COMPUTER-AIDED DESIGN & APPLICATIONS, 2016
VOL. 13, NO. 5, 637–646
http://dx.doi.org/10.1080/16864360.2016.1150710

A web repository to describe and execute shape oriented workflows

Marco Attene, Daniela Cabiddu, Stefano Gagliardo, Franca Giannini and Marina Monti

IMATI-CNR Genova, Italy

ABSTRACT
The effective use of advanced tools and geometric models across different applications frequently requires model processing to satisfy the application requirements. Besides being time consuming processes, these model adaptations normally employ advanced tools that need specific knowledge for their proper usage. Therefore, it is frequently difficult understanding which tools and operations better fit with the specific purposes.

This paper presents the capabilities of the Visualization Virtual Service (VVS) infrastructure for the creation and retrieval of shape processing workflows either to describe best practice process pipelines or to allow running sequences of web services.

KEYWORDS
Geometric processing; web repository; web services; workflows

1. Introduction

Nowadays the use of various systems for the creation and analysis of 3D digital models has moved most of the engineering applications into the digital world. CAD systems are de-facto standard for the creation of new product shapes. They are supported by specific application tools for the verification and simulation of all the product aspects, such as the layout, the behavior and functionality, and the related production operations. According to the specific type of analysis, different model representations are used to better fit the efficiency and functionality of the algorithms performing the analysis and simulations. Therefore, the effective use of geometric models in engineering applications frequently requires some model processing to satisfy the application requirements. This processing can involve conversion of the representation and format as well as modifications in object geometry and/or topology.

As an example, performing the finite element analysis during a product design not only requires the creation of a mesh model from the CAD B-Rep but also model adjustments, and a shape simplification involving both topological and geometric changes [11]. The few long and thin triangles produced by a tessellation algorithm are perfect for a visualization setting because they allow higher frame rates. Unfortunately, these triangles are definitely not appropriate for a FEA application, where the regularity of the mesh and its density in regions affected by particular stress are determinant to guarantee faithful simulation results. 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 [3]. Also, after remeshing and depending on the analysis type, the surface mesh can be used to produce a conforming tetrahedral mesh.

Another example is the process required to get correctly printable shapes. 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 [4]. Proper pipelines of geometric algorithms to repair raw digitized models have been defined in the literature [2] and can be implemented as automatic processing 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 [3]. 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 [2]. Thus, to make these pipelines available as executable workflows, the management of loops and conditional tasks should be included.

CONTACT Marina Monti Marina.Monti@ge.imati.cnr.it

© 2016 CAD Solutions, LLC, http://www.cadanda.com

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. Automatic process 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 simplification can depend on the rendering capabilities, whereas the amount of smoothing can depend on the specific rendering engine used.

Another example in which shape model adaptation is necessary is the use of CAD models in VR environments. VR is the combination of real-time presentation and immersive interaction for the modification and evaluation of models and processes within a computer-generated environment. VR techniques and tools have seen a big improvement in the last years, and their use has been recognized advantageous to replace expensive and work intensive physical prototypes by their digital representation. 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 high equipment costs. Additionally, the other main issue limiting VR usage is related to 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 [20]. Therefore, CAD models need to be converted in a VR compatible format, i.e. a polygonal representation. Various problems can be detected in this conversion [21]. 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 but still very important, 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 briefly described processes are generally performed according to steady sequences resulting from technological constraints and experience. Thus, understanding which tools and operations better fit with the specific purposes is frequently difficult for non-experts. The web provides plenty of documentation and tutorials on the use of the most disparate tools, but they are weakly organized and generally related to the specific software.

To overcome these limitations, in this paper, we present the capabilities of the Visualization Virtual Service (VVS) infrastructure for the creation and retrieval of shape processing workflows. Within the VVS, two types of workflows are considered: static workflows, which describe best practice process pipelines, and 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 for the preparation of CAD data for Virtual Reality environments.

In section 2 of this paper, we present the workflow ontology that formalizes the knowledge needed for the creation and retrieval of shape processing workflows within the Visualization Virtual Service (VSS) infrastructure. Section 3 explains how the executable workflow module can support geometry processing research activities. In section 4 the repository in which users may upload, search and browse workflows is illustrated. Finally, section 5 concludes the paper by summarizing the achievements.

2. The workflow ontology

The Visualization Virtual Service (VVS) (http://visionair.ge.imati.cnr.it/) developed within the VISIONAIR project extends the Digital Shape Workbench (DSW) from the AIM@SHAPE Network of Excellence [1]. It consists of ontologies and web-based repositories of Shapes, Tools and Workflows, together with an advanced Search Framework. The VISIONAIR project aims at federating and providing an unique access to advanced visualization resources available at research centers and universities from Israel and other countries in Europe [16]. Researchers can access the facilities, including Virtual Reality, Augmented Reality, Holography, High Quality Image Processing, etc., to test their research results or to carryout visualization trials. Within the VISIONAIR project, the VVS is aimed at supporting the experts in the preparation of experiments [5] by including the possibility of finding and storing both shape data and shape processing tools necessary to run tests and to create virtual environments. To better support the preparation of data, during the project a particular interest emerged to have available organized information on the processes for the preparation of VR environments and some shape processing capabilities to treat the data. To fulfil this requirement we extended the VVS framework to include capabilities for creating, storing and running two different types of workflows:

- Static workflows to describe best practice processes,
- Executable workflows to allow the execution of sequences of web services.

