
Title:
**A Web Repository to Describe and Execute Shape Processing Workflows**

Authors:
Marco Attene, Marco.Attene@ge.imati.cnr.it , IMATI-GE, CNR, Genova, Italy
Daniela Cabiddu, Daniela.Cabiddu@ge.imati.cnr.it , IMATI-GE, CNR, Genova, Italy
Stefano Gagliardo, Stefano.Gagliardo@ge.imati.cnr.it , IMATI-GE, CNR, Genova, Italy
Franca Giannini, Franca.Giannini@ge.imati.cnr.it , IMATI-GE, CNR, Genova, Italy
Marina Monti, Marina.Monti@ge.imati.cnr.it, IMATI-GE, CNR, Genova, Italy

Keywords:
Geometric processing, Web repository, Web services, Workflows

DOI: 10.14733/cadconfP.2015.303-307

Introduction:
Nowadays geometric models are ubiquitous, and models which are created in a given context often need to be reused in different scenarios. In these cases, a pre-processing is typically necessary to adapt the model to the new requirements (e.g. format conversion, geometric/topological modifications). As an example, performing the finite element analysis during a product design requires the creation of a mesh model from the CAD B-rep and also model adjustments and shape simplification involving both topological and geometric changes. The triangles produced by a tessellation algorithm are usually perfect for a visualization setting but not appropriate for a FEA application, where regularity of the mesh and its density in regions affected by particular stress are determinant to guarantee faithful simulation results. These processing are generally performed according to steady sequences resulting from technological constraints and experience. As a consequence, it is often difficult for non-experts to understand which tools and operations better fit with the specific purposes. In this case, a remeshing process is necessary to modify the shape of the triangles without modifying the overall shape of the model. Before remeshing, however, possible pre-processing might be necessary to guarantee that the mesh actually encloses a solid.

Another example in which shape model adaptation is necessary is the use of CAD models in VR environments. Even if VR offers advantages and new usage possibilities, it is mainly adopted in large companies, such as automotive and aerospace industry. A larger adoption of VR tools in wider engineering applications, and in particular in smaller companies, is still quite limited due to various reasons that include the time and skill required to properly adapt CAD models to be effectively used in VR applications. Some CAD vendors are providing integrated solutions for data acquisition and integration, but they are strictly coupled with their systems and do not fully automate the process. Design reviews and simulation in VR environments demands for high visualization capabilities obtained by processing polygon data, whereas CAD models are based on continuous surfaces. Moreover, engineering models are not created to be visualized in real time but to provide the effective detailed product shape to be manufactured or to serve as a schematic representation of the characteristics to be analyzed [7]. Therefore, CAD models need to be converted in a VR compatible format, i.e. a polygonal representation. Various problems can be detected in this. As in the previous examples, there is an inadequate treatment of the geometry with loss of precision leaving to inconsistent models with wrong surface orientation or cracks. In addition, the obtained models are too complex with unnecessary details, e.g. hidden areas, but at the same time, they miss realism. In fact, texture information is rarely associated to the CAD model, but it is quite important for truthful VR visualization. Finally, semantic information associated to each object, including its structure, is lost

and frequently needs to be recreated. To overcome these problems, several adjustments have to be performed by VR specialists using ad hoc tools. Thus, the required knowledge in choosing the most appropriate tool functionality in the correct sequence may further discourage engineers in adopting VR in their product development. The Web provides plenty of documentation and tutorials on the use of the most disparate tools, but they are weakly organized and mostly focused single specific software tools.

To overcome these limitations, we have developed new functionalities within the Virtual Visualization Service (VVS) infrastructure for the creation and retrieval of shape processing workflows [9], [1]. The VVS developed within the VISIONAIR project [8] extends the Digital Shape Workbench (DSW) created in the AIM@SHAPE Network of Excellence [1]. It consists of ontologies and web-based repositories of Shape Tools and Workflows, together with an advanced Search Framework [2]. Within the VVS, two types of workflows are considered: the so-called static workflows, which describe best practice processing pipelines, and the executable workflows, which allow running sequences of Web services. An ontology has been defined for their formal description and instances may be created for the specification of best practices in preparation of CAD data for Virtual Reality environments.

