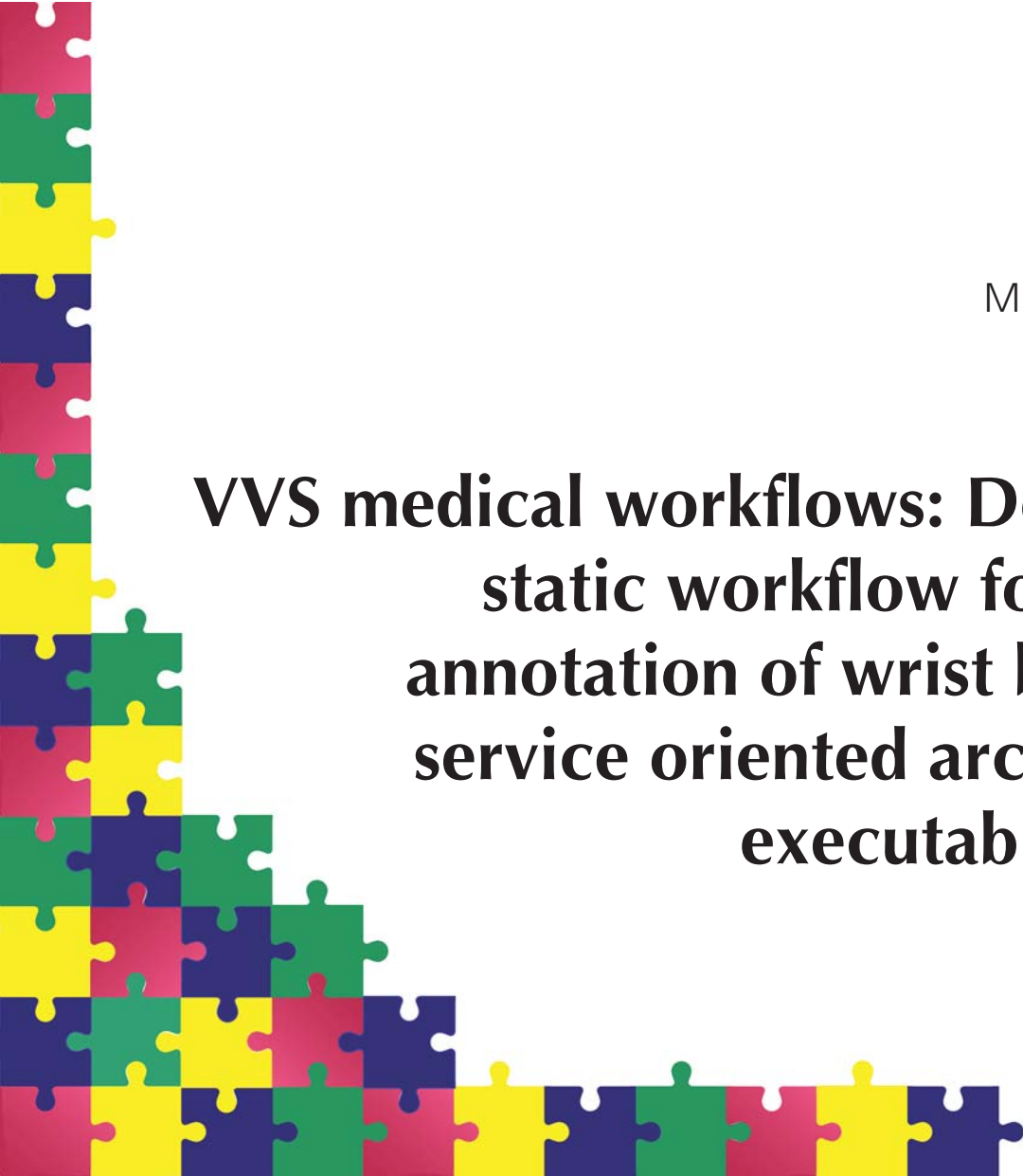


REPORT SERIES

M. Pitikakis, F. Giannini

A decorative graphic on the left side of the page consisting of a grid of colorful puzzle pieces in shades of red, green, yellow, and blue, arranged in a stepped pattern that tapers to the right.

**VVS medical workflows: Definition of a static workflow for part-based annotation of wrist bones & web service oriented architecture for executable workflows**

# IMATI REPORT Series

Nr. 17-07

March 2017

## **Managing Editor**

Paola Pietra

## **Editorial Office**

Istituto di Matematica Applicata e Tecnologie Informatiche "*E. Magenes*"

Consiglio Nazionale delle Ricerche

Via Ferrata, 5/a

27100 PAVIA (Italy)

Email: [reports@imati.cnr.it](mailto:reports@imati.cnr.it)

<http://www.imati.cnr.it>

Follow this and additional works at: <http://www.imati.cnr.it/reports>

---

Copyright © CNR-IMATI, 2017.

IMATI-CNR publishes this report under the Creative Commons Attributions 4.0 license.

**VVS medical workflows: Definition of a static workflow for part-based annotation of wrist bones & web service oriented architecture for executable workflows**

Marios Pitikakis, Franca Giannini



Marios Pitikakis  
Softeco Sismat S.r.l.  
Via De Marini, 1 – Torre WTC  
16149 Genova, Italy  
e-mail address: [marios.pitikakis@gmail.com](mailto:marios.pitikakis@gmail.com)

Franca Giannini  
Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes" – CNR  
Via De Marini, 6  
16149 Genova, Italy  
e-mail address: [franca.giannini@ge.imati.cnr.it](mailto:franca.giannini@ge.imati.cnr.it)

---

## **Abstract**

The report describes the extension of the Digital Shape Workbench version 5 (DSW5) for the inclusion of workflows for the analysis and processing of 3D models obtained from medical imaging data. The VVS Workflow Repository (WR) manages the creation, visualization and execution of shape processing workflows. The considered workflows are of two main categories: static and executable. The first category refers a kind of tutorial workflow/pipeline and is used for sharing expert knowledge e.g. for explaining the steps needed when dealing with a specific process. The second one represents chain of executable web services performing specific pipelines of geometric processing functions. The document describes the adopted architecture and structure for their specification and execution.

**Keywords:** *Shape processing, semantic web, on-line repositories*

---

*[page left intentionally blank]*

**VVS Medical Workflows:**

**Definition of a Static Workflow for  
Part-based Annotation of wrist bones**

**&**

**Web Service oriented architecture  
for Executable Workflows**

**Marios Pitikakis, Franca Giannini**

## Contents

1	Abstract.....	3
2	Introduction.....	4
3	Definition of a static workflow for part-based annotation .....	5
3.1	Categories of workflows in the Workflow Repository.....	5
3.2	Static Workflows .....	5
3.3	Description of the part-based annotation pipeline .....	6
3.4	Medical related tools inserted in the TR .....	11
4	Semantically Enriched Web Services .....	15
4.1	Ontology-Driven Service Discovery and Composition .....	16
5	Overview of the VVS Service-Oriented Architecture .....	17
5.1	Dynamic Workflow Composition and Execution of Web Services .....	20
5.2	Shape Processing Web Services and workflows .....	23
6	Implementation of Shape Processing Applications as Services .....	24
6.1	Development of single-step Web Services .....	24
6.2	Development of the Web Service Workflows .....	25
6.3	First Web Service Workflow Scenario.....	28
6.4	Second Web Service Workflow Scenario.....	29
7	The Web Services and workflows user interface .....	30
7.1	Single-Step Web Services.....	31
7.2	Web Services Workflows .....	33
8	Integration with the CAR2VR ontology .....	37
9	Technologies and tools used .....	40
9.1	OpenESB.....	40
	Acknowledgments.....	41

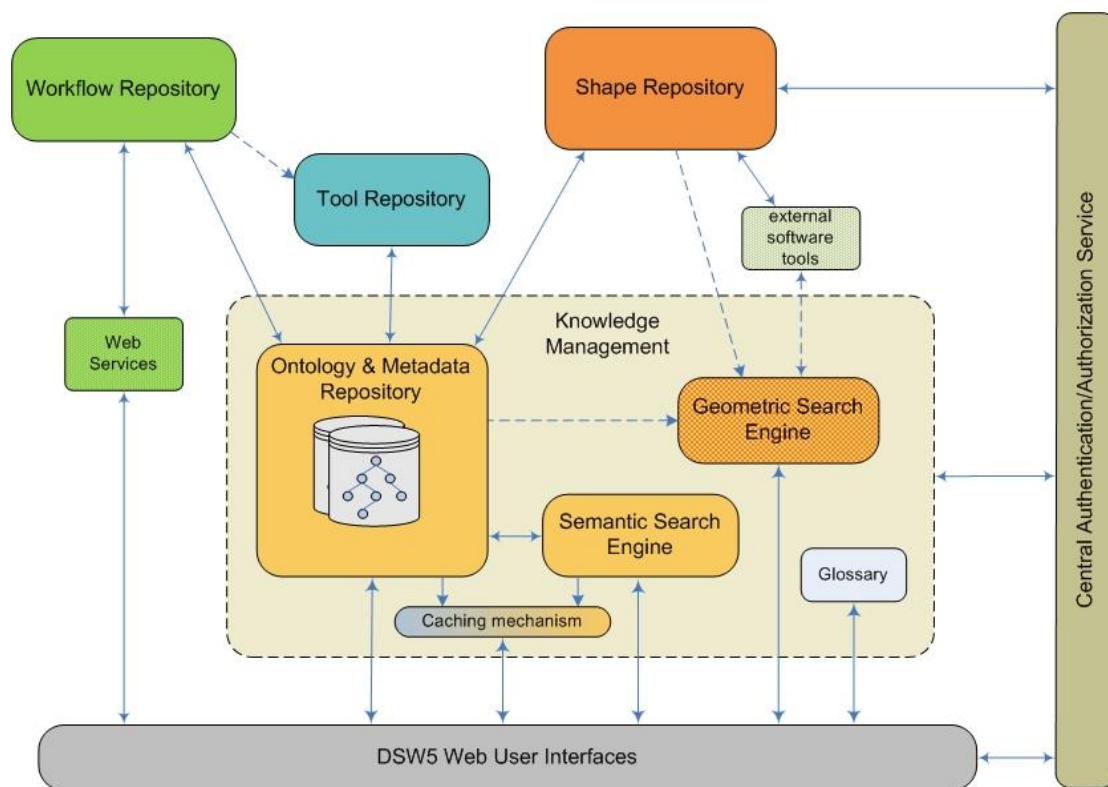


## 1 Abstract

The report describes the extension of the Digital Shape Workbench version 5 (DSW5) for the inclusion of workflows for the analysis and processing of 3D models obtained from medical imaging data. The VVS Workflow Repository (WR) manages the creation, visualization and execution of shape processing workflows. The considered workflows are of two main categories: static and executable. The first category refers a kind of tutorial workflow/pipeline and is used for sharing expert knowledge e.g. for explaining the steps needed when dealing with a specific process. The second one represents chain of executable web services performing specific pipelines of geometric processing functions. The document describes the adopted architecture and structure for their specification and execution.

## 2 Introduction

The Digital Shape Workbench version 5 (DSW5) is part of the VISIONAIR Virtual Visualization Service (VVS) and its primary goal is the formalization and sharing of knowledge about 3D digital shapes and their applications. DSW5 integrates resources and knowledge into a unified interface, and consists of the data repositories: the *Shape Repository* (SR), the *Tool Repository* (TR), the *Workflow Repository* (WR), the *Ontology & Metadata Repository* (which is the knowledge management system) and a number of different ways of browsing and searching for these resources. An overview of the architecture of DSW5 is shown in Figure 1.



**Figure 1.** Overview of the DSW5 system architecture.

The first part of this report describes the definition of a static workflow for the medical domain, and more specifically the pipeline from the medical image acquisition to the part-based semantic annotation of wrist bones. The static workflow is stored in the Workflow Repository (WR) and all the software tools required to support each step of the pipeline were inserted in the Tool Repository (TR).

The rest of the report introduces a semantic organization and description of Web Services for enhancing the ability to compose processing chains (workflows) of Web Services. Ontologies can play an important role in empowering Web Services with

semantics that can enable service discovery and composition. We adopt a knowledge-based approach that effectively utilizes the existing knowledge infrastructure of the VVS.

A generic service oriented middleware architecture was designed and developed for orchestrating composite services and a set of shape processing Web Services were implemented that can be easily combined into workflows.

### 3 Definition of a static workflow for part-based annotation

#### 3.1 Categories of workflows in the Workflow Repository

The VVS Workflow Repository (WR) manages the creation, visualization and execution of shape processing workflows. The description of the workflows is stored in the Workflow Ontology (WO), which is a process ontology that describes all the necessary information about workflows. The Common Tool Ontology (CTO) is imported by the WO, thus taking advantage of the already existing formalization of tools given by the CTO.

There are two main categories of workflows in the WR: ***static workflows*** and ***executable workflows***.

#### 3.2 Static Workflows

A *static workflow* represents a kind of tutorial workflow/pipeline and is used for sharing expert knowledge e.g. for explaining the steps needed when dealing with a specific process.

The WR and the underlying WO was developed with the aim of answering questions regarding the problem of CAD to VR transition, however, it can be reused and/or extended to also cover other domain like the medical domain in our case.

*Static workflows* are simple sequences of at least two steps, called ***activities***, while no parallel activities are allowed (i.e. steps that are performed at the same time). The main activities composing a workflow are called macro-activities and may be decomposed into sub-activities i.e. simple-activities. A macro-activity with no sub-activities is considered a “simple activity”. Simple-activities are required sub-activities of a macro-activity and the “simple” adjective is due to the fact that they are one-step activities that correspond to a functionality of the tool performing the activity.

For each *static workflow*, a domain of application can be defined. The knowledge base formalization has been designed in such a way that other shape-oriented

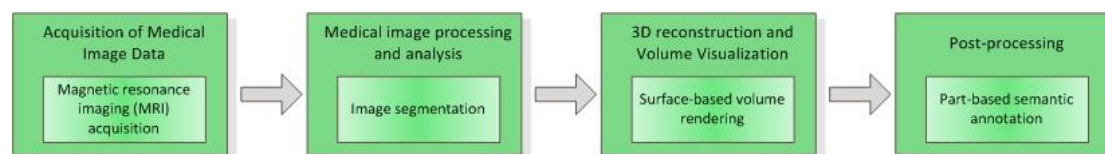
workflows with different domains can also be addressed. More specifically, the **medical** domain was added as an instance of *WorkflowDomain* class in the WO.

Tools performing an activity are not directly described in the WO, but are retrieved in an indirect way. Each simple-activity is associated to a *Functionality* (as described by the CTO ontology), that may be either an algorithm in the Shape Processing domain or a simple feature of a tool that is referred as *micro-functionality* (as defined in the WO). Macro-activities are not required to be directly associated to a functionality or a tool, unless they have no sub-activities. Typically, no single tool can perform a macro-activity but a set of tools may be required.