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, i.e. a catalogue of software tools for the creation, analysis and modification of shapes. Fig. 1 gives an overview of the ontology structure. Process languages and ontology are a subject studied by many authors in the last decades, such as in [14][16][20]; Gangemi et al. in [12] provide a good overview of them. We opted for the ontological approach to be compatible with the rest of the VVS. We decided not to use already existing process ontologies because of their complexity greater than the one needed for the VISIONAIR purposes. Available ontologies related to VR applications [19][10] are mainly devoted to support the behavioral modelling of the objects or to formalize the code for generating a Virtual World without considering any data adaptation aspect.

The main class of the WO is the Workflow class, where workflows are indeed instantiated. In particular, two subclasses have been created, WorkflowStatic and WorkflowExecutable, for the instantiation of documental (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, i.e. corresponding to a single functionality, can

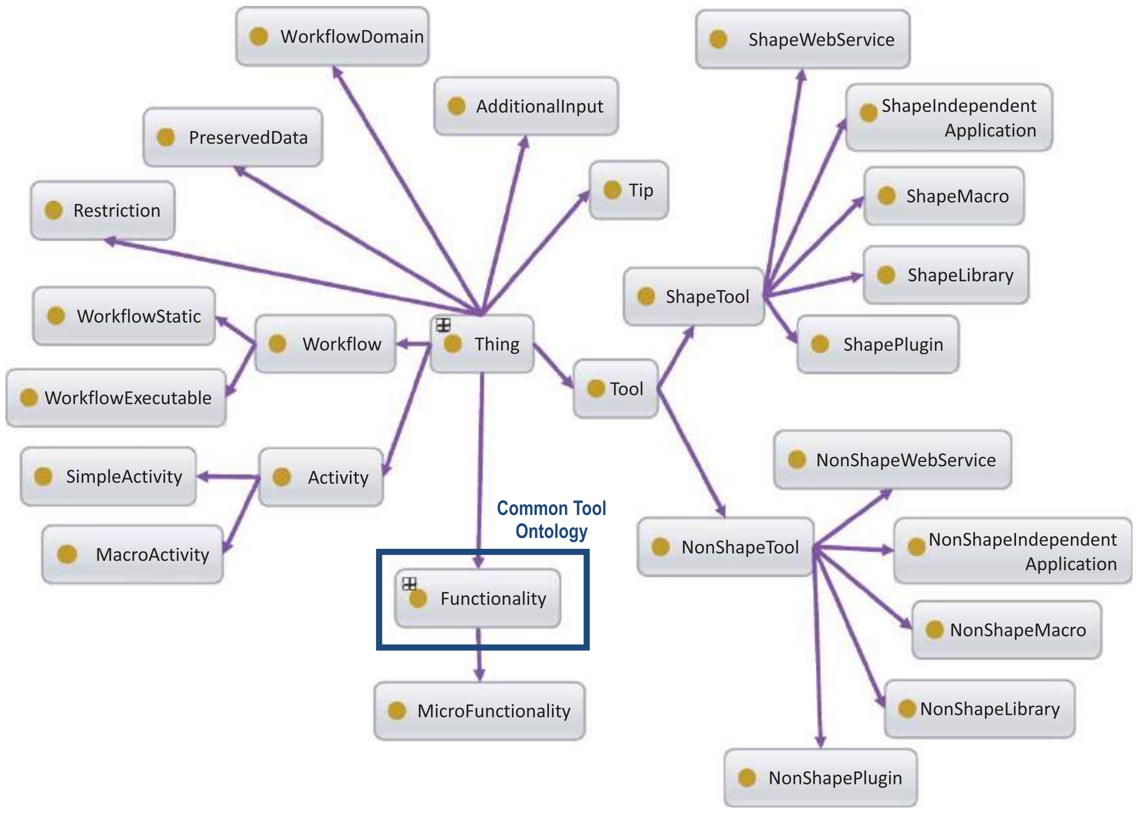


Figure 1. The Workflow Ontology Structure.

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 property WorkflowDomain of the Workflow class specifies the purpose of the workflow and its context of use. Each activity may be described in more details by specifying some additional information: hints and constraints on the correct performing of the activity can be provided as instances of the Tip and Restriction classes, respectively, and linked to the activity through the hasTip or hasRestriction property. Additional data to carry out the specific activity and preserved information during its execution are modelled as elements of the AdditionalInput and PreservedData classes, respectively, and are linked to the activity through the hasAdditionalInput and hasPreservedData object properties. In particular, tips and restrictions can be linked not only to the activity, but also to specific additional data, preserved information or tool related to it, using the tipOfInput, tipOfData and tipOfTool properties (analogous ones for restrictions).

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, such as text files with a deeper explanation of the activity or also images.

Information about the tools listed in the VVS that can perform an activity are indirectly obtained through the functionality an activity is linked to. Fig. 2 depicts the exploited connections between the Common Tool Ontology, used for the Tool repository of the VVS, and the Workflow ontology. As the tools defined in the Tool Repository are mainly meant to be shape-oriented tools, 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 includes, in addition to the common properties, the link to the .xml files that describe the workflow and are used by the engine, described in the next section, for the execution of the workflow itself.

