Visual Analysis for **E**xtremely **La**rge-**S**cale **S**cientific **Co**mputing

D4.3 – User Interaction Metaphors Specification

Version 1.0

Specification of the user interaction metaphors for the query formulation and the visual analysis process

Deliverable Information

Grant Agreement no	619439
Web Site	http://www.velassco.eu/
Related WP & Task:	WP4 – High performance visualization / scalable visual analysis T4.3 – Interaction metaphors to express queries on visualization features
Due date	30/06/2015
Dissemination Level	Public
Nature	Report
Author/s	Miguel Pasenau de Riera, Andreas Dietrich, Frank Michel
Contributors	Alvaro Janda

Approvals

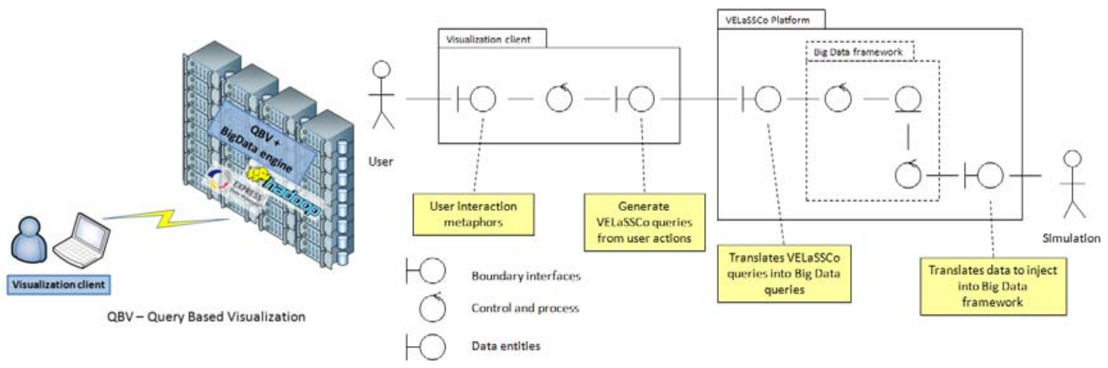
	Name	Institution	Date	OK
Author	Miguel Pasenau	CIMNE		
Task Leader	Miguel Pasenau	CIMNE		
WP Leader	Frank Michel	FRAUNHOFER		
Coordinator				
Contributor				
Contributor				

Table of Contents

Table of figures

1 Introduction

This document explains the user interaction with the VELaSSCo platform and the mapping of the user actions to VELaSSCo Vqueries. The point of contact between the user and the VELaSSCo platform is the visualization client, as show in Figure 1, which will translate the user actions and interactions into internal commands or will communicate with the VELaSSCo platform and issue one or several VQueries.

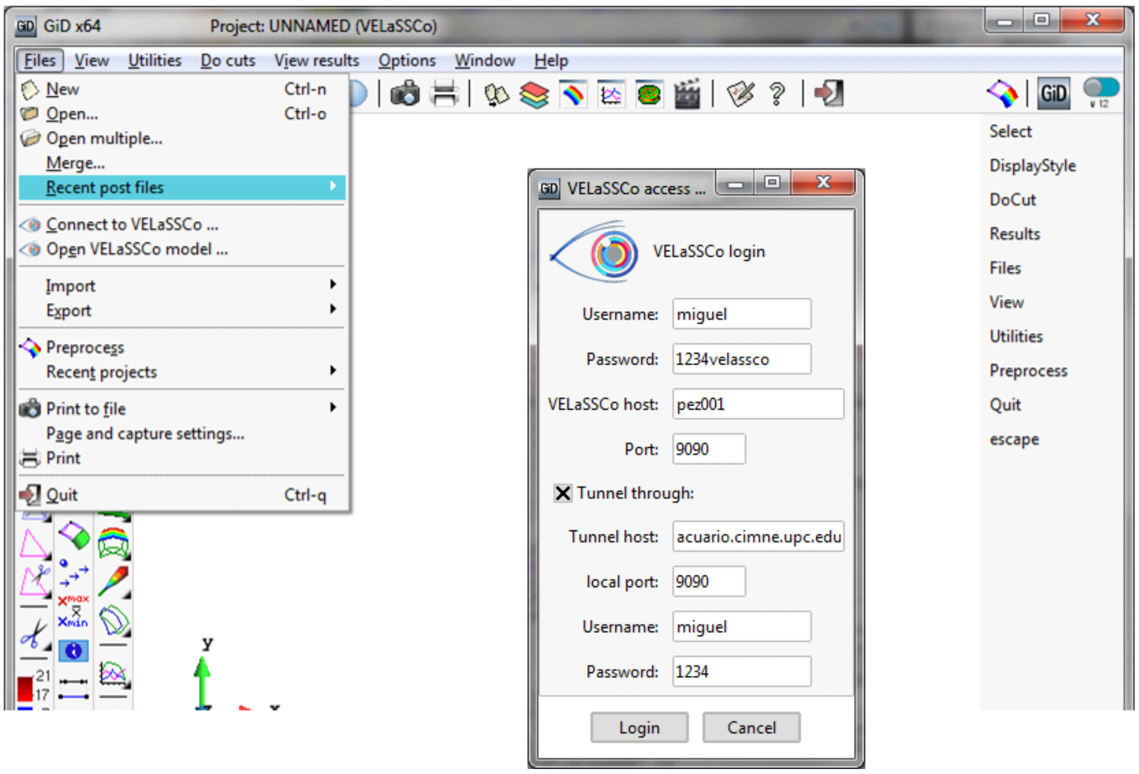


QBV + BigData engine
Visualization client
QBV – Query Based Visualization
User
Visualization client
VELaSSCo Platform
Big Data framework
User interaction metaphors
Generate VELaSSCo queries from user actions
Translates VELaSSCo queries into Big Data queries
Translates data to inject into Big Data framework
Simulation
Boundary interfaces
Control and process
Data entities

Figure 1. The user interacts with the visualization client, GiD or iFX, which will forward user requests to the VELaSSCo platform, using the Access Library, in order to access and analyze, i.e. post-process, the data ingested by the simulation program or already existing files.

From the user's point of view, the interaction with the simulation data stored in the VELaSSCo platform will be the same as the interaction with models stored locally. For instance if the user wants to visualize a colour map of the *Pressure* distribution over a surface, the visualization client will access the local data if the model selected by the user is stored locally or it will access the VELaSSCo platform to retrieve the simulation data, but the user will use the same *Contour Fill* icon or window to perform the action.

Still specific extensions to the visualization client will be needed, as the VELaSSCo tool is a remote service and the visualization client should be able to handle the connection and session information. Some of these extensions include new menu entries, options and windows, like the login and model selection windows.



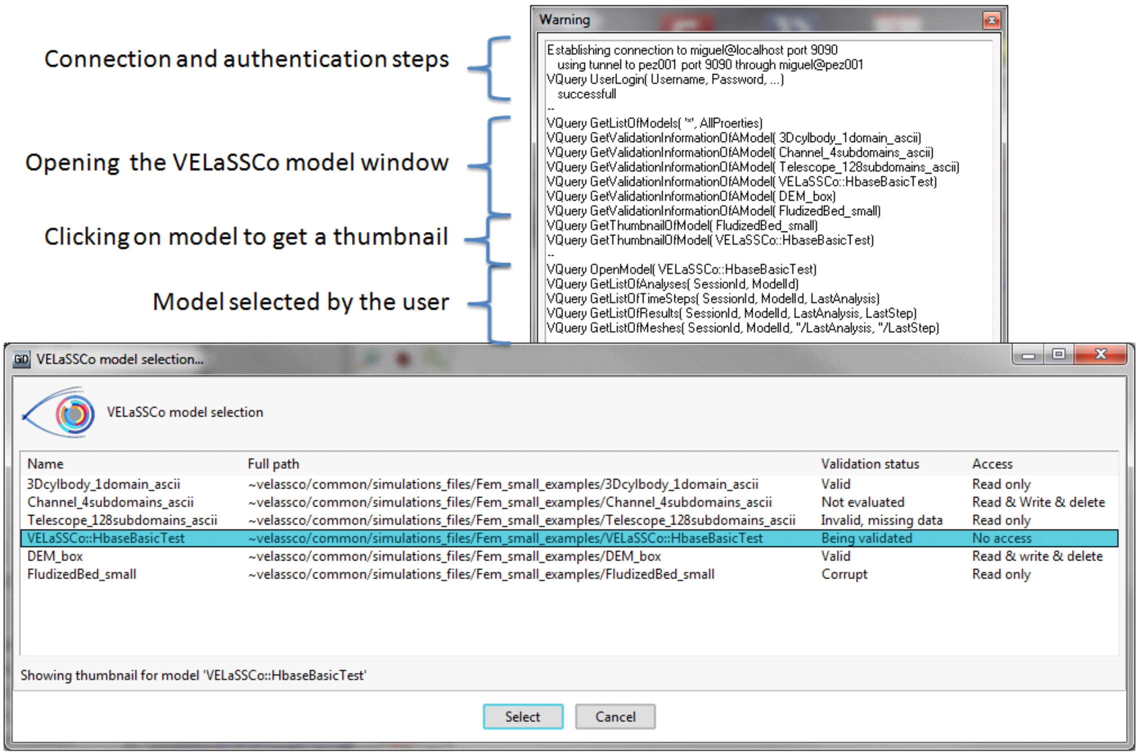
Username: miguel
Password: 1234velassco
VELaSSCo host: pez001
Port: 9090
[✓] Tunnel through:
Tunnel host: acuario.cimne.upc.edu
local port: 9090
Username: miguel
Password: 1234

Figure 2. Two new additions to the *Files* menu in the visualization client GiD, that allows the user to login to the VELaSSCo platform and to select a model selection from the platform. Also shown is the connection window.

Some of the user interactions will be solved in the same visualization client, like zooming, rotating or switching layers on and off, but other user actions will trigger one or several VELaSSCo queries.

For instance when the user selects a model to visualize, the sequence of issued VQueries from the visualization client can be:

1. OpenModel,
2. GetValidationInformationOfAModel,
3. GetBoundingBox, to have a global view of the model
4. GetListOfMeshes, to list the different layers of the model so that the user can switch then on or off,
5. GetListOfAnalyses, GetListOfTimeSteps and GetListOfResults, to show the user the output results of the simulation program and which can be selected to visualize and analyze,
6. and several GetBoundaryOfAMesh will be issued to render the boundary meshes of the model.


Connection and authentication steps
Opening the VELaSSCo model window
Clicking on model to get a thumbnail
Model selected by the user
Warning
VELaSSCo model selection...
VELaSSCo model selection
Select
Cancel

```
Establishing connection to miguel@localhost port 9090
   using tunnel to pez001 port 9090 through miguel@pez001
VQuery UserLogin( Username, Password, ...)
   successfull
--
VQuery GetListOfModels( '*', AllProerties)
VQuery GetValidationInformationOfAModel( 3Dcylbody_1domain_ascii)
VQuery GetValidationInformationOfAModel( Channel_4subdomains_ascii)
VQuery GetValidationInformationOfAModel( Telescope_128subdomains_ascii)
VQuery GetValidationInformationOfAModel( VELaSSCo::HbaseBasicTest)
VQuery GetValidationInformationOfAModel( DEM_box)
VQuery GetValidationInformationOfAModel( FludizedBed_small)
VQuery GetThumbnailOfModel( FludizedBed_small)
VQuery GetThumbnailOfModel( VELaSSCo::HbaseBasicTest)
--
VQuery OpenModel( VELaSSCo::HbaseBasicTest)
VQuery GetListOfAnalyses( SessionId, ModelId)
VQuery GetListOfTimeSteps( SessionId, ModelId, LastAnalysis)
VQuery GetListOfResults( SessionId, ModelId, LastAnalysis, LastStep)
VQuery GetListOfMeshes( SessionId, ModelId, "/LastAnalysis, "/LastStep)
```

Name	Full path	Validation status	Access
3Dcylbody_1domain_ascii	~velassco/common/simulations_files/Fem_small_examples/3Dcylbody_1domain_ascii	Valid	Read only
Channel_4subdomains_ascii	~velassco/common/simulations_files/Fem_small_examples/Channel_4subdomains_ascii	Not evaluated	Read & Write & delete
Telescope_128subdomains_ascii	~velassco/common/simulations_files/Fem_small_examples/Telescope_128subdomains_ascii	Invalid, missing data	Read only
VELaSSCo::HbaseBasicTest	~velassco/common/simulations_files/Fem_small_examples/VELaSSCo::HbaseBasicTest	Being validated	No access
DEM_box	~velassco/common/simulations_files/Fem_small_examples/DEM_box	Valid	Read & write & delete
FludizedBed_small	~velassco/common/simulations_files/Fem_small_examples/FludizedBed_small	Corrupt	Read only

Showing thumbnail for model 'VELaSSCo::HbaseBasicTest'

Figure 3. Just opening the VELaSSCo model window will trigger several VQueries, as well as clicking on different models in order to get a small picture. After the user selects a model, some information is retrieved to create the visualizations client menus and windows.

