VELaSSCo

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

D5.2. Architecture Evaluation

Version #1.0

Deliverable Information

Grant Agreement no	619439
Web Site	http://www.velassco.eu/
Related WP & Task:	WP5 - Usability and Effectiveness Evaluation Task 5.2 – Architecture Evaluation
Due date	30/11/2015
Dissemination Level	Public
Nature	Report
Author/s	Ivan Martinez, Miguel Angel Tinte
Contributors	Alavaro Janda, Giuseppe Filippone, Miguel A. Pasenau de Riera, Tomas Pariente

Approvals

	Name	Institution	Date	OK
Author	Miguel Angel Tinte, Ivan Martínez	**ATOS**	30/11/2015	
Task Leader	Ivan Martínez	**ATOS**	30/11/2015	
WP Leader	Ivan Martínez	**ATOS**	30/11/2015	
Coordinator	Abel Coll	**CIMNE**	30/11/2015	
Quality Check	Heidi E. I. Dahl	**SINTEF**	30/11/2015	

Table of Contents

1 Introduction

1.1 Purpose of the document

The main purpose of this document is to report on the validation of the architectural components of the VELaSSCo framework for simulation purposes and identify the list of corrective actions in case it is needed. To do so, this document follows the guidelines proposed on document D5.1 [1] where Goal Question Metric methodology for evaluation was introduced. The main objective, is applying full GQM cycle over Architecture and End User Functionalities dimensions in order to assess current development status as well as detecting new requirements and improvements. Current evaluation takes three main items as the basis for achieving the process: the cluster where VELaSCCo platform has been deployed, the tools selected to assess and measure the platform developed and two main scenarios or use cases to carry out this evaluation.

The completeness of this report will depend on the status of VELaSCCo modules development at the moment of evaluation, as well as reliability of VELaSCCo platform access and services availability. Therefore, architecture evaluation encompasses several aspects including technical functionality requirements, physical resources consumption, service integration interoperability, etc. All these aspects have been materialized in a list of metrics which have been described within a measurement plan.

Besides this, the document will provide a roadmap for future evaluations, clarifying current assessment process and future processes, which will be reported in future evaluations.

Finally, it is necessary to mention that evaluation focuses on two scenarios based on both simulation types present in the project, DEM and FEM simulation, aiming to provide an overall picture of evaluation status.

1.2 Structure of the document

The document is structured as follows:

Section 1 gives a brief introduction and outlines the major purpose of the document.

Section 2 recaps on the main aspects of the Architecture Evaluation planning describing in details the current status on GQM cycle for End User Functionalities and Architecture Dimensions as well as the Study set-up and methodology.

Section 3 is the core section of the document. It provides a detailed description about the Measurement Plan defined for the dimensions affected for the evaluation process extending the Measurement Plan defined in D5.1 [1] describing how to read and use the metrics listed in GQM tables.

Section 4 reports on Data collected from the execution of the two test scenarios evaluated.

Section 5 provides an interpretation of the Data collected in order to validate the soundness of the architectural components of the VELaSCCo framework for simulation purposes and take corrective actions in case it is needed.

Section 6 concludes with consolidated findings and reports on the next steps.

Section 7 contains the references.

2 Architecture Evaluation Planning

Deliverable D5.1 [1] provided an overview of the VELaSCCo Evaluation Methodology selected for the evaluation of the VELaSCCo framework. This section recaps on the main aspects of GQM and provides a picture of the main aspects to be taken into account to apply correctly GQM for the evaluation process.

2.1 GQM life-cycle for End User Functionalities and Architecture Dimensions

On the one hand, the current status of the progress applying the GQM cycle over the End User functionalities Dimension is the following:

- Identification of GQM goals: Step 1 finished. A set of technical goals related to business functionalities have been identified considering as goal each one of the tasks defined in the use cases definition described in Section 2.2.3.
- Development of a GQM plan and Derive Measurement Plan: Steps 2 and 3 finished. An extended Measurement Plan is provided as part of T5.2 in Section 3 including additional information to the GQM Metrics Table described in D5.1 [1] .
- Data collection and Interpret collected data: Steps 4 and 5 to be covered at the end of the Architecture Evaluation task or T5.2 and reported in the current document.

Figure 1 shows the GQM steps finished (in green) and pending (in orange) related the End User Functionalities Dimension.

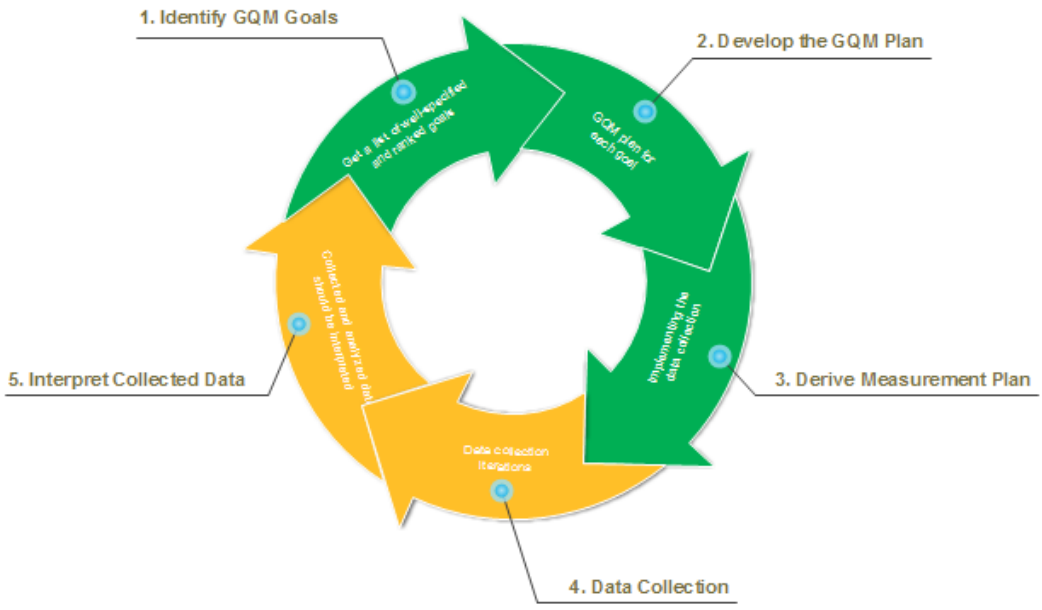

Figure 1. GQM cycle status over End-User Functionalities Dimension

On the other hand, the current status of the progress applying the GQM cycle over the SW Architecture Dimension is the following:

- Identification of GQM goals: Step 1 finished. The standard ISO-9126 was adopted to measure Architecture dimension considering goals as each one of the suggested software quality characteristics described in the standard.
- Development of a GQM plan and Derive Measurement Plan: Steps 2 and 3 finished. An extended Measurement Plan is provided in Section 3 including additional information to the GQM Metrics Table described in D5.1 [1] .
- Data collection and Interpret collected data: Steps 4 and 5 to be covered at the end of the Architecture Evaluation task or T5.2 and reported in the current document.

Figure 2 shows the GQM steps finished (in green) and pending (in orange) related the Architecture Dimension.

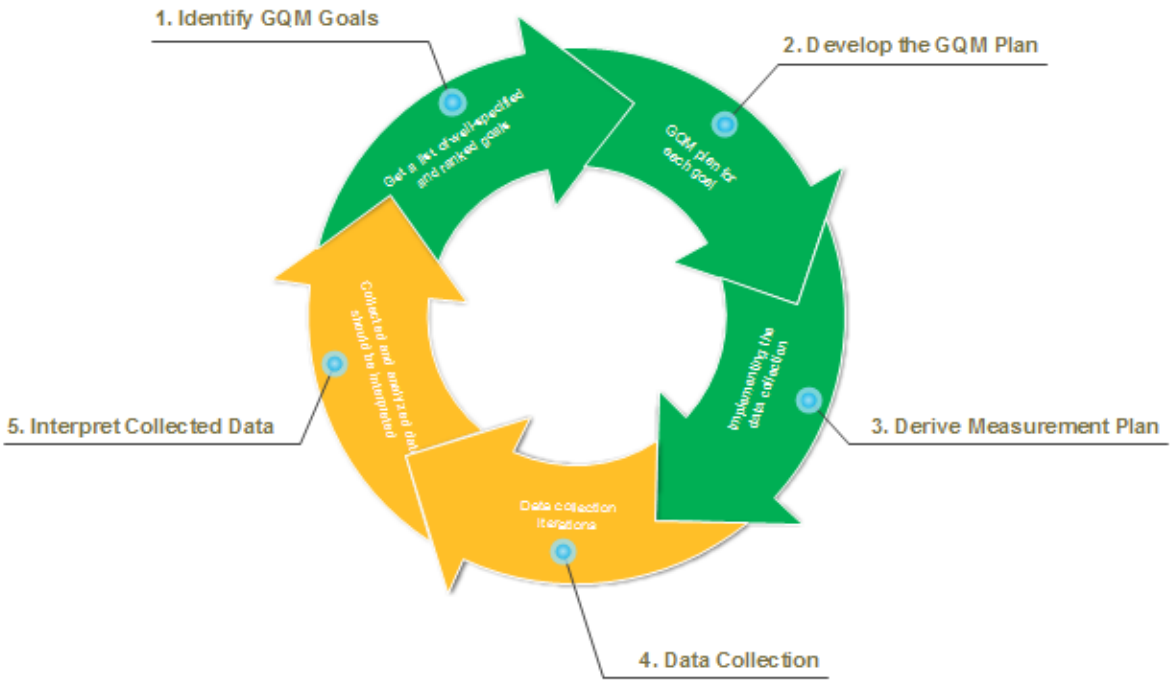

Figure 2. GQM cycle status over SW Architecture Dimension

2.2 Study set-up and methodology

2.2.1 Acuario Cluster

In the view of the deployment of the VELaSCCo platform in UEDIN's EDDIE cluster for the evaluation event, the current implementation is in the Acuario cluster, located at the CIMNE premises, in which 10 nodes are allocated for the VELaSCCo project.