3. Executable workflows

The Executable Workflow Module is designed to enable the actual processing of geometric models. Since a typical process (see Fig. 3) consists of considering an input model and performing a sequence of operations on it, 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 can be exploited individually or combined in complex geometry processing

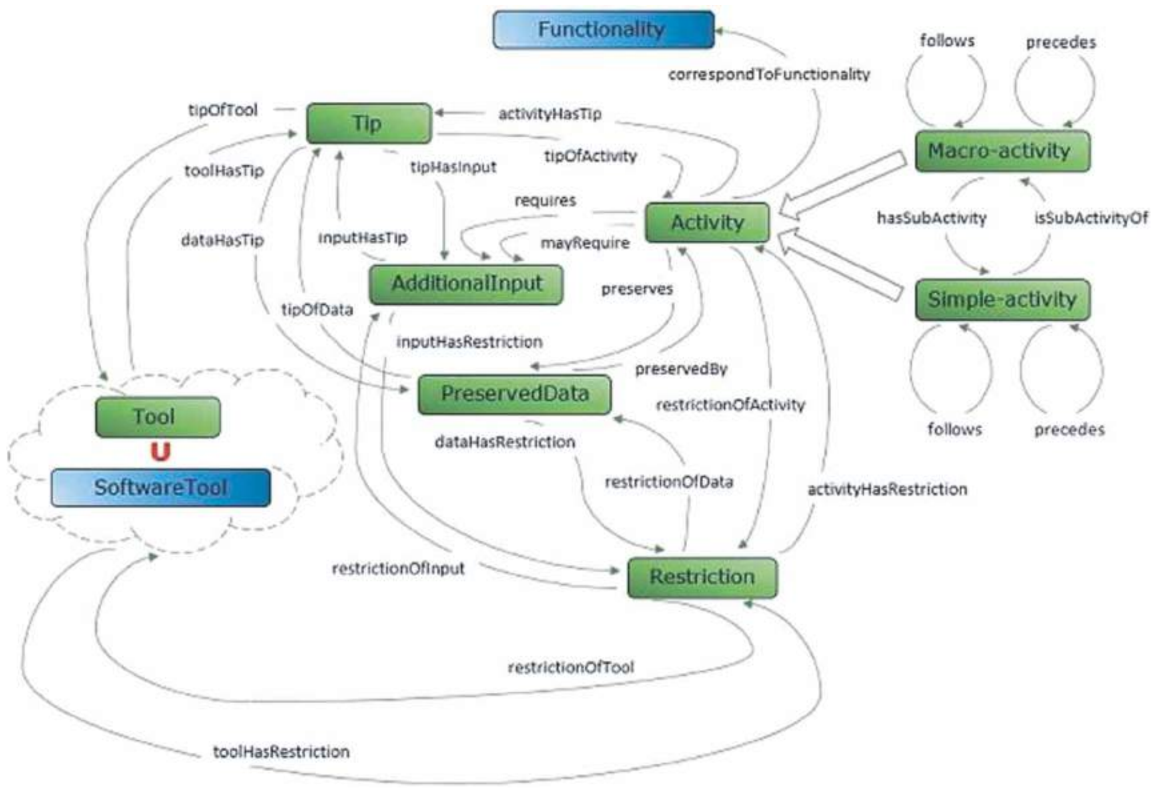


Figure 2. A zoomed view of the workflow ontology on the Activity and related classes with the linking properties.

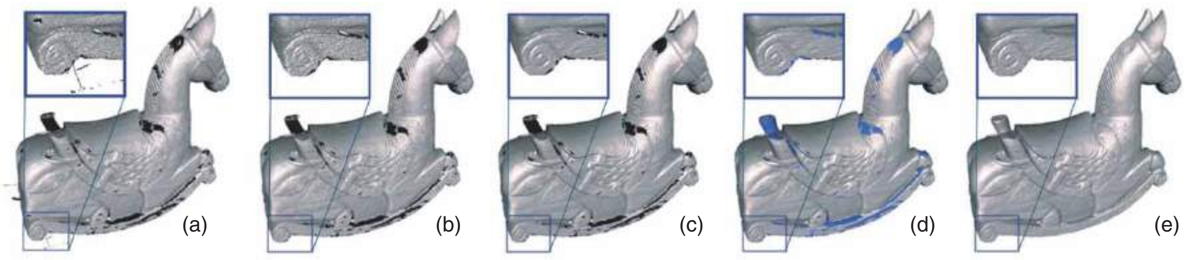


Figure 3. Local mesh repairing [4]: a typical example of geometry processing workflow. The input model (a) has 160 spurious disconnected components that are removed (b). Then one iteration of laplacian smoothing is applied (c) to enhance the surface and the 404 holes are patched (d). Finally, degenerate triangles are removed (e).