In the previous *Pressure* colour map example, when the user selects the *Contour Fill* icon for a VELaSSCo simulatied model, the VQuery GetResultFromVerticesID will be launched to get the result values to draw a colour map on the surface meshes.

After a summary of the VELasSCo architecture, the types of VQueries, the next sections describes the user interactions with the visualization clients and the effects these interactions has in the VELaSSCo platform.

1.1 User interaction in the VELaSSCo architecture

As stated in the previous deliverables (D2.2, D2.3, D2.4, D2.5 and D3.1), the VELaSSCo architecture is composed of different layers (two outside and two inside the platform). For the visualization process, the strategy consists in sending queries from the visualization client to the platform, to retrieve the information stored and produced by the platform.

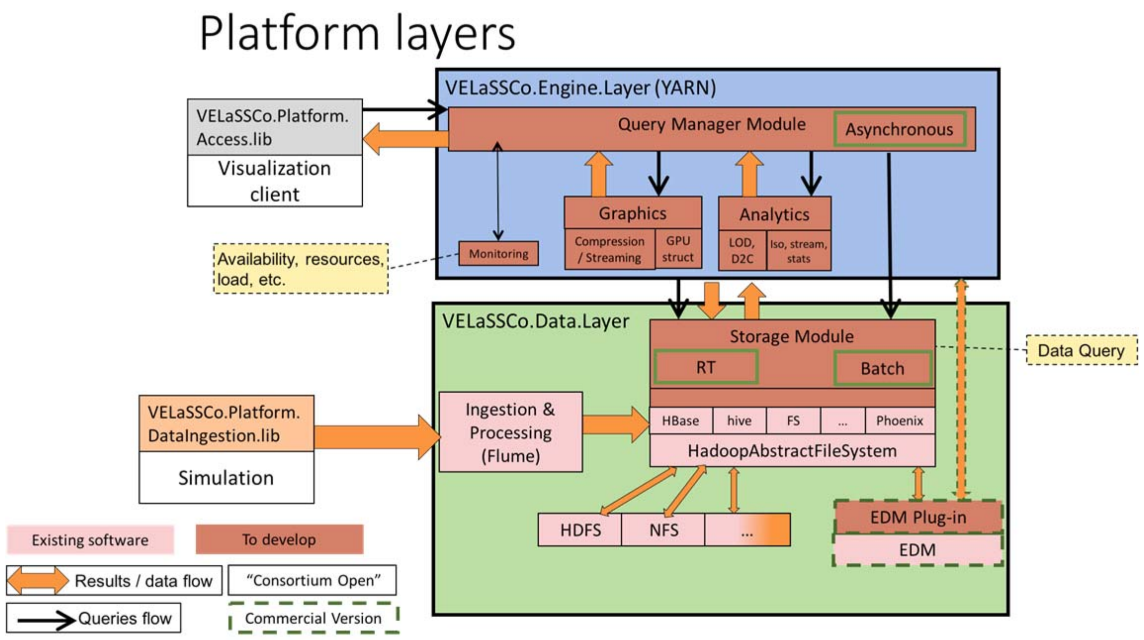


Figure 4. Schema of the VELaSSCo Architecture.

The visualization client will handle and translate the user's requests and interactions and call the provided API entry points by the Access Library. The library is responsible for issuing the corresponding VELaSSCo VQueries, do the necessary translation of its parameters and wait for results of the execution og the VQueries in the platform. The communication with the VELaSSCo platform will be done using the thrift protocol, as shown in Figure 5. Specifically the communication will be done with the Query Manager Module of the Engine Layer, which will manage the VQuery workflows inside the platform.

This process is described with more detail in section 3.3.

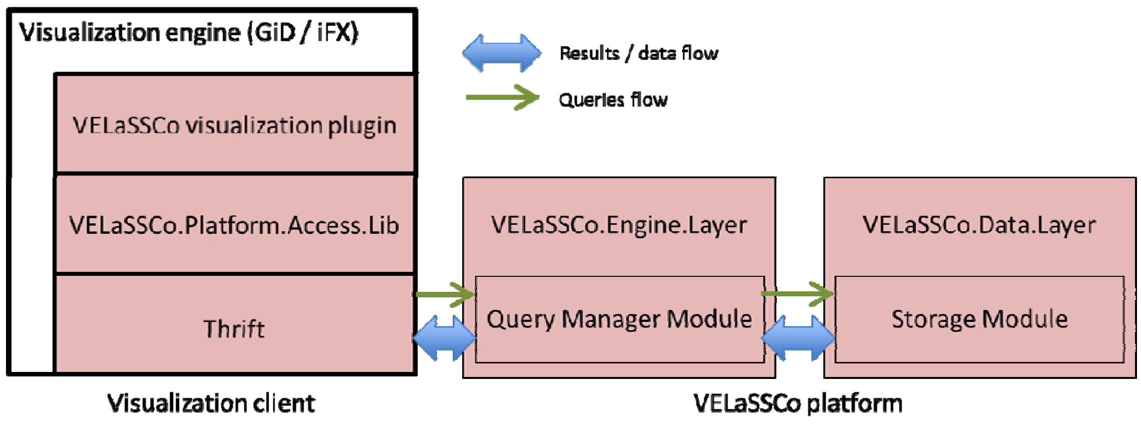


Figure 5. Query workflow triggered by the visualization engine.

1.2 User interaction classification

Different user interactions will trigger different processes in the VELaSSCo platform, from simple simulation data accesses to complex analytics. Deliverable D2.5 already classified the functionalities of the platform into VQuery families.

In this project, the workload execution is expressed by queries named: VELaSSCo queries (VQueries). A Vquery can express functionality at the user level and also at the data ingestion level. A VQuery is an aggregation of operations from several modules, which can evolve at different levels.

Depending on the functionalities involved and their mapping to VQueries, the user interactions can be grouped in:

- Session: which provides the frame for access to the simulation contents data, including user login and logout, model selection and access to the model's thumbnails and validation information, among others. These will be mapped to Session VQueries.
- Basic visualization: on-demand actions like render a global view of model, switch layers on and off, or renaming them, visualize surface meshes or render the boundary of volume models. These will be mostly mapped to Direct Result Queries and some of them to Result Analysis Queries, like *GetBoundingBox* or *GetBoundaryOfAMesh*.
- Advanced visualization: progressive display and streaming of simulation data like moving the camera from point A to point B, dynamic zoom actions or animating particles. These will trigger complex VQueries which will be designed after the release of the first prototype.
- Result visualizations: like render a colour map of a result, i.e. contour fill, draw some result labels, given a node list or an element list, visualize result graphs like point evolution, result on nodes or points. These will also be mostly mapped to Direct Result Queries and some of them to Result Analysis Queries, like *InterpolateResultForPointList*.
- Result feature extraction: visualize iso-surfaces, stream lines, cuts, interpolation over iso-surfaces and cuts, which will imply processing the simulation data. These interactions will be mapped to Result Analysis Queries.
- Complex result creations: discrete to continuum transformations, statistical results, which will also be mapped to Result Analysis Queries.

1.3 Ensuring feed-back and interaction:

Some of the above mentioned user interactions may represent a time-consuming and complex sequences of operations inside the VELaSSCo platform.

To ensure a feed-back to the user while the complex Vqueries are being performed, a preview of the output result will be send to the visualization client.

To ensure also a fast transfer of the big amount of information to be sent to the visualization client from a complex VQuery on a full detailed model, further simplification techniques will be used to reduce the network bandwidth and to transfer only the needed information for rendering.

Further details about these technique and how are they integrated in the VELaSSCo platform are explained in D2.5 and D4.1.

Following sections will describe each type of user interaction in detail and their repercussions in the platform regarding to the actions that will be triggered, modules-operations-components involved and what data is accessed and stored. Nevertheless the focus will be on the user interaction: user input and the client's feedback.

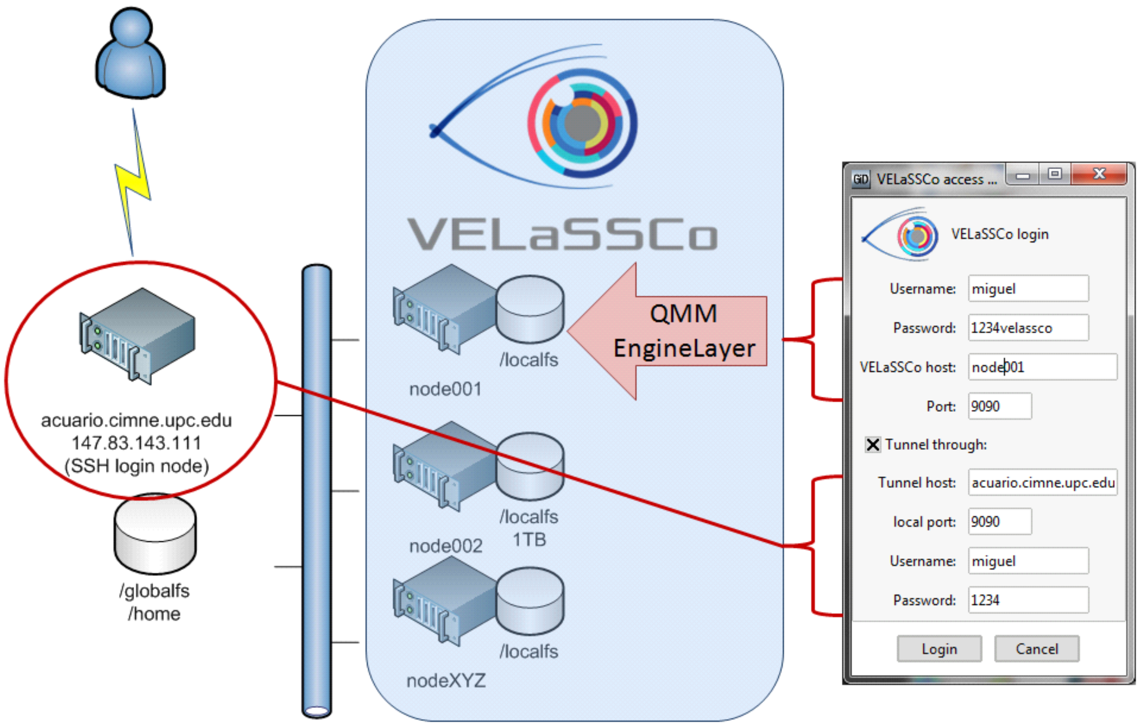
2 Session queries

2.1 User interaction

In order to allow the user the use of the VELaSSCo platform from the visualization clients, some modifications in the user interface are needed, and also internally to discriminate if the simulation result are local or the Access Library should be used to retrieve them from the VELaSSCo platform.

As shown in Figure 2, two new options have been added in GiD, one to allow the user to connect to the platform and another one to select the model ingested into the VELaSSCo platform to be visualized and analyzed. The connection window can also be seen in Figure 2. The different fields in the window reflects the nature of the VELaSSCo platform when it is deployed inside an HPC cluster where the access should be done through an specific login-node, which does not belong to the VELaSSCo platform.