### 3.3 Description of the part-based annotation pipeline

The defined static workflow for the medical domain describes the complete pipeline for part-based semantic annotation of wrist bones and consists of the following four activities and their sub-activities (as show in Figure 2):

- a) Acquisition of Medical Image Data: Magnetic resonance imaging (MRI) acquisition
- b) Medical image processing and analysis: Image segmentation
- c) 3D reconstruction and Volume Visualization: Surface-based volume rendering
- d) Post-processing:



**Figure 2.** Overview of the medical static workflow definition.

The full title of the static workflow is: “Patient specific part-based semantic annotation of 3D models”. The description of the static workflow was defined as follows:

- The main goal of this workflow is to identify and visualize prominent features from patient-specific 3D reconstructions of bones, annotate them with semantic concepts and define characterizations that could potentially support computer assisted diagnosis, therapy planning, bio-mechanical simulation, prosthesis fitting etc. Part-based annotations can couple patient-specific geometry and their semantics.

The description and functionality (according to the CTO) of each activity and sub-activity is shown on Table 1.

**Table 1: Descriptions and functionalities of all the defined activities.**

Activity/sub-activity name	Functionality	Description
Acquisition of Medical Image Data	(Acquisition) Measurement, Probing	In medical imaging there are various types of techniques, equipments or probes used to acquire images, called modalities, which can provide vital information about the structural, chemical and electrical properties of the human body. Each modality is based on different physical phenomena and thus captures different types of information.
Magnetic resonance imaging (MRI) acquisition	(Acquisition) Measurement, Probing	Magnetic resonance imaging (MRI) is based on measuring the resonating frequency (or spin) of atoms under magnetic fields and is able to capture cross-sectional images of the body using non ionizing radiation. Image contrast in MRI is determined due to a number of factors including the biochemical environment of water molecules, the movement and diffusion of fluid and the density of water molecules in tissue. Consequently, MRI provides much greater contrast between the soft tissues of the body than computed tomography. This makes MRI especially useful in musculoskeletal, oncological (cancer) and neurological imaging.
Medical image processing and analysis	Other functionality	Medical image processing and analysis techniques extract clinically relevant information from radiological data. Image analysis is crucial for many diagnosis and therapy planning tasks and is often carried out as a pipeline of individual steps in which quantification and visualization are the final goals.
Image segmentation	Segmentation	Image segmentation usually represents the core of image analysis. This process assigns labels (unique identifiers) of anatomical or pathologic structures to parts (segments) of the image data. Image segmentation is often a sophisticated, time-consuming process that produces geometric descriptions of the relevant structures. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

		Segmentation of organs/bones/tissues is necessary to determine important characteristics such as the size, shape or volume of anatomical structures.
3D reconstruction and Volume Visualization	Visualization	Imaging modalities such as CT, MR, PET and ultrasound can acquire not only projective 2D images but also cross-sectional images of the human body. 3D medical visualization is the process of generating images from raw medical data to gain insight into its qualitative and quantitative features. While clinicians can gain some insight from visualizing 3D data as a series of 2D slices, a 3D reconstruction of the data provides a much richer and highly comprehensive view of the structures contained in the image, due to our implicit ability to perceive 3D shapes.
Surface-based volume rendering	SurfaceMeshing	Volume data can be visualized by generating an intermediate representation, which is subsequently rendered i.e. indirect volume rendering (primarily surface-based visualization). Structures of interest in volumetric data are typically differentiated from the surrounding image data by a boundary; hence, the resulting surface on these voxels is called an isosurface. There are a lot of methods for surface-based visualization, like Contour Tracing, the Cuberille Voxel Representation and the Polygonal Isosurface Extraction (Marching Cubes algorithm).
Post-processing	Other functionality	Most modern Computer Aided Detection/Diagnosis (CAD) systems and software rely on post-processing operations on medical image volume data. Such operations include for example the calculation of diagnostic parameters, the evaluation of measurements, the annotation of anatomical structures etc.
Part-based semantic annotation	Annotation	Part-based annotations are focused on identifying and visualizing prominent features of anatomical structures and annotating/characterizing them with semantic concepts that could support the diagnosis, monitoring and follow-up of patients.

A screenshot of the medical static workflow when browsing the Workflow Repository is shown in Figure 3.

The screenshot shows the AIM@SHAPE v5.0 website interface. The main content area displays a workflow titled "Patient specific part-based semantic annotation of 3D models". The workflow is represented as a flowchart with the following steps:

- Acquisition of Medical Image Data**: Includes "Magnetic resonance imaging (MRI) acquisition".
- Medical image processing and analysis**: Includes "Image segmentation".
- 3D reconstruction and Volume Visualization**: Includes "Surface-based volume rendering".
- Post-processing**: Includes "Part-based semantic annotation".

Arrows indicate the flow from Acquisition to Processing, then to 3D reconstruction, and finally to Post-processing. A separate arrow also points from the 3D reconstruction step to the Post-processing step.

Metadata for the workflow includes:

- Description:** The main goal of this workflow is to identify and visualize prominent features from patient-specific 3D reconstructions of bones, annotate them with semantic concepts and define characterizations that could potentially support computer assisted diagnosis, therapy planning, bio-mechanical simulation, prosthesis fitting etc. Part-based annotations can couple patient-specific geometry and their semantics.
- Creator:** Marios\_Pitrikakis
- Creation date:** 2015-10-29 09.50.07
- Domain:** Medical

Additional metadata for the "Part-based semantic annotation" step includes:

- Description:** Part-based annotations are focused on identifying and visualizing prominent features of anatomical structures and annotating/characterizing them with semantic concepts that could support the diagnosis, monitoring and follow-up of patients.
- Metadata individual:** [Part-based semantic annotation 8 Patient specific part-based semantic annotation of 3D models 2015-10-29 09 50 07](#)
- Correspond to:** [Annotation](#)
- Tools:** [SemAnatomy3D](#), [ShapeAnnotator](#), [Smart Object Modeling plug-in for 3D Studio MAX](#)
- Input formats:** [off](#), [VRML](#), [vtk](#)
- Output formats:** [txt](#), [sem3D](#), [segs](#)

The left sidebar contains navigation links for Workflow Repository, Executable Web Services, User Group Management, and Credits. The top navigation bar includes links for Ontologies, Shapes, Tools, Workflows, and Glossary.

**Figure 3.** Browsing the Workflow Repository for medical workflows.

The medical workflow was inserted in the Workflow Repository using the web user interface as shown in Figure 4 and Figure 5.

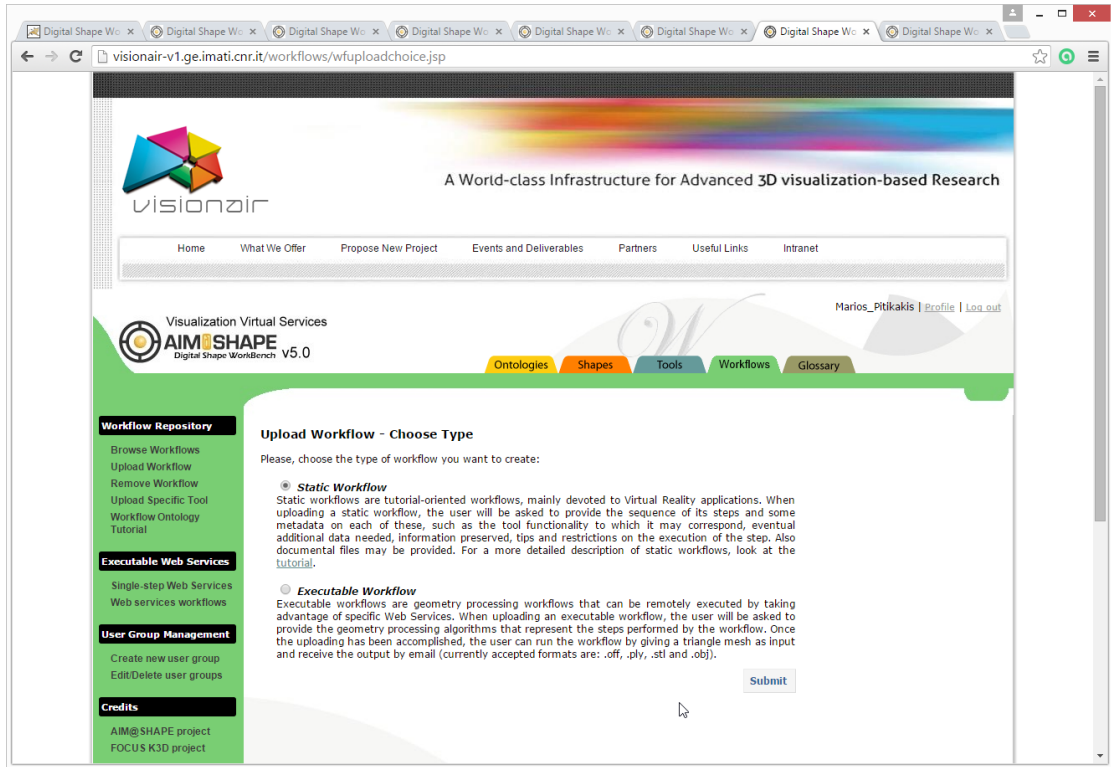


Figure 4. Inserting a static workflow from the web UI.

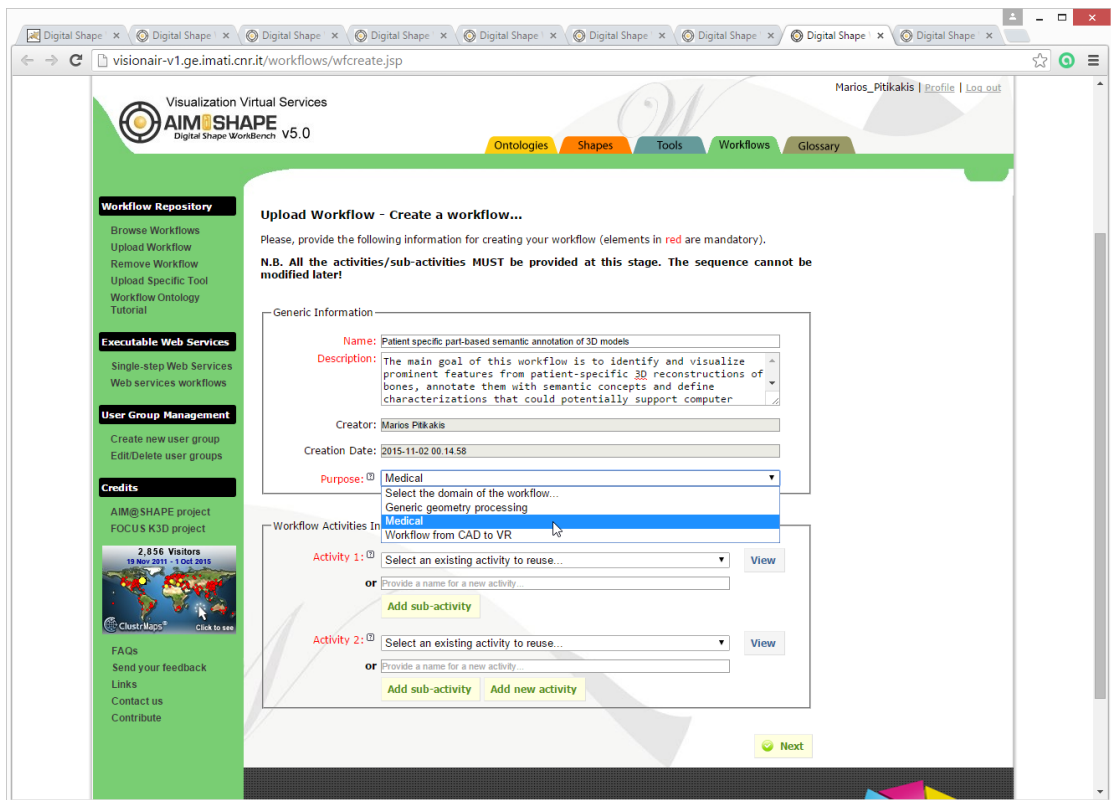


Figure 5. Filling the workflow information



### 3.4 Medical related tools inserted in the TR

To support the medical static workflow, several state-of-the-art tools were inserted in the *Tool Repository* (TR) (see Table 2) that can be used to process and manipulate digital shapes for research and visualization purposes. More specifically, eleven (11) tools were inserted: 3 instances of the *Library* class and 8 instances of the *IndependentApplication* class.

In addition, the following instances were inserted in the *ShapeFormat* class of the TR, which are related with the medical domain: dcm, dicom, nifti, nrrd, mha, mhd, vtk, avi, ps, png, tiff, sem3D, segs

**Table 2.** Detailed description of the inserted tool instances in the TR.

Category: Library
<p><b>hasName:</b> GDCM (Grassroots DICOM library)</p> <p><b>hasDescription:</b> Grassroots DICOM (GDCM) is an open source implementation of the DICOM standard so that researchers may access clinical data directly. GDCM includes a file format definition and a network communications protocol, both of which should be extended to provide a full set of tools for a researcher or small medical imaging vendor to interface with an existing medical database.</p> <p><b>hasFunctionality:</b> Visualization, Other</p> <p><b>hasInputFormat:</b> dcm, dicom, jpg</p> <p><b>writtenWithProgrammingLanguage:</b> C++</p> <p><b>hasDevelopmentStatus:</b> stable</p> <p><b>hasURL:</b> <a href="http://gdcm.sourceforge.net">http://gdcm.sourceforge.net</a></p> <p><b>hasExecutionPlatform:</b> Windows, Linux, Mac OS, other UNIX</p> <p><b>hasLicense:</b> Other (BSD, Apache license)</p> <p><b>hasSourceCodeURL:</b> <a href="http://sourceforge.net/projects/gdcm/">http://sourceforge.net/projects/gdcm/</a></p>
<p><b>hasName:</b> ITK (Insight Segmentation and Registration Toolkit)</p> <p><b>hasDescription:</b> Insight Segmentation and Registration Toolkit (ITK) is an open-source, cross-platform system that provides developers with an extensive suite of software tools for image analysis. Developed through extreme programming methodologies, ITK employs leading-edge algorithms for registering and segmenting multidimensional data.</p> <p><b>hasFunctionality:</b> Registration, Segmentation</p> <p><b>hasInputFormat:</b> dicom, nrrd, mha, mhd, vtk, bmp, jpg, png, tiff</p> <p><b>hasOutputFormat:</b> dicom, nrrd, mha, mhd, vtk, bmp, jpg, png, tiff</p> <p><b>writtenWithProgrammingLanguage:</b> C++</p> <p><b>hasDevelopmentStatus:</b> stable</p> <p><b>hasURL:</b> <a href="http://www.itk.org/">http://www.itk.org/</a></p> <p><b>hasExecutionPlatform:</b> Windows, Linux, Mac OS, other UNIX</p> <p><b>hasLicense:</b> Other (BSD, Apache license)</p> <p><b>hasSourceCodeURL:</b> <a href="http://www.itk.org/ITK/resources/software.html">http://www.itk.org/ITK/resources/software.html</a></p>
<p><b>hasName:</b> VTK (The Visualization Toolkit)</p> <p><b>hasDescription:</b> The Visualization Toolkit (VTK) is an open-source, freely available software system for 3D computer graphics, image processing, and visualization. It consists of a C++</p>

class library and several interpreted interface layers including Tcl/Tk, Java, and Python. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture, and volumetric methods, as well as advanced modeling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation. VTK has an extensive information visualization framework and a suite of 3D interaction widgets. The toolkit supports parallel processing and integrates with various databases on GUI toolkits such as Qt and Tk.