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, as shown in Fig. 4(a). 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

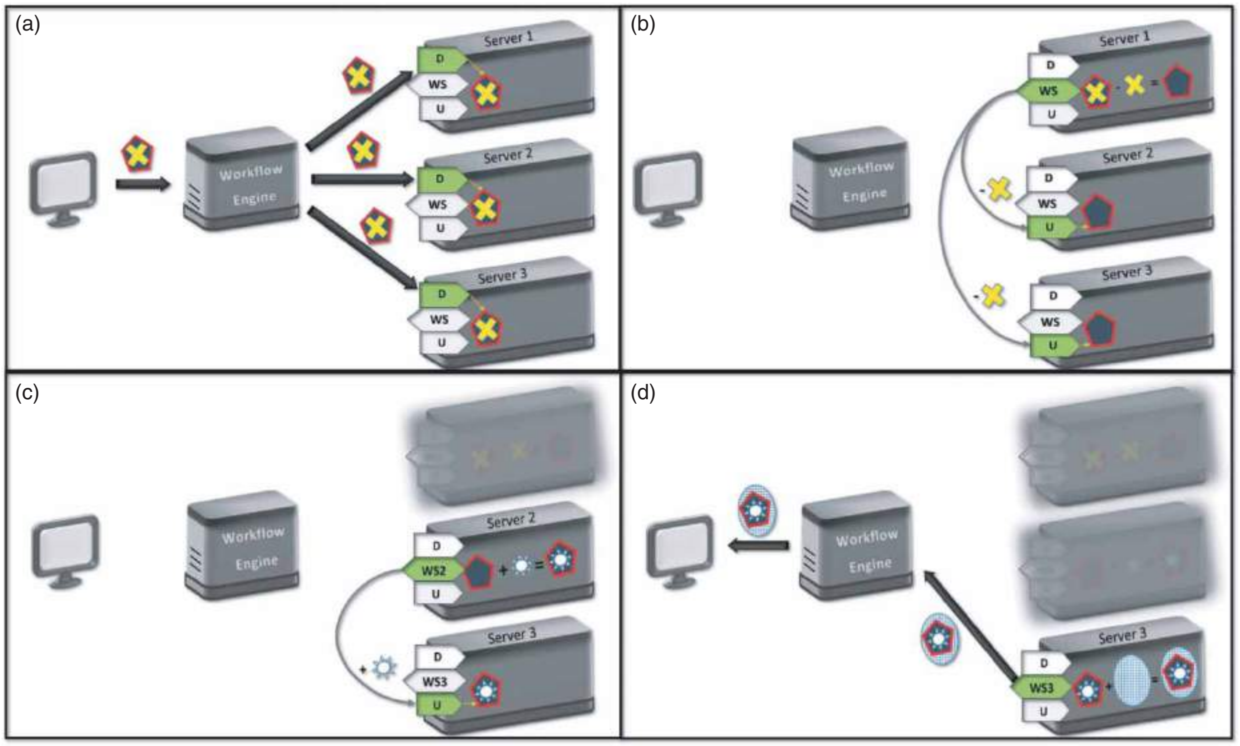


Figure 4. Mesh Transfer Protocol Example. (a) The three-layered architecture composed by a graphical user interface, a workflow engine and three servers, each of them exposing a Web service to support a specific operation and to modules able to download (D) meshes and update (U) the previously downloaded mesh by applying the corrections. The engine broadcasts the address of the input mesh towards all the involved servers that proceed with the download. (b) The first service runs the task and returns the address of the produced modifications to the engine that shares it with the following involved servers. Both download the file and apply the modifications. (c) The second service is invoked, runs its task and makes the list of applied editing operations available, so that the third involved server can update its local copy of the input. (d) The engine triggers the third service and waits for the corresponding output that will be forwarded to the user.

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.

In more detail, the runtime workflow execution works as follows. The Workflow Engine stores the list of Web services that are available and able to perform specific tasks. For each workflow task, the Engine selects and invokes the appropriate Web service. When triggered for execution, each Web service receives the address of the input mesh and possible input parameters. When the task terminates, the address of the generated output is returned to the Engine so that it can be forwarded to the next involved Web service. When the last Web service terminates its task, its output is returned to the Workflow Engine that forwards it to the user by publishing the corresponding link.

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 [6]. 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.

3.1. The mesh transfer protocol

We have observed that the transfer of large meshes from a server to another according to the aforementioned protocol constitutes a bottleneck in the workflow execution. In order to improve the transfer speed and efficiently support the processing of large datasets, we took advantage of the prediction/correction metaphor used in mesh compression to design an optimized mesh transfer protocol [7]. Our solution is based on the observation that numerous mesh processing algorithms simply transform an input mesh into an output by computing and applying local or global modifications. In many cases modifications can be only local or may modify both geometry and connectivity by minimally changing the overall shape. In all these cases, it is reasonable to assume that the modifications applied on the input can be more compactly encoded in form of list of applied editing operations than the explicit output mesh. The calling service may assume that the output will be identical to the input and may obtain the result by reapplying the encoded modifications.

Based on the aforementioned observation, the optimized mesh transfer protocol works as follows (an example is shown in Fig. 4). The user selects or defines an executable workflow and uploads an input mesh. The engine analyzes the workflow, locates the servers hosting web services able to perform each operation and broadcasts to them the address of the input mesh (Fig. 4(a)). Each server is triggered to download the input model and save it locally. At the first step of the experiment, the workflow engine invokes the suitable web service that runs the algorithm, produces the result, and locally stores the output mesh and the modification file (both compressed) (Fig. 4(b)). Their addresses are returned to the workflow engine that forwards them to all the subsequent servers involved in the workflow. Each server downloads the encoded modifications and applies them to the mesh it already has in memory in order to update the local copy of the model. Then, the workflow engine triggers the next service (Fig. 4(c)) for which an up-to-date copy of the mesh is readily available on its local server. At the end of the workflow execution, the engine receives the address of the output produced by the last invoked web service and returns it to the user interface (Fig. 4(d)), so that the user can proceed with the download.