A detailed correspondence between the physical layout and the fields of the window is shown in Figure 6.

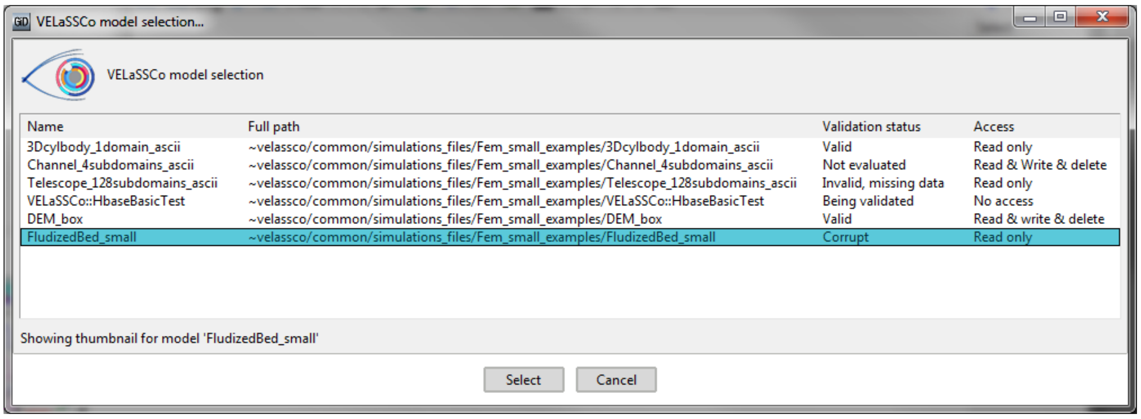


Username: miguel
Password: 1234velassco
VELaSSCo host: node001
Port: 9090
☒ Tunnel through:
Tunnel host: acuario.cimne.upc.edu
local port: 9090
Username: miguel
Password: 1234

Figure 6. Connection details to the VELaSSCo platform installed in an HPC cluster.

Once the user is logged in the VELaSSCo platform, he can select any model he has access to from the model selection window, in order to analyze and visualize its results in the visualization client.

This is achieved through the model selection window, which also shows some details about the simulated data models ingested in the VELaSSCo platform, as shown in Figure 7.



Name	Full path	Validation status	Access
3Dcylbody_1domain_ascii	~velassco/common/simulations_files/Fem_small_examples/3Dcylbody_1domain_ascii	Valid	Read only
Channel_4subdomains_ascii	~velassco/common/simulations_files/Fem_small_examples/Channel_4subdomains_ascii	Not evaluated	Read & Write & delete
Telescope_128subdomains_ascii	~velassco/common/simulations_files/Fem_small_examples/Telescope_128subdomains_ascii	Invalid, missing data	Read only
VELaSSCo::HbaseBasicTest	~velassco/common/simulations_files/Fem_small_examples/VELaSSCo::HbaseBasicTest	Being validated	No access
DEM_box	~velassco/common/simulations_files/Fem_small_examples/DEM_box	Valid	Read & write & delete
FludizedBed_small	~velassco/common/simulations_files/Fem_small_examples/FludizedBed_small	Corrupt	Read only

Showing thumbnail for model 'FludizedBed_small'

Figure 7. VELaSSCo model selection window.

Once the model is selected, then the visualization client will request information about several model properties and fill the corresponding menus and windows. Also a global view of the model will be rendered.

2.2 Mapping to the VELaSSCo architecture

User Login:

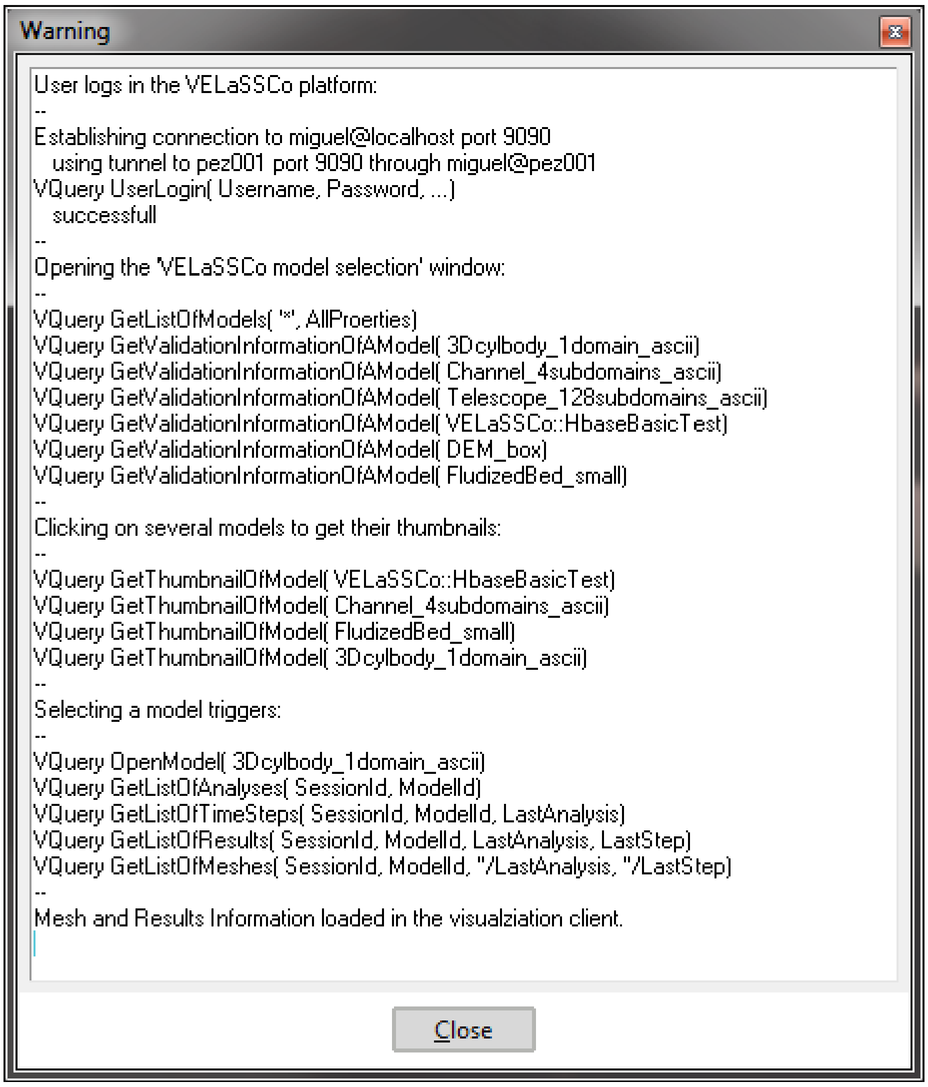
When the user fill the details in the VELaSSCo login window, the visualization client will eventually create a tunnel through the HPC login node to the VELaSSCo platform, specifically to the node in which the Engine Layer application is running, will connect to the Query Manager Module (QMM) and issue a *UserLogin* VQuery.

Model Selection window:

Just opening the window and filling the different fields with information about the models present VELaSSco platform will trigger the GetListOfModels and several GetValidationInformationOfAModel VQueries. Also as the user clicks on any model, the GetThumbnailOfAModel Vquery will be called in order to provide the user with a visual information about the selected model.

When the user selects a model to be opened by the visualization client, several menus and windows should be filled with information about the model to allow the user the selection of different analyses, time-step and result to be visualized.

A sequence example of the VQueries called by the visualization client, through the Access Library, from the VELaSSCo login step until the visualization client is ready for the user interaction with the selected simulation data can be seen in Figure 8.



```
User logs in the VELaSSCo platform:
--
Establishing connection to miguel@localhost port 9090
   using tunnel to pez001 port 9090 through miguel@pez001
VQuery UserLogin( Username, Password, ...)
   successfull
--
Opening the 'VELaSSCo model selection' window:
--
VQuery GetListOfModels( '*', AllProerties)
VQuery GetValidationInformationOfAModel( 3Dcylbody_1domain_ascii)
VQuery GetValidationInformationOfAModel( Channel_4subdomains_ascii)
VQuery GetValidationInformationOfAModel( Telescope_128subdomains_ascii)
VQuery GetValidationInformationOfAModel( VELaSSCo::HbaseBasicTest)
VQuery GetValidationInformationOfAModel( DEM_box)
VQuery GetValidationInformationOfAModel( FludizedBed_small)
--
Clicking on several models to get their thumbnails:
--
VQuery GetThumbnailOfModel( VELaSSCo::HbaseBasicTest)
VQuery GetThumbnailOfModel( Channel_4subdomains_ascii)
VQuery GetThumbnailOfModel( FludizedBed_small)
VQuery GetThumbnailOfModel( 3Dcylbody_1domain_ascii)
--
Selecting a model triggers:
--
VQuery OpenModel( 3Dcylbody_1domain_ascii)
VQuery GetListOfAnalyses( SessionId, ModelId)
VQuery GetListOfTimeSteps( SessionId, ModelId, LastAnalysis)
VQuery GetListOfResults( SessionId, ModelId, LastAnalysis, LastStep)
VQuery GetListOfMeshes( SessionId, ModelId, "/LastAnalysis, "/LastStep)
--
Mesh and Results Information loaded in the visualziation client.
```

Figure 8. VELaSSCo VQueries issued by the login and model selection steps.

2.3 Workflow in the VELaSSCo architecture

As already described in the D1.3 deliverable a typical user interaction session, which includes the login process, some result visualization and closing the session, may look as represented in following figure:

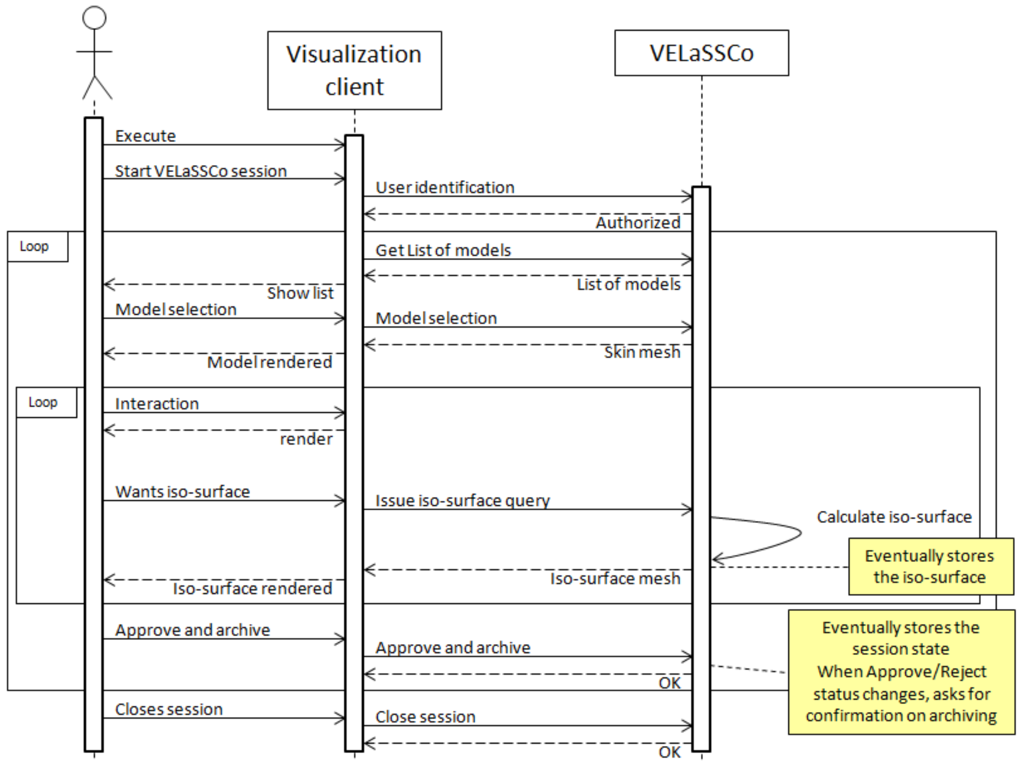


Figure 9. Sequence diagram of user interaction with the system.

