implementării standardelor OGC, sub formă de operaţii disponibile la nivelul interfeţei grafice a platformei GreenLand. Folosirea acestor servicii poate fi motivată prin oferirea unor metode comune de acces, căutare, vizualizare şi extragere a imaginilor satelitare existente în depozite de date localizate la distanţă.

Cele mai importante soluţii şi metode descrise pe parcursul acestei lucrări, fac referire la avantajele obţinute, din punctul de vedere al utilizatorilor, în raport cu abordările tradiţionale. Printre acestea se pot menţiona: căutarea automată a datelor spaţiale înmagazinate în diferite depozite de date, vizualizarea dinamică a imaginilor satelitare prin suprapunerea peste hărţi interactive, optimizarea operaţiei de extragere a datelor relevante prin specificarea interactivă a regiunii geografice de interes şi a parametrilor corespunzători procesului.

Referinţe

Aloisio, G., Cafaro, M. (2003) A Dynamic Earth Observation System. Parallel Computing, 29, 1357-1362.

Athan, T., Dassau, O., Ghisla, A. (2013) Quantum GIS 1.8.0 Geographic Information System User Guide. Open Source Geospatial Foundation,

http://docs.qgis.org/pdf/QGIS-1.8-UserGuide-en.pdf

Bektaş, B.F., Goksel, C., Sozen, S., Allenbach, K., Gvilava, M., Rahman, K., Gorgan, D.,

Mihon, D. (2012) Remote Sensing Services – ESIP Platform and Hot Spot Inventory Case Studies. enviroGRIDS Deliverable D2.11, http://envirogrids.net/index.php?option=com_jdownloads&Itemid=13&view=finish&cid=139&catid=11

Chen, Z., Chen, N., Yang, C., Di, L. (2012) Cloud Computing Enabled Web Processing Service for Earth Observation Data Processing. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 5, 1637-1649.

Dârdală, M., Reveiu, A., Felix, F.T. (2011) Model de realitate virtuală integrat în sisteme informatice geografice. Revista Română de Interacţiune Om-Calculator, 4(număr special), 115-118.

Descrierea produselor ASTER (2013),

http://www.satimagingcorp.com/satellitesensors/aster.html

Descrierea produselor Landsat (2013),

http://landsat.gsfc.nasa.gov/

Descrierea produselor SPOT (2013),

http://www.satimagingcorp.com/satellite-sensors/SPOT-5 Resolution Spectral Modes.pdf

Esri Maps (2012) An Esri White Paper,

http://www.esri.com/library/whitepapers/pdfs/esri-maps-for-ibm-cognos.pdf

Giuliani, G., Nativi, S., Lehmann, A., Ray, N. (2011) WPS Mediation: an Approach to Process Geospatial Data on Different Computing Backends. Computers and Geosciences, 47, 20-33.

Gorgan, D., Bacu, V., Stefanut, T., Rodila, D., Mihon, D. (2012) Earth Observation Application Development based on the Grid Oriented ESIP Satellite Image Processing Platform. Journal of Computer Standards and Interfaces, 34(6), 541-548.

Gorgan, D., Bacu, V., Stefanut, T., Rodila, D., Mihon, D. (2009) Grid Based Satellite Image Processing Platform for Earth Observation Application Development. Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 247-252.

Grigoriu, A., Buraga, S. C. (2011) Studii asupra vizualizării datelor şi infograficelor. Revista Română de Interacţiune Om-Calculator, 4(număr special), 25-28.

Mihon, D., Colceriu, V., Minculescu, A., Bacu, V., Gorgan, D. (2012) Descrierea diagramatică a prelucrării imaginilor satelitare în aplicaţia GreenLand. Revista Română de Interacţiune Om-Calculator, 5(2), 129-134.

Ramsey, P. (2004) User Friendly Desktop Internet GIS (uDIG) for OpenGIS Spatial Data Infrastructures. Refractions Research Inc.

Reeves, M., Zhao, M., Running, S. (2006) Applying Improved Estimates of MODIS Productivity of Characterize Grassland Vegetation Dynamics. Rangeland Ecology and Management, 59, 1-10.

Sextante (2011) A Versatile Open-Source Library for Spatial Data Analysis,

http://gvsigce.sourceforge.net/sextante web/pdf/SextantePaper.pdf

Standardul OGC (2004) Integrating Geospatial Standards and Standards Strategies into Business Process. OGC White Paper,

http://portal.opengeospatial.org/files/?artifact_id=5098&version=2&format=pdf

Standardul OGC (2007) OpenGIS Web Service Common Implementation Specification,

http://www.opengeospatial.org/standards/common

Wang, L., Sousa, W. P., Gong, P., Biging, G. S. (2004) Comparison of IKONOS and QuickBird Images for Mapping Mangrove Species on the Caribbean Coast of Panama. Remote Sensing of Environment, 432-440.

Wang, Y., Wang, S., Zhou, D. (2009) Retrieving and Indexing Spatial Data in the Cloud Computing Environment. 1st International Conference on Cloud Computing, 322-331

Proceedings of the

International Conference on Complex, Intelligent and Software Intensive Systems

15-18 February 2010 / Krakow, Poland

Los Alamitos, California

Washington • Tokyo

ADiS 2010 - First International Workshop on Autonomic Distributed Systems

S1: ADiS-2010 Session 1

S2: ADiS-2010 Session 2

ECDS 2010 - Fourth Workshop on Engineering Complex Distributed Systems

S1: Managing Complexity

Satellite Image Interpolation and Analysis through GreenView Application

Danut Mihon[1], Victor Bacu[1], Róbert Mészáros[2], Györgyi Gelybó[2], Dorian Gorgan[1]
[1]*Computer Science Department, Technical University of Cluj-Napoca*
[2]*Eötvös Loránd University, Department of Meteorology*
{vasile.mihon, victor.bacu, dorian.gorgan}@cs.utcluj.ro,
mrobi@nimbus.elte.hu, gyoresz@caesar.elte.hu

***Abstract*— Earth Science data and particularly the satellite images supplies important information on status and dynamics of earth covering vegetation. The development of environment applications is a challenge for Grid technologies and software development methodologies. This paper exemplifies and experiments the development of the GreenView application by the Grid Application Development Methodology. The paper highlights through the ESIP and gProcess platforms based methodology the steps of the GreenView implementation. The application reveals two main functionalities – refinement of the satellite image resolution, and the field measurements based calibration of the satellite image. The paper presents as well the experiments on the GreenView execution over the See-Grid infrastructure.**

I. INTRODUCTION

The aim of the GreenView applications is a refinement of surface and vegetation parameters in SEE region based on satellite images and field measurements. In the frame of GreenView applications, construction, usage and comparison of diverse satellite datasets are performed. High resolution satellite measurements can be used for numerous environmental studies (climate-related, surface-atmosphere exchange or air pollution modeling). These environmental studies require detailed input fields. Satellite data could be appropriate sources to create these input datasets for the regional models.

Environmental data have to be regridded to the grid of the model for which satellite data serve as input. This application is to make a satellite dataset ready to use by performing pre-processing of the original dataset according to the needs of the end-user. This includes the following steps: regridding the data to an arbitrarily chosen grid defined by the user and providing data series for a period chosen by the user, in time steps of the original data or averaged for a certain period.

In the frame of another GreenView application the calibration of BIOME-BGC ecosystem model has been performed. Carbon balance components e.g. gross primary productivity (GPP) is of great interest in satellite remote sensing. The MOD17 algorithm, the GPP estimation model developed for MODIS (Moderate Resolution Imaging Spectroradiometer) sensor onboard Terra and Aqua satellites is partially based on BIOME-BGC model. It is of great importance to use and calibrate the model based on the flux tower measurements. Once the BIOME-BGC model is calibrated, MOD17 can also be calibrated and extended to regional scale in order to obtain more accurate estimation of the carbon balance components.

The GreenView application is developed by using the ESIP platform [1]. The Environment oriented Satellite Data Processing Platform (ESIP) has been developed through the SEE-GRID-SCI (SEE-GRID eInfrastructure for regional eScience) [2].