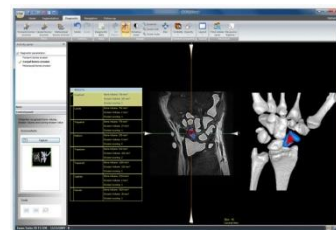
**hasFunctionality:** Visualization, Modeling, Meshing, Smoothing, Triangulation,  
**hasInputFormat:** 3ds, stl, obj, vtk, ply, xyz, dicom, mha, mhd, bmp, jpg, png, tiff  
**hasOutputFormat:** stl, obj, wrl, vtk, ply, avi, mha, mhd, bmp, jpg, png, tiff, ps  
**writtenWithProgrammingLanguage:** C++  
**hasDevelopmentStatus:** stable  
**hasURL:** <http://www.vtk.org/>  
**hasExecutionPlatform:** Windows, Linux, Mac OS, other UNIX  
**hasLicense:** Other (BSD license)  
**hasSourceCodeURL:** <http://www.vtk.org/download/>

### Category: IndependentApplication

**hasName:** RheumaSCORE


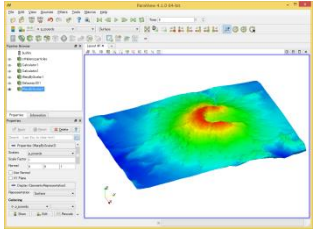
**hasDescription:** RheumaSCORE helps radiologists and physicians during the medical investigation and the diagnostic processes related to the management of clinical progression of patients suffering from Rheumatoid Arthritis (RA). RheumaSCORE leads the user during the three-dimensional segmentation process of the bones structure using the Geodesic Active Contour approach. After segmentation, RheumaSCORE provides automatical evaluation of the bones erosion scoring (using OMERACT RAMRIS criterion) and can visualize the generated 3D models. RheumaSCORE stores all the information related to a patient examination (e.g. acquired DICOM images, anatomical 3D segmented elements, three-dimensional features results, user annotations) in the system database for retrieval so that the user can visualize the follow-up of a patient. It also provides automatic comparison among the parameter results and evaluates the difference between pairs of contiguous values in time, showing interactive plots with highlighted temporal trends. RheumaSCORE was developed by Softeco Sismat S.r.l.


**hasFunctionality:** Segmentation, Visualization, SurfaceMeshing  
**hasInputFormat:** dcm, dicom  
**hasOutputFormat:** mha, mhd, vtk  
**writtenWithProgrammingLanguage:** C++  
**hasDevelopmentStatus:** stable  
**hasUIType:** Graphical UI  
**hasURL:** <http://www.research.softeco.it/rheumascore.aspx>  
**hasExecutionPlatform:** Windows  
**hasLicense:** Commercial



**hasName:** VolSeg

**hasDescription:** VolSeg is generic purpose segmentation tool that can assist users during the image segmentation and 3D model generation process. VolSeg can load all kind of DICOM data e.g. MRI, CT, PET, US etc. images and allows the user to experiment with different segmentation methods and parameters to achieve the desired results. VolSeg is intended for users with some experience in image segmentation techniques, algorithms and methods. The user can fine-tune several algorithm parameters, which provides a lot of flexibility, but this may require previous knowledge of the applied algorithm functionality and usage.

<p>VolSeg was developed by Softeco Sismat S.r.l.</p> <p><b>hasFunctionality:</b> Segmentation, Visualization, SurfaceMeshing</p> <p><b>hasInputFormat:</b> dcm, dicom, mha, mhd</p> <p><b>hasOutputFormat:</b> mha, mhd, vtk</p> <p><b>writtenWithProgrammingLanguage:</b> C++</p> <p><b>hasDevelopmentStatus:</b> Beta</p> <p><b>hasUIType:</b> Graphical UI</p> <p><b>hasExecutionPlatform:</b> Windows</p> <p><b>hasLicense:</b> Commercial</p>	
<p><b>hasName:</b> ITK-SNAP</p> <p><b>hasDescription:</b> ITK-SNAP is free, open-source, and multi-platform software application used to segment structures in 3D medical images. ITK-SNAP provides semi-automatic segmentation using active contour methods, as well as manual delineation and image navigation. In addition to these core functions, ITK-SNAP offers many other supporting utilities.</p> <p><b>hasFunctionality:</b> Segmentation, Visualization</p> <p><b>hasInputFormat:</b> dcm, dicom, nifti, mha, mhd</p> <p><b>hasOutputFormat:</b> dcm, dicom, nifti, mha, mhd</p> <p><b>writtenWithProgrammingLanguage:</b> C++</p> <p><b>hasDevelopmentStatus:</b> stable</p> <p><b>hasUIType:</b> Graphical UI</p> <p><b>hasURL:</b> <a href="http://www.itksnap.org">http://www.itksnap.org</a></p> <p><b>hasExecutionPlatform:</b> Windows, Linux, Mac OS, other UNIX</p> <p><b>hasLicense:</b> GPL</p> <p><b>hasSourceCodeURL:</b>  <a href="http://www.itksnap.org/pmwiki/pmwiki.php?n=SourceCode.SourceCode">http://www.itksnap.org/pmwiki/pmwiki.php?n=SourceCode.SourceCode</a></p>	
<p><b>hasName:</b> Paraview</p> <p><b>hasDescription:</b> ParaView is an open-source, multi-platform data analysis and visualization application. Users can quickly build visualizations to analyze their data using qualitative and quantitative techniques. The data exploration can be done interactively in 3D or programmatically using ParaView's batch processing capabilities.</p> <p><b>hasFunctionality:</b> Visualization</p> <p><b>hasInputFormat:</b> dcm, mha, mhd, vtk, xyz, ply, obj, wrl, bmp, jpg, png, tiff</p> <p><b>hasOutputFormat:</b> stl, obj, wrl, vtk, ply, avi, mha, mhd, bmp, jpg, png, tiff</p> <p><b>writtenWithProgrammingLanguage:</b> C++</p> <p><b>hasDevelopmentStatus:</b> stable</p> <p><b>hasUIType:</b> Graphical UI</p> <p><b>hasURL:</b> <a href="http://www.paraview.org/">http://www.paraview.org/</a></p> <p><b>hasExecutionPlatform:</b> Windows, Linux, Mac OS, other UNIX</p> <p><b>hasLicense:</b> Other (BSD license)</p> <p><b>hasSourceCodeURL:</b> <a href="http://www.paraview.org/download/">http://www.paraview.org/download/</a></p>	
<p><b>hasName:</b> 3D Slicer</p> <p><b>hasDescription:</b> 3D Slicer, is a free, open source software package for visualization (including volume rendering) and image analysis (including registration and interactive segmentation) of medical images and for research in image guided therapy. It supports multi-modality imaging including MRI, CT, US, nuclear medicine, and microscopy.</p> <p><b>hasFunctionality:</b> Visualization, Registration, Segmentation</p> <p><b>hasInputFormat:</b> dicom, dcm, nifti, nrrd, mha, mhd, vtk, stl, obj, bmp, jpg, png, tiff</p> <p><b>hasOutputFormat:</b> dicom, nifti, nrrd, mha, mhd, vtk, stl, bmp, jpg, png, tiff</p>	

<p><b>writtenWithProgrammingLanguage:</b> C++  <b>hasDevelopmentStatus:</b> stable  <b>hasUIType:</b> Graphical UI  <b>hasURL:</b> <a href="http://www.slicer.org/">http://www.slicer.org/</a>  <b>hasExecutionPlatform:</b> Windows, Linux, Mac OS, other UNIX  <b>hasLicense:</b> Free  <b>hasSourceCodeURL:</b> <a href="http://www.slicer.org/pages/SourceCode">http://www.slicer.org/pages/SourceCode</a></p>	
<p><b>hasName:</b> MITK (The Medical Imaging Interaction Toolkit)  <b>hasDescription:</b> The Medical Imaging Interaction Toolkit (MITK) is a free open-source software system for development of interactive medical image processing software. MITK combines the Insight Toolkit (ITK) and the Visualization Toolkit (VTK) with an application framework. As a toolkit, MITK offers those features that are relevant for the development of interactive medical imaging software covered neither by ITK nor VTK.  <b>hasFunctionality:</b> Visualization, Diffusion, Registration, Segmentation  <b>hasInputFormat:</b> dicom, nrrd, mha, mhd, vtk, bmp, jpg, png, tiff  <b>hasOutputFormat:</b> dicom, nrrd, mha, mhd, vtk, bmp, jpg, png, tiff  <b>writtenWithProgrammingLanguage:</b> C++  <b>hasDevelopmentStatus:</b> stable  <b>hasUIType:</b> Graphical UI  <b>hasURL:</b> <a href="http://mitk.org/wiki/The_Medical_Imaging_Interaction_Toolkit_%28MITK%29">http://mitk.org/wiki/The_Medical_Imaging_Interaction_Toolkit_%28MITK%29</a>  <b>hasExecutionPlatform:</b> Windows, Linux, Mac OS, other UNIX  <b>hasLicense:</b> Other (BSD license)  <b>hasSourceCodeURL:</b> <a href="http://mitk.org/wiki/Downloads#MITK_Source_Code">http://mitk.org/wiki/Downloads#MITK_Source_Code</a></p>	
<p><b>hasName:</b> SemAnatomy3D  <b>hasDescription:</b> SemAnatomy3D provides a rich set of tools for 3D shape analysis that supports semantic annotation of patient-specific 3D models with concepts derived from the ontology and quantitative attributes.  <b>hasFunctionality:</b> Annotation, 3D shape analysis, Erosion detection  <b>hasInputFormat:</b> off, vtk, vrml  <b>hasOutputFormat:</b> txt, sem3D, segs  <b>writtenWithProgrammingLanguage:</b> Java  <b>hasDevelopmentStatus:</b> Alpha  <b>isAvailableAtInfrastructure:</b> IMATI  <b>hasUIType:</b> Graphical UI  <b>hasURL:</b>  <b>hasExecutionPlatform:</b> Windows  <b>hasLicense:</b>  <b>hasSourceCodeURL:</b></p>	
<p><b>hasName:</b> OsiriX  <b>hasDescription:</b> OsiriX is an image processing software dedicated to DICOM images. OsiriX has been specifically designed for navigation and visualization of multimodality and multidimensional images: 2D Viewer, 3D Viewer, 4D Viewer (3D series with temporal dimension, for example: Cardiac-CT) and 5D Viewer (3D series with temporal and functional dimensions, for example: Cardiac-PET-CT). OsiriX supports a complete dynamic plugins architecture. The 3D Viewer offers all modern rendering modes: Multiplanar reconstruction (MPR), Surface Rendering, Volume Rendering and Maximum Intensity Projection (MIP). All these modes support 4D data and are able to produce image fusion between two different series (PET-CT and SPECT-CT display support). OsiriX is at the same time a DICOM PACS workstation for imaging and an image processing software for medical research (radiology</p>	

and nuclear imaging), functional imaging, 3D imaging, confocal microscopy and molecular imaging.

**hasFunctionality:** Visualization

**hasInputFormat:** dcm, dicom, jpg, png, tiff, avi

**hasOutputFormat:** dcm, dicom, jpg, tiff

**writtenWithProgrammingLanguage:** Objective-C

**hasDevelopmentStatus:** Stable

**isAvailableAtInfrastructure:**

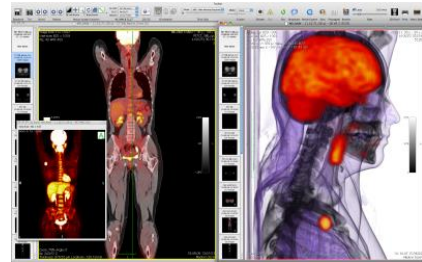
**hasUIType:** Graphical UI

**hasURL:** <http://www.osirix-viewer.com>

**hasExecutionPlatform:** MacOS

**hasLicense:** LGPL

**hasSourceCodeURL:**



## 4 Semantically Enriched Web Services

The semantic organization and description of Web Services is a crucial requirement for enabling the effective composition of Web services. Our primary objective is to demonstrate how formalizing, sharing and re-using expert knowledge can effectively improve shape processing and scientific computations in general. Our aim is to provide a framework that can support the development of sophisticated (“intelligent”) services which can be utilized according to user’s needs or context.

This can be achieved via a structured information space where resources are documented with reference to domain specific ontologies, enabling efficient discovery and access for both humans and computer programs. By linking workflow definitions back to the ontology, it is possible not only to find resources (data and programs/tools) that can be combined into workflows but also to find workflows which can produce data objects of a given class/category that are automatically annotated with metadata according to the AIM@SHAPE Common Shape Ontology.

Our approach towards a system architecture and implementation combines the principles of Web Services and the Semantic Web to support geometry processing tasks/activities for a given application domain. The specification of dynamic processes brings together, at runtime, a set of functional elements in order to implement an application-specific functionality.

The main benefits of our approach is mainly focused on the fact that data can be produced dynamically as a combination of other existing data and programs/tools, combined together as workflows. Hence, it is possible not only to discover and access the currently available objects but also to generate new ones by launching a computation. In addition, when a computation finishes the produced data can be

automatically annotated with metadata that capture its provenance, and can be stored in the ontology-driven knowledge base of the VVS.

What is equally important is the classification of the tools / workflows functionality at a semantic level. In our case, the focus is on assigning this functionality in components that are well-documented and at the same time easily accessible through Web Services. This is implemented via ontologies that contain concepts (classes) and instances (objects) of resources relevant to the particular application domain. The properties of these resources are described via metadata which also contain information about the location and protocols/interfaces through which they can be accessed or invoked. Such an organization greatly simplifies the task of navigation and searching in a large information space.

We can identify three kinds of semantic descriptions: data semantics, functionality semantics and user task (process) semantics. Data semantics of shape objects are captured by ontology-driven metadata. Functionality semantics are formalized by the application of (Semantic) Web Services. Processing task semantics are formalized through workflows.