2.4 Special considerations for the Hadoop and EDM scenarios.

The access to the models and the user permission are handled by the Query Manager Module of the Engine Layer. In the Hadoop scenario this information is stored in HBase tables. In the EDM scenario and additional authentication step may be required in the EDM engine which will require modifications in the user interface and which will be decided after the release of the first prototype.

3 Basic Visualization

As outlined in the VELaSSCo architecture overview (see D2.2), a user interacts with the VELaSSCo platform from a workstation which runs a visualization client. The visualization client consists of a visualization engine (e.g., iFX or GiD), which makes use of the VELaSSCo platform access library to send queries to the platform and to receive the result of the queries. Inspection of received data is done on the user's workstation with the visualization engines.

In this chapter, we are going to describe the different interaction modes in which a user can inspect the data that has been provided by the platform, how this translates to VELaSSCo queries, and what parts of the architecture are involved when executing these queries.

3.1 User Interaction

Typically, a user will first open a session and select a model via session queries (Chapter 2), then specify which information is to be retrieved with result or analysis queries (Chapter 5 and 0). Once the requested data has been transported to the visualization engine within the visualization client, the user can interact with it, whereas this interaction can trigger new queries to get new, additional, or updated data.

When working on a graphical workstation the received data is displayed as a 2D or 3D model. The graphical user interface of the visualization engines provides four basic modes to interact with the model:

3.1.1 Navigating

Navigating the model requires changing the viewport that displays a region of interest of the model. This is done by changing parameters of a virtual camera with the mouse (or any other appropriate input device). Supported standard viewing operations are:

- **Panning.** The model is moved within the camera projection plane (i.e., up/down and left/right).
- **Zooming.** The model is moved closer to the camera or farther away. In addition to controlling zoom with mouse gestures (e.g., moving the mouse forward/backward), there will also be the possibility to select a region in the window plane using a rubber-banding box. The client will then zoom the area up to the full window.
- **Rotating.** In a 3D scenario the model is rotated around the primary axes.

Depending on the use case, the client will perform the navigation either completely within the available data, or request new data from the platform. For example, when (virtually) walking through a large architectural structure (like the oil rig shown in D1.3) only the data that is visible from the current viewpoint needs to be loaded. When the camera is moved to other parts of the scene, the newly visible data can be loaded on demand.

3.1.2 Retrieving Information

Although both the original data that went into a simulation as well as the results are transformed into graphical primitives for visual inspection, it is beneficial to have access to the underlying numerical data. For example, in a particle simulation a user might be interested in the concrete velocity values of a specific particle.

Such operations will be supported by the GUI, which will allow the user to pick with the mouse pointer a specific part of the model. The visualization client can then display the numerical data in a separate window or as tooltips.

3.1.3 Selecting Parts

In case the user is not interested in the whole model and needs to focus only on a specific part, the user interface will allow for enabling/disabling a subset of the current dataset. This can be done by in several ways:

- **Layers.** Data that has been grouped beforehand (e.g., layers of a mesh) can be turned on and off per group.
- **Value ranges.** The user can define data values range. For example, only particles with attribute values in a certain range should be displayed.
- **Rubber-banding.** Similar to the use of rubber-banding to define a zoom area, this method can also be employed to select a region of interest of data that is displayed exclusively while any other data is hidden.

3.1.4 Modification

In addition to inspecting data coming from the platform, there is also the possibility to do modifications. Types of modification are:

- **Attribute modification.** This involves changing data that is attached to objects, meshes, and vertices. For example, this can be colors of particles, or names of layers.
- **Mesh modification.** This involves merging and splitting of meshes as well as the generation of new or additional meshes, e.g., creating the boundary of a volume mesh, splitting a mesh into halves, or merge unconnected components into a compound mesh.

Some of these modifications, like attribute changes, can be done locally on the client. Others, such as complicated mesh operations need to be performed on the platform. In all cases the outcome of changes can eventually be stored in the platform database.

3.2 Mapping to the VELaSSCo Architecture

When not used in a progressive or streaming mode (see next chapter), the actual visualization is performed *after* a query has been executed, i.e., after the result of the query has been transferred to the client (and the visualization engine). Therefore, all interaction modes that are only used for visualization (but do not lead to a modification) are working independently from the platform on the user's workstation, and do not trigger any queries or other platform operations.

As mentioned above, modification operations may require additional analysis queries (see Chapter 0) to the platform if the computational requirements are too heavy to be performed on the client side (one example is the splitting of a complete volume mesh). The outcome of analysis queries is usually automatically stored in the platform, without the need for the client to take care of storage operations.

3.3 Workflow in the VELaSSCo Architecture

Figure 10 shows the high-level workflow between the visualization client and the VELaSSCo platform. Operating the graphical user interface of the visualization engine can trigger different types of VQueries (session queries, direct result queries, or analysis queries). In all cases, the visualization engine calls functions of the platform access library, which will then send a query command to the engine layer that retrieves the requested result from the data layer. More detail of how this works can be found in D2.4 and D4.1.

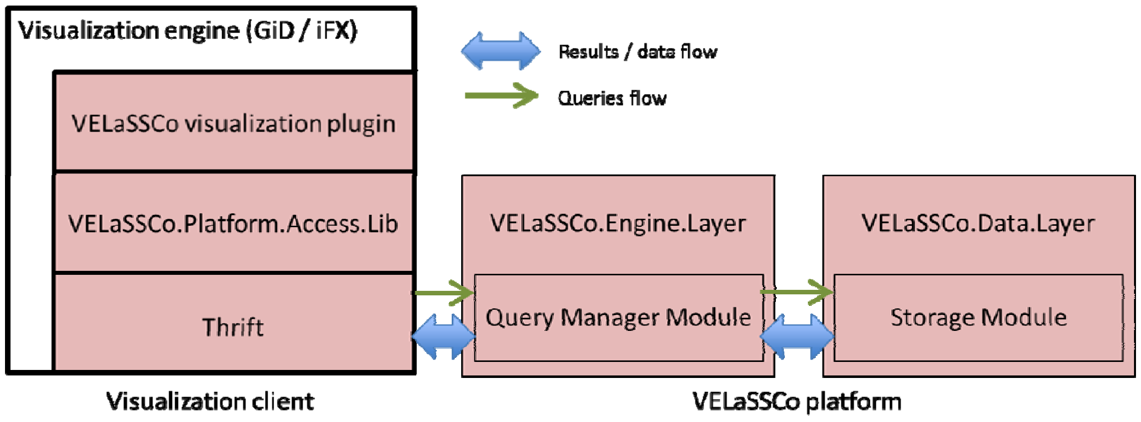


Figure 10. Query workflow triggered by the visualization engine.

In order to reduce bandwidth, query results are prepared in a suitable way so that the information can be displayed by the visualization engines at high speed with minimal latencies (see D4.1).

This is sufficient for scenarios where the results of a query can be quickly computed and also where all data can be stored as a whole on the client machine. However, there are cases where multi-resolution and streaming approaches have to be employed to ensure interactivity and fast feedback. How to deal with these situations will be described in the following chapter.

4 Advanced Visualization: Asynchronous Display, Multi-Resolution and Streaming

Due to the scale of simulations handled by the VELaSSCo platform, the amount of data produced by result and analysis queries can be extremely large. Often, such datasets exceed the storage capabilities of the visualization client (both on the CPU and GPU). Moreover, the time needed to retrieve or compute new data on the VELaSSCo HPC platform can be substantial, therefore further impeding immediate feedback to the client. To overcome this problem, i.e., to shorten the “time to picture”, a number of different strategies can be leveraged:

4.1 Asynchronous Display

Asynchronously display in this context means that visualization on the user’s workstation and retrieving (or computing) query results on the HPC server are timely overlapping. There are two main scenarios where asynchronous techniques can be useful:

- **Fast preview.** Post-processing operations on large data can require a considerable amount of time, even if done in an HPC environment. A fast preview mode quickly generates a preview result by performing post-processing on a simplified dataset (e.g., iso-surface computation can be done by working on a coarser mesh than original one). The coarse result can then be viewed on the client while the full result is computed simultaneously by post-processing the full dataset.
- **Streaming.** In case post-processing needs to be performed continuously (e.g., when a user would like to cycle through a range of iso-surface values in a volumetric mesh), data that is needed next can be computed/retrieved in the background, parallel to viewing the currently loaded data.

4.2 Multi-Resolution

Rather than generating output results at a fixed single resolution, the platform can also generate multiple levels-of-detail (LOD). In a typical multi-resolution scenario, a fully detailed dataset is encoded as a coarse representation plus additional detail, which can be added gradually to produce a progressively more detailed representation. An overview of LOD techniques can, e.g., be found in [1].

In the VELaSSCo architecture, the graphics module can create such an LOD representation from the original data if required. All LODs can be stored in the storage layer for later use. These LODs are then used in two scenarios:

- **LOD on the CPU/GPU.** Today, GPUs typically have less memory available on board than their host machine. 3D scenes with a high number of geometrical primitives may not fit into GPU memory, but can often reside in CPU memory as a whole. Using a LOD scene, the CPU will then upload only the required detail level to the GPU. In this use case, the VELaSSCo platform sends all levels to the client, which then manages the data autonomously.

- **Streaming.** LOD can also be combined with streaming. Here, all detail levels reside in the VELaSSCo platform. The client requests appropriate levels from the query manager as needed.

4.3 Streaming

When using streaming, data is sent from the platform to the client in parts instead of a single big block. Streaming can be combined with asynchronous and multi-resolution modes if beneficial. Subdivision of the original data for streaming can be done in different domains:

- **Time.** The most common use for streaming is sending data related to varying time steps. For example, a simulation may produce an animation where the trajectories of particles a computed over a specific time frame. Visualization of this animation on the client can be handled similar to movie streaming.
- **Space.** In this case, data is subdivided spatially. For example, consider an iso-surface with a great geometrical extend. Such a scene can be subdivided (e.g., by a regular grid). Depending on the camera settings of a virtual camera, only the area around the observer needs to be sent to the client. When the camera moves, the client requests the areas that newly become visible.
- **Detail.** Here, the platform progressively sends new detail to the client if requested, e.g., when zooming close to an iso-surface. This can be based on a pre-computed LOD representation, but it would also be possible to compute new detail levels in the analysis module on demand.

4.4 Mapping and Workflow in the VELaSSCo Architecture

Advanced visualization in the VELaSSCo architecture uses the same channels and modules as the basic visualization. The difference to the basic visualization is that a single monolithic query is separated into subqueries (e.g., to request a subset of the original data at a specific detail level). Also, a number of specific session queries need to be implemented to provide control over the platform (e.g., to request the current state of an asynchronous query).

The actual implementation of the advanced visualization mode is currently subject to research. The prototypic version of the VELaSSCo platform (due in M21) will have basic visualization implemented. The final version of the platform (due in M30) will feature the mentioned advanced modes.