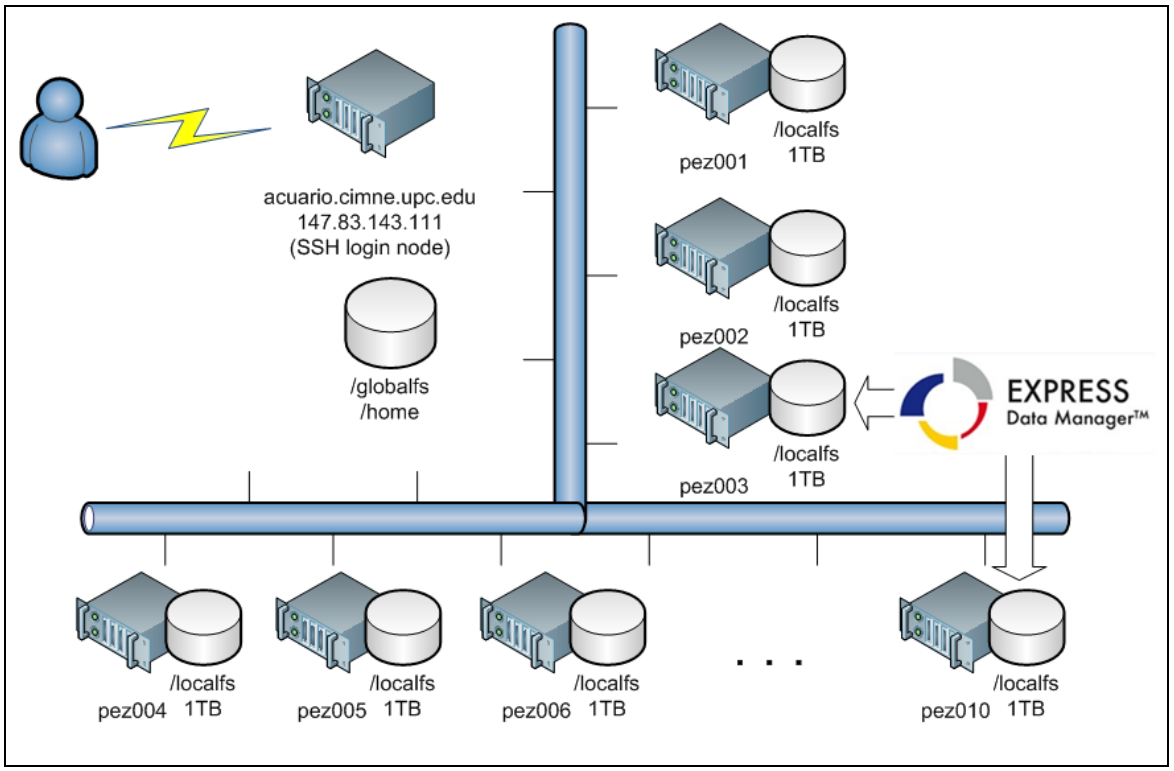

Figure 3. Configuration of the 10 nodes of CIMNE's Acuario cluster devoted to the VELaSSCo project

Figure 3 shows the configuration of the 10 nodes:

- 8 nodes are used to develop the open-source version (architecture) of the platform, using the Hadoop framework, and
- 2 nodes (pez003 and pez010) are used to develop the closed version of the platform, using JOTNE's EDM engine.

All nodes are connected through a 20Gbps InfiniBand network and their configuration is:

- CPU: 2 x Intel Quad Core E5410 @ 2.33 GHz (total of 8 cores)
- RAM: pez001: 32GB, the rest: 16 GB
- Network: DDR x4 InfiniBand (20Gb/s) + 1Gbps Ethernet
- Hard disk: 1 x 150 GB + 1 x 1TB

Deliverables D3.4 [6] , D4.2 [5] , D4.1 [7] and D2.4 [7] explain the two architectures in detail.

The current architectures for the two scenarios used to produce this prototype are depicted in the following figures (Figure 4 and Figure 5), which may have evolved slightly compared to earlier deliverables. Figure 4 depicts the architecture based on open source software, Figure 5 the one with Jotne's DBMS EDM.

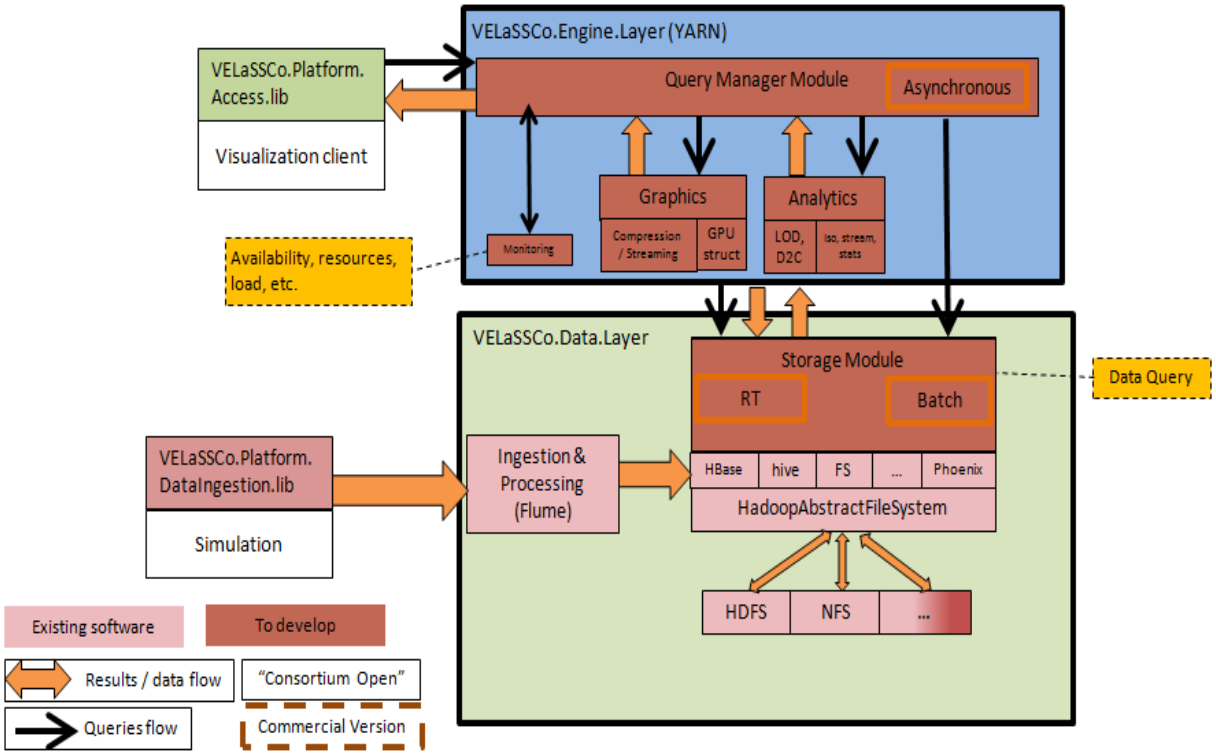

Figure 4: The VELaSSCo architecture used for the open source version of this prototype

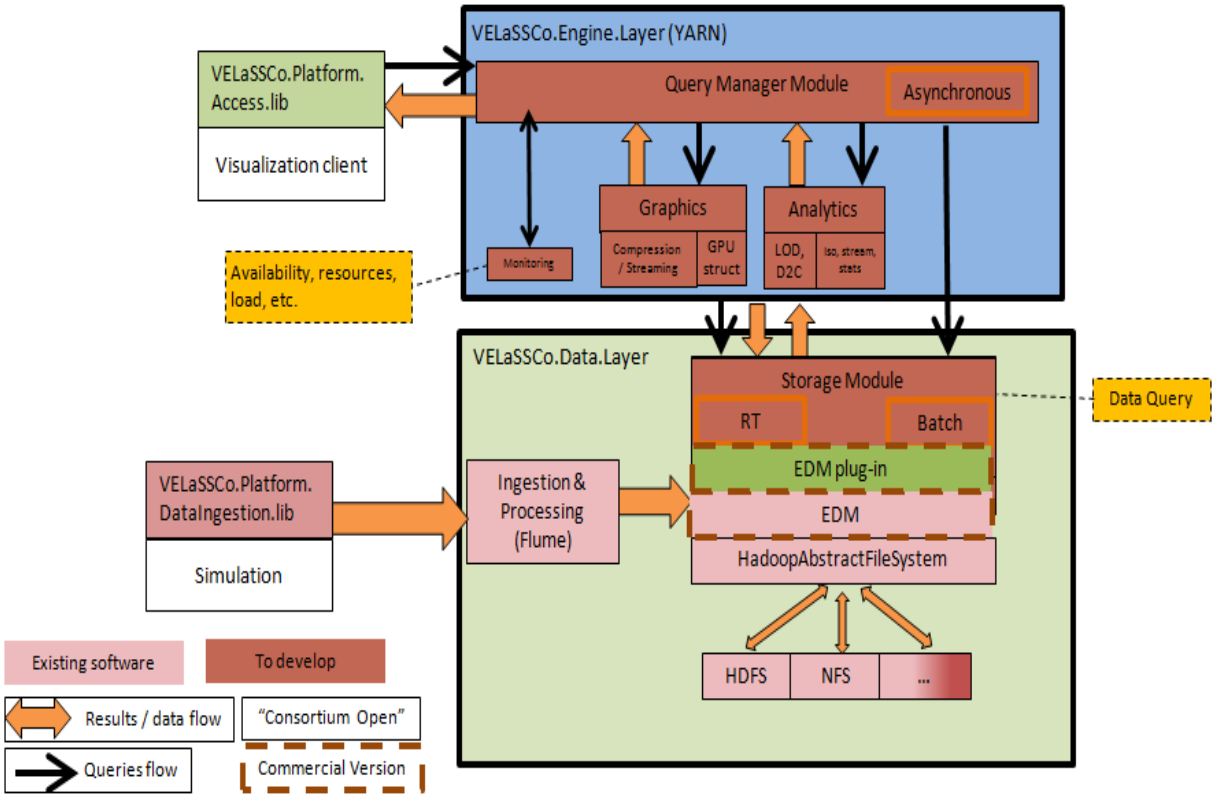

Figure 5: The VELaSSCo architecture used for the closed source version of this prototype

Deliverable D4.2 [5] relates in detail the modules of this architecture to the workflow involved in the simple VQueries.