The notion of a workflow, defined as an ordered sequence of tasks or activities, related by data and control flow relationships, is used to describe the computational aspect of the creation of processing pipelines. A task is typically performed by executing a program or invoking another process.

#### **4.1 Ontology-Driven Service Discovery and Composition**

By augmenting web services with semantic descriptions can result in a more automatic management of these services. Specifically, web service discovery, composition and mediation can become dynamic, with software agents able to reason about the functionalities provided by different web services, to locate the best ones for solving a particular problem and to automatically compose the relevant web services into workflows and build applications dynamically.

Service composition is not a trivial problem. It must be as less manual as possible (preferably fully automatic) and deal with discovering complex service combinations, conditions and preferences requirements. The purpose of discovery in the context of abstract workflow mediation is to provide mechanisms for finding services which can deliver the missing pieces of information or can bridge the identified incompatibilities by exploiting the knowledge stored in the system.

Enabling access to scientific applications and tools using a service-oriented approach allows the tools developers to focus on the domain science, and delegate the

management of the complex back-end resources to others who are more proficient in SOA middleware.

We address these problems by developing a conceptual and technological platform able to encapsulate 3D resources into a semantic layer for efficient storage, retrieval and reuse of these resources. This objective is achieved by providing access to existing scientific and computational resources and by allowing users to assemble these resources together in order to generate complex functionalities based on Web Service workflows. The capability to compose complex services strongly depends on the conceptualization of the domain. The conceptualization of the relevant concepts and the relations among them are represented by ontologies able to capture expert knowledge.

The development of the VVS knowledge-based system included the following:

- Formalization of domain knowledge through ontologies, starting from the basic building blocks (tools, services, applications, data sets), that allow a shared understanding of the domain and make explicit descriptions of generic functionalities.
- Grouping of basic functional elements in order to implement application-specific functionality. This is addressed by the development of Web Service workflows.
- Incorporation of semantics into service descriptions and service composition through the use of loosely coupled, reusable software components.
- Knowledge discovery that can help the users assemble relevant information for effective decision-making and semantically enriched services, to improve their capability to perceive and utilize system knowledge. This can also help the users discover and assemble services into processes for easier and better quality of workflow executions.

## **5 Overview of the VVS Service-Oriented Architecture**

Composing services together is a challenge for SOA middleware meeting e-Science environments and scientific computing. The variety of services requires the development of models, techniques and algorithms in order to create composite services and execute them. Composing services is technically performed by chaining interfaces using a syntactic or semantic method of matching.

The specification of a service composition requires dealing with two major issues: service description and orchestration. Services are perceived as black-box components with well-defined interfaces, they are accessed using XML message exchanges and metadata in the form of WSDL are used to describe abstract interfaces and concrete endpoints.

The execution order and conditions of such a workflow are specified through an orchestration model/language. The generation of composite services requires a clear and unambiguous description. In a SOA-enabled environment, we build application workflows by orchestrating various services in the right order (sequencing, conditional behavior etc.) via BPEL<sup>1</sup>. BPEL supports the description of abstract and executable processes.

The orchestration of executable processes is then deployed and executed by a workflow engine in a Java Business Integration (JBI<sup>2</sup>) enabled platform (see Figure 6 **Errore. L'origine riferimento non è stata trovata.**). JBI provides an open application integration framework, is built on a Web Services model and provides a pluggable architecture for different Service Engines (SE). It is a standard meta-container for integrating service containers that can host service producer and consumer components (service units). OpenESB<sup>3</sup> implements an Enterprise Service Bus (ESB) runtime using JBI as the foundation. This allows easy integration of web services to create loosely coupled enterprise class composite applications. It also provides various tools for the development, deployment, and management of composite applications.

The Java EE Service Engine acts as the bridge between Java EE applications and JBI. A Java EE application archive (ear/war/jar) can be packaged in a JBI composite application and be deployed in a JBI server. We are using the JBI runtime that has been integrated with our GlassFish application server.

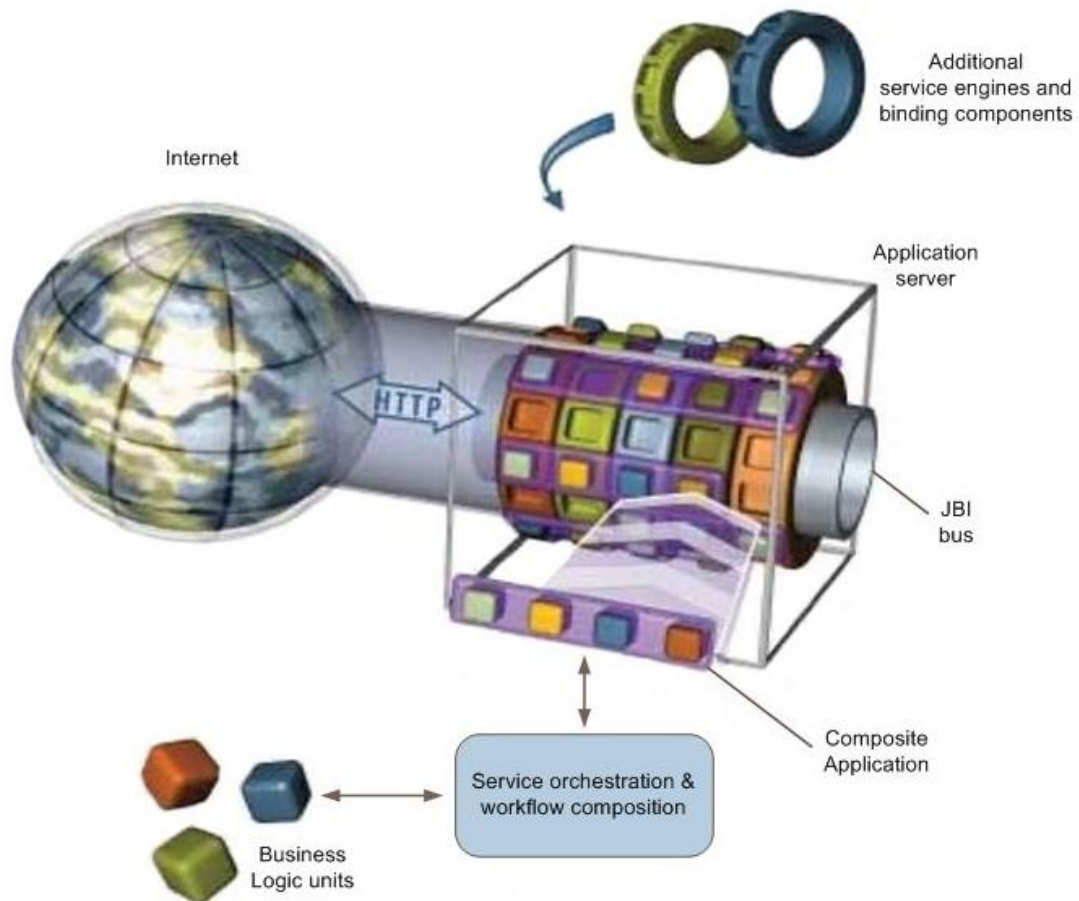
---

<sup>1</sup> [https://en.wikipedia.org/wiki/Business\\_Process\\_Execution\\_Language](https://en.wikipedia.org/wiki/Business_Process_Execution_Language)

<sup>2</sup> [https://en.wikipedia.org/wiki/Java\\_Business\\_Integration](https://en.wikipedia.org/wiki/Java_Business_Integration)

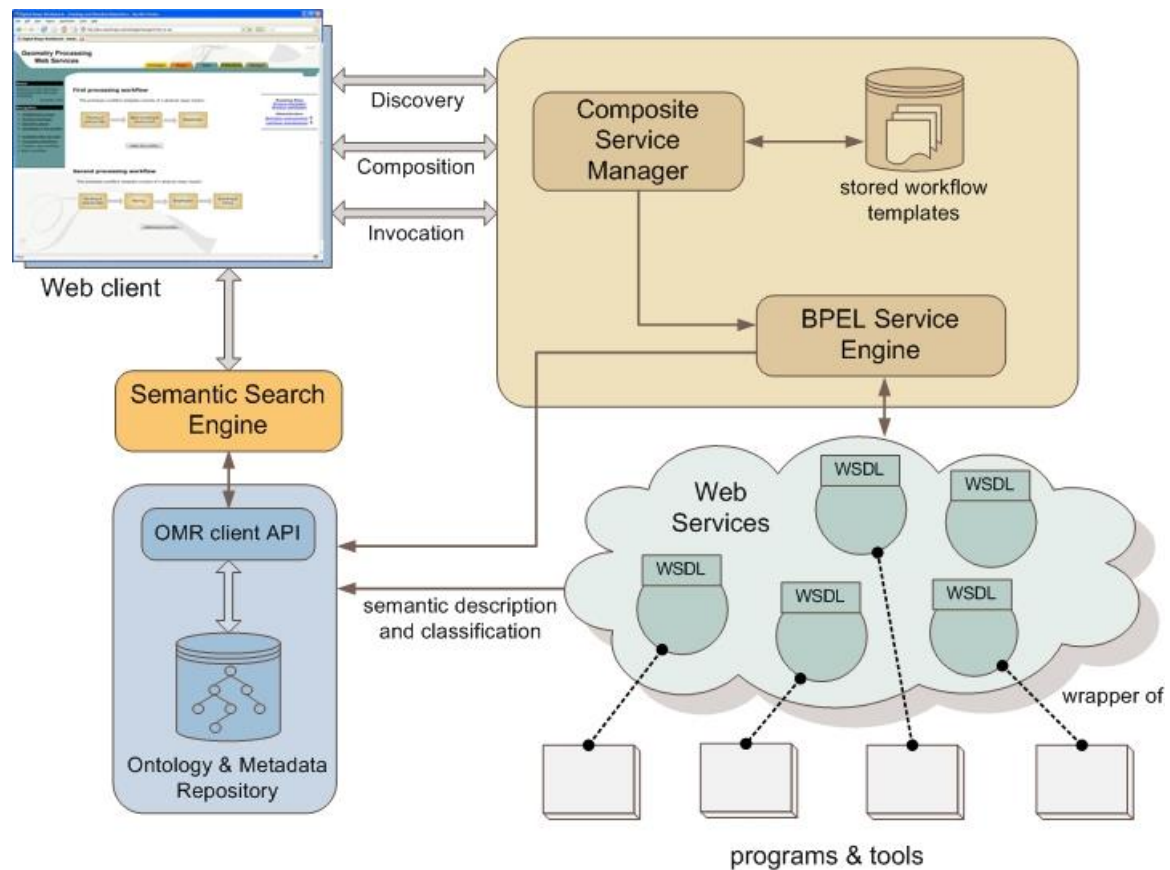
<sup>3</sup> [https://en.wikipedia.org/wiki/Open\\_ESB](https://en.wikipedia.org/wiki/Open_ESB)





**Figure 6.** Service oriented JBI architecture

Our architecture, presented in Figure 7, is organized into three layers. The first layer includes a set of scientific applications and tools. The second layer contains associated Web Services. These Web Services could be either simple or composite. Some of the simple Web Services just wrap applications and tools defined in the first layer. Composite services are generated as processing chains (workflows) of simple (single-step) Web Services. The third layer includes the service composition component, the BPEL service engine and a Web client GUI and manager. Three types of requests are submitted through the GUI: discovery, invocation, and composition of Web Services into workflows. During the service composition process, there is communication with the knowledge base for discovery and selection of services. It is also possible to utilize stored workflow templates (abstract workflow definitions) for the generation of new processing services. These abstract workflows are stored in the form of deployed BPEL modules.



**Figure 7.** The semantically enriched Web Service architecture

### 5.1 Dynamic Workflow Composition and Execution of Web Services

Dynamic selection and composition of Web Services is increasingly used to automate either scientific or business processes. There are three different ways to invoke a Web Service:

- static binding
- dynamic binding
- dynamic invocation

With static binding the client is compiled and binded at development time. This binding is tightly bound to one and only one service implementation. It provides the fastest performance but gives the least flexibility.

With dynamic binding the only part of the client that is compiled at development time is the interface to a service type (i.e. the WSDL *portType* definition). At runtime the client can bind to any service implementation that supports that *portType*. It generates a dynamic proxy from the service's WSDL binding at runtime and casts it to the interface. There is a slight performance reduction though, but in exchange you

win a lot of flexibility. Using this technique, the application is able to connect to any number of different service implementations without modification.

With dynamic invocation, there is no need to compile anything at development time. Instead everything is done at runtime. The application retrieves and interprets the WSDL at runtime and dynamically constructs calls. This method gives the most flexibility, but also requires a much more sophisticated client and there is a decrease in terms of performance, which occurs on each invocation.

*Partner links* describe interfaces (messages and operations), transport protocol, and most importantly, the location of each service to be used. BPEL processes call these external services using information stored in the *partner links*. The partner links define operations and message types that make up the interface to the service using *portTypes* in WSDL. *portTypes* also indirectly define the transport used to communicate with the service (bindings) and the location of the service (service address).

Commonly, process designs in BPEL include static *partner links* that refer to a single external process selected by the developer at design time. This approach is appropriate for most systems; however, scientific workflow processes are more complex. They could interact with multiple external services and define multiple *partner links*, and some of these *partner links* might not be known at design time. As a result, all potential callouts and logic for deciding which *partner links* to use is usually built inside the business process itself, unnecessarily complicating that process. Furthermore, as additional services are added, the resulting process grows more and more unwieldy, as any changes to the *partner links* require modification of the entire process.

In our implementation we are using the **dynamic binding** technique. This approach eliminates the need to anticipate and manage all relationships at design time. The BPEL language supports the concept of dynamic binding of *partner links* by shielding processes from web service changes and letting the system manage *partner links* dynamically at runtime.

The WS-Addressing<sup>4</sup> standard provides a mechanism called endpoint references (EPR) that allows selecting one of the available services or even defining new services at runtime. The process statically depends on the interface information defined in the *portType* whereas an endpoint reference (which maps the binding to

---

<sup>4</sup> Web Services Addressing 1.0 – Core <http://www.w3.org/TR/ws-addr-core/>

the service) allows us to redefine the service location dynamically. In essence, the endpoint reference is a dynamic alternative to the static service element defined in the WSDL. In our case, the process/workflow designer can remain isolated from the decision about which services to call as long as those services conform to a standard (common) interface.

In our approach we use our knowledge-based framework for dealing with service composition and orchestration. Instead of creating hard-coded service discovery and routing logic, we utilize dynamic knowledge-driven service selection and binding mechanisms, triggered by predefined knowledge captured in the ontology.