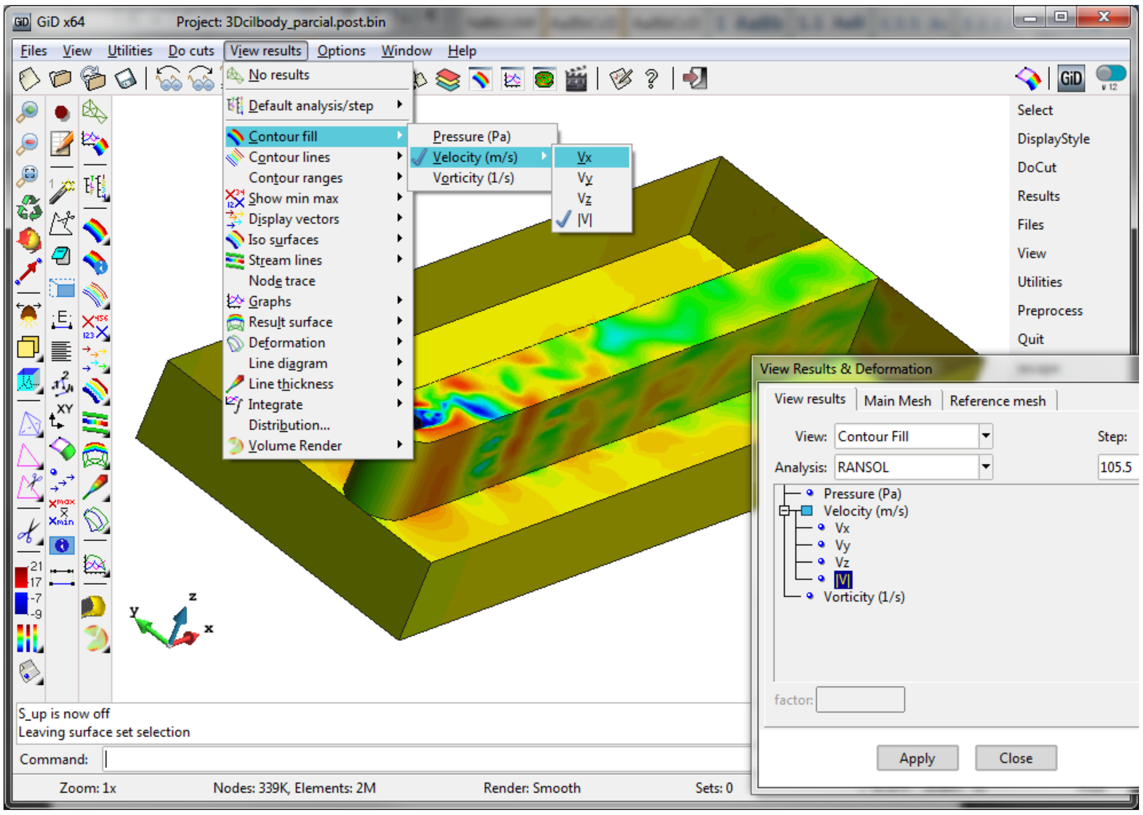
5 Result queries

5.1 User interaction

As with any post-processing program, the user will be able to select different result visualizations and feature extraction from the visualization client. The visualization client will then interact with the VELaSSCo platform using the *Access Library* to get the needed data to render the information the user has requested, see scheme at Figure 10.

User interactions that results in the extraction of features like cut-planes, iso-surfaces, stream-lines or in the creation of new results like discrete 2 continuum transformation or results statistics, they are explained in the next chapter.

Typical results visualization like *Contour Fill*, which draws the results values using a colour map, Display Vectors, Deformations or Result surface, which extrudes the surface mesh according to the values of the selected results, can be chosen using the usual icons, menus or windows, as shown in Figure 11. For a full detailed explanation on how to select and adjust these visualizations types please refer to the GiD user's manual [2].

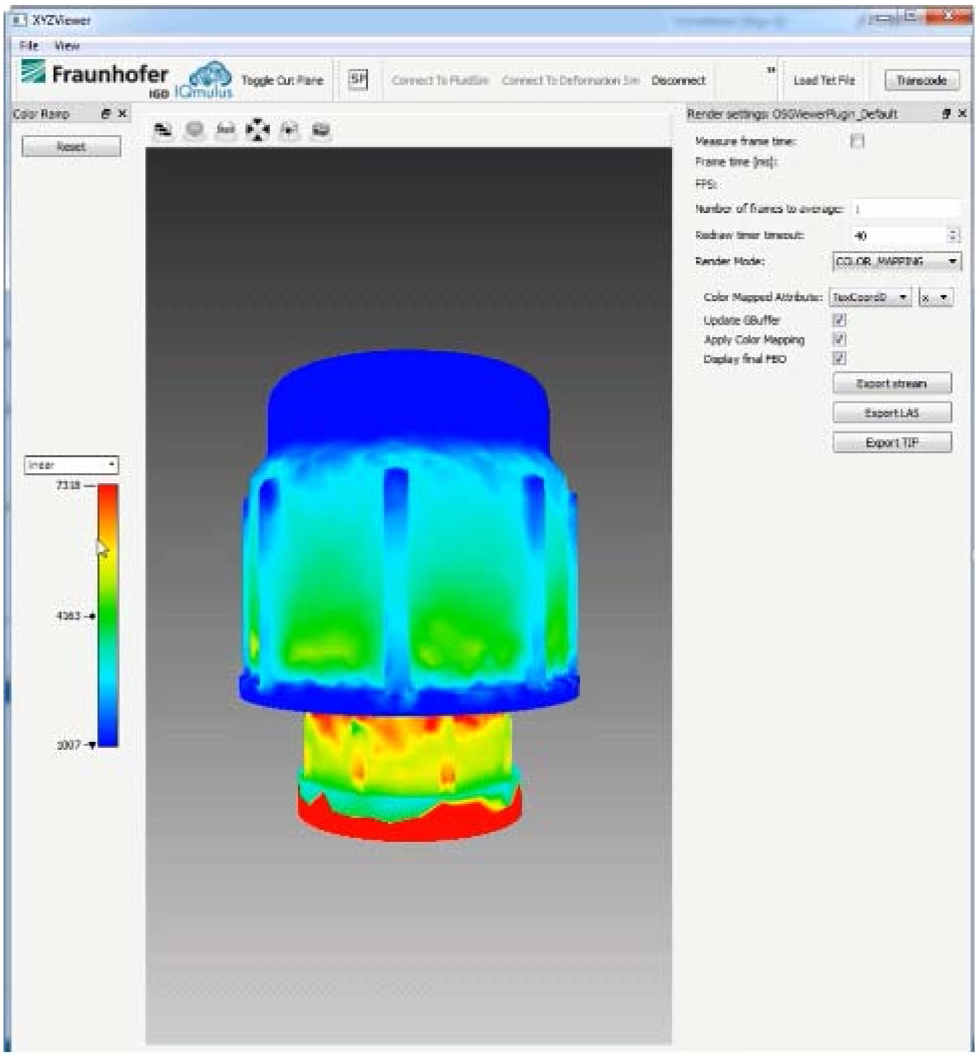


View: Contour Fill
Analysis: RANSOL
Step: 105.5

S_up is now off
Leaving surface set selection
Command:
Zoom: 1x   Nodes: 339K, Elements: 2M   Render: Smooth   Sets: 0

Figure 11. Result visualization selection with the vertical icon bar, on the left, using top-down menus, in the middle, or through the View Results window.

In order to draw these visualization types, the visualization client only needs the surface meshes and the simulation results values, located at the nodes, i.e. vertices of the mesh, or at the elements of the mesh.



Render settings: OSGViewerPlugin_Default
Measure frame time: ☐
Frame time [ms]:
FPS:
Number of frames to average: 1
Redraw timer timeout: 40
Render Mode: COLOR_MAPPING
Color Mapped Attribute: TexCoord0 x
Update GBuffer ✓
Apply Color Mapping ✓
Display final FBO ✓

Figure 12. Contour Fill result visualization on iFX.

Once a result visualization is selected by the user for a certain result, and their result values retrieved from the VELaSSCo platform, the other visualization of the same result do not need to retrieve any new values, as the visualization client already has them. These results visualizations include *Contour Fill*, *Contour Lines*, *Result Surface*, *Deformation*, *Display vectors*. Adjusting the visualization parameters, like the number of colours to be used in *Contour Fill*, or the colour map, or changing the deformation factor of the vectors sizes does not incur in new VELaSSCo VQueries.

Also when the user wants to display some labels for a certain result, if the result is being visualized then the client does not need to connect to the platform to retrieve these. Only when the labels of a new result are to be drawn, these values are to be retrieved.

The visualization clients may also show results values as graphs like *Point Evolution* or *Line Graph* as show in Figure 13 and Figure 14 . Depending on the graph type selected by the user not only result values should be retrieved for some mesh vertices for a certain time-step but for all time-step of the simulation, *point evolution*, or they need to be interpolated for spatial points inside the mesh elements, *line graph*.

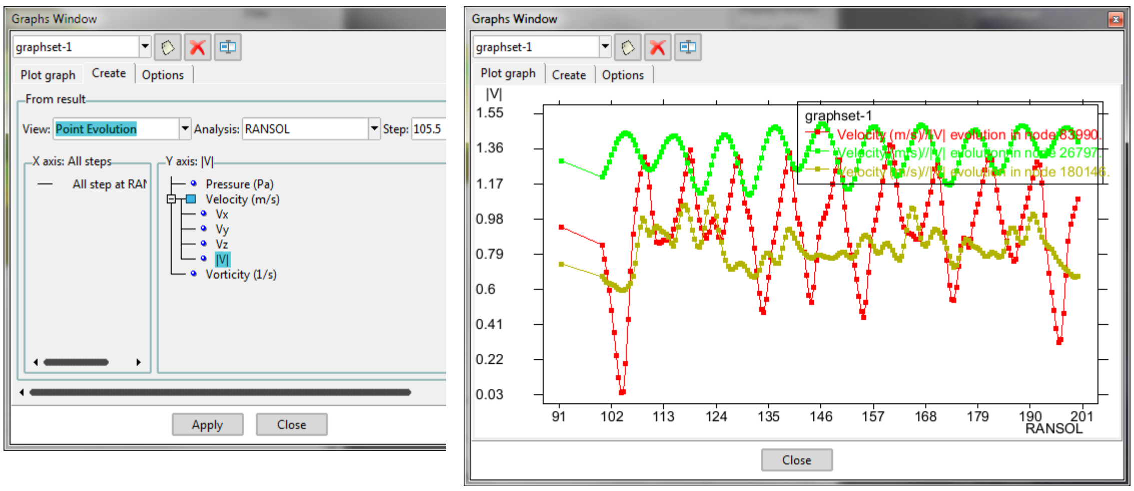


Figure 13. Graph example in the visualization client GiD: evolution of the *Velocity modulus* for some selected nodes.

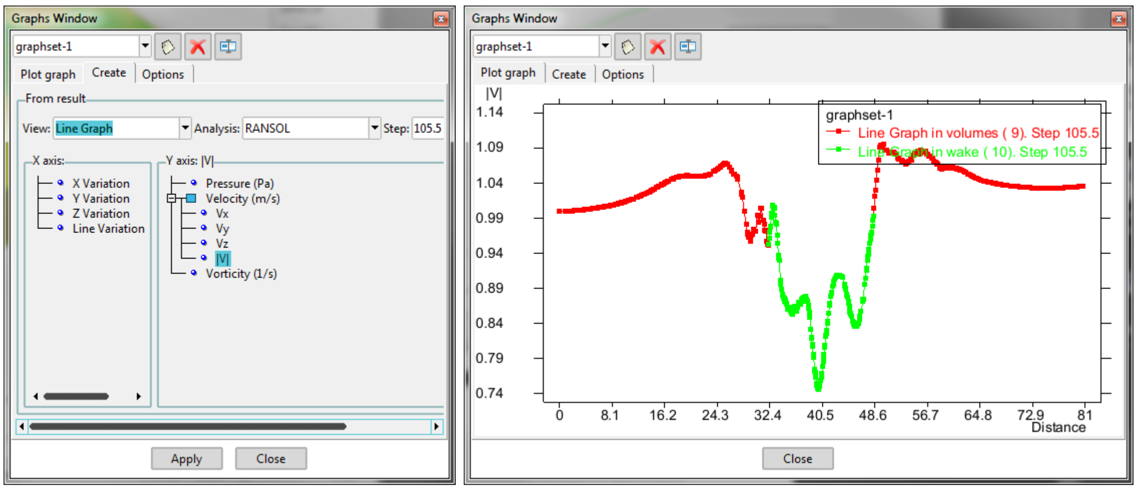


Figure 14. Another graph example: the *Velocity modulus* distribution along the diagonal of the fluid volume.