The Workflow Ontology:

The Workflow Ontology (WO) is the knowledge base that allows describing formally both documental and executable workflows. The ontology is built on top of the Common Info Ontology (CIO) and of the Common Tool Ontology (CTO), which organize the information about the users and the tool repository of the VVS (a catalogue of software tools for the creation, analysis and modification of shapes). Fig. 1 gives an overview of the ontology structure; in green the classes of the WO, in yellow and cyan the ones of the CIO and CTO, respectively. We decided not to use already existing process ontologies because of their complexity greater than the one needed for the VISIONAIR purposes, in which the main issue was to assist users in the preparation of data for the transition from one application environment to another, considering the starting and final data format and/or the software tools.

The main class of the WO is the Workflow class, where workflows are indeed instantiated. In particular, two subclasses have been defined, WorkflowStatic and WorkflowExecutable, for the instantiation of documental workflows for best practices (that are static), and executable workflows, respectively. Static workflows are meant as sequences of at least two activities that are elements of the Activity class. Simple activities correspond to a single functionality and can be grouped in macro-activities when they contribute to a unique logical action, which is normally performed by using the same software system. They are elements of the SimpleActivity and MacroActivity classes, respectively, both subclasses of the Activity class. The WorkflowDomain class is defined to specify for each element of the Workflow class the purpose of the workflow and its context of use. Other important classes are used to indicate tips and constraints or additional data required to carry out the specific activity. Additionally, the URL of documentation files may be linked to an activity or to one of its tips or restrictions for providing more detailed information. As the tools defined in the Tool Repository are mainly meant to be shape-oriented, the creation of new tools more devoted to specific application domain (such as tools managing sounds for VR) is allowed as instances of the Tool class of the WO.

The formalization of executable workflows is at a preliminary stage and includes, in addition to the common properties, the link to the .xml file that describes the workflow and is used by the engine described in the last section for the execution of the workflow itself.

The Workflow Repository:

The WO is the knowledge base of the Workflow Repository, a web platform where users can upload, remove, search and browse workflows. To facilitate the upload of new workflows, a dedicated user-friendly interface has been developed, which supports the creation of the instances of the required ontology classes or of the web service sequence for the executable workflow. It is a step-by-step uploading procedure that guides the user through the creation of a workflow without the need to know how the metadata he/she is inserting are stored in the underlying ontology. In the case of static workflows, it is possible to reuse existing elements by choosing them from dropdown lists; a “View” button is provided to visualize the metadata of the selected instance in order to verify if the

considered element is suitable to the user needs. No limit to the number of activities and sub-activities composing a workflow is given. When the user creates a new activity, he/she is prompted to insert the activity metadata, which include name, description, corresponding functionality, additional inputs, preserved data, tips and restrictions; specific tools for which they apply may be associated, as well as documental files may be uploaded. The browsing interface allows the user seeing all the workflows stored in the repository. The browsing is based on the metadata stored in the ontology. \

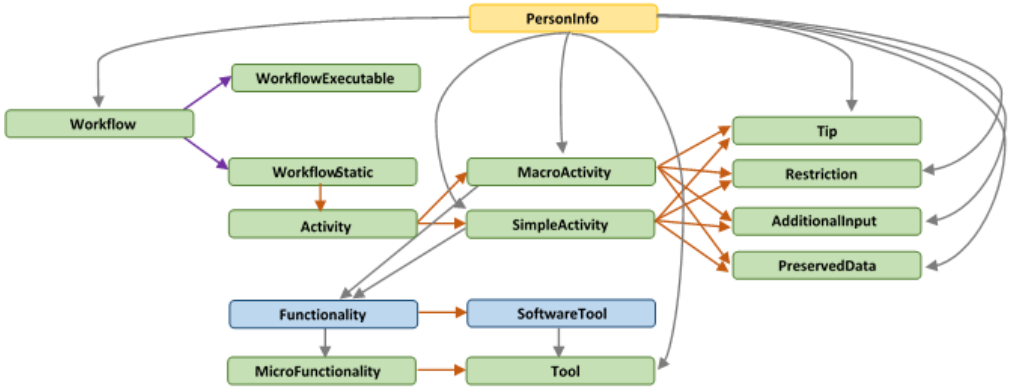


Fig.1: The Workflow Ontology Structure.