This framework is used for:

- (a) Service selection to control how concrete service endpoint references are assigned to abstract process activities. Specific policies for service endpoint selection could be used to find the best suited service instead of manual selection.
- (b) Controlling runtime rebinding by user-defined service preferences and constraints to enable automatic knowledge-based selection of service endpoints. Usually policies attached to process activities are matched with services in order to find an optimal configuration that satisfies required user preferences and constraints. The selection could cover several QoS criteria such as performance, accuracy, reliability, availability, cost etc.

Ontology-driven knowledge controls the way discovery, selection, and binding are managed at runtime. Dynamic (late) binding is the process of transparently mapping an abstract service to a concrete service instance at runtime. Web Services registries do not provide support for dynamic binding. Hence, the service client is responsible for service selection and rebinding. Typically, the client queries the registry based on certain criteria, retrieves a list of services and then manually selects one service and tries to invoke it. The limitations of this approach become evident especially when dynamic rebinding becomes necessary. The main problem is that the service client is responsible for taking all corrective actions (e.g., re-querying the registry for new service bindings, polling the registry for possibly new services, etc.). This requires considerable client-side code since current registries (and their APIs) do not provide support for implementing such dynamic binding.

Furthermore, the logic for discovering and selecting the best service (according to some criteria) cannot be easily encoded to the registry. Most registries are organized according to some kind of taxonomy and are using keyword search for service discovery. In addition, most registries focus on business / commercial Web Services

and are usually insufficient for scientific Web Services where relationships between classes need to be defined or semantic searching is required. Because of these limitations, we are not using UDDI or ebXML registries. Instead, we rely on our own ontology-based repository (OMR) for managing and querying the services metadata.

Our approach aims to address the aforementioned limitations. Dynamic binding is handled transparently in combination with the service discovery. The service selection and (re)binding are enforced by our knowledge-based middleware. The latter queries our repository to find services matching the given criteria. Based on the available services and knowledge stored to or inferred from the ontology, the user selects the service that best fulfils the functional constraints.

Our runtime platform relies on the OpenESB<sup>5</sup> BPEL service engine to execute a process, is able to trigger dynamic binding when needed to support composite scientific workflows and can be easily integrated with the NetBeans IDE and GlassFish application server.

## 5.2 Shape Processing Web Services and workflows

Due to the intrinsic complexity of 3D models, ontology-driven metadata are necessary in order to reach a sufficient level of expressiveness and semantic impact. Metadata provide a thorough characterization of models by capturing information related to the processing history of an object, the possible actions that can be performed to it (e.g. smoothing, simplification, enhancement etc.), or the tools/services that can use it as input or produce it as output.

In addition, the formalization of the relations among tasks/activities (e.g. workflow steps), programs/tools (e.g. Web Services) and data (e.g. 3D models) is necessary to specify processing pipelines. The Common Tool Ontology (CTO) can support the description of workflows by using the concept of *SoftwareTool*.

The concept *Functionality* represents all the possible different tasks/activities a tool can provide (e.g. Triangulation, Voxelization, Filtering etc). Instances of these functionalities can have references to specific tools in the Tool Repository or executable Web Services.

Typically, a requestor/user applies a specific task on a particular shape, whereas service providers publish generalized descriptions of their service capabilities that will enable clients to find them. For effective service selection to occur, functionality

---

<sup>5</sup> <http://www.open-esb.net/>

queries should be more abstract and include information about how the task should be achieved, and under what constraints/conditions. This use of abstract functionality descriptions is one of the reasons we developed the *Functionality* class hierarchy of the *Common Tool Ontology*.

## 6 Implementation of Shape Processing Applications as Services

Our objective is to demonstrate how formalizing, sharing and re-using expert knowledge can effectively improve scientific computing in general and shape processing in particular, especially focused on the medical domain. By embedding semantics and domain knowledge in the different stages of processing, we enhance the 3D processing pipeline, allow the re-use of valuable resources (e.g., existing content, processing tools/services, workflows), contribute to efficient resource discovery and support the composition and execution of workflows. All of the above can improve the different kinds of medical diagnosis procedures, can contribute in the monitoring and treatment of patients, and ultimately assist and support medical professionals in a wide range of clinical decision-making, research, teaching and learning activities.

### 6.1 Development of single-step Web Services

With the use of the MeshLab<sup>6</sup> mesh processing system, we exported several 3D shape processing tools as Web Services. We selected a subset of the most commonly used filters in geometry processing in general, which are also used in medical shape processing.

MeshLab is a free and open-source general-purpose mesh processing system, which was developed by ISTI-CNR in the framework of the EPOCH Network of Excellence. It is designed to help the flow and adaptation of 3D models that typically occur in the pipeline when processing 3D data. MeshLab provides many mesh processing functionalities.

To enable web access to a selection of MeshLab tools, we wrapped them into Web Services using the command line version of MeshLab (meshlabserver) which takes as input an XML formatted "filter script" i.e. sequences of filtering actions.

More specifically, the following thirteen (13) Web Services are currently implemented and available:

---

<sup>6</sup> <http://meshlab.sourceforge.net/>

- **Cleaning:** Remove Duplicate Faces, Remove Duplicate Vertex
- **Smoothing:** Laplacian Smooth, Smooth Face Normals, Two Step Smooth
- **Reconstruction:** Poisson Surface Reconstruction
- **Simplification:** Clustering Decimation, Quadric Edge Collapse Decimation
- **Sampling:** Clustered Vertex Subsampling, Regular Recursive Sampling, Stratified Triangle Sampling
- **Meshing:** Marching Cubes (APSS), Marching Cubes (RIMLS)

Each of the developed Web Services is described by a set of metadata information and is classified according to the *Functionality* hierarchy of the *Tool Common Ontology* (TCO). Every service is associated to a *Software Tool* instance in the TCO ontology and more specifically to the *WebService* class.

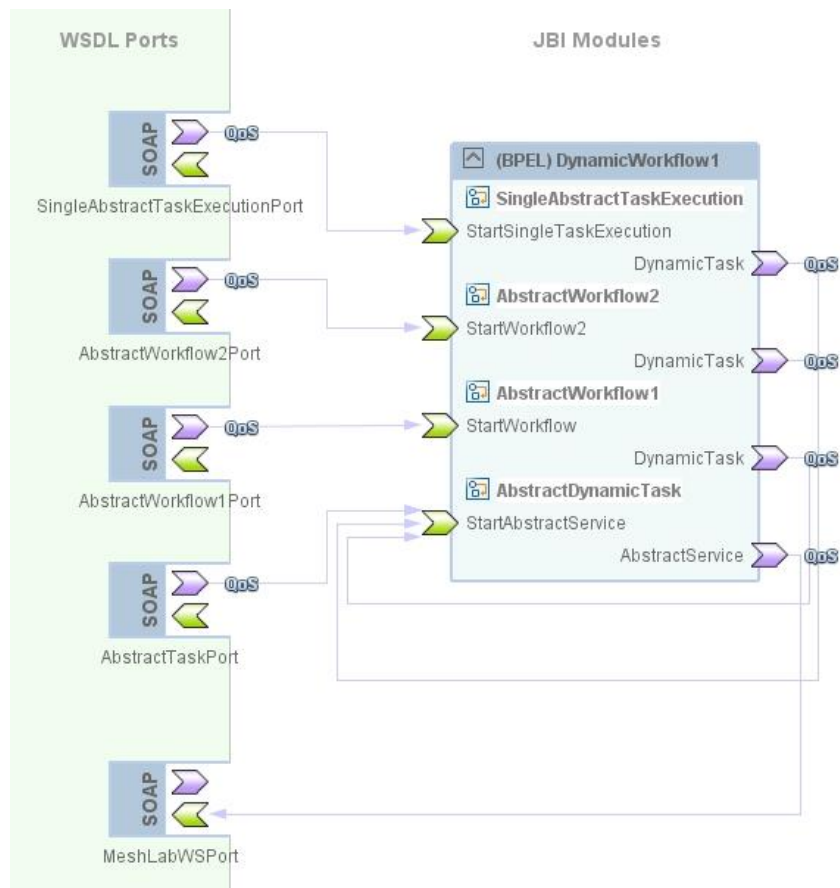
## 6.2 Development of the Web Service Workflows

Composing services into a workflow, given specific target functionalities, implies to deal with the following:

- the services discovery and selection;
- the service composition and orchestration according to a specific functionality;
- to automate the composition process as much as possible.

We used the SOA module of the NetBeans IDE (integrated with OpenESB and GlassFish) for the development of our Web Services, the BPEL process modules, the composite applications and the web user interfaces. The Web Services and all the other web modules are deployed using the GlassFish application server which provides not only server-side infrastructure for deploying and managing services, but also client-side API for invoking those services.

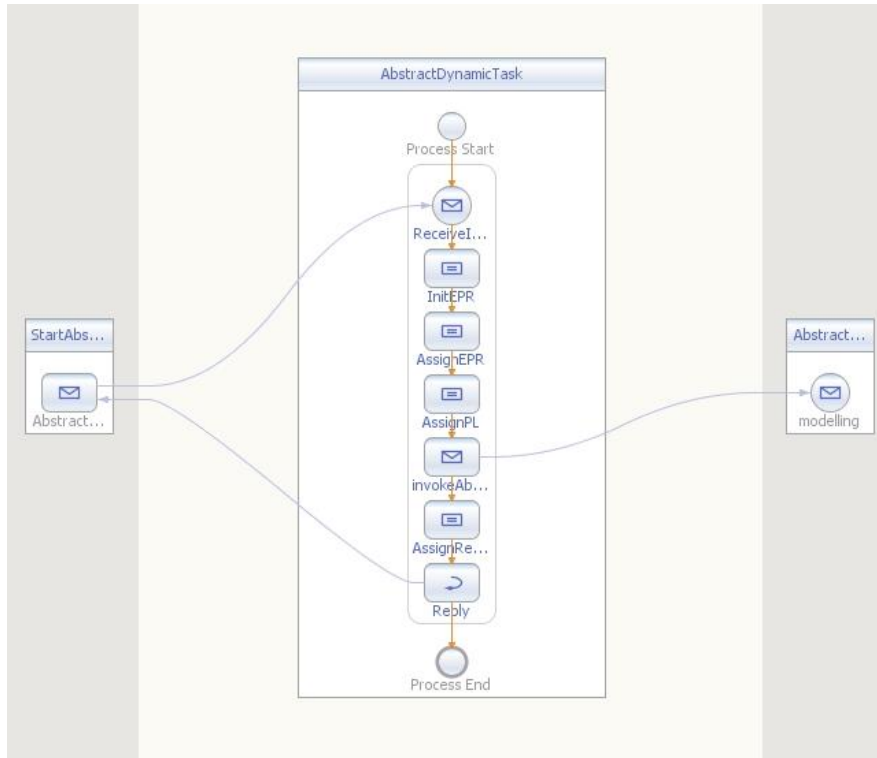
Figure 8 gives an overview of the developed service endpoints (WSDL ports), the JBI module with all our BPEL processes and the connections between them (WSDL bindings). This design view has been created using the Composite Application Service Assembly (CASA) editor provided with NetBeans.



**Figure 8.** Design view of the developed BPEL processes.

The designer views of our BPEL process diagrams are shown in the following figures. Figure 9 **Errore. L'origine riferimento non è stata trovata.** illustrates the designer view (BPEL editor from NetBeans) of our abstract task implemented with dynamic endpoint reference (dynamic binding). The purpose of this abstract task is to be able to invoke any kind of Web Service implementing the same interface.

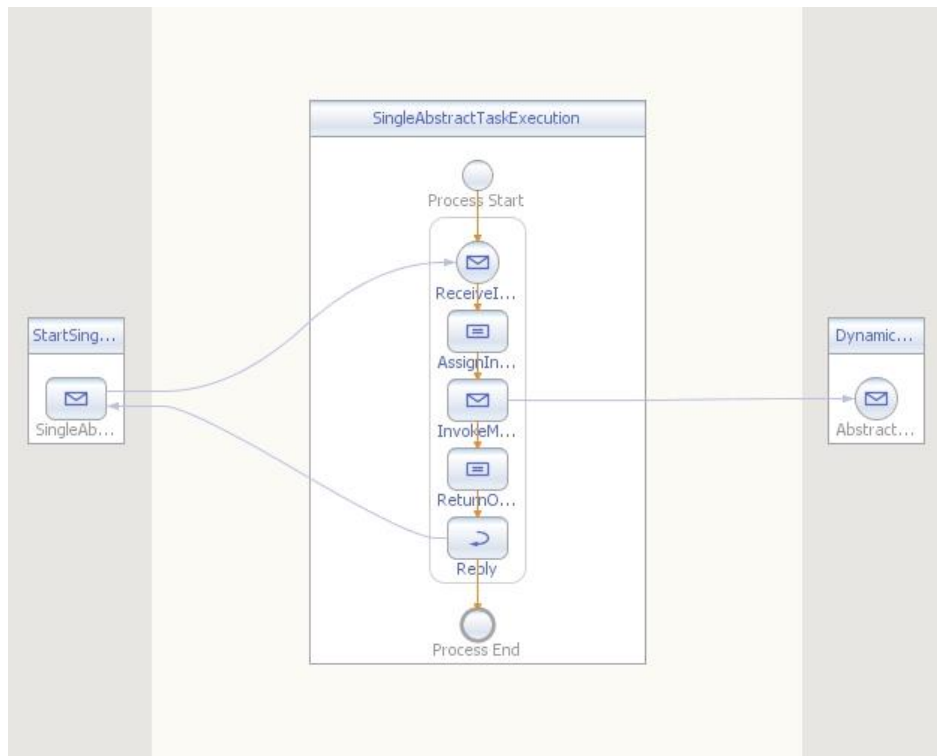




**Figure 9.** BPEL editor view of our abstract task with dynamic binding.

Figure 10 gives the design of a single-step execution service which invokes the abstract task process shown in Figure 9, i.e. the *partner link* on the right in Figure 10 corresponds to the *partner link* on the left in Figure 9. This can be considered as a special case workflow containing only one step.

The designer's views of our two workflow scenarios are given in the following sections.



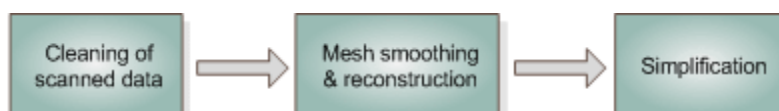
**Figure 10.** BPEL editor view of a single abstract task execution step.

Two geometry processing workflows were developed (more details are given in the following subsections) where service discovery and selection is done semi-automatically. The user can select appropriate services, using a functionality-based discovery method, for each abstract task (or *Activity* class, as it is defined in the *Workflow Ontology*) of the workflow.