II. RELATED WORKS

Because more and more applications process large amount of data, parallel distributed computing system for information extraction and processing were developed. Projects like G-POD [3] and GENESI-DR [4] allow manually and automatic data secure access through web services.

Complex Grid applications are using workflows for the algorithms description. Several solutions are available, mostly of them are using DAGs (direct acyclic graph representation). Pegasus [5], Taverna [6] or GridFlow [7] are using a DAG description for defining Grid workflow. Taverna uses a XML based description language for defining workflows, called SCUFL (Simple Conceptual Unified Flow Language) [8]. It allows iterations on datasets based on some specific strategies. In GridFlow the workflows are represented as a flow of activities. The workflow management is based on a hierarchical scheduling system.

III. ALGORITHM DESCRIPTION

The GreenView application offers three different functionalities: *coarse to fine interpolation*, *fine to coarse interpolation,* and *GPP computing*. Each of these functionalities implements one or more algorithms. These algorithms are presented further into this section. The applications can be useful for environmental or climate-change related studies among others.

A. Coarse to fine interpolation

The first functionality of the application is the '*coarse to fine interpolation'*. It is necessary to have data fields of different environmental and biophysical variables on the same grid in order to efficiently use them in complex studies

or models. This common grid is chosen based on the application. This procedure performs spatial interpolation of a coarsely gridded dataset to the (finer) target grid. There is a variety of interpolation methods available in the literature developed specifically for certain applications. In order to choose the most appropriate one, issues like precision, computation resources needed on a specific data set, parallelization possibilities and others must be considered.

As the main goal of the process is to obtain environmental variables on the satellite grid, a nonlinear cosine function based distance weighted interpolation [9] has been chosen. In this study for demonstration purposes, we used the 0.5°×0.5° resolution climate dataset of the PRUDENCE (*P*rediction of *R*egional scenarios and *U*ncertainties for *D*efining *E*uropea*n* *C*limate change risks and *E*ffects) project as coarsely gridded data, supposing a hypothetic climate change impact study performed by the end-user. The target grid is the 1 km resolution MOD17 grid. At the end of the analysis phase, the algorithm should be described (in this case) as mathematical formulas.

B. Fine to coarse interpolation

This functionality from the GreenView application is the opposite of the *coarse to fine* method, meaning that its purpose is to obtain a data field with a lower resolution from one with higher resolution. In this case a simple pixel averaging routine is performed to retrieve the given variable (e.g. average, minimum, maximum etc.) on the target grid. The given values of fine grid are calculated to the corresponding coarse grid cells. The value in coarse grid cell is calculated from values of fine grid cells, where the centre of fine grid cells are inside the coarse grid cell.

C. Ecosystem model calibration

Third functionality of the GreenView application is a calibration of an ecosystem model which estimates the productivity of plants. This term can be well described by gross primary production (GPP, the total amount of carbon taken up by vegetation via photosynthesis during a specific time interval). GPP is measured several places worldwide using the so-called eddy covariance technique. In this study, the measurements at Hegyhátsál, Hungary have been used [10]. This micrometeorological method provides information about the carbon cycle of the vegetation in a small spatial scale (orders of a few square kilometers). The MOD17 product that is derived from the data of the MODIS sensor onboard satellites Terra and Aqua provides an estimate for the gross photosynthesis of terrestrial vegetation at 1 km spatial resolution. The tall tower based direct (eddy covariance) measurements provide an attractive method for the calibration of the official MOD17 (GPP) data product. Besides GPP, MOD17 also provides information about net primary production (NPP). As the official MOD17 algorithm [11] is based on the BIOME-BGC model it is of great importance to use and calibrate the BIOME-BGC model based on the tall tower measurements.

During the calibration procedure the algorithm needs to generate a large number of datasets. For every dataset, a merit function is computed obtaining a numerical value that shows how good the calibration is. After a number of N datasets are generated, we end up with N values for the merit function, from which the minimum value is selected. This minimum value represents the best calibration that can be done by using those generated datasets.

Computing GPP value is a difficult thing because it implies the usage of a mathematical formula, of an already defined model (BIOME-BGC model) and the data obtained must comply with some restrictions imposed by the model. The GPP computation is based on measured data and obtained or simulated data (using BIOME-BGC model). The entire GPP computing algorithm is divided into four steps:

1. Generation of a large number of random value sets
2. Simulation of the BIOME-BGC model
3. Comparison between simulated and ground measured data for GPP parameter
4. Selection of a single generated value set, based on merit function.

The aim of the third functionality of the application is to generate a pseudo colored map, in which every pixel represents a GPP value. In order to do this, first we need to generate a large number of random values for some specific parameters. These parameters are contained in files and for each one is specified an interval. Generating a random value for a parameter must be done in the specified parameter's interval and that value must comply with the restrictions, otherwise that value is not good and therefore a new one must be generated. The application generates random values for 33 parameters that are organized in three types of files.

The simulation of BIOME-BGC model is using the random generated values as input data set. After the simulation is complete, a text file is generated (gpp.txt) that contains the values for GPP (Gross Primary Production) parameter over the Hegyhátsál region. After a large number of set values are generated, only one set is selected: the one that minimizes the merit function. The BIOME-BGC model is an already defined model, written in C programming language, which is used to generate the GPP values based on sets of random values.

D. Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

IV. GRID APPLICATION DEVELOPMENT METHODOLOGY

The ESIP platform uses a development methodology based on incrementally defining and developing components. The methodology steps used to define and develop Grid application are presented in the following paragraphs. This methodology is used to develop the GreenView application.

A. *Algorithm identification and analysis*

Specialists in the application domain field must conduct the first step of the methodology, as well as specialists in mathematics and informatics. This first step establishes the problem domain and identifies possible solutions. Different criteria, like parallelization capability, performance, resources are used in the algorithm selection process. It is an important step because the selected algorithms will have a great impact on the overall system performance.

B. *Data model definition*

After identifying a solution, or an algorithm we must identify the data that will be used. In the environmental applications, possible input data are satellite images, environmental data, like temperature, or different field measurements. The data management and storage solution is also important for the algorithm performance.

C. *Identify atomic parts of the computation*

In order to gridify the algorithm we must identify the atomic parts. These atomic parts are later on developed as executable applications, or web services. They can be also already developed and can be available by different libraries. We can choose different strategies for gridifying the application. One solution is to divide the algorithm in subcomponents that will represent the job execution unit, or we can parallelize the application by applying the same algorithm on multiple datasets.

D. *Algorithm implementation*

Grid or Web services, procedures, separate applications can be used to implement the atomic components. In order to improve the performance different programming languages can be chosen to implement the algorithms, for example C++ or Java.

E. *ESIP based process description*

The ESIP platform uses process description graphs to define processing algorithms. A process description graph (PDG) is represented like a DAG, direct acyclic graph, by composing basic operators, services and sub-graphs. The operators are representing the atomic components. Some components can be already developed as Grid or Web services, and by combining this functionality in the processing graph, we can benefit of reusability of code. In some cases, a subset of a graph can be used in many other parts. We can create a hyper-graph by combining sub-graphs. These sub-graphs, at the execution step, are expanded, and the basic operators or services are executed.

The PDG represents a pattern; it does not specify the particular inputs. We are specifying only the input type, for example satellite image, or a float value, but not the concrete value or the physical file. By attaching input resources to the PDG, we obtain an IPDG (Instantiated Process Description Graph). Only the IPDGs can be executed.

Different components, editor, manager, executor, that have a web service interface and a user interaction component, make part of the ESIP platform. The web services are used to expose different functionality and the interaction components are used to build up the user interface. The *EditorWS* provides information about available resources (e.g. lists of operations, sub-graphs, services, satellite image types, data types etc). Information about workflows (i.e. PDG, IPDG) and fetching and uploading resources (e.g. workflow, operators, services, sub-graphs, data) are retrieved by the *ManagerWS*.

The *ExecutorWS* executes instantiated workflows (IPDGs) over the Grid. Another feature of the ExecutorWS is the monitoring of workflows execution.

The *EditorIC* supports the user's editing operations for workflow development. The *ManagerIC* instantiates workflows for particular satellite data and manages the model resources (e.g. operators, services, sub-graphs, satellite data). The *ViewerIC* displays the input and output data (e.g. initial and processed satellite images) on the client site, and gets and displays the monitoring data.