For static workflows, it allows visualizing not only the complete ones, but also those sub-parts that are made up by at least two activities or even those activities with more than one sub-activity. Complete and non-complete workflows are shown in different colors and it is possible in any case to filter the workflows in such a way that only the complete ones are shown. Moreover, the user can choose to filter the workflows using different searching criteria: such as by domain/purpose, or by input/output tools or formats. By clicking on a workflow, the user can access a new tab, which provides a more detailed view on it, where a tree representation of the workflow is given together with the main information about it. By clicking on the box representing one of its activities, the user may also get all the information on the activity itself, from the tools stored in the VVS Tool repository performing it, to its inputs/outputs and the tips and restrictions (Fig. 2).

Executable Workflows:

The Executable Workflow Module is designed to support geometry processing research activities. Since running experiments is one of the main tasks in this field and a typical experiment consists of performing a sequence of operations on a starting model and analyzing the results, our framework allows running state-of-the-art algorithms by using only a standard Web browser, without struggling with software installations, compatibility issues, or hardware requirements. Algorithms may be exploited individually or combined in complex geometry processing workflows. Furthermore, researchers in other fields than geometry processing who need to exploit geometry algorithms to run their experiments can easily take advantage of our system since no expertise in programming and geometry modelling is required.

The framework architecture is organized in three layers. On one side, a Web-based user interface allows choosing the desired algorithm among the available ones or defining complex geometry processing pipelines by combining a set of available operations. On the other side, a set of Web services is available. Web services may be considered as black boxes, each of them able to run a specific geometry processing algorithm on an input model using possible input parameters and returning the generated output address. The Workflow Engine is the interface between the two sides and is responsible of the pipeline runtime execution. It receives the specification of a geometry processing workflow, which can be either a new one or the identifier of one of the previously defined pipelines, and the address of an input mesh. When all the data is available, the Engine sequentially

invokes the various Web services, manages the flow of data among them and returns the address of the eventual result to the user interface.

The Workflow Engine also supports the execution of workflows that include conditional tasks and loops by delegating the evaluation of the condition to specific Web services able to evaluate mesh qualities. These conditional services receive from the Workflow Engine the address of the input mesh, evaluate a specific mesh quality and return a Boolean value to indicate if the condition is satisfied. The Engine is responsible to select the workflow operations that should be executed after the condition evaluation, according to the obtained result. The aforementioned protocol allows to remotely run geometry processing algorithms on 3D models, but the transfer of large-size meshes may constitute a bottleneck in the workflow execution. In order to avoid this situation, we designed an optimized mesh transfer protocol, based on the observation that many algorithms simply apply local modifications to the input. Our optimized protocol supports the processing of large meshes by sensibly reducing the overall elaboration time.

Geometric Pipelines Implementable as Executable Workflows:
The following examples illustrate geometric pipelines that can be realized as executable workflows.

*Mesh repairing for 3D printing*

Today fabricating an appropriate 3D model using a low-cost 3D printer is as easy as printing a textual document, but creating a 3D model, which is actually "appropriate" for printing, is definitely complicated. A 3D model can be produced either from scratch by using traditional CAD software, or from real-world objects using 3D digitizers. In both cases, the raw model is likely to have a number of defects and flaws that make it unsuitable for printing [3].