In this scenario, the entire input mesh is broadcasted only once at the beginning of the process, and the final result is transmitted only once at the end. In between, only the modifications are broadcasted to the subsequent servers. Thus, when the corrections are actually smaller than the partial results, this procedure produces significant benefits. In any case, each web service produces both the modifications and the actual result so, when the latter is smaller than the former, the subsequent web services can directly download the output instead of the list of applied modifications.

In our settings, each web service is required to run a geometry processing algorithm, to keep track of the editing operations and to save them along with the final result. To do this, the algorithm itself must be enriched with proper code to stream such operations into the modification file. In our current implementation we support atomic operations to encode the insertion, removal and modification of single simplexes of any order (i.e. vertices, edges and triangles). When an algorithm terminates, the produced sequence of operations is further compressed through arithmetic coding to minimize redundancy [22]. Note that the application of the editing operations by the subsequent web services requires less computational efforts and time than the rerun of the algorithm because its analysis part and the operation precondition checks are not needed anymore.

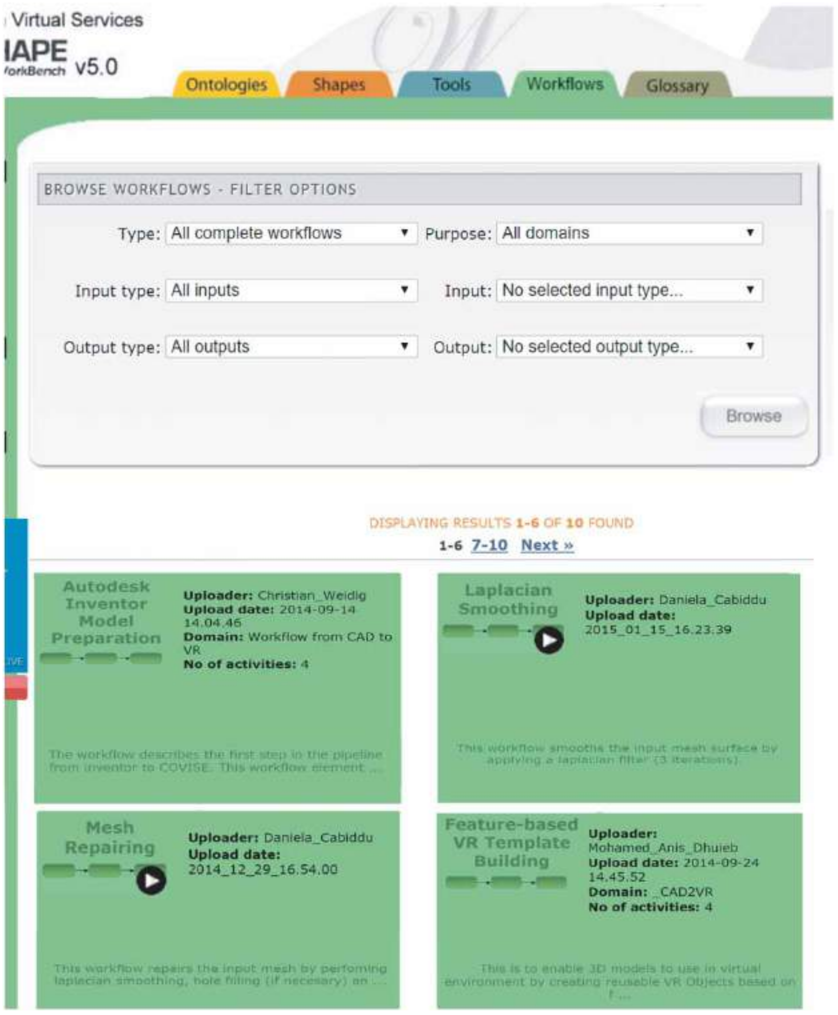
In addition to allowing the production of adapted models, our executable workflows may provide benefit to 3D model repositories and for geometric processing experimentation replication, as described in the following sections.

3.2. Executable workflows and 3D model repositories

3D model repositories are already key instruments for researchers in the area of geometry processing and product design, but endowing them with executable workflow capabilities is expected to further boost their efficacy. While Stanford's repository [24] was focused on models coming from 3D digitization, other collections exist that deal with synthetic CAD models [25] or that focus on specific algorithms, such as the Princeton Shape Benchmark [23], which is aimed to evaluate shape query and retrieval algorithms. VISIONAIR's Visualization Virtual Services try to encapsulate most of the functionalities provided by previous repositories, and allows registered users to upload resources (e. g. 3D test models, prototype tools, algorithm results). The VVS requires a huge amount of space to explicitly store 3D models and is strongly redundant since outputs of geometric processes, which are often very similar to each other, are individually uploaded with no attention to avoid possible duplicated data. The possibility to reproduce these processes on the fly through executable workflows strongly reduces the storage space required by the repository, where only the original data and the results of the most complex processes (which would take too long to be reproduced online) are stored explicitly.