2.2.2 Testing Tools

2.2.2.1 Nagios

Nagios[1] is an open source software monitor tool which enables monitoring your entire IT infrastructure to ensure systems, applications, services, and business processes are functioning properly. In the event of a failure, Nagios can alert technical staff of the problem, allowing them to begin remediation processes before outages affect business processes, end-users, or customers.

In VELaSCCo project, we will use Nagios to monitor cluster resources in order to detect any failure of servers, network connection, etc. as well to obtain performance metrics valuable for evaluation purposes.

Figure 6 below display Cluster nodes with default services monitored to check connectivity issues, resources consumption, etc.:

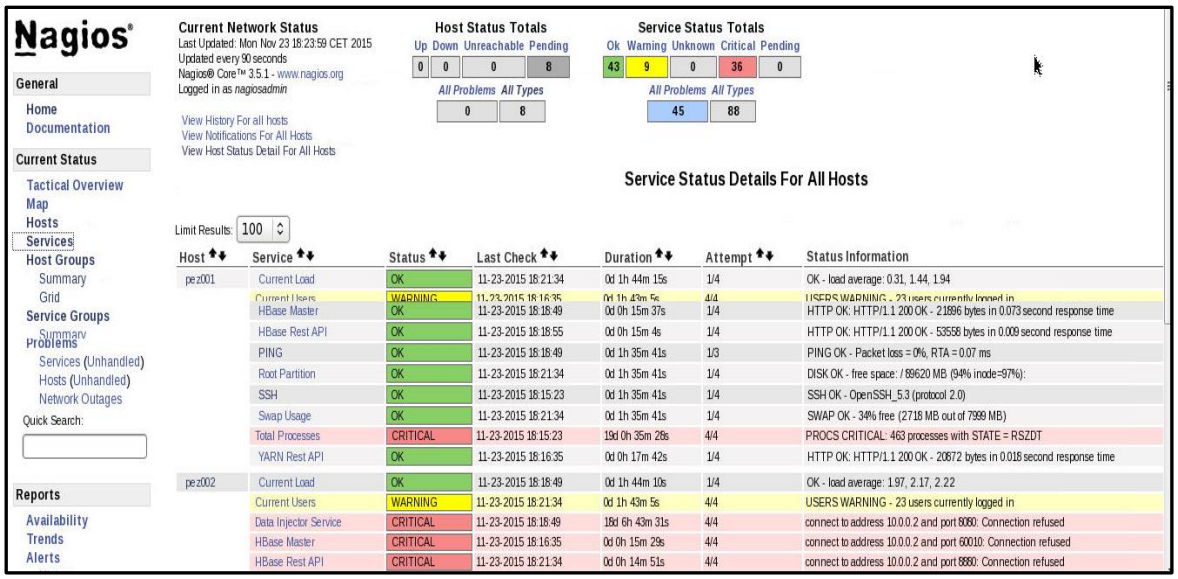

Figure 6. Nagios monitor tool on CIMNE Cluster (pez001-pez009)

Nagios generates a periodic report as well as automatic alerts regarding status of the services, which are displayed via web browser: green colour indicates good results for threshold defined, yellow reflects some warning respect the values expected and red colour notify about some malfunction affecting the service.

[1] https://www.nagios.org/

2.2.2.2 TestComplete

TestComplete[2] Platform helps you create accurate and repeatable automated tests across multiple devices, platforms, and environments easily and quickly. It will be used within VELaSCCo project to reproduce Use Cases steps, in order to measure performance for each necessary step to complete a VQuery (VELaSCCo Query) and to ensure the proper working of simulation data access and visualization. To do so, TestComplete software will record every step perform over VELaSCCo visualization clients (GiD[3] and iFX[4])

Figure 7 shows the way TestComplete works and how it can be used simulate GiD and iFX Uses Cases execution:

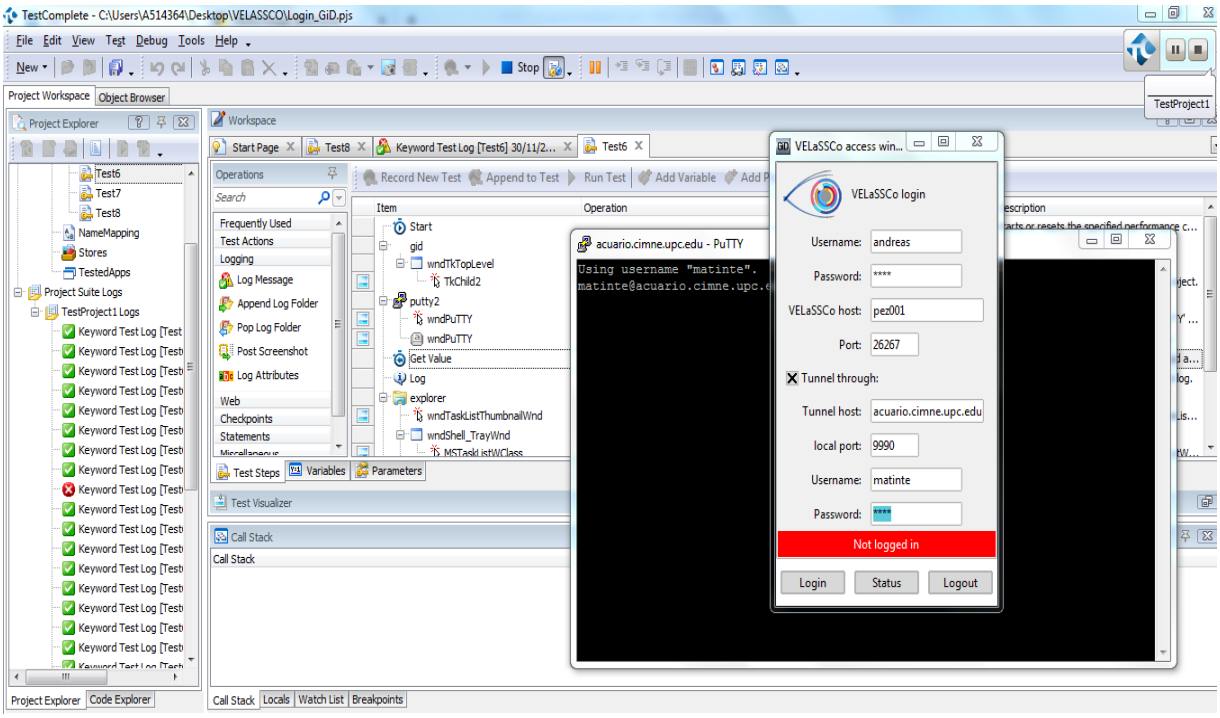

Figure 7. Automatizing Use Cases testing with TestComplete.

2.2.2.3 Query Manager and Access Library Logs

These modules have been developed within VELaSCCo project [4] and they are the intermediary modules between the visualization client and the VELaSCCo platform, being in charge of the external communication. Another task of these modules is to analyse queries sent by users and decompose them into sub queries. These sub queries will interact with the VELaSCCo platform at different levels: analytics, or storage. Besides this, this module is also in charge of applying multi resolution queries in order to provide a dataset in an interactive way.

[2] http://smartbear.com/product/testcomplete/overview/
[3] http://www.gidhome.com/
[4] https://www.igd.fraunhofer.de/en/Institut/Abteilungen/IET/Projekte/iFX-Visualization

2.2.3 Testing Scenarios and Use cases description

Once mentioned the tools to be used for testing, current subsection aims to describe the main scenarios and Use Cases to be evaluated and how to apply previous tools to achieve a fully evaluation description.

We consider as pre-condition for the definition uses cases at task level that the **data have been injected previously**. Given this premise, the evaluation tasks defined for Telescope Use case (FEM) and Fluidized Bed Use case (DEM) are described below:

- *FEM Evaluation tasks (T1…10):*

  - T1: Connect to VELaSSCo
  - T2: Open a simulation model (*model FEM.M1.)*
  - T3: Select coarser mesh.
    - T3.1: *Select coarser mesh for all time steps.*
  - T4: Rotate model.
  - T5: Select original mesh
  - T6: Get the evolution of a result on a node over time.
    - T6.1: *Get the **pressure** value of **node number 5** for **all time steps***
  - T7: Visualize a contour fill of a result.
    - *T7.1: Visualize the contour fill of **pressure** in the skin of the volume mesh in **time step 8***
  - T8: Do a cut in the volume mesh.
    - *T8.1: Do and visualize a cut in the volume mesh, parallel to AA direction, and passing through (x, y, z) coordinates.*
  - T9: Visualize a result onto the cut plane
    - T9.1: Visualize the velocity vectors onto the cut plane in time step 7.
  - T10: Logout.