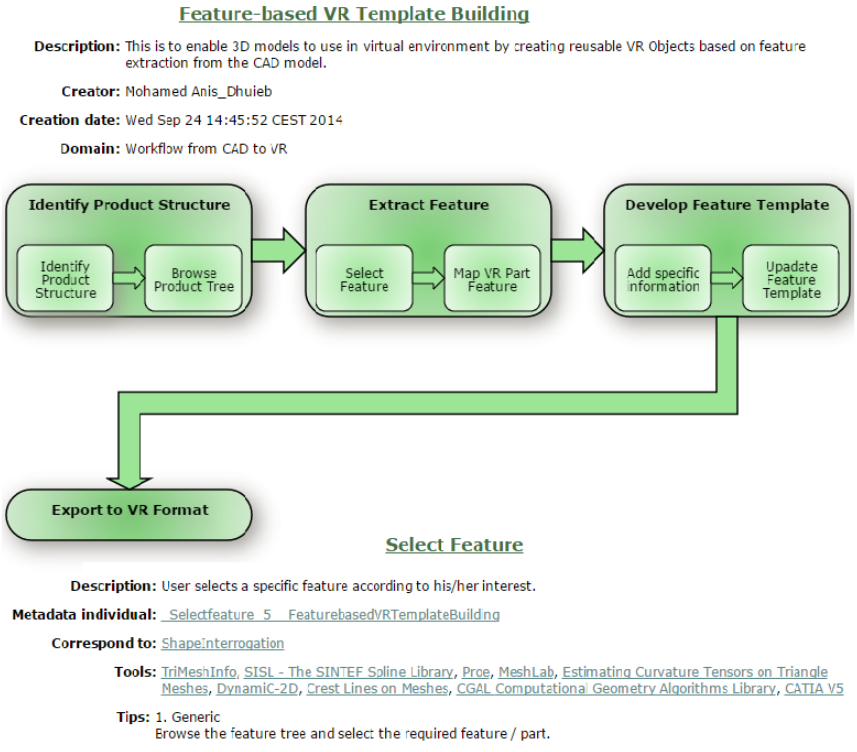


Fig. 2: Visualization of a static workflow in the Workflow Repository.

Proper pipelines of geometric algorithms to repair raw digitized models have been defined in the literature [4] and can be implemented as executable workflows. Similarly, but using different sequences of basic algorithms, meshes produced by tessellation of assembled CAD models can be automatically fixed to make them printable [5]. In most cases the repairing process can take place in a completely automatic manner, that is, without user intervention. However, some workflows may need to iterate the execution of one or more basic algorithms to converge to an eventual clean result. Thus, to make these pipelines available as executable workflows, the system must provide support for loops and conditional tasks.

*Mesh improvement for visualization*

When a mesh model must be visualized it is often important to convey a clear unbiased picture of the object. This requirement is in contrast with the characteristics of typical raw models coming from 3D digitization sessions, where a number of surface holes are commonplace just as surface noise, tiny disconnected components, gaps, and so on. Mesh visualization is not as demanding as 3D printing, but all the aforementioned defects should be removed or reduced to produce a nice and informative rendering. Geometric pipelines including surface smoothing, hole filling, gap closing and possibly simplification are therefore necessary [6]. Executable workflows can be implemented in this case too, and a number of variations can be provided depending on the target visualization device (e.g. a powerful graphics workstation, a desktop PC, a smartphone). For example, the level of the possible simplification can depend on the rendering capabilities, whereas the amount of smoothing can depend on the specific rendering engine used.

Acknowledgements:
This work has been partially supported by the European Commission under grant agreement 262044 VISIONAIR and by the PO CRO Fondo Sociale Europeo Regione Liguria 2007-2013 Asse IV "Capitale Umano" Ob. Specifico I/6, project "Tecniche di visualizzazione avanzata di immagini e dati 3D in ambito biomedicale". The authors want to thank the VISIONAIR partners participating to JRA 9 and, in particular, Stefano Mottura, Christian Weidig, Lionel Roucoules and Walter Terkaj.
Additional thanks to Marios Pitikakis for the technical support and to Prof. Bianca Falcidieno and all the participants to the AIM@SHAPE NoE for making possible the initial version of the DSW.

References:
[1] AIM@SHAPE , FP6 IST NoE 506766, http://www.aimatshape.net
[2] Attene M.; Giannini F.; Pitikakis M.; Spagnuolo M.: The VISIONAIR Infrastructure Capabilities to Support Research, Computer-Aided Design and Applications, 10(5), 851 - 862, 2013. http://dx.doi.org/10.3722/cadaps.2013.851-862
[3] Attene M.; Campen M.; Kobbelt L.: Polygon mesh repairing: an application perspective. ACM Computing Surveys, 45(2), Art. 15 (33 pages), 2013. http://dx.doi.org/10.1145/2431211.2431214
[4] Attene M.: A lightweight approach to repairing digitized polygon meshes. The Visual Computer, 26(11), 1393-1406, 2010. http://dx.doi.org/10.1007/s00371-010-0416-3
[5] Attene M.: Direct repair of self-intersecting meshes. Graphical Models, 76, 658-668, 2014. http://dx.doi.org/10.1016/j.gmod.2014.09.002
[6] Cabiddu D.; Attene M.: Distributed Triangle Mesh Processing In Procs. of 22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2014 (WSCG), Plzen (CZ), 2-5 June 2014
[7] Raposo, A.; Corseuil, E. T. L.; Wagner, G. N.; dos Santos, I. H. F.; Gattass, M.: Towards the use of cad models in VR applications. In Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications (VRCIA '06). ACM, New York, NY, USA, 67-74. http://dx.doi.org/10.1145/1128923.1128935
[8] VISIONAIR: VISION Advanced Infrastructure for Research, http://www.infra-visionair.eu/
[9] VVS - http://visionair.ge.imati.cnr.it/