3.3. Geometric experiment replication

Besides enabling a more flexible storage organization, our executable workflows also allow reproducing geometric experiments much more easily and fairly than current solutions. A typical geometry processing experiment consists of performing a sequence of operations on an input polygon mesh and analyzing the results. An



BROWSE WORKFLOWS - FILTER OPTIONS

Type: All complete workflows
Purpose: All domains
Input type: All inputs
Input: No selected input type...
Output type: All outputs
Output: No selected output type...

DISPLAYING RESULTS 1-6 OF 10 FOUND
1-6 7-10 Next »

Autodesk Inventor Model Preparation
Uploader: Christian_Weidig
Upload date: 2014-09-14 14.04.46
Domain: Workflow from CAD to VR
No of activities: 4
The workflow describes the first step in the pipeline from inventor to COVISE. This workflow element ...

Laplacian Smoothing
Uploader: Daniela_Cabiddu
Upload date: 2015_01_15_16.23.39
This workflow smooths the input mesh surface by applying a laplacian filter (3 iterations).

Mesh Repairing
Uploader: Daniela_Cabiddu
Upload date: 2014_12_29_16.54.00
This workflow repairs the input mesh by perfoming laplacian smoothing, hole filing (if necesary) an ...

Feature-based VR Template Building
Uploader: Mohamed_Anis_Dhuieb
Upload date: 2014-09-24 14.45.52
Domain: _CAD2VR
No of activities: 4
This is to enable 3D models to use in virtual environment by creating reusable VR Objects based on f ...

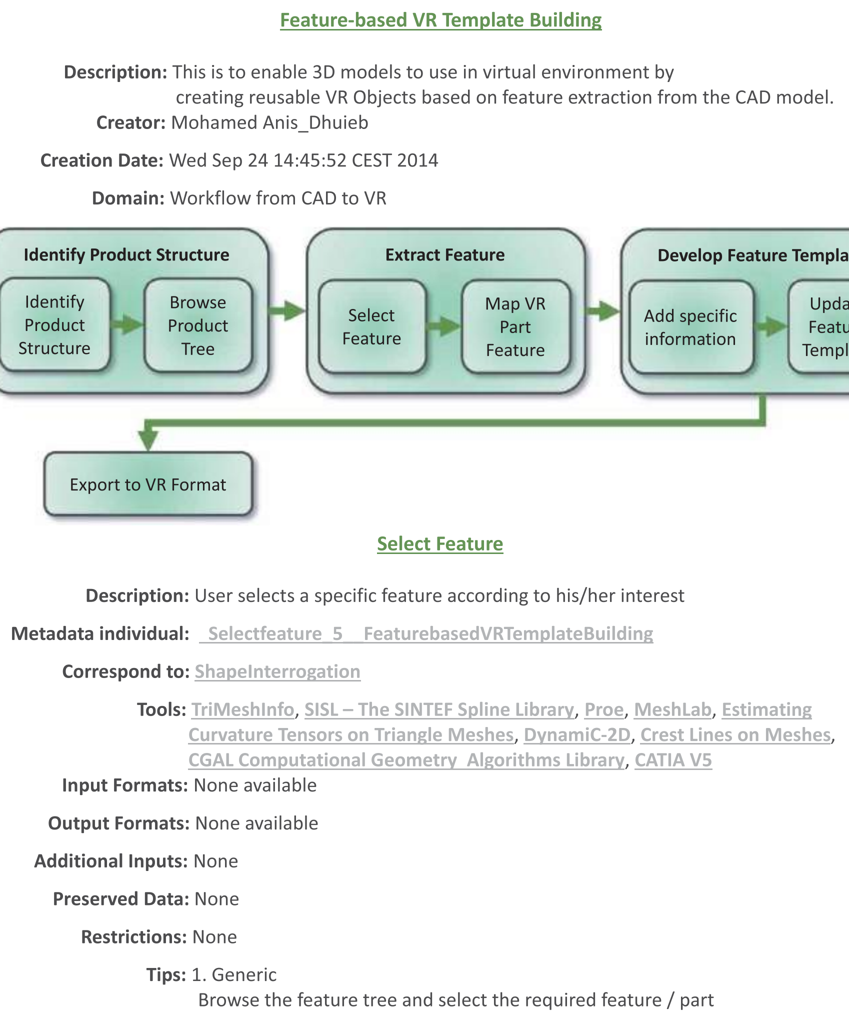
Figure 5. Browsing of workflows (both static and executable) in the Workflow Repository.

example is shown in Fig. 3. Mesh editing tools such as MeshLab [9] and OpenFlipper [18] allow interactively editing a mesh and save the sequential list of executed operations, so that it can be re-executed locally from the tool user interface. Unfortunately, to reproduce such an experiment, a researcher needs a similarly performing machine, and needs to install the same software tool (e.g. MeshLab) and possible plugins. To get rid of any specific software, hardware, and operating system, Campen and colleagues published an online service called WebBSP [8] which is able to remotely run a few specific geometric operations. The user is required to upload an input mesh from a standard web browser and select a single geometric algorithm from a set of available operations. The algorithm actually runs on the server and a link to download its output is returned. In this case, however, the available operations are not customizable by users and only one of them can be run at each call. Conversely, our executable workflows are fully dynamic (i.e. new web-services can be made available by users), can stack several operations, can include conditional tasks and loops, and can exploit the computing power of several nodes in a distributed network.