- *DEM Evaluation tasks (T1…11):*

  - T1: Connect to VELaSCCo
  - T2: Open a simulation model (*model DEM.M1.)*
  - T3: Visualize a contour fill of a particle result.
    - *T3.1: Visualize the **velocity-Y** in the skin of the particles for **time step 2939000***
  - T4: Rotate model.
  - T5: Get the evolution of a result on a particle over time.
    - *T5.1Get the velocity y-component value for:*
      - *Analysis = **DEM***
      - *Coordinates = **Particles***
      - *Time-steps: **ALL***
      - *Result = **Velocity-Y***
      - *Node number **2724***
  - T6: Visualize p2p contacts.
    - *T6.1: Visualize the **p2p contacts** mesh for **time step 2939000***
  - T7: Visualize a contour fill of a p2p result.
    - *T7.1: Visualize the **Force-Y** in the skin of the p2p contacts for **time step 2939000***
  - T8: Compute d2c of the model
    - *T8.1: Compute discrete to continuum for:*
      - *Static mesh = d2c_1*
      - *D2C analysis name = d2C_FB2*
      - *Time-step options = ALL*
      - *Coarse-graining method = Gaussian*
      - *Coarse-graining options:*
      - *Width = 0.003*
      - *Cut-off factor = 3*
      - *Process contacts = True*
      - *Do temporal averaging = True*
      - *Temporal averaging options = ALL*
  - T9: Do a cut plane in the d2c mesh.
    - *T9.1: Do and visualize a cut in the d2c mesh, parallel to Y direction, and passing through (0, 0, 0) coordinates.*
  - T10: Visualize a result of d2c onto the cut plane
    - T10.1: Visualize the **Velocity-Y** onto the cut plane for computed d2c in time step ***0***
  - T11: Logout.

3 Measurement Plan

Taking as input the GQM Metrics for End User Functionalities and Architecture Dimension defined in D5.1 [1] , we provide an extension of GQM Metrics table that we named as “GQM Metrics Extended Table”.

As a prelude to the specification of the extended tables for the two dimensions considered we provide a description about how to read and use the metrics that appear in the GQM Metrics Extended Table. The following information is given for each metric in the table:

- **Metric**: Code that represent a concrete metric for each one of the Dimension. The code is composed by M.XY#N where M means metrics, XY is the codification of the dimension (EU = End User, AR= Architecture, AL = Algorithms, NI = Navigation and Interaction and VI = Views) and finally N is the number of the metric.
- **Description**: Briefly description of the Metric.
- **Purpose of the metric**: This is expressed as the question to be answered by the application of the metric.
- **Measurement, formula and data element computations**: Provides the measurement formula and explains the meanings of the used data elements.
- **Interpretation of measured value**: Provides the range and preferred values.
- **Metric scale type**: Type of scale used by the metric. Scale types used are; Nominal scale, Ordinal scale, Interval scale, Ratio scale and Absolute scale.
- **Measure type**: Types used are; Size type (e.g. Function size, Source size) , Time type (e.g. Elapsed time, User time) , Count type (e.g. Number of changes, Number of failures), Credentials (e.g. User/Password, Public/Private key), Velocity (e.g. Rotation model velocity)
- **Tool**: Software tool or mechanism to calculate results.
- **Technique**: method to be applied in order to obtain empirical results for metrics.

3.1 End User Functionality Extended Table

From the conceptual point of view the metrics included in Table 1, we can group the metrics into the following groups:

- **Injection Metrics**: metrics related with injection of simulation data into VELaSCCo Big Data Platform. The metrics codes listed in Table 1 are: M.EU#1, M.EU#2, M.EU#3 and M.EU#4.
- **Simulation Configuration Metrics**: metrics on what characterizes a simulation, such as the simulation file size, the number of particles, the number of time

steps or the number of results at particle level. The metrics codes listed in Table 1 are: M.EU#5, M.EU#6, M.EU#7, M.EU#8, M.EU#9 and M.EU#10.

- **Security Metrics**: related to secured access, log in or log out into the Platform. The metrics codes grouped in are: M.EU#11, M.EU#12 and M.EU#22.
- **Performance Metrics**: metrics focus on the query performance of the VELasCCo queries involved in the first prototype of the Platform and listed in Table 1 with the following codes: M.EU#13, M.EU#14, M.EU#15, M.EU#16, M.EU#17, M.EU#18, M.EU#19, M.EU#20 and M.EU#21.

The complete Measurement Plan defined for the End User Dimension is shown inTable 1:

Metric	Description	Purpose	Formula	Interpretation	Metric Scale Type	Measure Type	Tool	Technique
M.EU#1	Number of files which the simulation is composed by.	Measure the number of partitions associated to the simulation data files.	Non-applicable	Non-applicable	Absolute	Count	Linux Command: ls | wc –l	Scan the input simulation data dir in "Acuario Cluster".
M.EU#2	Number of Data Events generated.	Measure the number of data events generated during the injection process associated to a concrete simulation.	Non-applicable	Initially in terms of generated data events we consider: **Small** = hundred, **Medium** size = thousand and **Large** = millions.	Absolute	Count	curl localhost:*agent_monitoring_port*/metrics	Run Data Injector using data simulation files.
M.EU#3	Number of Hbase nodes	Measure the distributed degree of HBase.	Non-applicable	*Suggested as minimum* (for a production deployment): • 1x HDFS NameNode • 1x JobTracker / Secondary NameNode • 3x ZK Nodes • 3x DataNode / RegionServer nodes (And if you want to run MapReduce, TaskTracker) • 1x Thrift Server (Only if accessing HBase from outside of the network it is	Absolute	Count	Physical Cluster	Execute HBase Monitoring service.

				running on)				
M.EU#4	Injection Time	Measure the data injection performance.	**TDI= CT - TDIS**, where CT = Current Time and TDIS = Data Injection Start Time	Performance Injection Time considered will be the following: • Bad: <1000 w/s in Hbase • Acceptable :>1000 w/s and < 10000 w/ sec in Hbase • Good :> 10000 w/sec in Hbase where w/s means writes/sec.	Absolute	Time	Nohup and curl.	Run Data Injector using data simulation files with nohup and search into nohup.out for the execution time.
M.EU#5	Simulation File Size	Measure the size of the simulation.	Non-applicable	Initially, in terms of simulation size we consider: **Small** <1GB, **Medium** from 1GB to 20GB and **Large** > 20GB).	Absolute	Size	Linux Command: du -bsh /file or folder	Scan the input simulation data dir in “Acuario Cluster”.
M.EU#6	Number of particles (p3p)	Measure the size of the particles involved in the simulation.	Non-applicable	Initially in terms of particles we consider: **Small** = hundred, **Medium** size = thousand and **Large** = millions.	Absolute	Count	Hbase shell command: get ‘Simulations_Data', “row_key", {FILTER => "(ColumnPrefixFilter('c000001_')"}	Filter Query to Hbase table named “Simulations_Data”
M.EU#7	Number of contacts (p3c and p3w)	Measure the size of the contacts at particle level involved in the simulation.	Non-applicable	Initially in terms of particles we consider: **Small** = hundred, **Medium** size = thousand and **Large** = millions.	Absolute	Count	Hbase shell command: get ‘Simulations_Data', “row_key", {FILTER => "(ColumnPrefixFilter('c000002_')"}	Filter Query to Hbase table named “Simulations_Data”
M.EU#8	Number of computational time steps	Measure the number of time steps that simulation is	Non-applicable	Initially in terms of timesteps we consider: **Small** <1000, **Medium** from 1000 to 15000 and **Large** > 15000.	Absolute	Count	Hbase shell command: count 'Test_Simulations_Data', {FILTER => "PrefixFilter('row_key_witho	Filter Query to Hbase table named “Simulations_Data”

		composed by.					ut_timesteps	
M.EU#9	Number of results at particle level	Measure the number of results at particle level.	Non-applicable	Initially in terms of number of results we consider: **Small** <3, **Medium** from 3 to 10 and **Large** > 10.	Absolute	Count	Hbase shell command: count 'Test_Simulations_Data', {FILTER =>"(ColumnPrefixFilter('c000001_')"}	Filter Query to Hbase table named "Simulations_Data"
M.EU#10	Number of results at contact level	Measure the number of results at contact level.	Non-applicable	Initially in terms of number of results we consider: **Small** <3, **Medium** from 3 to 10 and **Large** > 10.	Absolute	Count	Hbase shell command: count 'Test_Simulations_Data', {FILTER => "(ColumnPrefixFilter('c000002_')"}	Filter Query to Hbase table named "Simulations_Data"
M.EU#11	User Credentials	To know what User is connected to the Platform.	Non-applicable	User and password associated to a concrete VELaSCCo Platform User.	Absolute	Credentials	VELaSSCo Excell Log file.	Search for User Name and Security Token in Access Library associated log file.
M.EU#12	Security Token	To provide a secured access to the Platform to a concrete User.	Non-applicable	Security Token associated to a concrete User Session into the Platform.	Absolute	Credentials	VELaSSCo Excell Log file.	Search for User Name and Security Token in Access Library associated log file.
M.EU#13	Time of opening model query execution	Measure the performance of opening a model.	**TOM = CT - TOMS**, where CT = Current Time and TOMS = Opening Model Start Time	Performance Time Scale considered will be the following: • Bad: >10 sec • Acceptable :> 5 and < 10 sec • Good :< 5 sec	Absolute	Time	Linux Command: grep "TOM=.* sec" log_file	Search for trace **"TOM = Value sec"** into Query Manager Log File. Should be user independent.
M.EU#14	Time of getting simplified	Measure the performance of getting a	**TSM = CT - TSMS**, where TSMS = Simulation Mesh	Performance Time Scale considered will be the following:	Absolute	Time	Linux Command: grep "TSM=.* sec" log_file	Search for trace **"TSM = Value sec"** into Query Manager