F. *User interface development*

In this last step is built up the GUI and mapped the user

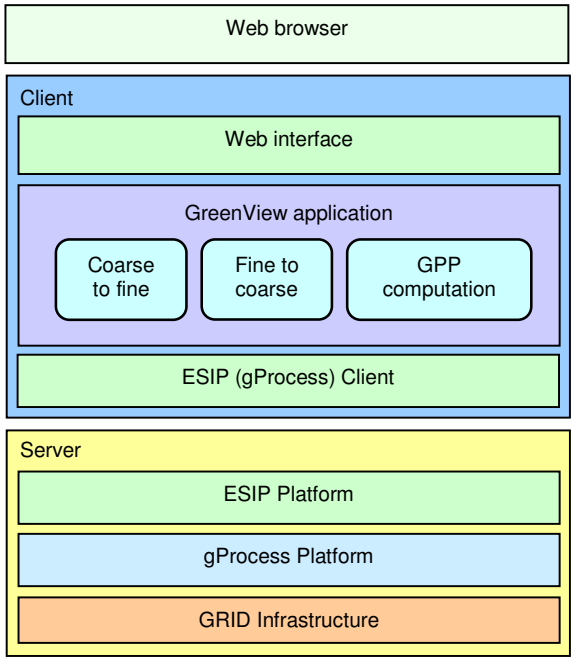


Fig. 1. GreenView related architecture.

interaction techniques over the Grid computation components.

V. GREENVIEW ARCHITECTURAL COMPONENTS

The GreenView is a client-server application that is accessible through a web browser. The GreenView is divided into three major components (Figure 1): Coarse to fine, Fine to coarse, and GPP computing. The interface is personalized for each component, by using different sets of buttons, text inputs, combo boxes and other combined tools. On the client side gProcess offers the possibility to describe and instantiate the workflow of a process execution on the Grid. ESIP is based on gProcess platform, which runs on the server and it is used to plan and manage the job execution over the Grid infrastructure. Therefore, the input resources are fetched to the Grid worker node, and after that the jobs are executed. At specific time intervals, the job status is retrieved and the server database is updated according to those changes. When the job finishes, the results are displayed to the user, through the interface.

VI. GREENVIEW USER INTERFACE

The GreenView user interface offers an easy access to the backhand functionality. The entire interface is created by using Flex 3.3, a free Adobe technology, along with BlaseDS, that connects the server functionality with the user's needs. The most important advantages of Adobe Flex are: free technology, the use of MXML for GUI development and action script for the backhand interactivity, predefined components, such as: buttons, input text, data grid, etc., platform independent, the application is converted to swf file that runs in Flash Player.

The GreenView application is divided into four different components, coarse to fine, fine to coarse, calibration and user manual.

A. *Coarse to fine interpolation component*

In order to launch a new executing process on the Grid, the user should follow four steps:

1. Select a geographical area, which can be entered by two ways. The user can specify this area through an hdf file or using the map component of the GreenView application. In the second case, the user has to know the geographical coordinates of that region, or he can select it by drawing a contour with the mouse.
2. Select a time interval. This interval can be chosen by specifying a year and month interval (for example: January 1961 – March 1961), or for the same month select a year interval (January, 1961 – 1963).
3. The user can start a new execution on the Grid, based on the given inputs, the ESIP platform will create the Grid jobs and will also monitor the execution.
4. The interface offers the possibility to view the status of an already executed or executing process by using the view process status facility.

B. *Fine to coarse interpolation component*

The main functionalities of this component are:

- Allowing the input dataset selection in a friendly manner. Therefore, the input TIFF images can be

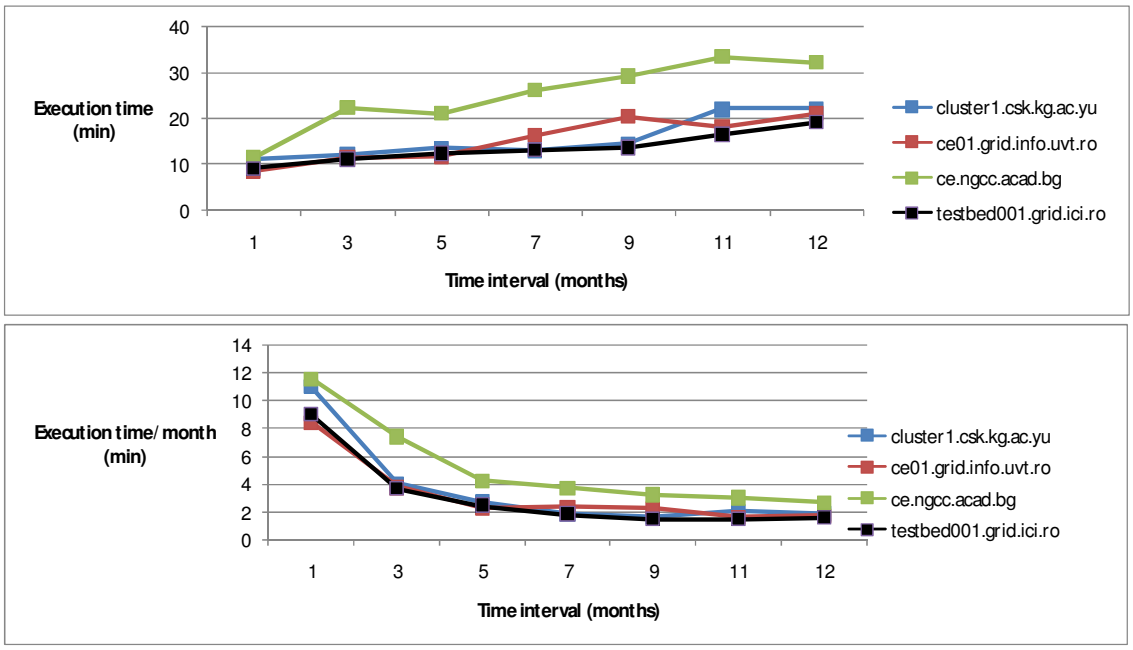


Fig. 2. Coarse to fine interpolation Grid execution results.

selected through an open dialog box. It is important to mention that multiple selections are allowed. These are added to a table, where the user has the possibility to remove any image that he desires.

- Upload functionality is available for copying the TIFF images to the server. On upload action some validations are require. In this way, if at least one image does not contain the correct metadata an error message is generated.
- After all the images are selected and uploaded to the server, the user should specify an output image resolution. This resolution has to be positive value.
- Launching a new process that runs over the Grid infrastructure, considering the specified input data.

C. *GPP computing component*

By using this component, the user should be able to calibrate the vegetation satellite measurements based on the BIOME-BGC model. In order to do this, the user can use he's own input files, or can use some predefined set of files. For the predefined files, only the view content action is available. For the other files, the user can change, upload and modify them. The second step is related to the selection of the simulation number for the BIOME-BGC model. The greater this number is, the better the calibration becomes.

D. *User manual component*

The interface allows the user to download a user manual to its local machine, or to consult it online by clicking the view user manual link. At this moment, the manual is displayed page by page, and the user has the possibility to navigate through pages or to use zoom in or zoom out facilities.