4. Workflow repository

The WO is the knowledge base of the Workflow Repository, a web platform where users can upload, remove, search and browse workflows; some of these capabilities, in particular uploading and removing, are allowed only for registered users, while any potential user can search or browse workflows.

To facilitate the upload of new workflows, a dedicated user-friendly interface has been developed to support the creation of the instances of the needed 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.


Feature-based VR Template Building

Description: This is to enable 3D models to use in virtual environment by creating reusable VR Objects based on feature extraction from the CAD model.
Creator: Mohamed Anis_Dhuieb

Creation Date: Wed Sep 24 14:45:52 CEST 2014

Domain: Workflow from CAD to VR

Select Feature

Description: User selects a specific feature according to his/her interest

Metadata individual: _Selectfeature_5__FeaturebasedVRTemplateBuilding

Correspond to: ShapeInterrogation

Tools: TriMeshInfo, SISL – The SINTEF Spline Library, Proe, MeshLab, Estimating Curvature Tensors on Triangle Meshes, DynamiC-2D, Crest Lines on Meshes, CGAL Computational Geometry Algorithms Library, CATIA V5
Input Formats: None available

Output Formats: None available

Additional Inputs: None

Preserved Data: None

Restrictions: None

Tips: 1. Generic
Browse the feature tree and select the required feature / part

Figure 6. Visualization of a static workflow in the Workflow Repository.

In the case of static workflows, the user is asked at first to provide the information directly related to the workflow, such as its name, description and list of activities. The user may either create a new activity by simply providing a name, or reuse an existing one by exploiting the given dropdown list. To get more information on an activity and to understand if it is suitable to the user needs, a "View" button is provided to access its metadata. 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 specify the required information such as the activity description, correspondent functionality, additional inputs, preserved data, possible tips and restrictions, possibly associating the specific tool for which they apply, as well uploading documental files.

In the case of executable workflows, in addition to the general information (name, description, creator and creation date), the user has to specify the list of web services the workflow is made up, as well as the required parameters of the added web service (if needed). If and while conditional blocks can be defined. Once the workflow has been uploaded, the user can execute the workflow by clicking on the provided "Run Workflow" button.

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. For static workflows, it allows visualizing all the existing complete workflows, those workflow sub-parts that are made up of at least two activities or even one activity with more than one sub-activity. User defined workflows and their sub-workflows are shown in different colors. It is possible to filter the workflows in such a way that only the complete ones are shown (Fig. 5). Further filtering options for the workflow visualization are types (static, executable), the domain/purpose, input/output tools and formats.

In the example of Fig. 5, the system lists the first four results of a user query for all the complete workflows available in the repository; no specific domain or input/output type are given. Among the displayed results there are two static and two executable workflows; the listed executable workflows allow user running web services for mesh smoothing and mesh repairing. The static workflows describe the required sequence of activities respectively to appropriately transfer models from CAD Inventor to COVISE virtual environment and to create reusable feature-based objects in VR environment starting from 3D CAD models.

By clicking on a workflow, the user can access to a more detailed view on it: a flow 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. 6).

5. Summary and conclusions

In this paper, the need of shape processing operations for a proper and effective use of geometric models in diverse applications has been outlined. Some examples have been presented in which the shape processing is generally performed according to steady sequences of operations resulting from technological constraints and experience. In most of the cases understanding which tools and operations better fit with the specific purposes is difficult for non-experts. Analyzing the various application scenarios, it is possible to identify some repeated and common model treatments, which can be somehow automated. To support engineers and researchers in performing these processing pipelines we developed a web-based repository of geometric processing workflows and a tool that allows the execution of workflows of web services for shape adaptation. In fact, we considered both static and executable workflows. The former are similar to tutorials to describe the sequences of activities and functionalities to be used to achieve specific objectives, while indicating tips for their execution by means of specific tools. The latter offer the advantage of directly running processes without the need of installing the related programs or delivering core data. Currently the repository contains some executable and several tutorial workflows resulting from the experience of the VISIONAIR partners in the preparation of CAD models for VR applications, but the adopted schema and the developed creation and search facilities can accommodate any kind of pipeline and context treating shapes. Registered user is allowed to create new workflows and make them available for the community. The integration with the Tool repository of the VVS infrastructure allows an automatic retrieval of the tools stored in the Tool repository that can be applied in the pipeline, thus providing an idea of the usable software tools to non-expert users.

Acknowledgments

This work has been partially supported by the European Commission under grant agreement 262044 VISIONAIR (http://www.infra-visionair.eu/) and by the PO CRO Fondo Sociale Europeo Regione Liguria 2007–2013 Asse IV "Capitale Umano" Ob. Specifico I/6 through the 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, to 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.