5.2 Mapping to the VELaSSCo architecture

Analysis, time-step and Result selection:

In order to fill the menus and the windows, as shown in Figure 11, from where the user chooses the analysis, time-step and results to visualize, the visualization client needs to retrieve the list of available result from the VELaSSCo platform. After the model is selected by the user the visualization client may retrieve all the results metadata, that is name, components, type and location, for all time-steps and for all analyses or this metadata may be retrieved when the users selects another analysis or time-step to visualize.

The related VELaSSCo VQueries are: *GetListOfAnalyses*, *GetListOfTimeSteps* and *GetListOfResults* which will access the Data Layer to retrieve the information.

Result visualization: Contour Fill, Vectors, etc.:

To render the user selected result visualization and if the result values are not already present, the visualization client will retrieve them from the VELaSSCo platform using the *GetResultFromVerticesID* or the *GetResultFromElementsIDs* VQueries depending on the location of the data. The *VerticesID* and *ElementsIDs* are already known by the visualization client as they are needed to render simulation model's mesh, and were requested with the VQueries *GetListOfMeshes*, *GetMeshVertices*, *GetMeshElements* or *GetMeshDrawData*, depending on the visualization, see previous chapter.

As commented in the previous section, changing the visualization type of the same result may reuse the already retrieved result values. From instance if the users changes the visualization from *Result Surface* to *Contour Fill* for the same result, the VELaSSCo platform will not be contacted to retrieve the result values. But if the user changes from *Contour Fill* to *Display Vectors*, although the result selected is the same, the visualization client will contact the VELaSSCo platform to retrieve the values for all *x*, *y* and *z* components of the result to draw the vector, as only one component is used to render the *Contour Fill*.

Labels:

When the user selects some mesh vertices to draw the result values as floating labels, these actions may trigger the *GetResultFromVerticesID* or *GetResultFromElementsIDs* VQueries in the VELaSSCo platform, as with the result visualization, but only if the visualization client does not already have these values.

Graphs:

As mentioned in the previous section, depending on the Graph type selected by the user different actions will be needed to draw the requested information:

- Point evolution:
  - if mesh vertices are selected then their result values are retrieved with the *GetResultFromVerticesID* VQuery, but for all time-steps of the analysis.
  - If spatial points are selected then their result values should be interpolated in the VELaSSCo platform by performing the *GetResultForPoints* VQuery.
- Line graph: the results values should be interpolated for the user selected segment that crosses the meshes in the VELaSSCo platform using the *GetSegmentWithResult* VQuery.

Other complex result visualizations will rely on new, still not designed VQueries. These new requirements will be specified after the release of the first prototype and its evaluation by the user-panel.

5.3 Workflow in the VELaSSCo architecture

In the next figure a schema of the triggered VQueries depending on the user actions is represented. Including the information requested by the visualization client after the users selects a model to analyze their simulated results, a colour map representation of a result and a line graph selection of another result.

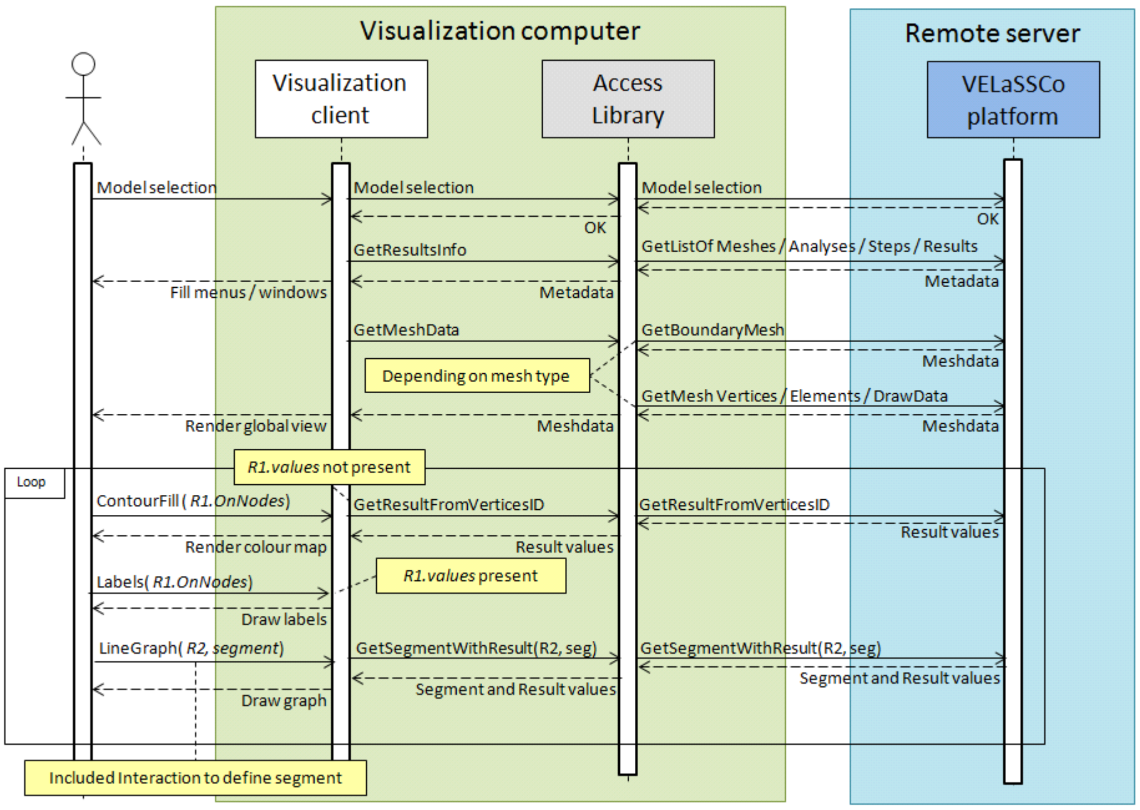


Figure 15. Workflow showing the interaction between the user, visualization client, access library and the VELaSSCo platform for three different actions: model selection and rendering, *Contour Fill* representation and *Line Graph* visualization after selecting a segment across the volume meshes.

For a detailed workflow of the VQueries across the VELaSSCo platform, please look into the D3.1 deliverable.

5.4 Multi-resolution version

For the first prototype the management of the multi-resolution model will be done at the visualization client. After the evaluation of the first prototype this management will be translated to the *Query Manager Module* of the *Engine Layer* in the VELaSSCo platform. A tighter relation between the visualization client, its hardware and the Access Library is needed for the platform to do an *automatic* multi-resolution management. At this early stage of the platform, the decision was taken to let the visualization client decide if a *Preview* of the data to be visualized is needed in order to avoid long waiting times for the user.

If the visualization client foresees that the execution of VQueries requires an appreciable amount of time, or the graphics capabilities too modest, first a request to the access library will be done for a preview of the data and, in parallel, the request for full resolution model will be issued. It can be expected that the VQueries on the coarser model will finish before the same VQueries over the full resolution model, and so their output will be received and render by the visualization client before the output of the Vqueries over the full-resolution model.

6 Analysis queries

6.1 User interaction

The analysis queries implemented in the VELaSSCo environment can be categorized into two main groups: feature extraction and data creation.

The set of **feature extraction** queries covers all the analysis that start from an initial set of simulation data and builds derived values intended to provide new meaningful information that can help to the engineering decision process. In particular, the specific analyses that belong to the feature extraction area are iso-surfaces data extraction, stream lines, cut planes and results interpolation over the original model or over iso-surfaces and cut planes.

The **data creation** queries set mainly regards the analysis of processing DEM simulation data both temporally and spatially in order to project their result into a continuum field. In particular, this covers the implementation of the discrete to continuum transformation that outputs new simulation results similar to FEM simulation data. Moreover, the extraction of the statistical properties from the DEM and FEM results is also considered as data creation.

In the following sections, a quick overview of the main analysis queries to be implemented in VELaSSCo platforms is provided. Moreover, these sections are mainly focused on describing how the user will interact with the Graphical User Interface in order to run the different analysis queries.

6.1.1 Isosurfaces data extraction

An isosurface is a three-dimensional surface (or two-dimensional line) that represents points of a constant value (e.g. pressure, temperature, velocity, density) within a volume (or surface) of space. In other words, an isosurface visualization is done by passing through all the points which have the same result's value inside a volume mesh or surface mesh (see **Figure 16. Example of an isosurface visualization [2].**Figure 16).

The first step to create an isosurface is the selection of a result between the list of simulation results available for the simulation model, analysis and time-step. After choosing a result or a result component, the user can select between several options:

- **Exact:** the user can input several fixed values for which the isosurfaces will be computed.
- **Automatic:** the user is asked for the number of isosurfaces to be created. The values for the isosurfaces between the minimum and the maximum of the selected result are automatically calculated by the platform.
- **Automatic Width:** the user is asked for a width. This width is used to create as many isosurfaces as are needed between the minimum and maximum values of the selected result.

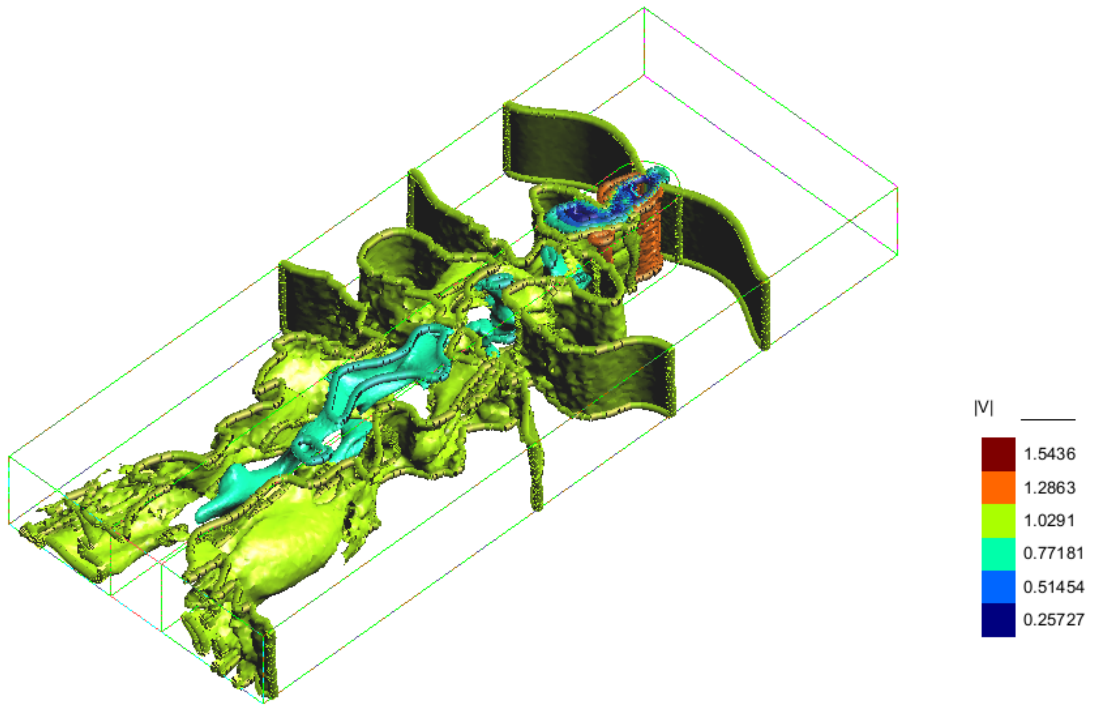


Figure 16. Example of an isosurface visualization [2].

6.1.2 Streamlines data extraction

Streamlines are a family of curves that are tangent to the velocity vector of the flow. They are independent of time and describe the direction of the flow field at a given instant in time. In VELaSSCo platform, the stream lines will be computed within single volume meshes, i.e. they cannot jump from one volume mesh to the next volume mesh, even if they are close neighbors. An example of streamlines visualization is shown in Figure 17.