VII. EXPERIMENTS

The experiments addressed the performance evaluation of the GreenView application from the execution time point of view. We tested our application on different Computing Elements (CE) sites from the SEE-GRID-SCI infrastructure. We have used the following CEs:

```
ce01.grid.info.uvt.ro
ce.ngcc.acad.bg
testbed001.grid.ici.ro
cluster1.csk.kg.ac.yu
```

A. *Coarse to fine*

The testing variable for this application is the time interval for interpolation of the temperature file. We tested the application for a range of 1 to 12 months interval. The GreenView application creates an IPDG description file. For every month from the selected interval, it is created a grid job that will be executed on the SEE-GRID-SCI gLite

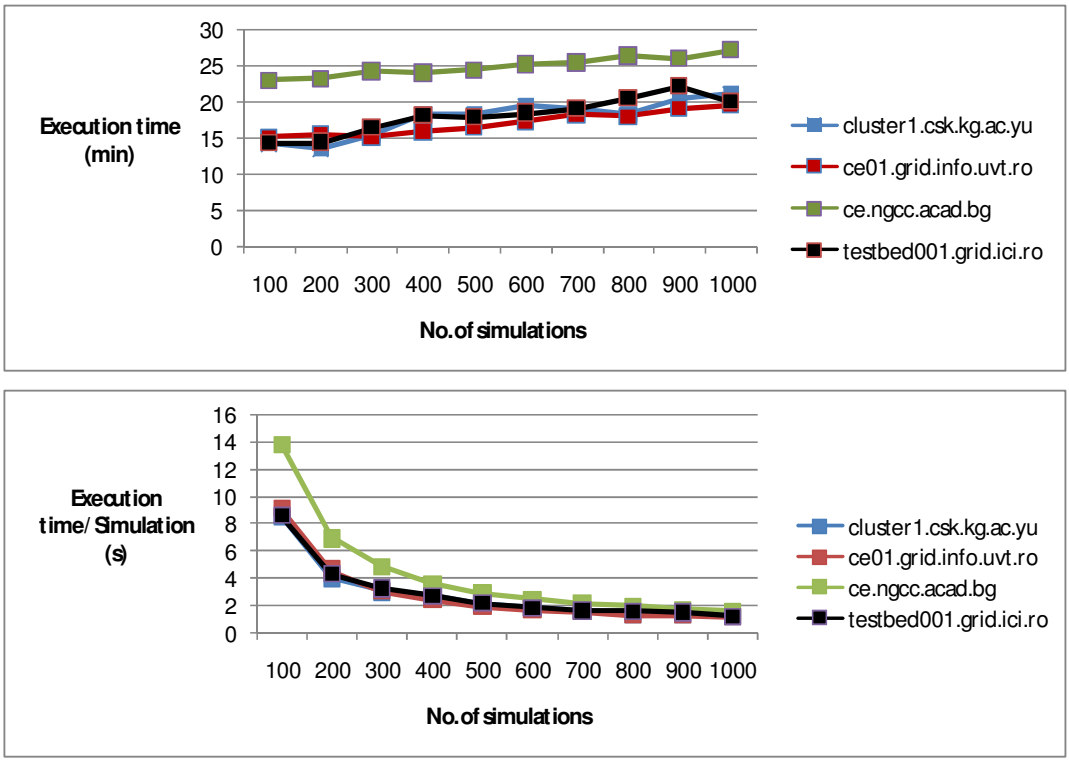


Fig. 3. Calibration Grid execution results.

infrastructure. In the Figure 2 is presented the overall execution time on different CE sites. The job execution time is also presented in the Figure 2.

B. Calibration

The number of simulation represents the testing variables. We used a range from 100 to 1000 simulation steps. Because these simulations are independent one on each other we can run the simulations in parallel. We create bulk jobs of 100 simulations. In this way, we avoid sending jobs that complete in less time than the time needed for submitting jobs, matching to a free computing element, etc. The Figure 3 presents the results for the calibration computation.

C. Interpretation of results

On the Grid infrastructure, the execution time of a job means the time to search free nodes of the Grid, time to send the required information (job application and required inputs), time required to execute the job and also time required to retrieve the results from the Grid.

The job overall execution time is increasing with the number of processing jobs. In contrast, the job execution time is decreasing. In general Grid computation is suitable for large amount of jobs and also the individual job execution time must be greater that the Grid time overhead.

The differences between different sites come from searching free resources for job execution, job submission time, computation power of the execution nodes or on network traffic and bandwidth.

VIII. CONCLUSIONS

The Grid Application Development Methodology presented in this paper offers guiding recommendations to the development process of a Grid application. The methodology is exemplified by the GreenView application development phases. The experiments regard the execution over the SEE-GRID-SCI infrastructure to prove the main functionalities from the GreenView application.

ACKNOWLEDGMENTS

This research is supported by SEE-GRID-SCI (SEE-GRID eInfrastructure for regional eScience) project, funded by the European Commission through the contract nr RI-211338.

Research is also supported by Hungarian Scientific Research Fund (OTKA K68253). Gy. G. acknowledges support by the Hungarian State Eötvös Fellowship.

Climate change data have been retrieved from the PRUDENCE data archive funded by the EU through contract EVK2-CT2001-00132.

MODIS data have been produced and distributed by NASA through the EOS Data Gateway system.

Biome-BGC version 4.1.1 was provided by the Numerical Terradynamic Simulation Group (NTSG) at the University of Montana. NTSG assumes no responsibility for the proper use of Biome-BGC by others.

Authors would also like to thank Dr. Galina Churkina (Leibniz-Centre for Agricultural Landscape Research) for her help in this study.

REFERENCES

[1] Gorgan D., Bacu V., Stefanut T., Rodila D., Mihon D., Grid based Satellite Image Processing Platform for Earth Observation Applications Development. IDAACS'2009 IEEE 5th International Workshop on "Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications", 21-23 September, Cosenza, Italy (accepted for publication in IEEE Computer Press), 2009

[2] SEE-GRID-SCI project consortium, SEE-GRID eInfrastructure for regional eScience, Available at http://www.see-grid-sci.eu/, 2009

[3] Fusco, L., Cossu, R. and Retscher. C.: Open Grid Services for Envisat and Earth observation applications. In High Performance Computing in Remote Sensing, A. Plaza, C. Chang (Eds.), Chapman & Hall/CRC, Taylor & Francis Group, p. 237–280, 2008

[4] GENESI-DR project consortium.
Available at http://www.genesi-dr.eu, 2009

[5] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi, M. Livny. Pegasus: Mapping Scientific Workflow onto the Grid. Across Grids Conference 2004, Nicosia, Cyprus, 2004.

[6] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver and K. Glover, M.R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 20(17):3045-3054, Oxford University Press, London, UK, 2004.

[7] J. Cao, S. A. Jarvis, S. Saini, G. R. Nudd. GridFlow:Workflow Management for Grid Computing. In 3rd International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan, IEEE CS Press, Los Alamitos, May 12-15, 2003.

[8] Johan Montagnat Tristan Glatard. Implementation of turing machines with the scufl datafloow language. Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid, Pages 663-668, ISBN:978-0-7695-3156-4, 2008.

[9] Zhao, M., F. A. Heinsch, R. R. Nemani, and S. W. Running 2005. Improvements of the MODIS terrestrial gross and net primary production global data set Remote Sensing of Environment. 95: 164–176. 2005

[10] Hungarian Tall Tower and Aircraft measurements at Hegyhátsál, Available at http://nimbus.elte.hu/hhs/

[11] Steven W. Running S. W., Nemani R., Glassy J.M., Thornton P. E., MODIS Daily Photosynthesis (PSN) and Annual Net Primary Production (NPP) Product (MOD17), Algorithm Theoretical Basis Document. Version 3.0, 29 Apr., 1999
Available at http://modis.gsfc.nasa.gov/data/atbd/atbd_mod16.pdf

Proceedings

2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing

Cluj-Napoca, Romania
August 26-28, 2010

Edited by
Ioan Alfred Letia

Considerations on the Grid Oriented Environmental Application Development Framework

Danut Mihon, Victor Bacu, Teodor Stefanut, Dorian Gorgan
Computer Science Department, Technical University of Cluj-Napoca
{vasile.mihon, victor.bacu, teodor.stefanut, dorian.gorgan}@cs.utcluj.ro

Abstract — **Environmental applications play an important role in the modern society due to the accuracy and the quality of the services they provide (weather forecast, meteorological or temperature prediction). The complexity of these applications makes them very hard to implement in a general, but still easy to use framework. Because of the large volume of processing data, most of the environmental models must be implemented as Grid applications, used for speeding up the execution process. This paper highlights some considerations in developing a framework capable of supporting different type of application, different input datasets (e.g. MODIS, Landsat, Thunderbird satellite images), but in the same time flexible enough to be understood by any kind of user. This could be achieved by using graphs for the workflow description of the application.**

Keywords-component; Grid; environmental application; flexible interface; workflow composition

I. INTRODUCTION

Environmental applications have a day by day increasing impact over modern society, through the services they provide, like: wheatear forecast, meteorological prediction, pollution measurement models, and other useful for the human kind community.