ORCID

Marco Attene http://orcid.org/0000-0002-9012-7245
Daniela Cabiddu http://orcid.org/0000-0001-5797-4189
Franca Giannini http://orcid.org/0000-0002-3608-6737
Marina Monti http://orcid.org/0000-0002-1627-3551

References

[1] AIM@SHAPE Advanced and Innovative Models And Tools for the development of Semantic-based systems for Handling, Acquiring, and Processing knowledge Embedded in multidimensional digital objects, Contract FP6 IST NoE 506766
[2] Attene M.: A lightweight approach to repairing digitized polygon meshes, The Visual Computer, 26(11), 2010, 1393–1406. http://dx.doi.org/10.1007/s00371-010-0416-3
[3] Attene M.: Direct repair of self-intersecting meshes, Graphical Models, 76, 2014, 658–668. http://dx.doi.org/10.1016/j.gmod.2014.09.002
[4] Attene M.; Campen M.; Kobbelt L.: Polygon mesh repairing: an application perspective, ACM Computing Surveys, 45(2), Art. 15, 2013. http://dx.doi.org/10.1145/2431211.2431214
[5] Attene, M.; Giannini, F.; Pitikakis, M.; Spagnuolo M.: The VISIONAIR Infrastructure Capabilities to Support Research. In: Computer-Aided Design and Applications, 10(5), 2013, 851–862. http://dx.doi.org/10.3722/cadaps.2013.851-862
[6] Attene, M.: Surface mesh qualities. In GRAPP/IVAPP, pages 79–85, 2013.
[7] 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
[8] Campen, M.: WebBSP 0.3 beta. http://www.graphics.rwthaachen.de/webbsp, 2010.
[9] Cignoni, P.; Corsini, M.; Ranzuglia, G.: Meshlab: an open-source 3d mesh processing system. ERCIM News, 73, 2008, 45–46.
[10] De Troyer, O.; Bille, W.; Romero, R.: Stuer, P.: On Generating Virtual Worlds from Domain Ontologies. In Proceedings of the 9th International Conference on Multimedia Modeling, Taipei, Taiwan, 2003, 279–294.
[11] Foucault, G.; Cuillière, J.-C.; François, V.; Léon, J.-C.; Maranzana, R.: Adaptation of CAD model topology for finite element analysis, Computer-Aided Design, 40(2), 2008, 176–196. http://dx.doi.org/10.1016/j.cad.2007.10.009
[12] Gangemi, A.; Borgo, S.; Catenacci, C.; Lehmann, J.: Task Taxonomies for Knowledge Content D07. Metokis Project. 2005
[13] Graf, H.; Brunetti, G.; Stork, A.: A methodology supporting the preparation of 3D-CAD data for design review in VR, International Design Conference – Design 2002, Dubrovnik, May 14–17, 2002
[14] Haller, A.; Oren, E.; Marmolowski, M.; Gaaloul, W.: A Process Ontology for Business Intelligence. DERI Technical report 2008-04-012008
[15] Hamri, O.; Léon, J.C.; Giannini, F.; Falcidieno, B.: Using CAD models and their semantics to prepare F.E. simulations. DETC '05 - CIE '05 - Design Engineering Technical Conferences & Computers and Information in Engineering Conference (Long Beach (CA), USA, 24–28 September 2005). Proceedings, 1–10. ASME, 2005. http://dx.doi.org/10.1115/DETC2005-84867
[16] Kopecki, A; Wossner, U; Mavrikios,; Rentzos, L.; Weidig, C.; Roucoules, L.; Ntofon, O.D.; Reed, M.; Dumont, G. D; Bundgens, D.; Mileck, A.; Baranyi, P.; Noel, F.; Masclet, C.; Attene, M.; Giannini, F.; Spagnuolo, M.: VISIONAIR VISION Advanced Infrastructure for Research, SBC Journal on 3D Interactive Systems, 2(2), 2011, 40–43. http://seer.ufrgs.br/jis/article/view/22520
[17] Lando, P.; Lapujade, A.; Kassel, G.; Furst, F.: Towards a general ontology of computer programs. In: Proceedings of the 2nd International Conference on Software and data Technologies (ICSOFT 2007).
[18] Möbius, J.; Kobbelt, L.: Openflipper: An open source geometry processing and rendering framework. In Proc. of the 7th International Conference on Curves and Surfaces, 488–500, Berlin, Heidelberg, 2012, http://dx.doi.org/10.1007/978-3-642-27413-8_31
[19] Pellens, B.; De Troyer, O.; Bille, W.; Kleinermann, F.: Conceptual Modeling of Behavior in a Virtual Environment, Special issue of International Journal of Product and Development, 4, 2007, 626–645.
[20] Pouchard, L. C.; Cutting-Decelle, A. F.; Michel, J. J.; Gruninger, M.: ISO 18629 PSL : A standardised language for specifying and exchanging process information, World Congress, 16(1), 2005, 1524–1524. http://dx.doi.org/10.3182/20050703-6-CZ-1902.01525
[21] 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
[22] Said, A. Introduction to arithmetic coding - theory and practice. In Lossless Compression Handbook, 101–152. Academic Press, 2002.
[23] Shilane, P.; Min, P.; Kazhdan, M.; Funkhouser, T.: The Princeton shape benchmark. In Proc. of SMI'04, 2004 167–178, Washington, DC, USA. http://dx.doi.org/10.1109/SMI.2004.1314504
[24] The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3dscanrep, 1996.
[25] 3D CAD browser. http://www.3dcadbrowser.com/, 2001.