The user will be able to create the streamlines by selecting the corresponding option in the menu. After choosing a vector result (typically velocity or momentum field), the user will be asked for a set of starting points for computation of the streamlines. The user will be able to use several ways to choose the initial points for the streamlines:

- **Clicking on the screen:** the user will be able to select one or several points by clicking on the screen of the GUI. The points will be the intersections between the lines orthogonal to the screen and the plane parallel to the screen and containing the centre of rotation.
- **Selecting nodes of existing meshes:** The position of the selected nodes will be used as starting points for the streamlines.
- **Along a line:** the user will be able to define a segment of a line along which several starting points for the streamlines will be created. After defining the segment (coordinates for the initial and end point), the user will be asked for

the number of starting points along the segment. The position of the starting points will be calculated based on this user input and the previous definition of the segment.

- **In a quad**: the user will be able to define a quadrilateral area by entering four lines with the mouse. The defined quadrilateral are will be used to a M x N matrix of points. Thus, once the user has defined the quadrilateral area, the GUI will ask for the numbers M and N of starting points for the streamlines to be computed.

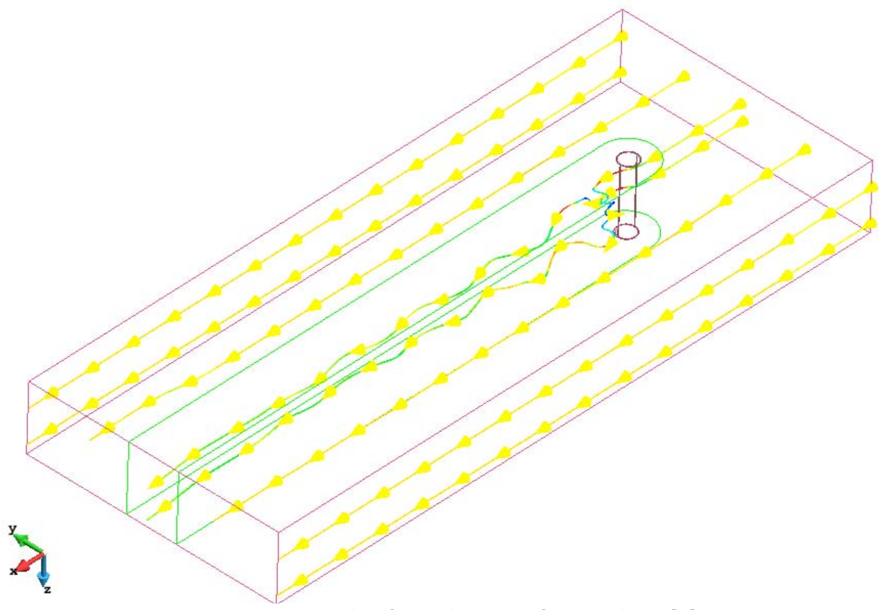


Figure 17. Example of visualization of streamlines [2].

6.1.3 Cutting planes

A cutting plane is defined as the plane that cuts through a simulation model by computing the intersection with a 3D box. The result is a truncated polygonal mesh that shows internal details that of the original model. The resulting cutting plane can be saved as an independent geometry. Figure 18 shows an example of a model where different cutting planes has been created.

The user can choose between different options to specify the cutting plane:

- **2 Points**: the user can define the plane by two points (introducing the coordinates xyz of the points or just clicking on the screen) and the visual direction orthogonal to the screen.
- **3 Points**: the user defines the plane by three points (introducing the coordinates xyz of the points or just clicking on the screen).
- **Succession:** This option allows the user to specify an axis (X, Y or Z) that will be used to create cut planes orthogonal to the specified axis. After

specifying the axis, the user will be asked for the number of cutting planes to be created.

.

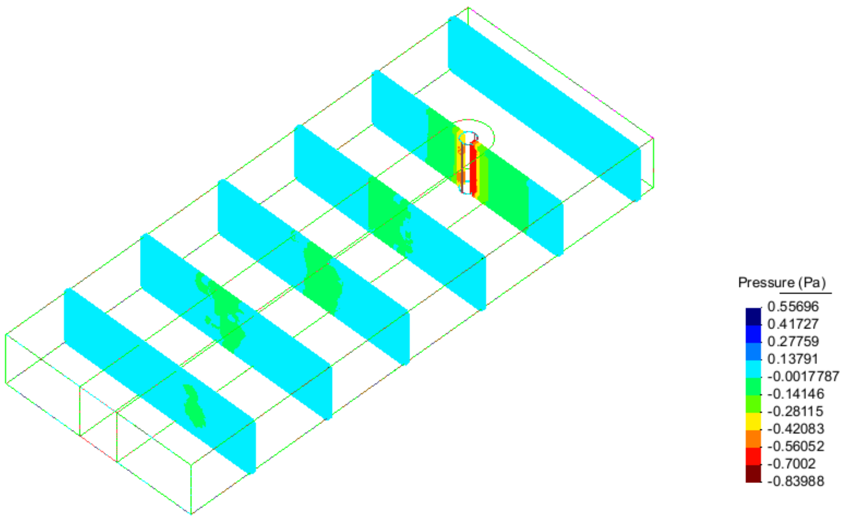


Figure 18. Example of visualization of different cutting planes in a simulation model [2].

6.1.4 Results interpolation for points

This capability of VELaSSCo platform consists on interpolating the values of a simulation model result for a set of points (x, y, z coordinates) at a specific time-step.

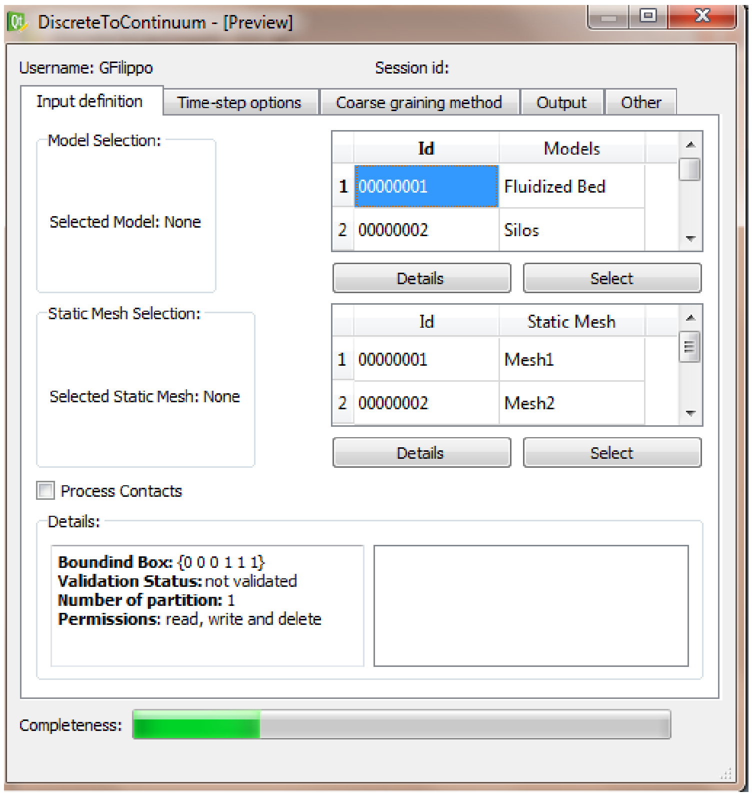
First of all, the user will need to select from the menu a specific result or component result of a simulation model, analysis and time-step. Afterwards, the user will be able to use different options to specify the set of points at which he/she wants to obtain the interpolation of the selected result:

- **Clicking on the screen:** the user will be able to select one or several points by clicking on the screen of the GUI. Alternatively, the user could introduce the coordinates x,y,z for each of the points in a terminal.
- **Along a line:** the user will be able to define a segment of a line along which several points for results interpolation will be created. After defining the segment (coordinates for the initial and end point), the user will be asked for the number of points along the segment. The position (x,y,z coordinates) of the points will be calculated based on this user input and the previous definition of the segment.

6.1.5 Discrete to continuum transformation

The discrete to continuum (D2C) transformation consist on spatial averaging methodology of DEM simulation data related to the particles and contacts. By means of this methodology, the new bulk properties and results are computed from the discrete DEM data and projected into a continuum field (static mesh) similar to FEM simulation data.

The D2C GUI main window is used to define the input for the algorithm (simulations data and static mesh), averaging parameters and the different output options. In particular, this GUI is composed of five contextual menus: “Input definition”, “Time Averaging”, “Coarse-graining method”, “Output definition” and “Other”. An example of the GIU is shown in Figure 19. The following subsections describes in details each of the contextual menus and the user interaction with them.



Username: GFilippo

Session id:

Selected Model: None

	Id	Models
1	00000001	Fluidized Bed
2	00000002	Silos

Selected Static Mesh: None

	Id	Static Mesh
1	00000001	Mesh1
2	00000002	Mesh2

Boundind Box: {0 0 0 1 1 1}
Validation Status: not validated
Number of partition: 1
Permissions: read, write and delete

Figure 19. Example of the “Input definition” section of the GIU for the Discrete to Continuum transformation.

Input definition menu:

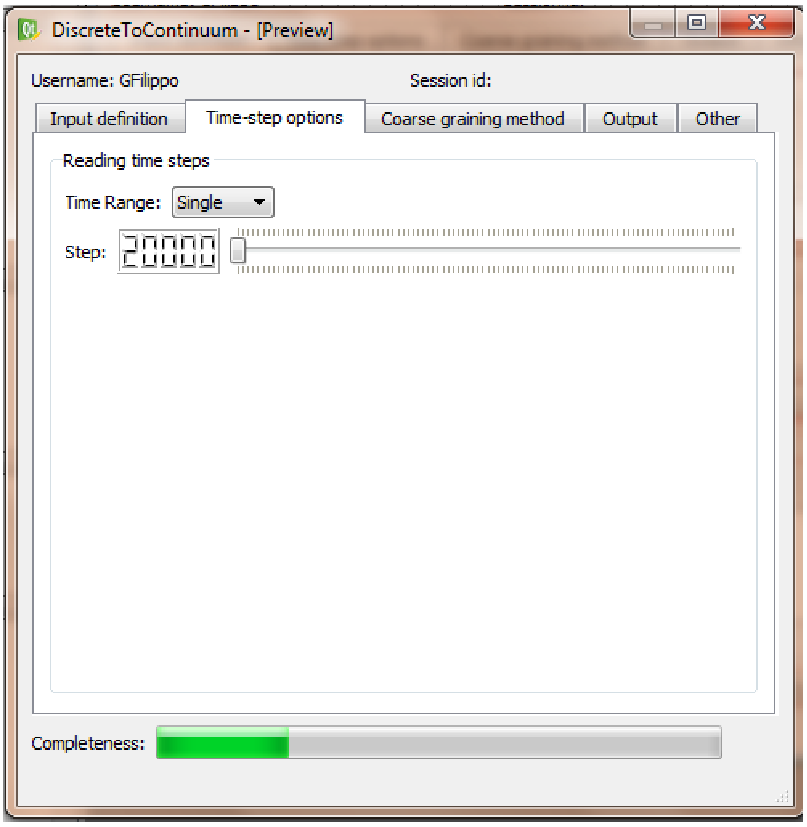
The first step for the user is to define an input **Model and a Static Mesh** by selecting them from the **table views** on the right side of the form, where all the models and static meshes stored into the platform are shown (see Figure 19). When a given model (or a static mesh) has the focus, the user can click on the **“details”** button in order to get more details about the selected object or click on **“Select”** in order to choose it as an input for the query. When user asks for details, theses besides a screenshot preview of the selected model are shown in the bottom area.

The process contacts check box is an option that activates/deactivates the processing of particle-particle and particle wall contact.

Time-step options menu:

The time-step menu allows the user to define the input time-step options for the D2C query (see Figure 20). With the "time range" combo box the user can choose between different time-step options to process the results of the input model:

- "ALL": all the time-steps of the input simulation model will be processed.
- "SINGLE": a single time-step of the input simulation model will be processed. In this case, the user can select the value of the time-step.
- "INTERVAL": an interval of the time-step of the input simulation model will be processed. This interval is defined by the user by introducing a value for "Time Start" and for "Time End".
- "SET": only the time-step defined by the user will be processed.