Every environmental application has a characteristic set of features, corresponding to the applicability domain (e.g. meteorology, soil pollution, water pollution, etc). These applications require large input data sets, that mainly consists of satellite images, like: MODIS (Moderate Resolution Imaging Spectroradiometer), Landsat, Meris or Thunderbird (some of them up to 1Gb in size). Another important aspect regards the fact that all the applications in the Earth Science domain are using algorithms based on big sets of parameters that have to be combined in a certain way in order to obtain the correct results.

This means that on a standalone machine, obtaining reasonable execution times is almost impossible. Grid infrastructure offers the solution to this problem, by providing parallel and distributed computation methods. This means, that an application is divided into atomic components that could be executed on more than one Grid nodes. In order to run a Grid based application, all the input datasets and dependency files should be sent to a Storage Element inside the Grid infrastructure. Then the execution step follows, right after the files are copied with success.

Not every application should be implemented to run on the Grid infrastructure, but only those that require a large volume of data processing. This could be explained by the simple fact that, for an application that completes its' execution in a few seconds, the time required to send the input dataset to the Grid nodes is greater than the execution time on the Grid. In this case it recommended to run the application on a single machine, rather than on the Grid infrastructure.

Basically all the applications, environmental or not, are composed of algorithms that have the following functionality: based on some input dataset, the algorithm generates an output result. Internally all kinds of operations are allowed, and these operations are executed in a predefined order. This algorithm could be easily associated with an oriented graph (called PDG – Processing Description Graph), where the nodes represent the input dataset and the operations, and the edges show the execution flow of the algorithm. The creation and execution of the PDG is supported through the ESIP and gProcess platforms [1].

The graph that represents the workflow of an algorithm provides a flexible structure in extending its' functionality. It also provides the possibility to include a graph inside another graph, in other words to combine two different algorithms and execute them as a single one.

For the environmental applications there are a variety of algorithms that could be used in different purposes. This paper brings in a solution that allows the users to execute on the Grid infrastructure different kinds of such algorithms it the Earth Science domains and allows the user to build the corresponding interface at runtime. This flexible interface assumes the existence of some predefined components that could be combined in every possible way to correspond to the application workflow.

II. RELATED WORKS

The utility of the Grid infrastructure aroused the interest of new environmental and social domains, like: seismic activity (through the DORII project [2]), water and atmospheric pollution (CrossGrid project [3]) or air traffic control (integrated in the BREIN application [4]). BEinGRID [5] is another project that experiments the Grids feasibility in a range of various European business sectors (entertainment, financial, industrial, chemistry, retail, textiles, etc).

GreenView and GreenLand [6] are two environmental applications, used in the Earth Science domains for temperature and vegetation growth monitorization and prediction. Also, some ESIP based experimental tests and related conclusions (found in [7]) could be analyzed for a better understanding of the two applications.

Grid applications are, in general, very complex and time consuming. Keeping in mind the distributed nature of the Grid, the applications were divided into atomic components (jobs), which can be executed in parallel on the Grid nodes. There are many languages for the description of the environmental applications that allow the user to specify the characteristics of their applications to be run on the Grid. These languages divided in two main categories: script-based and workflow-based. There are several languages used to describe these characteristics: JSDL (Job Submission Description Language), GXML (Guideline XML) or JDL (Job Description Language). Most of these languages assume that the applications can be represented as a set of jobs. This entire workflow is graph-structure based, where the nodes represent the jobs and the edges describe the execution path of the application. There are several types of such applications, used for describing the workflows: Pegasus [8], Taverna [9] or GridFlow [10].

III. SYSTEM RELATED ARCHITECTURE

This section presents the architecture developed as a support platform for the previous mentioned functionalities of the system. Most of environmental applications have a client-server architecture. Our solution relies on ESIP platform (Environment Oriented Satellite Image Processing). On the client side are developed only the interface components of this application, and the server side relies on the Grid infrastructure and the ESIP and gProcess platforms, where the ESIP is based on gProcess platform (Figure **1**).

gProcess platform consists of several components (like: editor, manager, executor and viewer). This platform also offers the possibility to create execution workflows in an interactive way. In other words gProcess is present on both client side and server side, where a server component (called web service) has a corresponding client one called interactive component (e.g. the viewer component is used to retrieve the execution result from the server and display it to the user).

The workflow of an algorithm execution could be described as a PDG (Process Description Graph) or as an iPDG (instantiated PDG). The first one is an abstract graph that contains only the operations and the connection between them, but has no knowledge about the input resources. So, a concrete graph is needed. In the gProcess language this graph is called iPDG. The iPDG encapsulates the functionality of the first one, and adds the possibility to specify the input resources (for every node in the graph) on execution time. All the resource information is stored in an xml file.

The creation and manipulation of a PDG or iPDG is simplified by the graph editor tool that allows the user to arrange the nodes in a visual manner.

Before creating the execution workflow it is important to have an overview of the available resources. The EditorWS [11] is used in this scope. It searches through the database and provides information about types of satellite images, list of operators, etc. General information about the workflow is

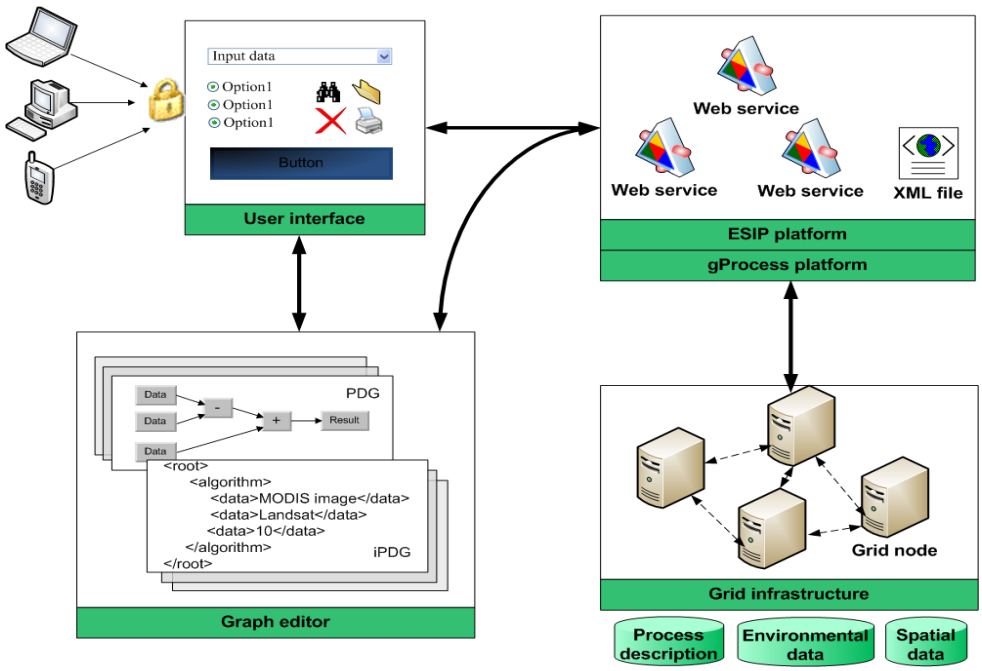


Figure 1. General system architecture

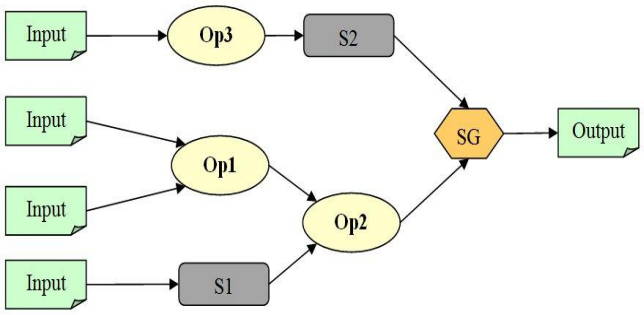


Figure 2. General iPDG example

given by the ManagerWS component that together with the EditorWS resides on the server side. Mapping the input dataset to the graph nodes, uploading them to the Storage Element on the Grid or the maintenance of the workflow is another important aspect treated by the ManagerWS component.