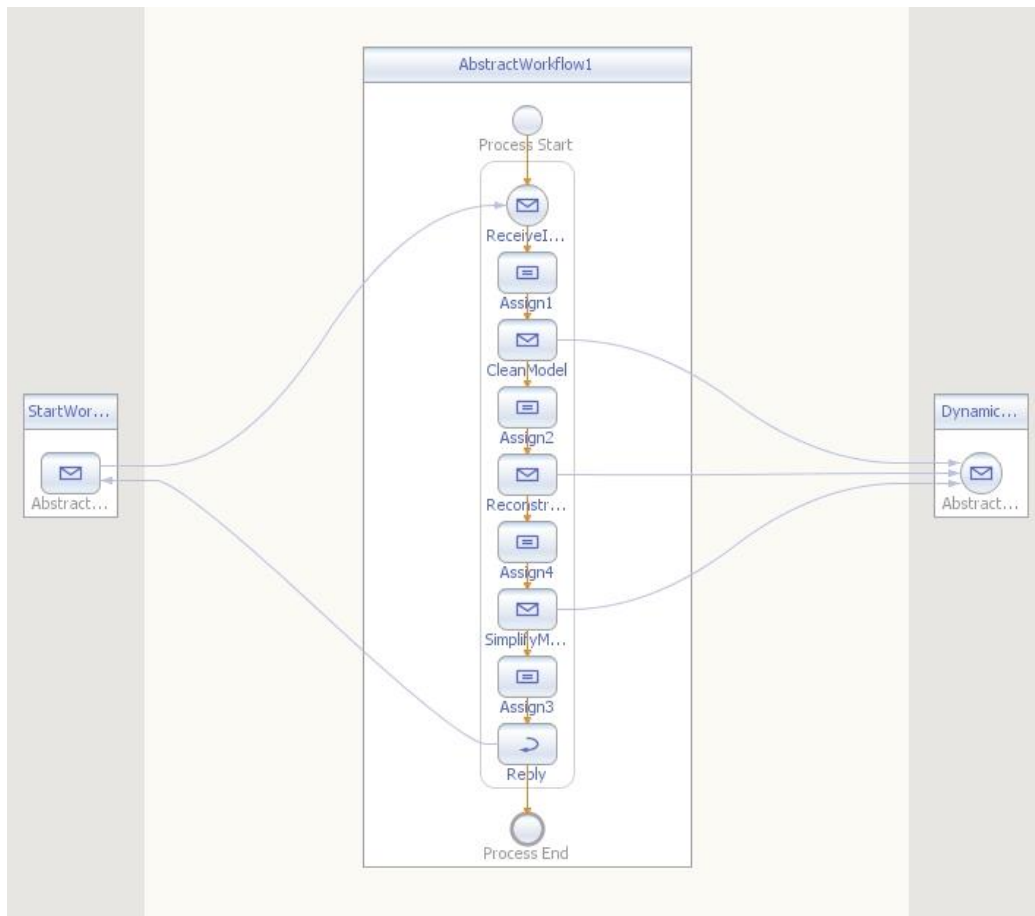
The following scenarios demonstrate the generic concepts of pre-processing, polygonal surface generation or reconstruction and post-processing, using two different pipelines.

### 6.3 First Web Service Workflow Scenario

The first workflow scenario was defined as follows:



This is the first of the predefined workflow templates that can be found in the VVS and represents a way to capture expert knowledge. The designer view of the above abstract workflow scenario (BPEL process diagram) is shown in Figure 11.



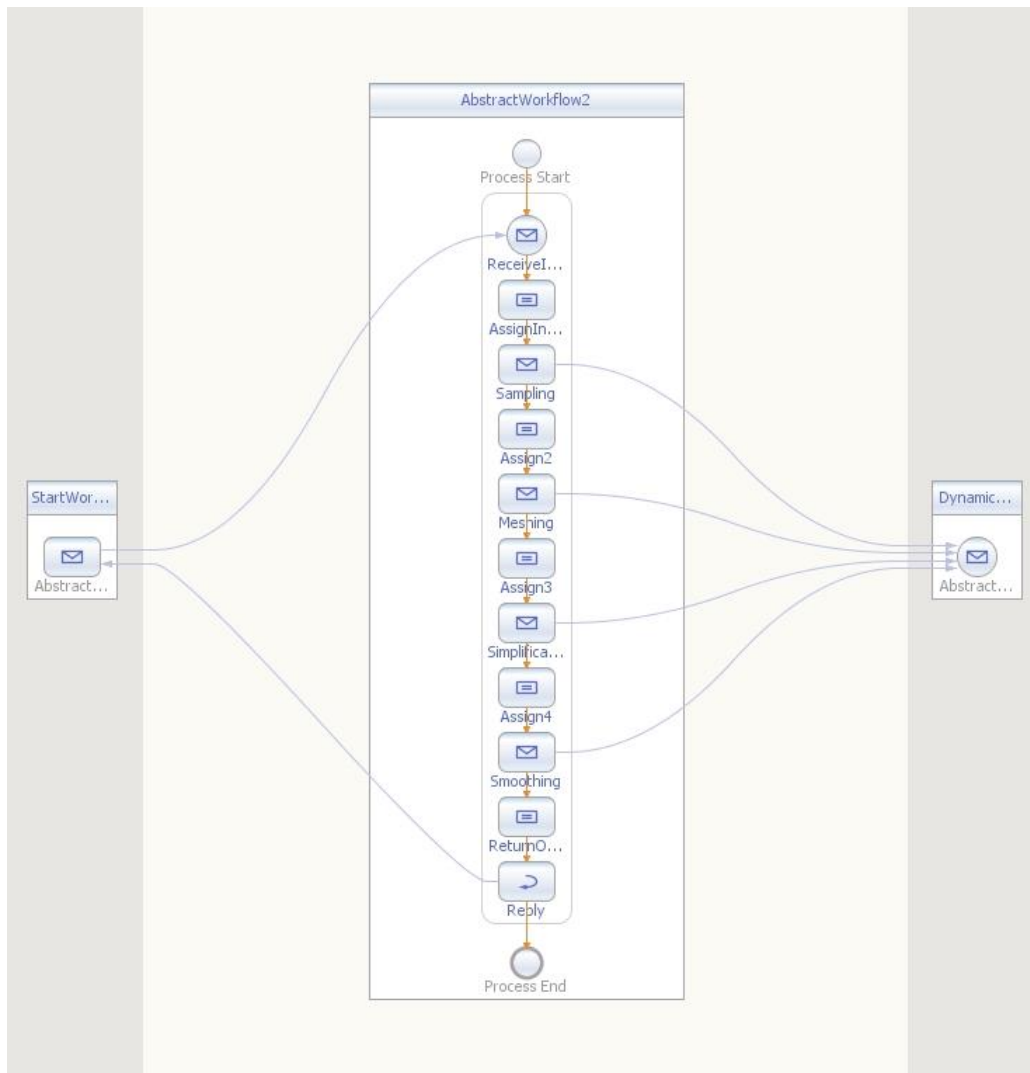
**Figure 11.** BPEL editor view of our first abstract workflow scenario.

#### 6.4 Second Web Service Workflow Scenario

The abstract workflow description for the second scenario was defined as follows:



The designer view of the above abstract workflow scenario (BPEL process diagram) is shown in Figure 12.



**Figure 12.** BPEL editor view of our second abstract workflow scenario.

## 7 The Web Services and workflows user interface

The Web Services UI provides a way to dynamically execute the available Web Services and Web Service workflows is currently available here: <http://visionair-v1.ge.imati.cnr.it/ws/>

A new subsection called “Executable Web Services” was added at the left side menu of the Workflows Repository. There are two main options: a) to execute a single-step web service and b) to execute one of the pre-defined dynamic web service workflows.

**Workflow Repository**

- Browse Workflows
- Upload Workflow
- Remove Workflow
- Upload Specific Tool
- Workflow Ontology
- Tutorial

**Executable Web Services**

- Single-step Web Services
- Web services workflows

## 7.1 Single-Step Web Services

The list of currently available Geometry Processing Web Services is shown in Figure 13. The user interface dynamically generates this list from the instances of the TCO class *WebService*. Additional information about each service is provided when the mouse pointer goes over the service name (tooltips). This information is also dynamically generated from the values of the datatype property called *hasDescription*. Additional information are also available by clicking on the “see more details” link, where the user is redirected to the web service description at the Tool Repository. By selecting a single-step service and pressing the execution button, the specific Web Service is invoked.

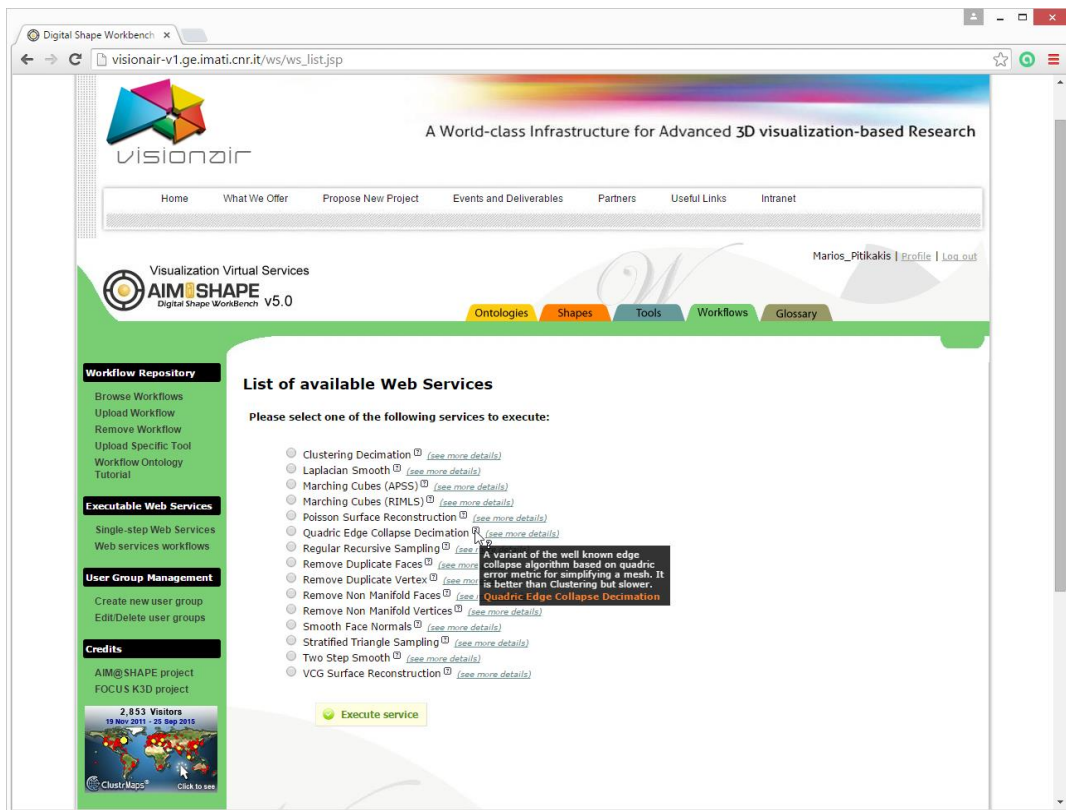


Figure 13. The user interface of a single-step Web Service selection.

After the Web Services selection, the user uploads the input model, or selects an existing model from the Shape Repository (SR), and initiates the workflow execution (see Figure 14). More details about acceptable input formats are given at the web service page. The list of existing model from the Shape Repository is dynamically generated according to the allowed input formats of the selected web service.

It is also possible to save the output model to the Shape Repository. If the user selects this option (see Figure 15), all the necessary metadata are automatically generated.

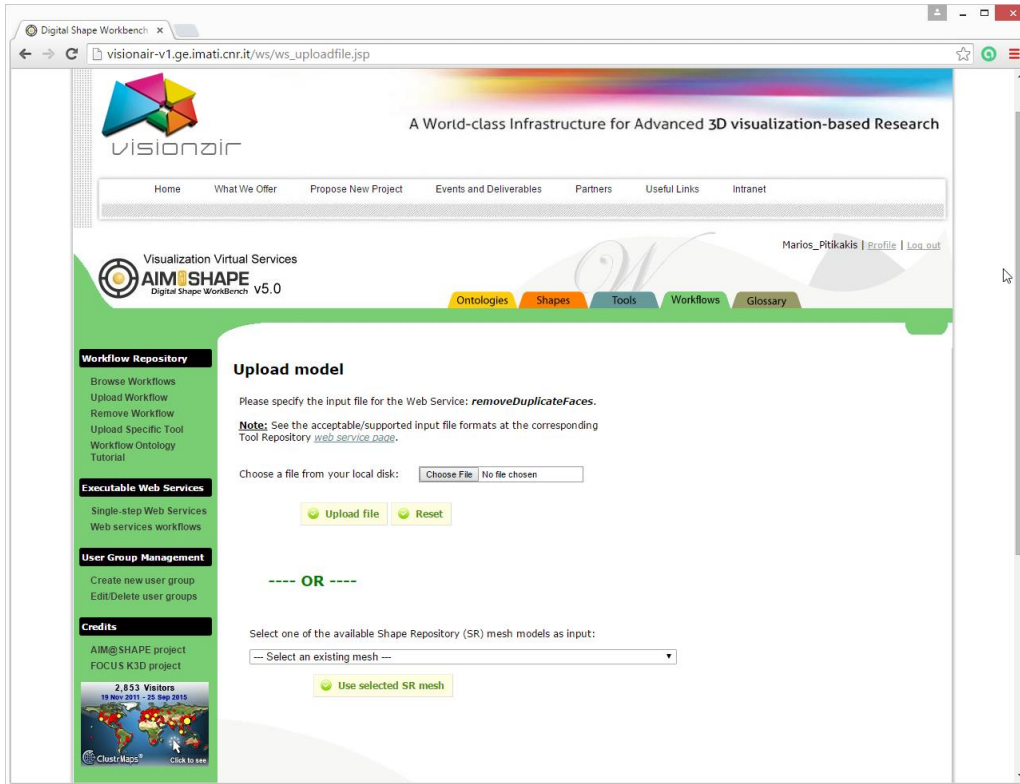


Figure 14. Upload an input model or select an existing model from the SR.