	mesh query execution	simplified mesh.	Start Time and Ct = Current Time	• Bad: >30 sec • Acceptable :> 10 and < 30 sec • Good : < 10 sec				Log File. Should be user independent.
~~M.EU#15~~	GiD Model Rotation Velocity	Measure the Model Rotation Velocity on GiD.	**VRM(GiD) = NRA / RT**, where NRA = Number of Rotated Angles and RT = Rotation time in sec.	Rotation Velocity Scale considered will be the following: Bad:--, Acceptable :--, Good :--	Absolute	Velocity	Linux Command: grep "VRM=.* sec" local_user_log_file	Search for trace **"VRM = Value sec"** into GiD Local Log File. Should be user dependent.
~~M.EU#16~~	IFX Model Rotation Velocity	Measure the Model Rotation Velocity on IFX	VRM(IFX) = NRA / RT, where NRA = Number of Rotated Angles and RT = Rotation time in sec.	Rotation Velocity Scale considered will be the following: Bad:--, Acceptable :--, Good :--	Absolute	Velocity	Linux Command: grep "VRM=.* sec" local_user_log_file	Search for trace **"VRM = Value sec"** into IFX Local Log File. Should be user dependent.
M.EU#17	Time of getting original mesh query execution	Measure the performance of getting the original mesh.	**TORM = CT - TORMS**, where and CT = Current Time and TORMS = Original Mesh Data Start Time.	Performance Time Scale considered will be the following, considering **n** the number of elements of the original mesh: • Bad: TT > (n / 5.000) sec, QET > 10s. • Acceptable : TT between (n / 5.000) and (num of elements / 10.000) sec, 3 s. < QET < 10 s. • Good : TT < (n / 10.000) sec, QET < 3 s. There are 2 timings: QET (Query Execution Time) + TT	Absolute	Time	Linux Command: grep "TORM=.* sec" log_file	Search for trace **"TORM = Value sec"** into Query Manager Log File. Should be user independent.

				(Transfer Time): to get the first one execute the query with a small mesh (i.e. < 1000 elements), then to get the second, use a big mesh (>1,000,000 or bigger).				
M.EU#18	Time of getting result on a vertex over time	Measure the performance of getting result on a node over time.	**TRON = CT - TRONS**, where and CT = Current Time and TRONS = Result on a Node Start Time.	Performance Time Scale considered will be the following: • Bad: QET > 10 sec. , TT < 100Kresults/s • Acceptable: 3s. < QET < 10s. 100Kres/s < TT < 500Kres/s • Good: QET < 3s. TT > 500Kres/s AS before there are 2 timings: QET + TT In this case TT depends also in the amount of data to be transferred (number of time-steps per vertex?)	Absolute	Time	Linux Command: grep "TRON=.* sec" log_file	Search for trace **"TRON = Value sec"** into Query Manager Log File. Should be user independent.
M.EU#19	Time of getting the contour fill for a concrete result	Measure the performance of getting the contour fill for a result.	**TCFR = CT - TCFRS**, where CT = Current Time and TCFRS = Contour Fill for a Result Start Time.	Performance Time Scale considered will be the following, considering **n** the number of elements (triangles) onto which the contour fill is drawn: • Bad: >n/5.000 • Acceptable : between n /10.000 and n/5.000	Absolute	Time	Linux Command: grep "TCFR=.* sec" log_file	Search for trace **"TCFR = Value sec"** into Query Manager Log File. Should be user independent.

				• Good : < n/10.000				
M.EU#20	Time of getting a cut in a volume mesh	Measure the performance of getting a cut in a volume mesh.	**TCVM = CT - TCVMS**, where CT = Current Time and TCFRS = Cut Volume Mesh Start Time.	Performance Time Scale considered will be the following, considering **n** the number of elements in the volume mesh: • Bad: >n/5.000 • Acceptable : between n /10.000 and n/5.000 • Good : < n/10.000	Absolute	Time	Linux Command: grep "TCVM=.* sec" log_file	Search for trace **"TCVM = Value sec"** into Query Manager Log File. Should be user independent.
M.EU#21	Time of getting a cut in a volume mesh with results	Measure the performance of getting a cut in a volume mesh with results.	**TCVMR = CT - TCVMRS**, where CT = Current Time and TCVMRS = Cut Volume Mesh with Results Start Time.	Performance Time Scale considered will be the following: • Bad: same as M.EU#14 • Acceptable : same as M.EU#14 • Good : same as M.EU#14	Absolute	Time	Linux Command: grep "TCVMR=.* sec" log_file	Search for trace **"TCVMR = Value sec"** into Query Manager Log File. Should be user independent.
M.EU#22	User session logout trace	Validate that the User is log out into the Platform.	Non-applicable	Log out trace for a concrete User Session into the Platform.	Absolute	Credentials	VELaSCCo Excell Log file.	Search for User Name and Security Token in Access Library associated log file.

Table 1. GQM Metrics Extended Table for EU Dimension

The metrics M.EU#15 and M.EU#16 will not be evaluated because they depend solely on the implementation of visualization clients GiD and IFX, but not VELaSCCo architecture platform.

3.2 Architecture Extended Table

The GQM process for architecture dimension is based on ISO 9126 [9] . From the conceptual point of view the metrics included in Table 2, we can group the metrics into the following groups:

- **Reliability Metrics** [9] : "*related to the behaviors of the system of which the software is a part during execution testing to indicate the extent of reliability of the software in that system during operation. Systems and software are not distinguished from each other in most cases*". The reliability metrics are divided into:
    a. **Maturity Metrics** [9] : "*related to attributes as the software freedom of failures caused by faults existing in the software itself*". The metrics codes grouped in are: MAR#3, MAR#4, MAR#5, MAR#6, MAR#7 and MAR#8.
    b. **Fault Tolerance Metrics** [9] : "*related to the software capability of maintaining a specified performance level in cases of operation faults or infringement of its specified interface*". The metrics codes grouped in are: MAR#11.
    c. **Recoverability Metrics** [9] : "*to measure such attributes as the software with system being able to re-establish its adequate level of performance and recover the data directly affected in the case of a failure*". The metrics codes grouped in are: MAR#16, MAR#17, MAR#18, MAR#19, MAR#20 and MAR#21.
- **Efficiency Metrics** [9] : "*to measure such attributes as the time consumption and resource utilization behavior of computer system including software during testing or operations*".
    a. **Time Behavior Metrics** [9] : "*to measure such attributes as the time behavior of computer system including software during testing or operations*". The metrics codes grouped in are: MAR#22, MAR#23, MAR#24, MAR#25, MAR#26 and MAR#27.
    b. **Resource Utilization Metrics** [9] : "*to measure such attributes as the utilized resources behaviour of computer system including software during testing or operating*". The metrics codes grouped in are: MAR#28, MAR#29, MAR#30, MAR#32, MAR#33, MAR#34, MAR#35, MAR#36, MAR#37 and MAR#38.

- **Maintainability Metrics** [9] : "*to measure such attributes as the behaviour of the maintainer, user, or system including the software, when the software is maintained or modified during testing or maintenance"*.

    a. **Analyzability Metrics** [9] : "*to measure such attributes as the maintainer's or user's effort or spent of resources when trying to diagnose deficiencies or causes of failures, or for identifying parts to be modified*". The metrics codes grouped in are: MAR#39, MAR#40, MAR#41, MAR#42, MAR#43 and MAR#44.
    b. **Testability Metrics** [9] : "*to measure such attributes as the maintainer's or user's effort or spent of resources when trying to diagnose deficiencies or causes of failures, or for identifying parts to be modified*". The metrics codes grouped in are: MAR#51 and MAR#52.

- **Portability Metrics** [9] : "*to measure such attributes as the behaviour of the operator or system during the porting activity*".

    a. **Adaptability Metrics** [9] : "*to measure such attributes as the behaviour of the system or the user who is trying to adapt software to different specified environments*". The metrics codes grouped in are: MAR#53, MAR#54 and MAR#55.
    b. **Installability Metrics** [9] : "*to measure such attributes as the behaviour of the system or the user who is trying to install the software in a user specific environment*". The metrics codes grouped in are: MAR#56 and MAR#57.
    c. **Co-existence Metrics** [9] : "*to measure such attributes as the behaviour of the system or the user who is trying to use the software with other independent software in a common environment sharing common resources"*. The metrics codes grouped in are: MAR#58.
    d. **Replaceability Metrics** [9] : "*to measure such attributes as the behaviour of the system or the user who is trying to use the software in place of other specified software in the environment of that software*". The metrics codes grouped in are: MAR#59, MAR#60 and MAR#61.