When all the resources are copied on the Grid nodes, the ExecutorWS is called to execute the application. Also this component is used for monitoring the progress of the execution, by periodically interrogate the status of the process.

The client components of the gProcess platform offer support for generating the PDG and iPDG graphs (using EditorIC component) or display to the user the results obtained after a process ends its' execution over the Grid infrastructure (ViewerIC component).

Taking into account all these aspects, a flexible interface should be created, that corresponds with the workflow layout generated by the user. Our solution is based on some predefined components that could be combined in such way to map over the generated workflow structure. The predefined tool set includes: basic components (text field, text areas, combo boxes, etc) and some complex components (specific for an entire environmental application).

IV. ALGORITHM IMPLEMENTATION

This section describes the method of creating new algorithms, by extending the functionality of the existing ones. The communication between the interface and the workflow management layer represents another aspect discussed in this paper. This section ends with a short use case scenario that highlights tree main environmental application development steps: building the interface for a specific environmental algorithm, generating the corresponding workflow and executing this workflow over the Grid infrastructure.

A. Workflow composition

Every workflow, described through a PDG or iPDG, has an xml structure. The only significant difference in the two graphs is that only the iPDG is used in the execution process, while the PDG offers an abstract description of the same process being executed. Figure **2** shows an example of an iPDG, containing different resources (input fields), operators, services or sub graphs.

The xml structure of the graph is a big impediment for the user to create algorithm workflows, due to the errors that could occur when manually editing the xml file. The graph editor tool provides the solution for this problem. Now, the only concern of the user in building personalize workflows is to specify the correct set of operators and the relations between them. The editing of the nodes and edges of the graph is much easier now, and could be performed with the mouse. When the user save its' current workflow, an xml is automatically generated. Loading an external workflow into the editor is another available option.

The editor framework offers the user the possibility to choose a specific resource from all the available resources stored in the database, and to map that resource to the desired node of the graph as an input data.

An environmental application, generally speaking, has more than one atomic component. An atomic component is the smallest subdivision of the application that can be executed atomically and independent from other atomic components. Every one of this component is regarded as an operator in the workflow composition, and the connection between them as dynamic relationships. So, in order to build a workflow of the application, first we need to identify all the atomic components of that application and then to establish an execution path within the graph nodes.

The workflow composition could be used when we have several applications that function in an independent manner, and we want to integrate them and create a single execution block, where, for example, the output of an application provides the input for other ones. The representation of the workflow as an oriented graph (PDG or iPDG) is key role of this problem, due to the fact that it allows sub graph integration inside the main graph. This sub graph could represent, if needed, an entire workflow of an environmental application.

In other words, the functionality extension of an application is realized by adding new nodes to the main workflow, nodes like: simple operators (addition or subtraction of numerical values), services (web service results), resources (numerical values, satellite images, text files, etc) or sub graphs (contains the result of another workflow execution).

The composition of the workflows offers a flexible structure in storing the layout of the PDG or iPDG, because this structure allows multiple sub graphs levels of imbrications. This means that we could have a graph that contains a sub graph and the last one contains another sub graph, and so on. The structure of the workflow is represented by an xml file, stored in the application database.

The next paragraph highlights an example used in vegetation classification based on GEMI (Global Environmental Monitoring Index) vegetation index. This example represents one of the functionalities of the GreenLand application that works with Landsat satellite images. Landsat images have seven frequency bands, and a combination of bands should be performed in order to obtain valid results for

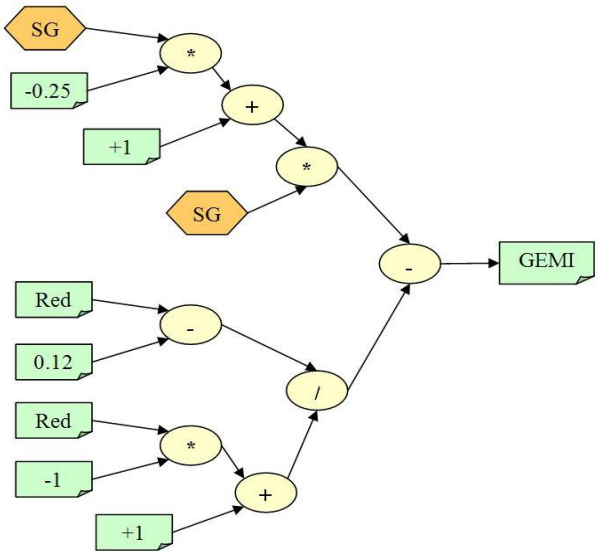


Figure 3. PDG structure for the GEMI vegetation index computation

vegetation classification, atmospheric pollution, temperature distribution and other Earth Science aspects.

The entire process of vegetation classification, based on GEMI vegetation index, could be reduced to the following two formulas:

$$\eta = \frac{2 \cdot (NIR^2 - R^2) + 1.5 \cdot NIR + 0.5 \cdot R)}{NIR + R + 0.5} \quad (1)$$

$$GEMI = \eta \cdot (1 - 0.25 \cdot \eta) - \frac{R - 0.12}{1 - R} \quad (2)$$

The (1) and (2) formulas combine different spectral bands (NIR – Near InfraRed band and R – Red band) in order to perform the vegetation classification. The structure of the algorithm that computes the GEMI index value could be divided in two atomic components: the first one used to calculate the η parameter, and the second uses this value in order to obtain the GEMI parameter.

Figure 3 shows the structure of the PDG used in the vegetation classification process. The PDG nodes consist of: operators (ellipse nodes), input data (rectangle nodes) and sub graphs (rectangles with rounded corners). It is worth to mention that the operators (+, -, *, /) are working in the satellite image filed, rather that with numerical values (e.g, the Red – 0.12 operation subtracts 0.12 from every pixel in the red spectral band of the Landsat satellite image). The sub graph that appears as one of the PDGs' node represents the value of the η parameter, computed with the first formula. As can be seen the integration of a sub graph inside another graph consists of adding a new node and link it in the right way to the general path of the workflow.

Every PDG has an associated iPDG. For the PDG description of the GEMI vegetation index (Fig.3), the corresponding iPDG has the following structure:

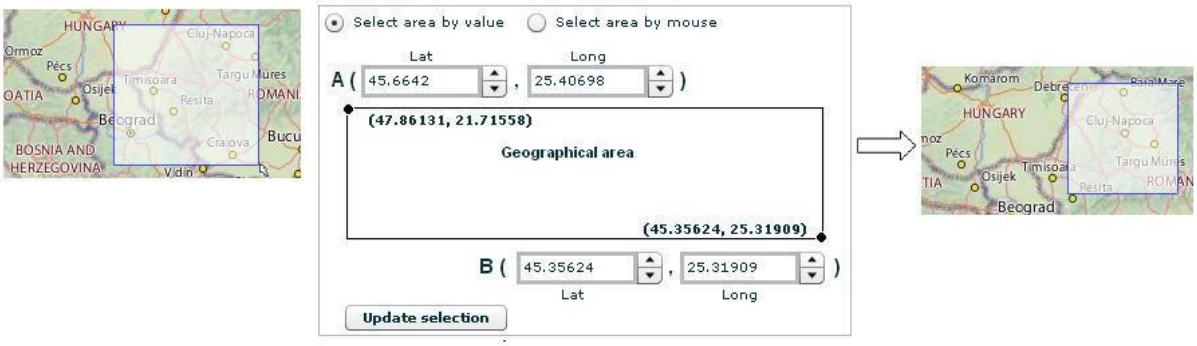
```
<?xml version="1.0" encoding="UTF-8"?>
<Workflow>
    <Nodes>
        <Resource id="1" name="B4" description="NIR spectral band">
            <LocalResource path="romania2_B4.tif" />
                <PostConditions>
                    <Output idTypeDB="1" />
                </PostConditions>
        </Resource>
        <Resource id="8" name="Constant" description="">
            <ValueResource value="1"/>
                <PostConditions>
                    <Output idTypeDB="6" />
                </PostConditions>
        </Resource>
        <Subgraph id="12" name="GEMI Subgraph" description="GEMI Vegetation Index computation" idDB="7">
            <Preconditions>
                <Input id="1"/>
                <Input id="2"/>
                <Input id="3"/>
                <Input id="4"/>
            </Preconditions>
        </Subgraph>

    </Nodes>
    <Groups>
    </Groups>
</Workflow>
```

This is only a short extraction of the entire iPDG, that highlights the storage and integration of resources, operators, constants and sub graphs.