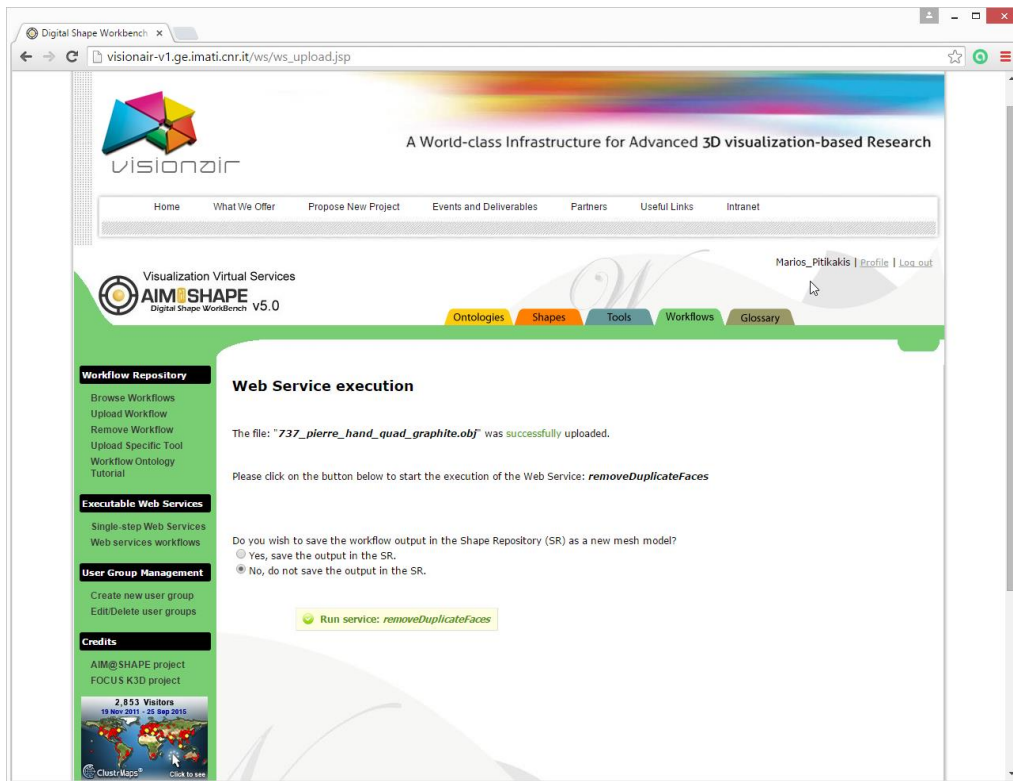
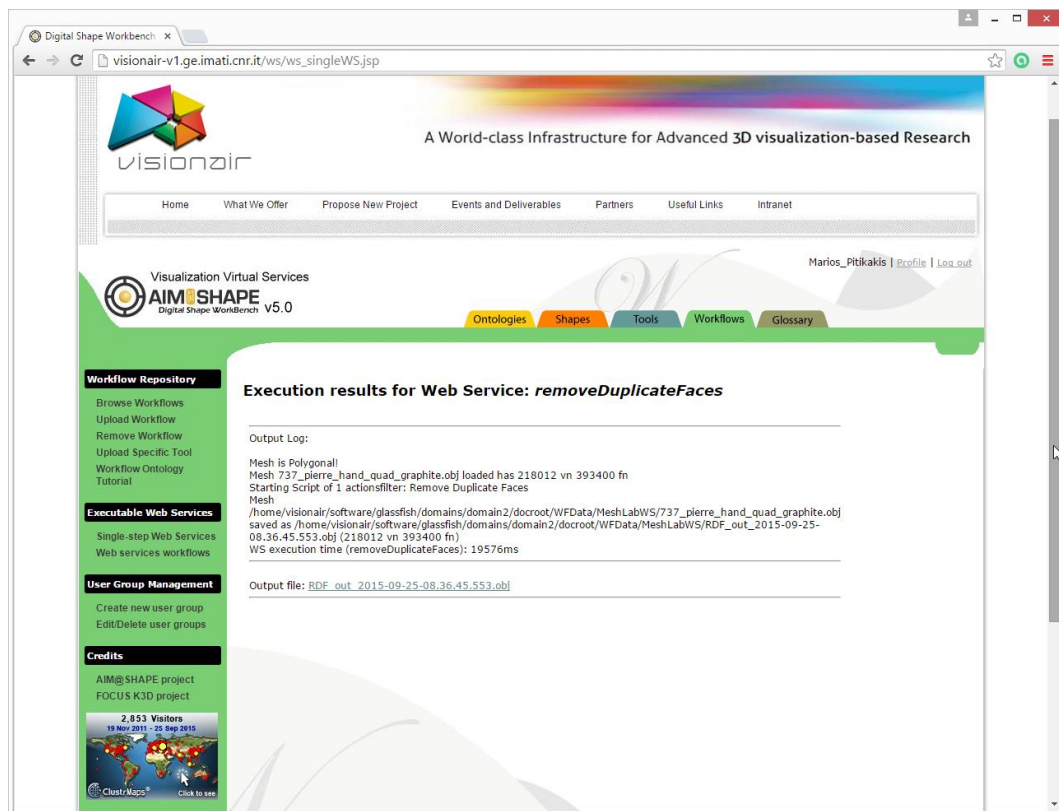


Figure 15. Ready to execute the selected web service

An example web service execution result page is shown in Figure 16. A summary of the output log is displayed (as generated by MeshLab) and the resulting output file is provided as a link and can be downloaded by the user.



**Figure 16.** Single-step web service execution result page (without saving the resulting model to the SR).

## 7.2 Web Services Workflows

The Web Services workflows page has currently two pre-defined dynamic workflows (see Figure 17), as described in the previous section.

The web interface of the first workflow scenario shows the abstract workflow definition (functionality descriptions) in a diagram and the user is prompted to assign concrete service instances to each task/activity of the abstract process. The selection of the specific Web Service instances is done from drop-down menus that are dynamically generated using the *Functionality* property of the *Software Tool* class i.e. the user selects specific web services that can be used to perform an abstract task (see Figure 18 for the first workflow and Figure 19 for the second workflow).

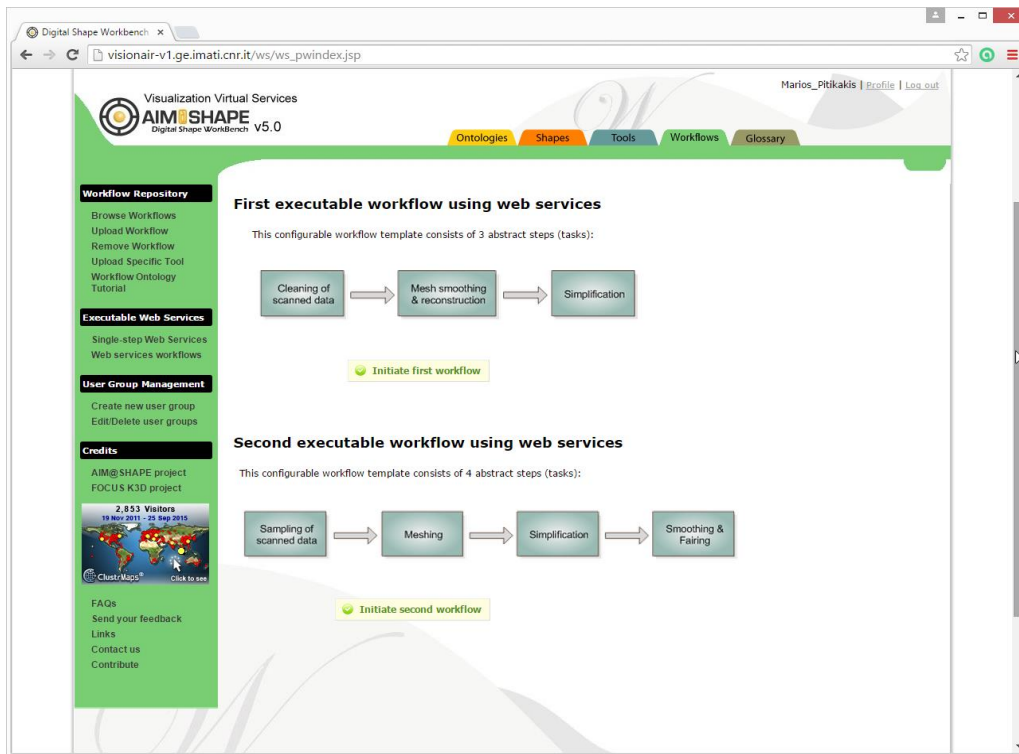


Figure 17. The web services workflows initial page.

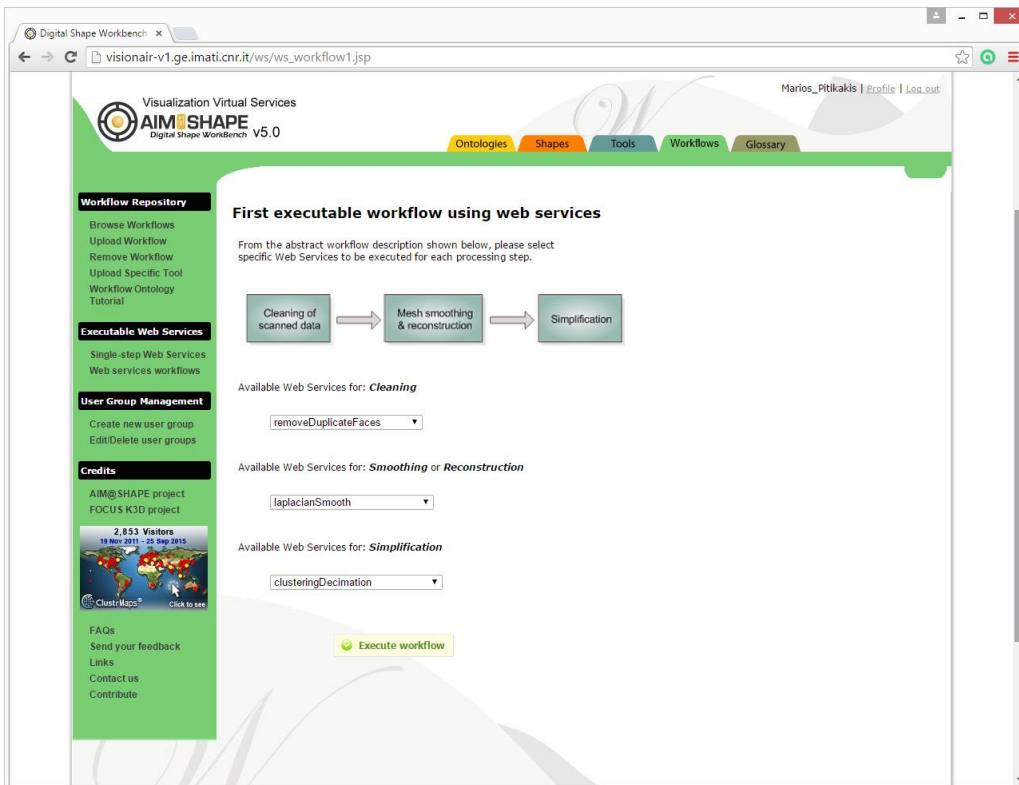
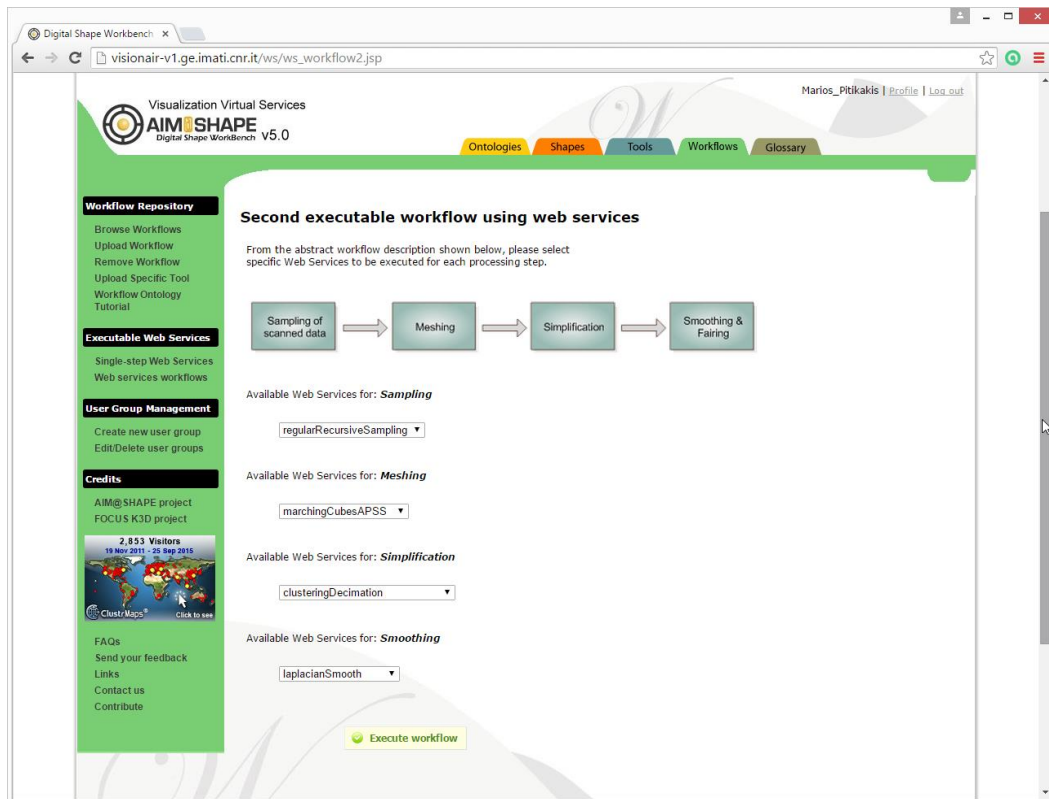


Figure 18. Selecting concrete web services for the execution of the first workflow.