Metric	Description	Purpose	Formula	Interpretation	Metric Scale Type	Measure Type	Tool	Technique
M.AR#3	Estimated latent fault density	How many problems still exist that may emerge as future faults?	**X= {ABS(A1 - A2)} / B** where, A1 = total number of predicted latent faults in Platform, A2 = total number of actually detected faults and B= Platform size	0<=X It depends on stage of testing. At the later stages, smaller is better.	Absolute	Count/Size	All SW Module Log files defined in VELaSCCo Platform. Architecture Diagram (B= Number of SW Modules)	Count the number of faults detected during a defined trial period and predicts potential number of future faults using a reliability growth estimation model.
M.AR#4	Failure density against test cases	How many failures were detected during defined trial period?	**X= A1 / A2**, where A1 = number of detected failures, and A2 = number of performed test cases.	0<=X It depends on stage of testing. At the later stages, smaller is better.	Absolute	Count/Size	All SW Module Log files defined in VELaSCCo Platform.	Count the number of detected failures and performed test cases
~~**M.AR#5**~~	Failure resolution	How many failure conditions are resolved?	**X= A1 / A2**, where A1 = number of resolved failures, and A2 = total number of actually detected failures.	0<=X<= 1 The closer to 1.0 is better as more failures are resolved.	Absolute	Count	Bug Tracking tool is needed.	Count the number of failures that did not reoccur during defined trial period under the similar conditions.
M.AR#6	Fault density	How many faults were detected during defined trial period?	**X= A / B**, where A = number of detected faults, and B = VELaSCCo Platform Size.	0<=X It depends on stage of testing. At the later stages, smaller is better.	Absolute	Count/Size	All SW Module Log files defined in VELaSCCo Platform.	Count the number of detected faults and compute density.
~~**M.AR#7**~~	Fault removal	How many faults have been corrected?	**a) X= A1 / A2**, where A1 = number of corrected faults,	0<=X<= 1. The closer to 1.0 is better as fewer faults remain.	Absolute	Count	Bug Tracking tool is needed.	Count the number of faults removed during testing and compare

			and A2 = total number of actually detected faults **b) Y= A1 / A3**, where A3 = total number of predicted latent faults in the software release.	0<=Y. The closer to 1.0 is better as fewer faults remain				with the total number of faults detected and total number of faults predicted.
M.AR#8	Mean time between failures	How frequently does the software fail in operation?	**a) X = T1 / A** **b) Y = T2 / A** T1 = operation time T2 = sum of time intervals between consecutive failure occurrences A = total number of actually detected failures (Failures occurred during observed operation time)	0<X, Y. The longer is the better. As longer time can be expected between failures.	Ratio	Time/Count	Nagios, and SW Module Log files defined in VELaSCCo Platform.	Count the number of failures occurred during a defined period of operation and computes the average interval between the failures.
M.AR#11	Breakdown avoidance	How often the software causes the breakdown of the total deployment environment?	**X= 1- A / B**, where A= Number of breakdowns, and B= Number of failures	0<=X<= 1. The closer to 1.0 is the better.	Absolute	Count	Nagios and SW Module Log files defined in VELaSCCo Platform.	Count the number of breakdowns occurrence with respect to number of failures.
M.AR#16	Availability	How available is the system for use during the specified period of time?	**a) X= { To / (To + Tr) }** **b) Y= A1 / A2** To = operation time Tr = time to repair	0<=X<=1. The larger and closer to 1.0 is better, as the user can use the software for more time. 0<=Y<=1. The larger and	Absolute	X is Time, and Y is Count	Nagios	Test system in a production like environment for a specified period of time performing all user operations.

			A1= total available cases of user's successful software use when user attempt to use A2= total number of cases of user's attempt to use the software during observation time. This is from the user callable function operation view.	closer to 1.0 is the better.				Measure the repair time period each time the system was unavailable during the trial. Compute mean time to repair.
M.AR#17	Mean down time	What is the average time the system stays unavailable when a failure occurs before gradual start up?	**X= T / N** T= Total down time N= Number of observed breakdowns The worst case or distribution of down time should be measured.	0<X. The smaller is the better; system will be down for shorter time.	Ratio	Time/Count	Nagios	Measure the down time each time the system is unavailable during a specified trial period and compute the mean time
M.AR#18	Mean recovery time	What is the average time the system takes to complete recovery from initial partial recovery?	**X= Sum(T) / B** T= Time to recovery downed software system at each opportunity N= Number of cases which observed software system entered into recovery.	0<X. The smaller is the better.	Ratio	Time/Count	Nagios	Measure the full recovery times for each of the time the system was brought down during the specified trial period and computes the mean time.

M.AR#19	Restartability	How often the system can restart providing service to users within a required time?	**X = A / B** A= Number of restarts which met to required time during testing or user operation support B= Total number of restarts during testing or user operation support	0<=X<=1. The larger and closer to 1.0 is better, as the user can restart easily.	Absolute	Count	Nagios	Count the number of times the system restarts and provides service to users within a target required time and compare it to the total number of restarts, when the system was brought down during the specified trial period
M.AR#20	Restorability	How capable is the software in restoring itself after abnormal event or at request?	**X= A / B** A= Number of restoration cases successfully done B= Number of restoration cases tested as per requirements.	0<=X<=1. The larger and closer to 1.0 is better, as he product is more capable to restore in defined cases.	Absolute	Count	Nagios	Count the number of successful restorations and compare it to the number of tested restoration required in the specifications.
M.AR#21	Restore effectiveness	How effective is the restoration capability?	**X= A / B** A= Number of cases successfully restored meeting the target restore time B= Number of cases performed	0<=X<=1. The larger and closer to 1.0 is the better, as the restoration process in product is more effective.	Absolute	Count	Nagios	Count the number of tested restoration meeting target restoration time and compare it to the number of restorations required with specified target time.
M.AR#22	Response Time	What is the time taken to complete a specified VQuery?	**RT(VQ) = CT - VQST** where CT = Current Time and VQST= VQuery Start Time.	RT(VQ) > 0	Absolute	Time	Linux Command: grep “VQ.* sec” log_file	Run a specified VQuery. Measure the time it takes to complete the operation. Keep a record of each attempt in Query

								Manager Log File using the format **"VQXXX = Value sec".** VQueries considered are: **VQ002, VQ010, VQ012, VQ217, VQ114, VQ100, VQ214, VQ215, VQ216**.
M.AR#23	Mean Time to response	What is the average wait time the user experiences after issuing a request until the request is completed within a specified system load in terms of concurrent tasks and system utilization?	**X = Tmean / TXmean**, where Tmean = ∑(Ti) / N, (for i=1 to N) TXmean = required mean response time, Ti= response time for i-th evaluation (shot), and N= number of evaluations (sampled shots)	0 <= X. The nearer to 1.0 and less than 1.0 is the better.	Absolute	Time	Linux Command: grep "VQ.* sec" log_file	Execute a number of scenarios of concurrent VQueries. Measure the time it takes to complete the selected VQuery. Keep a record of each attempt in Query Manager Log File using the format **"VQXXX = Value sec"** and compute the mean time for each scenario.
M.AR#24	Worst case response time	In the worst case, can user still get reply from the software within a time short enough to be tolerable for user?	**X= Tmax / Rmax**, where Tmax= MAX(Ti) (for i=1 to N), Rmax = required maximum response time, MAX(Ti)= maximum response time among evaluations, N= number of evaluations (sampled	0 < X. The nearer to 1 and less than 1 is the better.	Absolute	Time	Linux Command: grep "VQ.* sec" log_file	Emulate a condition whereby the system reaches a maximum load situation. Run application and keep a record of each attempt in Query Manager Log File using the format **"VQXXX = Value sec"**

			shots), and Ti= response time for i-th evaluation (shot)					
M.AR#25	Throughput	How many VQueries can be successfully performed over a given period of time?	**X = A / T** A = number of completed tasks T = observation time period	0 < X. The larger is the better.	Ratio	Count/Time	Linux Command: grep "VQ.* sec" log_file	Start several job VQueries. Measure the time it takes for the measured task to complete its operation. Keep a record of each attempt in Query Manager Log File using the format **"VQXXX = Value sec".**
M.AR#26	Mean amount of Throughput	What is the average number of concurrent VQueries the system can handle over a set unit of time?	**X = Xmean / Rmean** Xmean = ∑(Xi)/N Rmean = required mean throughput Xi = Ai / Ti Ai = number of concurrent tasks observed over set period of time for i-th evaluation Ti = set period of time for i-th evaluation N = number of evaluations	0 < X. The larger is the better.	Absolute	Count	Linux Command: grep "VQ.* sec" log_file	Execute a number of concurrent Vqueries. Measure the time it takes to complete the selected VQuery in the given traffic. Keep a record of each attempt in Query Manager Log File using the format **"VQXXX = Value sec".**
M.AR#27	Worst case throughput ratio	What is the absolute limit on the system in terms of the number and handling of concurrent VQueries as throughput?	**X = Xmax / Rmax** Xmax = MAX(Xi) (for i = 1 to N) Rmax = required maximum	0 < X. The larger is the better.	Absolute	Count	Linux Command: grep "VQ.* sec" log_file	Emulate the condition whereby the system reaches a situation of maximum load. Run job VQueries concurrently and

			throughput. MAX(Xi) = maximum number of job tasks among evaluations Xi = Ai / Ti Ai = number of concurrent tasks observed over set period of time for i-th evaluation Ti = set period of time for i-th evaluation N= number of evaluations.					monitor result(s).
M.AR#28	I/O devices utilization	Is the I/O device utilization too high, causing inefficiencies?	**X = A / B** A = time of I/O devices occupied B = specified time which is designed to occupy I/O devices	0 <= X <= 1. The less than and nearer to the 1.0 is the better.	Absolute	Time	Nagios	Execute concurrently a large number of VQueries, record I/O device utilization, and compare with the design objectives.
M.AR#29	I/O loading limits	What is the absolute limit on I/O utilization in fulfilling a function?	**X = Amax / Rmax** Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum I/O messages MAX(Ai) = Maximum number of I/O messages from 1st to i-th evaluation. N= number of	0<= X. The smaller is the better.	Absolute	Count	Nagios	Emulate a condition whereby the system reaches a situation of maximum load. Run application and monitor result(s).

			evaluations.					
M.AR#30	I/O related errors	How often does the user encounter problems in I/O device related operations?	**X = A / T** A = number of warning messages or system failures T = User operating time during user observation	0 <= X. The smaller is the better.	Ratio	Count/Time	Nagios	Emulate a condition whereby the system reaches a situation of maximum I/O load. Run the application and record number of errors due to I/O failure and warnings.
M.AR#32	Maximum memory utilization	What is the absolute limit on memory required in fulfilling a function?	**X = Amax / Rmax** Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum memory related error messages MAX(Ai) = Maximum number of memory related error messages from 1st to i-th evaluation N= number of evaluations	0<= X. The smaller is the better	Absolute	Count	Nagios	Emulate a condition whereby the system reaches a situation of maximum load. Run application and monitor result(s)
M.AR#33	Mean occurrence of memory error	What is the average number of memory related error messages and failures over a specified length of time and a specified load on the system?	**X = Amean / Rmean** Amean = $\sum$(Ai)/N Rmean = required mean number of memory related error messages Ai = number of	0<= X. The smaller is the better	Absolute	Count	Nagios	Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to memory failure and warnings.