B. Interface development

Building general interfaces for the environmental applications is a difficult task, especially when the user has the possibility to combine these applications. We develop our solution around the assumption of the existence of a set that contains some predefined components tools, like: text input, text area, combo boxes and some complex tools build to support the execution of an entire environmental application (e.g. vegetation classification – GreenLand application, temperature monitoring and distribution – GreenView application)[12].

The user completes several steps in order to create the interface that supports the structure of the execution workflow. There is no restriction in going backwards and modify one of the previous steps. First of all the user creates a new interface for the application that he wants to execute over the Grid infrastructure. All the time, a toolbox of predefined components is available for the user's needs. The following step is to establish the right set of components to be used for the current interface development. Arranging these components



description	Node name	Start server time	End server time	Status	Options
Calibration	21_Calibration	2010-02-10 21:26:59	2010-02-10 22:29:39	DONE	
Calibration	20_Calibration	2010-02-10 21:26:59	2010-02-10 22:23:49	DONE	
Calibration	19_Calibration	2010-02-10 21:26:59	2010-02-10 22:28:20	DONE	
Calibration	18_Calibration	2010-02-10 21:26:59	2010-02-10 22:24:30	DONE	
Calibration	17_Calibration	2010-02-10 21:26:59	2010-02-10 22:23:20	DONE	
Calibration	16_Calibration	2010-02-10 21:26:59	2010-02-10 22:23:43	DONE	
Calibration	15_Calibration	2010-02-10 21:26:59	2010-02-10 23:04:37	DONE	
Calibration	14_Calibration	2010-02-10 21:26:59	2010-02-10 23:04:25	DONE	

Figure 4. Geographical area selection through direct map manipulation techniques; Visualize and mange the Grid execution results

onto the main interface window is the last step in building the interface. The selection and positioning of the components is done in a drag and drop manner, from a toolbox into the interface layout.

Until now the interface layout and the execution workflow are independent from one another. In order to develop a unitary platform, there must be a communication between the two layers. This is realized through the graph editor tool. It allows the user to describe the graph and also to specify the resources for the nodes in the graph. This means that an available resource list is displayed to the user in the editor window. This list contains all the resources specified by the user in the application interface, through the predefined components.

Some particular features regarding the interface development refer to the existence of an interactive map, in order to ease the user work in selecting the geographical area of interest (Figure **4**). Already implemented in GreenView and GreenLand applications these interactive techniques are very appreciated among the user community. They allow area selection by manually entering the latitude and longitude coordinates, or by using the mouse action in the geographical area selection process. This interactive map supports, for now, only 2D geographical space visualization, but switching in 3D mode is another upcoming feature.

Because of the large volume of data processing, the interface framework must offer support to the user in the process of the result analyses. This means that the output must be available in multiple formats for downloading (e.g. JPEG image, ASCII, NetCdf satellite image format, etc). For example in the temperature distribution applications, it is easier to analyze the results based on JPEG images, than on the ASCII files. This facility assumes that the environmental application supports different formats conversion on runtime, without the user knowledge (e.g. transform a TIFF image into a JPEG one or from an ASCII temperature file generate a corresponding pseudo colored JPEG image).

C. Workflow execution over the Grid infrastructure

When the workflow of an environmental application is complete, the next step is to execute it on the Grid nodes. The completion of the workflow means that the PDG and iPDG is correctly described and that all the needed resources are available for the Grid processing. There are three main steps identified for the execution of the application.

The first one regards the upload action of the resources to the Storage Element inside the Grid infrastructure. When executing application over the Grid, the best way is to copy the needed input dataset on large storage devices that resides in the Grid infrastructure. Using this method some possible communication errors with an outside infrastructure is avoided. This step is completed through the gProcess platform, or the ManagerWS component to be more precise. This component grabs the resources from the server database and fetched them to the Grid nodes, where the execution takes place. All these operations are executed without the user knowledge. If there are large resource files, then the uploading step could take a few seconds to complete.

Once the resources are available inside the Grid infrastructure, the application execution is performed through the web services developed for the ExecutorWS component. Before the execution, all the atomic components (called jobs) of the application should be identified. Depending on the number of those components, the ExecutorWS will establish the number of Grid nodes (basically every job is executed on a

	1 month	3 months	5 months	7 months	9 months	11 months	12 months
ACAD	11.59	7.44	4.24	3.73	3.25	3.05	2.69
UPB	10.03	4.11	3.08	2.11	1.34	1.17	1.04
RENAM	11.02	4.02	2.71	1.87	1.61	2.00	1.84
ICI	9.05	3.68	2.46	1.85	1.50	1.49	1.58
UVT	8.43	3.78	2.31	2.33	2.26	1.16	1.75

Figure 5. Execution time measurements for the *coarse to fine* component of the GreenView application

different station). This has a big impact on the overall execution time, through the parallelization and distribution capabilities. It is worth to mention, that the execution start from a point (or points in the case of concurrent applications) in the workflow description, represented in this case by the iPDG.

If the execution path of the workflow reaches a node that doesn't have all the resources available, then this node is skipped and the next one is taken for execution. Of course that, this node will be executed as soon as it disposes of all the required resources. If no other node in the iPDG is available, then the algorithm waits for other Grid nodes to finish their execution and to provide the required resources.

Another important aspect of the environmental application execution over the Grid infrastructure is the fact that the entire Grid activity could be monitorized with the MonitorWS component integrated within the gProcess platform. There are lots of useful things to monitor for every process that takes part in the execution phase (e.g. a process status - starting from waiting status up to done status, cpu load, memory usage, number of stations and stations description engaged in the Grid processing, etc).

The monitoring component is tightly connected with the user interface, because at specific periods of time the overall execution status over the Grid infrastructure is interrogated and this status is displayed to the user in a friendly and easy to visualize structure (interactive table for example).

On the entire execution of a process, the user has full control over the Grid processing, meaning that all the information displayed in the interface is up to date with the real execution status on the Grid nodes. At every time the user could stop the execution of a process by using an interface button action. Another important aspect integrated within the environmental application development techniques is that not all the users have full access to all the processes executed or in execution phase over the Grid infrastructure. At the beginning, the user, that is the author of the current process, could choose between a private access (accessible only for the current user) or public access (open for all the users in the Earth Science community) for its process. This is an important factor in searching processes by different criteria, such as: description keywords, processes with a certain status or searching processes by the starting date of the execution.

V. EXPERIMENTS

All the experiments the cover this section highlights the necessity for an infrastructure that could deal with large volume of data processing, without considerably affecting the overall execution time .Environmental models are a particular case of such high power consumption application, and the Grid is the required infrastructure to store and execute these applications.

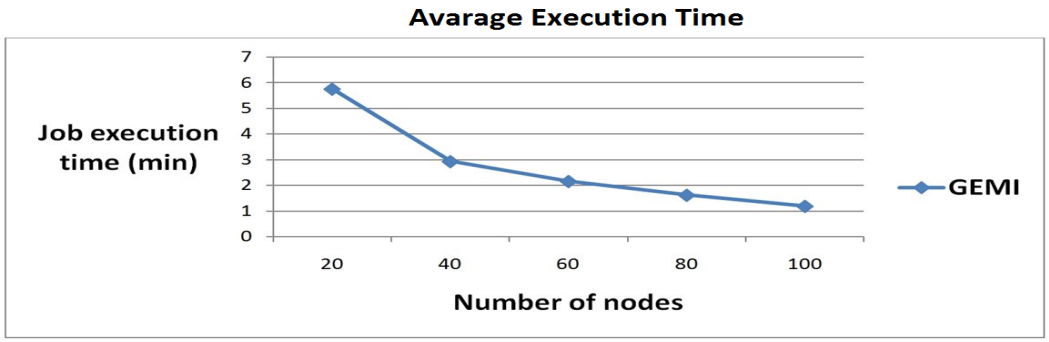


Figure 6. Execution time measurements for the vegetation classification based on GEMI vegetation index

	2000 iterations	3000 iterations	4000 iterations	5000 iterations	6000 iterations	7000 iterations	8000 iterations	9000 iterations	10000 iterations
ACAD	964.50	683.33	1139.00	414.40	421.50	349.86	286.13	381.11	290.70
UPB	2361.00	1138.00	1249.25	879.60	734.83	728.14	732.25	652.00	584.60
RENAM	1456.50	1138.33	830.50	559.00	526.67	500.43	473.63	427.33	777.00
ICI	1445.00	1043.33	1028.00	812.60	718.00	641.57	493.38	586.33	732.90
UVT	1275.50	964.67	736.25	1086.80	693.00	517.86	480.50	906.22	469.60

Figure 7. Execution time measurements for the calibration component of the GreenView application

If all the previous sections describe the general platform for building complex environmental applications (using a flexible interface and the workflow composition techniques), the experiments in this section were conducted based on the GreenView and GreenLand, two of the applications in the Earth Science domain.