DiscreteToContinuum - [Preview]

Username: GFilippo

Session id:

Reading time steps

Time Range: Single

Step: 20000

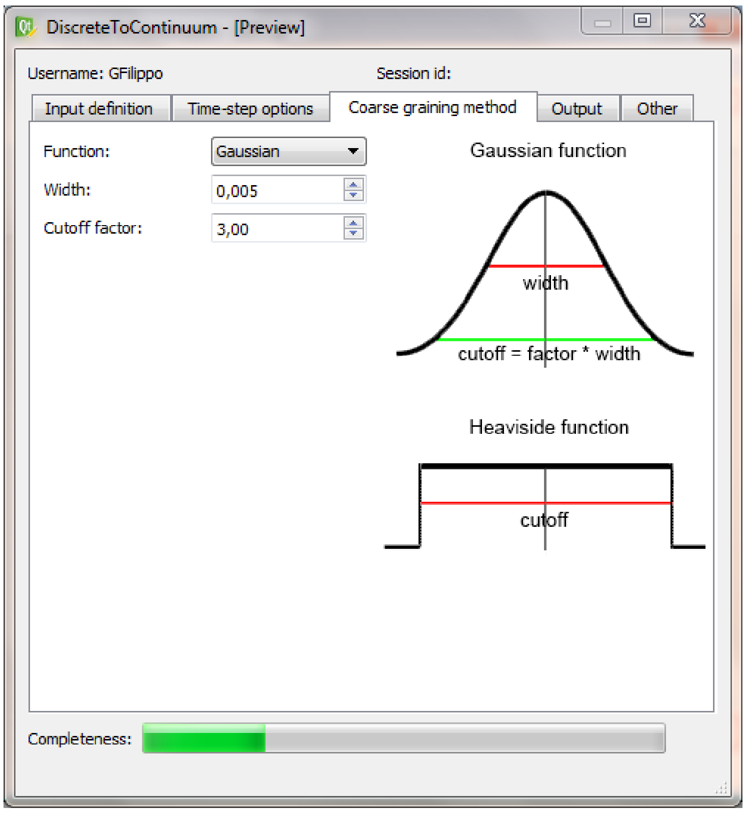
Completeness:

Figure 20. Example of the "Time-step options" section of the GIU for the Discrete to Continuum transformation.

Coarse graining method menu:

This menu section is used for the selection of the coarse-graining method and definition of the input parameters to compute the spatial averaging of the D2C transformation (see Figure 21). It is composed of the following fields:

- **"Function":** this combo box allows the user to select the spatial averaging function: "Heaiside" or "Gaussian".
- **"Width"**: this field corresponds to the coarse-graining length scale for the spatial averaging when the Gaussian is selected in the field "Function". In the case, that "Heaviside" is selected, the field "Width" is not shown to the user.
- **"Cuttoff" or "Cuttoff factor"**: this field corresponds to the maximum length over which the spatial averaging is computed. In the case of "Gaussian" function, the value specified by the user corresponds to a factor of the defined width.


DiscreteToContinuum - [Preview]

Username: GFilippo
Session id:

Function: Gaussian
Width: 0,005
Cutoff factor: 3,00

Completeness:

Figure 21. Example of the "Coarse-graining" section of the GIU for the Discrete to Continuum transformation.

Output:

The output menu provides a friendly user interface for defining the VELaSSCo platform required information to store the result. The user has to specify an analysis name and select the set of result values that have to be computed during the D2C run (Mass density, Momentum, Velocity field, Stress field, …).

To execute the query, the user clicks on the "run query button" and it will be sent to VELaSSCo platform for computation. Another important feature provided in this menu is the execution progress bar that shows in real-time about the current progress of the computation by receiving feedback from the Query Manager module of the platform.

After the D2C of the model has been processed and the output generated, the visualization capabilities of the rendering engine and visualisation software can be used to explore the new results. Moreover, the structure of the output of D2C transformation is similar to a FEM simulation model and as consequence the user can take advantage of all the analytics queries of VELaSSCo platform for further analysis of the results (iso-surfaces, stream lines, cutting planes, interpolation, statistical results, etc…).

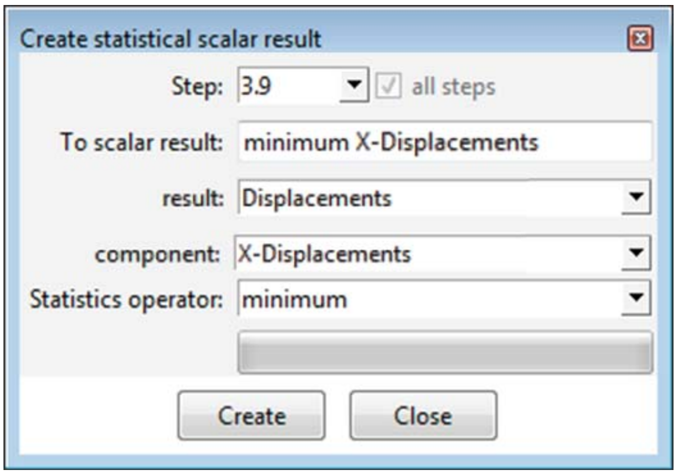
6.1.6 Statistical results

Statistical results queries regard the implementation of statistical methodologies related to the results or components results of the nodes for both single and multiple time-steps. The statistical results can be the average value, minimum and/or maximum values, accumulated value and standard deviation. It will also include the generation of graphs to analyse the results in the mesh such as histograms.

The user will be able to create statistical scalar results using the corresponding option of the menu. Afterwards, a window similar to the one shown in Figure 22 will be opened. In this new window the user will be able to select the different options to compute the statistical results:

- **"Step":** the user will be able to select the time-step or time-steps to be used in calculation of the statistical results.
- **"To scalar result":** the user can introduce in this field a name for the statistical results to be computed.
- **"Result":** the user needs to select one of the results of the simulation model.
- **"Component":** the user will be able to select one of the components of the selected result for the calculation.

- **"Statistic operator":** the user can select the type of statistical operator to be used for the calculation: average, minimum, maximum, accumulated, standard deviation.


Create statistical scalar result

Step: 3.9 ☑ all steps

To scalar result: minimum X-Displacements

result: Displacements

component: X-Displacements

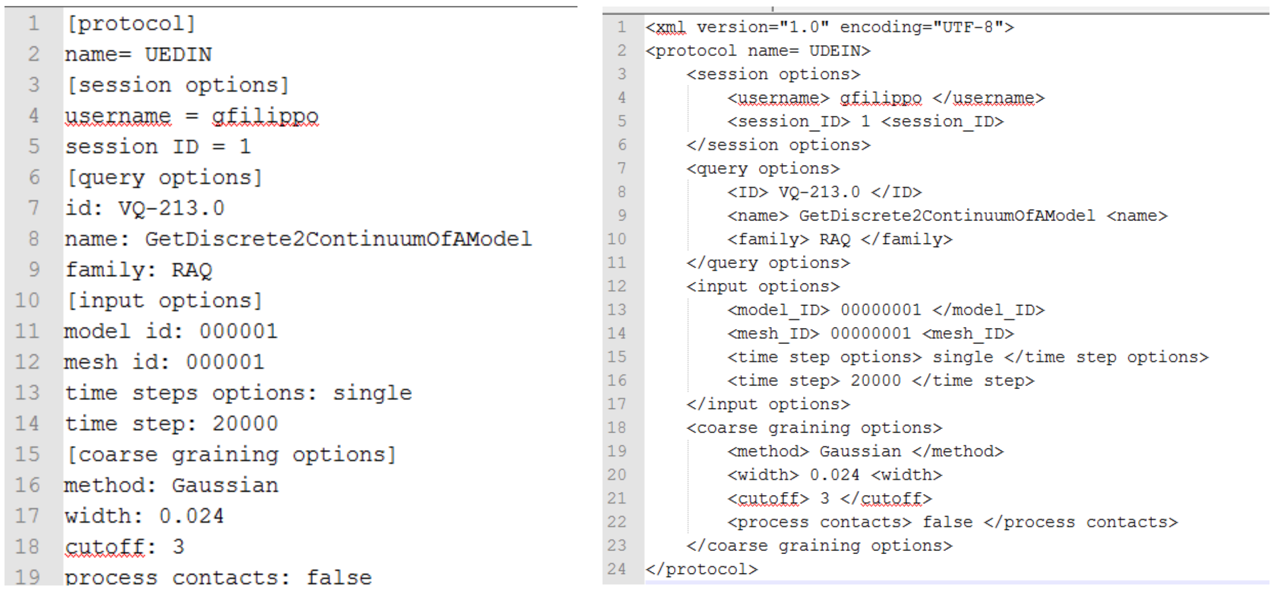
Statistics operator: minimum

Figure 22. Example of the GIU for creating statistical results [2].

6.2 Mapping to the VELaSSCo architecture

The interaction (error detection and result data) between client and server can be handled by a communication protocol between the GUI and the VELaSSCo platform. When the user selects the corresponding result visualization or operation (cuts, iso-surface, discrete 2 continuum transformation, etc.) the visualization client will gather all parameters defined by the user in the graphical user interface components, and these values will be organized and encapsulated, and eventually stored into an INI file or XML file (see figure 8). This structured data will be mapped into a protocol buffer and sent as an asynchronous request message to the query manager module.

The platform will receive the message in the buffer, will parse it and will translate it into a VELaSSCo query. What follows are two examples of the GUI output both in INI and XML format that can represent a D2C query request to send to the server (see Figure 23).



INI format

```
[protocol]
name= UEDIN
[session options]
username = gfilippo
session ID = 1
[query options]
id: VQ-213.0
name: GetDiscrete2ContinuumOfAModel
family: RAQ
[input options]
model id: 000001
mesh id: 000001
time steps options: single
time step: 20000
[coarse graining options]
method: Gaussian
width: 0.024
cutoff: 3
process contacts: false
```

XML format

```
<xml version="1.0" encoding="UTF-8">
<protocol name= UDEIN>
    <session options>
        <username> gfilippo </username>
        <session_ID> 1 <session_ID>
    </session options>
    <query options>
        <ID> VQ-213.0 </ID>
        <name> GetDiscrete2ContinuumOfAModel <name>
        <family> RAQ </family>
    </query options>
    <input options>
        <model_ID> 00000001 </model_ID>
        <mesh_ID> 00000001 <mesh_ID>
        <time step options> single </time step options>
        <time step> 20000 </time step>
    </input options>
    <coarse graining options>
        <method> Gaussian </method>
        <width> 0.024 <width>
        <cutoff> 3 </cutoff>
        <process contacts> false </process contacts>
    </coarse graining options>
</protocol>
```

Figure 23. Example of the INI and XML format that represent query request.

At the end of the computation, for all the different query (Discrete 2 continuum transformation, iso-surfaces, cutting planes, stream lines, …) results will be stored in HBase tables server-side. The platform will send back to the client the results of the VQuery executions. In case of error the server will send the client a special message containing information about the detected error.

6.3 Workflow in the VELaSSCo architecture

In this section, a possible scenario for a D2C query request is treated. It regards the D2C execution process from the query run request to the visualization of results. The actors involved in this example are:

- The **user**, which is the software end user.
- The **client**, which is the Visualization Client Software.
- The **server**, which is the VELaSSCo platform and specifically the Query Manager module.

The whole activity can be divided into the following sequential operations:

- The User (by interacting with the GUI) defines the input parameters for the D2C query.
- The User executes the query.
- The Client maps the query parameters from the GUI format into a structured format.
- The Client sends a request (with the query input parameters) to the Server.

- The Server receives the request from the Client and bypasses it to the Query Manager Module.
- The Server (Query Manager Module) parses the message into a Query Manager Module friendly format and executes the specific query.
- The Server sends back to the Client a message that contains the query outcome and information where results are stored
- The Client sends a request to the Server to retrieve data results
- The Server send back to the Client the required results data
- The Client receives from the Server the results data and visualize it

7 References

[1] David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshney, Benjamin Watson, and Robert Huebner. Level of Detail for 3D Graphics. Morgan Kaufmann, 2002.

[2] CIMNE (2014). GiD – The personal pre and postprocessor. User Reference Manual.