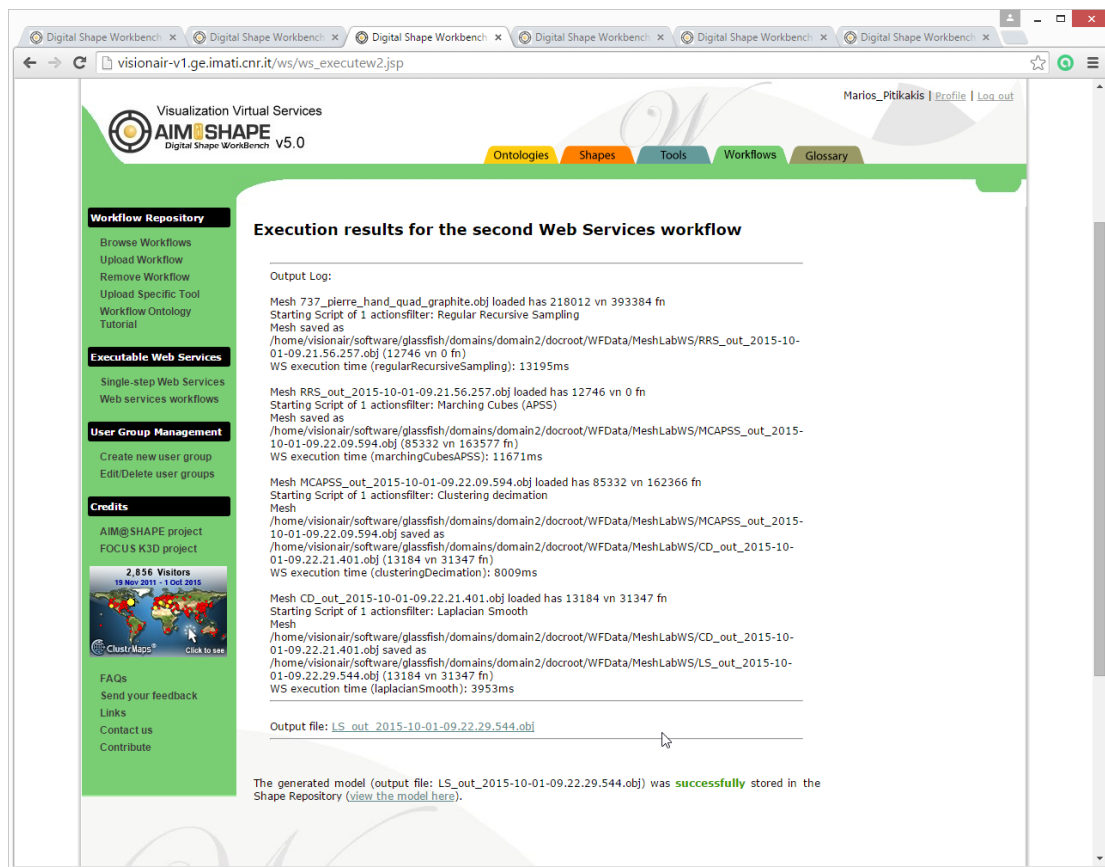
**Figure 19.** Selecting concrete web services for the execution of the second workflow.

After the Web Services selection for each workflow abstract task, the user uploads the input model, or selects an existing model from the Shape Repository, and initiates the workflow execution (similarly to the single-step web service). As described in section 4.1, the services are dynamically bound to each task of the abstract BPEL process.

Finally, a short summary of the execution log is displayed and the resulting output model is provided for download. In addition, as with the single-step web service, the workflow execution can automatically produce the appropriate metadata for the resulting (output) model i.e. processing history, documentation and other details (e.g. number of vertices, number of faces, model origin, file size and file format, location and URL etc.) and the resulting model and its metadata can automatically be stored back to the knowledge base if the user selects the corresponding option (see Figure 20).

Note that saving the generated model to the Shape Repository (SR) involves a number of steps: a) create a new ontology instance in the appropriate class/category of the Common Shape Ontology (e.g. *ManifoldSurfaceMesh*) and a new *FileInfo* instance, b) calculate and fill automatically some of the metadata (e.g. number of vertices, number of faces etc), c) update the SR cache table, d) generate

automatically a new thumbnail (applicable only for mesh models), e) compute the MT and signature of the model and added them in the Geometric Search Engine (GSE) database.

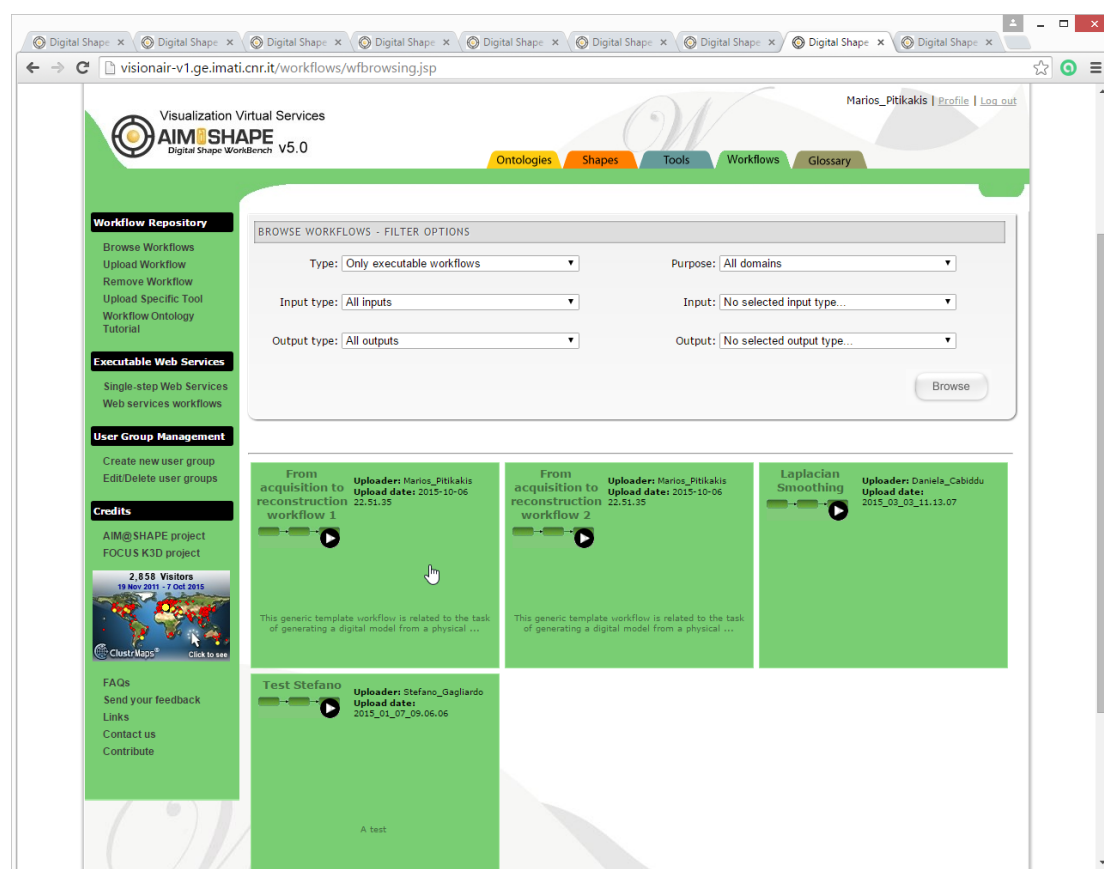


**Figure 20.** Result page of the second executable web services workflow and saving the resulting model to the SR.

## 8 Integration with the CAR2VR ontology

The two web service workflows described in the previous sections were inserted in the **Workflow Ontology** as instances of the class **WorkflowExecutable**, along with their workflow tasks (as instances of the **Activity** class). These two workflows mainly deal with the generation of a digital model from a physical object. The digital model may be further processed and several post processing tools can be applied to the digital model, depending on the application domain.

These new workflows can also be found in the *Workflow Repository* through the “Browse workflows” page (see Figure 21). The wf\_cache tables were updated as well.



**Figure 21.** Browsing for executable workflows in the *Workflow Repository*.

In addition, two new domains were added as instances to the **WorkflowDomain** class: “Generic geometry processing” and “Medical”.

Table 3 below shows all the instances created in the **Workflow Ontology** to support these new web service workflows.

**Table 3.** Detailed description of the inserted instances in the *Workflow Ontology*.

Instances of <i>WorkflowExecutable</i> class
<p><b>hasName:</b> From acquisition to reconstruction workflow 1</p> <p><b>hasDescription:</b> This generic template workflow is related to the task of generating a digital model from a physical object (from the acquisition to the final reconstruction of the model). Several post processing actions can be applied to the digital model, depending on the application domain. This workflow contains the following categories of activities: Cleaning of scanned data, Mesh smoothing &amp; reconstruction and Simplification. Upon execution, the user selects specific available web services that can be used to perform each of these abstract activities.</p>
<p><b>hasName:</b> From acquisition to reconstruction workflow 2</p> <p><b>hasDescription:</b> This generic template workflow is related to the task of generating a digital model from a physical object (from the acquisition to the final reconstruction of the model). Several post processing actions can be applied to the digital model, depending on the application domain. This workflow contains the following categories of activities: Sampling of scanned data, Meshing, Simplification and Smoothing &amp; Fairing. Upon execution, the user selects specific available web services that can be used to perform each of these abstract activities.</p>
Instances of <i>SimpleActivity</i> class
<p><b>hasName:</b> Cleaning of scanned data</p> <p><b>hasDescription:</b> Removing unwanted data, regions or elements (faces, vertices etc) from a model.</p> <p><b>isActivityOf:</b> From_acquisition_to_reconstruction_workflow_1</p> <p><b>correspondToFunctionality:</b> GeometryImprovement</p> <p><b>precedes:</b> Mesh_smoothing_and_reconstruction</p> <p><b>follows:</b></p>
<p><b>hasName:</b> Mesh smoothing and reconstruction</p> <p><b>hasDescription:</b> Building a surface mesh using a specific surface reconstruction method/approach. Before the reconstruction the model can be smoothed first.</p> <p><b>isActivityOf:</b> From_acquisition_to_reconstruction_workflow_1</p> <p><b>correspondToFunctionality:</b> Reconstruction, Smoothing</p> <p><b>precedes:</b> Simplification</p> <p><b>follows:</b> Cleaning_of_scanned_data</p>
<p><b>hasName:</b> Simplification</p> <p><b>hasDescription:</b> A method/algorithm for simplifying a mesh i.e. reducing the number of faces used in a surface mesh while keeping the overall shape, volume and boundaries preserved as much as possible.</p> <p><b>isActivityOf:</b> From_acquisition_to_reconstruction_workflow_1</p> <p><b>correspondToFunctionality:</b> GeometricSimplification</p> <p><b>precedes:</b></p> <p><b>follows:</b> Mesh_smoothing_and_reconstruction</p>
<p><b>hasName:</b> Sampling of scanned data</p> <p><b>hasDescription:</b> Selecting only a subset of the scanned data. The desired number of samples is usually smaller than the mesh size and can vary according to the chosen sampling strategy.</p> <p><b>isActivityOf:</b> From_acquisition_to_reconstruction_workflow_2</p> <p><b>correspondToFunctionality:</b> Sampling</p> <p><b>precedes:</b> Meshing</p> <p><b>follows:</b></p>
<p><b>hasName:</b> Meshing</p>

<p><b>hasDescription:</b> Meshing algorithms compute a triangular mesh approximating of a surface. The algorithm computes a set of sample points on the surface, and extract an interpolating surface mesh from the three dimensional triangulation of these sample points, until some size and shape criteria on the elements of the surface mesh are satisfied.</p> <p><b>isActivityOf:</b> From_acquisition_to_reconstruction_workflow_2</p> <p><b>correspondToFunctionality:</b> Meshing</p> <p><b>precedes:</b> Simplification_2</p> <p><b>follows:</b> Sampling_of_scanned_data</p>
<p><b>hasName:</b> Simplification_2</p> <p><b>hasDescription:</b> A method/algorithm for simplifying a mesh i.e. reducing the number of faces used in a surface mesh while keeping the overall shape, volume and boundaries preserved as much as possible.</p> <p><b>isActivityOf:</b> From_acquisition_to_reconstruction_workflow_2</p> <p><b>correspondToFunctionality:</b> GeometricSimplification</p> <p><b>precedes:</b> Smoothing_and_Fairing</p> <p><b>follows:</b> Meshing</p>
<p><b>hasName:</b> Smoothing and Fairing</p> <p><b>hasDescription:</b> Smoothing a polygon mesh improves the quality of the model (e.g. Denoising, Filtering etc). Mesh Fairing reduces the variation in curvature.</p> <p><b>isActivityOf:</b> From_acquisition_to_reconstruction_workflow_2</p> <p><b>correspondToFunctionality:</b> Smoothing, Fairing</p> <p><b>precedes:</b></p> <p><b>follows:</b> Simplification_2</p>
Instances of <b>WorkflowDomain</b> class
<b>hasName:</b> Generic geometry processing
<b>hasName:</b> Medical

## 9 Technologies and tools used

The enabling technologies used for the development and deployment of this application are summarized in Table 4.

**Table 4: Enabling technologies for development and deployment of the web application.**

Product/technology	Version	Role
Java EE	8u45	Platform for the Java programming language
Netbeans (or OpenESB IDE 2.3.1)	6.5 or 7.0	Integrated Development Environment (IDE) for software developers.
GlassFish with JBI	2.1.1	Application server with integrated JBI service engines.
OpenESB	v2	Tools for building Integration and SOA applications.
BPEL	2.0	Web service orchestration.

### 9.1 OpenESB

For the development of web services and web service workflows, the **OpenESB<sup>7</sup>** was used. OpenESB is a Java-based open source enterprise service bus. It is used as a platform for both enterprise application integration and service-oriented architecture (SOA). OpenESB is open-source and relies on standard JBI (Java Business Integration), XML, XML Schema, WSDL, BPEL and Composite application that provide simplicity, efficiency and long-term durability.

The JBI specification defines two component types: The services engine (SE) and the binding component (BC):

- Binding components act as the interface between the outside world and the bus, being able to generate bus messages upon receipt of stimuli from an external source, or generate an external action/interaction in response to a message received from the bus.

---

<sup>7</sup> <http://www.open-esb.net/>

- Service engines receive messages from the bus and send messages to the bus. SE's have no direct contact with the outside world. They rely on the bus for interaction with other components, whether binding components or other service engines.

## **Acknowledgments**

The work has been carried out within the Regional PAR-FAS project “I-REUMA Imaging non invasivo dedicato per diagnosi precoce e follow-up delle patologie reumatiche del distretto mano-polso”. The authors thank Marina Monti, Stefano Gagliardo for the work carried out during the CAD2VR ontology specification within the VISIONAIR project, and Giuseppe Patanè, Marco Attene, Daniela Cabiddu, Michela Spagnuolo for the useful discussions on the workflow specifications in the medical domain.

---

**Recent titles from the IMATI-REPORT Series:****2017**

**17-01:** *BPX preconditioners for isogeometric analysis using analysis-suitable T-splines*, D. Cho, R. Vázquez.

**17-02:** *Initial-boundary value problems for nearly incompressible vector fields, and applications to the Keyfitz and Kranzer system*, A. P. Choudhury, G. Crippa, L.V. Spinolo.

**17-03:** *Quantitative estimates on localized finite differences for the fractional Poisson problem, and applications to regularity and spectral stability*, G. Akagi, G. Schimperna, A. Segatti, L.V. Spinolo.

**17-04:** *Optimality of integrability estimates for advection-diffusion equations*, S. Bianchini, M. Colombo, G. Crippa, L.V. Spinolo.

**17-05:** *A mathematical model for piracy control through police response*, G.M. Coclite, M. Garavello, L.V. Spinolo.

**17-06:** *Uncertainty Quantification of geochemical and mechanical compaction in layered sedimentary basins*, I. Colombo, F. Nobile, G. Porta, A. Scotti, L. Tamellini.

**17-07:** *VVS medical workflows: definition of a static workflow for part-based annotation of wrist bones & web service oriented architecture for executable workflows*, M. Pitikakis, F. Giannini.