GreenView application is a refinement of surface and vegetation parameters (coarse to fine component) in SEE region based on satellite images and field measurements (GPP – Gross Primary Production - calibration component). In the frame of GreenView applications, construction, usage and comparison of diverse satellite datasets are performed. High resolution satellite measurements can be used for numerous environmental studies (climate-related, surface-atmosphere exchange or air pollution modeling).

GreenLand application is used to classify the Landsat satellite images in the process of creating thematic maps. A thematic map contains information about the distribution of specific elements in nature (e.g. soil, water, air, vegetation, etc).

This offers useful information for the classification of the ecological and habitat diversity, monitoring of fires, risk management (e.g. avalanches, floods, landslides, etc.) and control of vegetation health [13].

To highlight the Grid advantages regarding the high processing volume of data, different experiments were conducted over several sites in the Grid infrastructure, like: ce01.grid.info.uvt.ro, ce.ngcc.acad.bg, testbed001.grid.ici.ro, cluster1.csk.kg.ac.yu. This also shows the geographical diversity of the Grid platform at worldwide level.

Figure 5 shows the overall execution time for the coarse to fine component integrated into the GreenView application. A time interval, expressed as months, is one of the inputs for this component that monitors the temperature distribution over

	2000	3000	4000	5000	6000	7000	8000	9000	10000
Stage Out	0.29	0.34	0.35	0.28	0.28	0.30	0.29	0.30	0.52
Running	23.13	23.29	20.61	24.12	31.51	31.61	27.91	38.22	36.47
Waiting	2.02	2.06	31.45	2.02	2.01	2.01	2.02	2.04	2.05
Submit	1.54	1.77	1.61	2.66	1.69	1.96	1.66	1.86	1.67
Stage In	0.28	0.30	0.54	0.57	0.36	0.45	0.35	0.55	0.47

Figure 8. Time spend by the calibration processes in every status of the execution

certain geographical areas. The experiment starts with 1 month up to 12 months in length. As the time interval grows in size, the execution time slightly decreases (e.g. in the 1 month interval case, a job finish its execution in about 10 seconds, while for the right headed time interval the total execution time for every job of the process is around 2 seconds).

This could be explained due to the fact that the coarse to fine component benefits from the parallel and distributed capabilities of the Grid infrastructure. The same set of experiments on a standalone machine will execute in order of tens of seconds. Another time consuming application regards the vegetation classification performed by the GreenLand application. The complexity of this case takes into account the large input data sets (represented through the Landsat satellite images) and the large number of the graph's nodes.

Figure **6** shows the results of the GEMI experiment. The X axis represents the number of nodes in the structure of the PDG, while the vertical one measures the total execution time for every job of the process.

The Calibration component, also called GPP computation (Gross Primary Production) developed inside the GreenView environmental application is a perfect Grid application, because it exploits very well the parallel capability of the Grid. This component is based on the BIOME-BGC model. The accuracy of the calibration is closely related with the number of model simulations. The reason why this component is so efficient on the Grid infrastructure is that the entire simulations are divided into bulks of 500 pieces that are executed independently on different Grid nodes. At the end of the execution a new operator collects all the results and computes the best calibration value for the GPP parameter.

Figure **7** presents the results obtain during the GPP computation experiment. The first test uses 2000 simulation of the BIOME-BGC model, and describes an increasing slope up to 10000 simulations. As in the case of the coarse to fine component, the overall execution time for every job of the process was measured. The conclusion on the result is the same: the bigger the simulation number, the lower the execution time for the job.

The first status of a Grid based process execution is the Submit status that corresponds to uploading all the required resources to the Storage Element, for the current Grid processing.

The second one is the Waiting status. In this case the execution process waits for previous nodes to end the running stage in order to provide the needed input for the current. A process could stay in waiting status when the ExecutorWS component searches available nodes in the Grid infrastructure.

The most time consuming is the Running status, in which the job is executed as a unitary and independent algorithm on the Grid infrastructure.

Figure **8** details all the previous mentioned features of a process execution on the Grid nodes. This experiment was performed for the calibration component and highlights the time spent by every job in one of the three main statuses: Submit, Waiting and Running.

It is worth to mention that the total execution time over the Grid infrastructure could be computed as a sum regarding: time needed to upload all the resources to the Storage Element inside the Grid, time required to search for available execution nodes, time to execute the process and the time to store the results into a local database.

VI. CONCLUSIONS

This paper tries to offer a general solution in building frameworks for the environmental applications, and takes into account the interface development and the workflow composition techniques. Some of the principles of this solution were already implemented, but other ones are only in the design stage. The Grid infrastructure has an important role in this solution, due to the huge amount of storage and high power computation capabilities. On the other hand the Grid offers an elegant technique for execution the applications in a parallel or distributed manner. The last issue has visible benefits over the execution time for the environmental applications.

REFERENCES

[1] Gorgan D., Environment VO - GreenView and ESIP. Presentation to "SEE-Grid-SCI: Environmental VO training in Cluj-Napoca", 13 Jan 2009, Clun-Napoca, http://seegrid3.lpds.sztaki.hu/indico/ (2009).

[2] DORII project, http://www.dorii.eu/

[3] CrossGrid project, http://www.eu-crossgrid.org/

[4] BREIN project, http://www.gridsforbusiness.eu/

[5] BEInGRID project, http://www.beingrid.eu/be7.html/

[6] Mihon D., Bacu V., Gorgan D., Mészáros R., Gelybó G., "Practical Considerations on the GreenView Application Development and Execution over SEE-GRID". SEE-GRID-SCI User Forum, Bogazici University, Istanbul, Turkey, December 9-10, pp. 167-175, (2009)

[7] Gorgan D., Bacu V., Rodila D., Pop Fl., Petcu D., "Experiments on ESIP - Environment oriented Satellite Data Processing Platform". SEE-GRID-SCI User Forum, Bogazici University, Istanbul, Turkey, December 9-10, , pp. 157-166, (2009)

[8] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi, M. Livny. Pegasus: Mapping Scientific Workflow onto the Grid. Across Grids Conference, Nicosia, Cyprus, (2004)

[9] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T.Carver and K. Glover, M.R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. Bioinformatics, 20(17):3045-3054, Oxford University Press, London, UK, (2004).

[10] J. Cao, S. A. Jarvis, S. Saini, G. R. Nudd. GridFlow:Workflow Management for Grid Computing. In 3rd International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan, IEEE CS Press, Los Alamitos, May 12-15, (2003).

[11] Bacu V., Stefanut T., Rodila D., Gorgan D., Process Description Graph Composition by gProcess Platform. HiPerGRID - 3rd International Workshop on High Performance Grid Middleware, 28 May, Bucharest. Proceedings of CSCS-17 Conference, Vol.2., ISSN 2066-4451, pp. 423-430, (2009)

[12] Gorgan D., Stefanut T., Bcu V., Mihon D., Grid based Environment plication Development Methodology, SCICOM, 7th International Conference on "Large-Scale Scientific Computations", 4-8 June, Sozopol, Bulgaria (2009)

[13] Mihon D., Bacu V., Gorgan D., Mészáros R., Gelybó G., Practical Considerations on the GreenView Application Development and Execution over SEE-GRID, 20-23 November, Istanbul, Turcia, (2009)