			memory related error messages for i-th evaluation N = number of evaluations					
M.AR#34	Ratio of memory error/time	How many memory errors were experienced over a set period of time and specified resource utilization?	**X = A / T** A = number of warning messages or system failures T = User operating time during user observation	0<= X. The smaller is the better	Ratio	Count/Time	Nagios	Emulate a condition whereby the system reaches a situation of maximum load. Run the application and record number of errors due to memory failure and warnings.
M.AR#35	Maximum transmission utilization	What is the absolute limit of transmissions required to fulfil a function?	**X = Amax / Rmax** Amax = MAX(Ai), (for i = 1 to N) Rmax = required maximum number of transmission related error messages and failures MAX(Ai) = Maximum number of transmission related error messages and failures from 1st to i-th evaluation. N= number of evaluations	0<= X. The smaller is the better	Absolute	Count	Nagios	Evaluate what is required for the system to reach a situation of maximum load. Emulate this condition. Run application and monitor result(s).
M.AR#36	Mean occurrence of transmission	What is the average number of transmission related	**X = Amean / Rmean** Amean = $\sum$(Ai)/N	0<= X. The smaller is the better	Absolute	Count	Nagios	Emulate a condition whereby the system reaches a situation of

	error	error messages and failures over a specified length of time and specified utilization?	Rmean = required mean number of transmission related error messages and failures Ai = Number of transmission related error messages and failures for i-th evaluation N = number of evaluations					maximum load. Run the application and record number of errors due to transmission failure and warnings.
M.AR#37	Mean of transmission error per time	How many transmissions -related error messages were experienced over a set period of time and specified resource utilization?	**X = A / T** A = number of warning messages or system failures T = User operating time during user observation	0<= X. The smaller is the better	Ratio	Count/Time	Nagios	Emulate a condition whereby the system reaches a situation of maximum transmission load. Run the application and record number of errors due to transmission failure and warnings.
M.AR#38	Transmission capacity utilization	Is software system capable of performing tasks within expected transmission capacity?	**X = A / B** A = transmission capacity B = specified transmission capacity which is designed to be used by the software during execution	0 <= X <= 1. The less than and nearer to the 1.0 is the better.	Absolute	Size	Nagios	Execute concurrently specified tasks with multiple users, observe transmission capacity and compare specified one.
M.AR#39	Audit trail capability	Can user identify specific operation	**X= A / B** A= Number of data	0<=X. The closer to 1.0 is the better.	Absolute	Count	Nagios and SW Module Log files	Observe behaviour of user or maintainer who

		which caused failure? Can maintainer easily find specific operation which caused failure?	actually recorded during operation B= Number of data planned to be recorded enough to monitor status of software during operation.				defined in VELaSCCo Platform.	is trying to resolve failures.
M.AR#40	Diagnostic function support	How capable are the diagnostic functions in supporting causal analysis? Can user identify the specific operation which caused failure?	**X= A / B** A= Number of failures which maintainer can diagnose (using the diagnostics function) to understand the cause-effect relationship B= Total number of registered failures	0<=X<= 1. The closer to 1.0 is the better.	Absolute	Count	Nagios and SW Module Log files defined in VELaSCCo Platform.	Observe behaviour of user or maintainer who is trying to resolve failures using diagnostics functions.
M.AR#41	Failure analysis capability	Can user identify specific operation which caused failure? Can maintainer easily find cause of failure?	**X=1- A / B** A= Number of failures of which causes are still not found B= Total number of registered failures	0<=X<= 1. The closer to 1.0 is the better.	Absolute	Count	Nagios and SW Module Log files defined in VELaSCCo Platform.	Observe behaviour of user or maintainer who is trying to resolve failures.
M.AR#42	Failure analysis efficiency	Can user efficiently analyze cause of failure?	**X= Sum(T) / N** T= Tout - Tin Tout = Time at which the causes of failure are found out (or reported back to user)	0<=X. The shorter is the better.	Absolute	Time/count	Nagios and SW Module Log files defined in VELaSCCo Platform.	Observe behaviour of user or maintainer who is trying to resolve failures.

			Tin = Time at which the failure report is received N= Number of registered failures					
M.AR#43	Status monitoring capability	Can user identify specific operation which caused failure by getting monitored data during operation?	**X= 1- A / B** A= Number of cases which maintainer (or user) failed to get monitor data B= Number of cases which maintainer (or user) attempted to get monitor data recording status of software during operation.	0<=X<= 1.The closer to 1.0 is the better.	Absolute	Count	Nagios and SW Module Log files defined in VELaSCCo Platform.	Observe behaviour of user or maintainer who is trying to get monitored data recording status of software during operation.
M.AR#51	Availability of built-in test function	Can user and maintainer easily perform operational testing without additional test facility preparation?	**X= A / B** A= Number of cases in which maintainer can use suitably built-in test function B= Number of cases of test opportunities	0 <= X <=1. The larger and the closer to 1.0 is the better.	Absolute	Count	SVN Unit test code.	Observe behaviour of user or maintainer who is testing software system after maintenance.
M.AR#52	Re-test efficiency	Can user and maintainer easily perform operational testing and determine whether the software is ready for operation or not?	**X= Sum(T) / N** T= Time spent to test to make sure whether reported failure was resolved or not N= Number of resolved failures	0<X. The smaller is the better.	Ratio	Time/Count	SVN Unit test code, Nagios and SW Module Log files defined in VELaSCCo Platform.	Observe behaviour of user or maintainer who is testing software system after maintenance.

M.AR#53	Adaptability of data structures	Can user or maintainer easily adapt software to data sets in new environment?	**X = A / B** A = The number of data which are operable and but are not observed due to incomplete operations caused by adaptation limitations B= The number of data which are expected to be operable in the environment to which the software is adapted	0<=X<=1.The larger and closer to 1.0 is the better.	Absolute	Count	Acuario Cluster staff report, EDIIE Cluster staff report and ATOS staff report installing IFX and GiD clients.	Observe user's or maintainer's behaviour when user is trying to adapt software to operation environment.
M.AR#54	Hardware environmental adaptability	Can user or maintainer easily adapt software to environment? Is software system capable enough to adapt itself to operation environment?	**X= 1 - A / B** A= Number of operational functions of which tasks were not completed or not enough resulted to meet adequate levels during combined operating testing with environmental hardware B= Total number of functions which were tested	0<=X<=1.The larger is the better.	Absolute	Count	Acuario Cluster staff report, EDIIE Cluster staff report and ATOS staff report installing IFX and GiD clients.	Observe user's or maintainer's behaviour when user is trying to adapt software to operation environment
M.AR#55	System software environmental adaptability	Can user or maintainer easily adapt software to environment?	**X= 1 - A / B** A= Number of operational functions	0<=X<=1. The larger is the better.	Absolute	Count	Acuario Cluster staff report, EDIIE Cluster staff report and ATOS staff report	Observe user's or maintainer's behaviour when user is trying to adapt software to

		Is software system capable enough to adapt itself to operation environment?	of which tasks were not completed or were not enough resulted to meet adequate level during combined operating testing with operating system software or concurrent application software B= Total number of functions which were tested				installing IFX and GiD clients.	operation environment.
M.AR#56	Ease of installation	Can user or maintainer easily install software to operation environment?	**X = A / B** A = Number of cases which a user succeeded to in changing the install operation for his/her convenience B = Total number of cases which a user attempted to change the install operation for his/her convenience	0<=X<=1. The larger is the better.	Absolute	Count	Acuario Cluster staff report, EDIIE Cluster staff report and ATOS staff report installing IFX and GiD clients.	Observe user's or maintainer's behaviour when user is trying to install software to operation environment
M.AR#57	Ease of setup retry	Can user or maintainer easily re-try set-up installation of software?	**X = 1 - A / B** A = Number of cases in which user fails in re-trying set-up during set-up operation B = Total number of	0<=X<=1. The larger is the better.	Absolute	Count	Acuario Cluster staff report, EDIIE Cluster staff report and ATOS staff report installing IFX and GiD clients.	Observe user's or maintainer's behaviour when user is trying to re-try set-up installation of software?

			cases in which user attempt to re-try setup during set-up operation.					
M.AR#58	Available coexistence	How often user encounters any constraints or unexpected failures when operating concurrently with other software?	**X = A / T** A = Number of any constraints or unexpected failures which user encounter during operating concurrently with other software T = Time duration of concurrently operating other software	0<=X. The closer to 0 is the better.	Ratio	Count/Time	Nagios and SW Module Log files defined in VELaSCCo Platform.	Use evaluated software concurrently with other software which user often uses.
M.AR#59	Continued use of data	Can user or maintainer easily continue to use the same data after replacing this software to previous one? Is software system migration going on successfully?	**X = A / B** A = number of data which are used in other software to be replaced and are confirmed that they are able to be continuously used B = number of data which are used in other software to be replaced and planned to be continuously reusable	0<= X <=1.The larger is the better.	Absolute	Count	Nagios and SW Module Log files defined in VELaSCCo Platform.	Observe user's or maintainer's behaviour when user is replacing software to previous one.
M.AR#60	Function inclusiveness	Can user or maintainer easily continue to use	**X = A / B** A = number of functions which	0<= X <=1.The larger is the better.	Absolute	Count	Nagios and SW Module Log files defined in	Observe user's or maintainer's behaviour when user is replacing

	similar functions after replacing this software to previous one? Is software system migration going on successfully?	produce similar results as previously produced and where changes have not been required B = number of tested functions which are similar to functions provided by another software to be replaced		VELaSCCo Platform.	software to previous one.

Table 2. GQM Metrics Extended Table for AR Dimension.

The metrics M.AR#5 and M.EU#7 will not be evaluated because they require the use of a Bug Tracking tool to manage the software development process which is not in the scope of the VELaSCCo project. The Changeability Metrics MAR#45, MAR#46, MAR#47 and MAR#48 and the Stability Metrics MAR#49 and MAR#50 nor will be evaluated because of its dependence with the software development process management.

4 Data Collection

This chapter focuses on obtaining empirically values associated to architecture metrics defined above. To do so, several architecture metrics are having been chosen as well as specific tools and techniques to calculate them.

Architecture metrics aims to measure a wide set of features related to specific use cases below. At this stage of project development, some of them will be retrieved after using some tools (logs, TestComplete, Nagios, etc.) and other will be calculated in next evaluation iterations, which will be indicated with acronym TBC* (To Be Calculated). This issue will depend on VQueries development for future iterations.

4.1 DEM Use Case: Fluidized Bed (Small)

This simulation is composed of three different files: FluidizedBed_small.p3c, FluidizedBed_small.p3p and FluidizedBed_small.p3w. Metrics below summarize results after ingestion of three files in a unique model on HBase and therefore some metrics can be calculated querying to HBase tables. Besides this, some metrics are related to Visualization Clients time responses: the clients used to measure so are GiD and iFX.

Table 3 displays data collected for DEM Use Case:

Metric	Description	Dimension	SW Components Involved	Testing Tool	Value
M.EU#1	Number of files which the simulation is composed by.	End User Funct.	Non-applicable	Non-applicable.	3
M.EU#2	Number of Data Events generated.	End User Funct.	Data Injector, Flume Agents	Flume monitoring service.	3683
M.EU#3	Number of Hbase nodes	End User Funct.	Hbase	Hbase monitoring service.	9
M.EU#4	Injection Time	End User Funct.	Data Injector, Flume Agents and Hbase	Data Injector REST service.	3 sec.
M.EU#5	Simulation File Size	End User Funct.	Non-applicable	Non-applicable.	115 Mb
M.EU#6	Number of particles (p3p)	End User Funct.	HBase	Non-applicable	11880
M.EU#7	Number of	End User Funct.	Non-applicable	Non-applicable	3500

	contacts (p3c and p3w)				
M.EU#8	Number of computational time steps	End User Funct.	HBase	Non-applicable	100
M.EU#9	Number of results at particle level	End User Funct.	HBase	Non-applicable	6
M.EU#10	Number of results at contact level	End User Funct.	Non-applicable	Non-applicable	3
M.EU#11	User Credentials	End User Funct.	GiD/IFX and AccessLib	TestComplete, VELaSCCo Logs	velassco/***** GiD: 6024 ms iFX: 9544 ms
M.EU#12	Security Token		AccessLib	TestComplete, VELaSCCo Logs	TBC*
M.EU#13	Time of opening model query execution	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	On-going
M.EU#14	Time of getting simplified mesh query execution	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#15	GiD Model Rotation Velocity	End User Funct.	GiD, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#16	IFX Model Rotation Velocity	End User Funct.	IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#17	Time of getting original mesh query execution	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#18	Time of getting result on a vertex over time	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule,	TestComplete, VELaSCCo Logs	TBC*

			HBase		
M.EU#19	Time of getting the contour fill for a concrete result	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#20	Time of getting a cut in a volume mesh	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#21	Time of getting a cut in a volume mesh with results	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#22	User session logout trace	End User Funct.	GiD/IFX, AccessLib	TestComplete, VELaSCCo Logs	"User velassco logged out" GiD: 1023 ms iFX: 3478 ms
M.AR#3	Estimated latent fault density	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#4	Failure density against test cases	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#6	Fault density	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#8	Mean time between failures	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#11	Breakdown avoidance	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*

M.AR#16	Availability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#17	Mean down time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#18	Mean recovery time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#19	Restartability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#20	Restorability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#21	Restore effectiveness	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#22	Response Time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#23	Mean Time to response	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#24	Worst case response time	Architecture	GiD/IFX, AccessLib,	Nagios	TBC*

			QueryManager, StorageModule, HBase, Flume and Data Injector		
M.AR#25	Throughput	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#26	Mean amount of Throughput	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#27	Worst case throughput ratio	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#28	I/O devices utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#29	I/O loading limits	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#30	I/O related errors	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#32	Maximum memory utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#33	Mean occurrence of memory error	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule,	Nagios	TBC*

			HBase, Flume and Data Injector		
M.AR#34	Ratio of memory error/time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#35	Maximum transmission utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#36	Mean occurrence of transmission error	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#37	Mean of transmission error per time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#38	Transmission capacity utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#39	Audit trail capability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#40	Diagnostic function support	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#41	Failure analysis capability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*

M.AR#42	Failure analysis efficiency	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#43	Status monitoring capability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#51	Availability of built-in test function	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#52	Re-test efficiency	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#53	Adaptability of data structures	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#54	Hardware environmental adaptability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#55	System software environmental adaptability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#56	Ease of installation	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#57	Ease of setup retry	Architecture	GiD/IFX, AccessLib,	Nagios	TBC*

			QueryManager, StorageModule, HBase, Flume and Data Injector		
M.AR#58	Available coexistence	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#59	Continued use of data	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#60	Function inclusiveness	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*

Table 3. Data collection Table for DEM Use Case

4.2 FEM Use Case: Telescope (Small)

FEM Use case is composed by two types of files: Mesh files (.msh) and Result files (.res). These two types contain all information needed to calculate metrics in table below. Another particularity of FEM Use case is that several partitions files are created in order to avoid handle large simulation files which could affect to performance.

Table 4 displays all values calculated for metrics associated to FEM Use Case:

Metric	Description	Dimension	SW Components Involved	Testing Tool	Value
M.EU#1	Number of files which the simulation is composed by.	End User Funct.	Non-applicable	Non-applicable.	256
M.EU#2	Number of Data Events generated.	End User Funct.	Data Injector, Flume Agents	Flume monitoring service.	283718
M.EU#3	Number of Hbase nodes	End User Funct.	Hbase	Hbase monitoring service.	9

M.EU#4	Injection Time	End User Funct.	Data Injector, Flume Agents and Hbase	Data Injector REST service.	29:25 min.
M.EU#5	Simulation File Size	End User Funct.	Non-applicable	Non-applicable.	8.4Gb
M.EU#6	Number of particles (p3p)	End User Funct.	HBase	Non-applicable	23,870,544 tetrahedrons
M.EU#7	Number of contacts (p3c and p3w)	End User Funct.	Non-applicable	Non-applicable	Non-applicable for FEM simulations.
M.EU#8	Number of computational time steps	End User Funct.	HBase	Non-applicable	19 time-steps
M.EU#9	Number of results at particle level	End User Funct.	HBase	Non-applicable	2 (Pressure and Velocity (vector))
M.EU#10	Number of results at contact level	End User Funct.	Non-applicable	Non-applicable	Non-applicable for FEM simulations.
M.EU#11	User Credentials	End User Funct.	GiD/IFX and AccessLib	TestComplete, VELaSCCo Logs	Velassco/***** GiD: 7833 ms iFX: 10514 ms
M.EU#12	Security Token		AccessLib	TestComplete, VELaSCCo Logs	TBC*
M.EU#13	Time of opening model query execution	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	On-going
M.EU#14	Time of getting simplified mesh query execution	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#15	GiD Model Rotation Velocity	End User Funct.	GiD, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#16	IFX Model Rotation Velocity	End User Funct.	IFX, AccessLib, QueryManager, StorageModule,	TestComplete, VELaSCCo Logs	TBC*

			HBase		
M.EU#17	Time of getting original mesh query execution	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#18	Time of getting result on a vertex over time	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#19	Time of getting the contour fill for a concrete result	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#20	Time of getting a cut in a volume mesh	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#21	Time of getting a cut in a volume mesh with results	End User Funct.	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase	TestComplete, VELaSCCo Logs	TBC*
M.EU#22	User session logout trace	End User Funct.	GiD/IFX, AccessLib	TestComplete, VELaSCCo Logs	"User velassco logged out" GiD: 1422 ms iFX: 3524 ms
M.AR#3	Estimated latent fault density	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#4	Failure density against test cases	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#6	Fault density	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#8	Mean time	Architecture	GiD/IFX,	Nagios	TBC*

	between failures		AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector		
M.AR#11	Breakdown avoidance	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#16	Availability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#17	Mean down time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#18	Mean recovery time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#19	Restartability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#20	Restorability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#21	Restore effectiveness	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#22	Response Time	Architecture	GiD/IFX, AccessLib, QueryManager,	Nagios	TBC*

			StorageModule, HBase, Flume and Data Injector		
M.AR#23	Mean Time to response	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#24	Worst case response time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#25	Throughput	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#26	Mean amount of Throughput	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#27	Worst case throughput ratio	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#28	I/O devices utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#29	I/O loading limits	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#30	I/O related errors	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and	Nagios	TBC*

			Data Injector		
M.AR#32	Maximum memory utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#33	Mean occurrence of memory error	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#34	Ratio of memory error/time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#35	Maximum transmission utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#36	Mean occurrence of transmission error	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#37	Mean of transmission error per time	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#38	Transmission capacity utilization	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#39	Audit trail capability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*

M.AR#40	Diagnostic function support	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#41	Failure analysis capability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#42	Failure analysis efficiency	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#43	Status monitoring capability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#51	Availability of built-in test function	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#52	Re-test efficiency	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#53	Adaptability of data structures	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#54	Hardware environmental adaptability	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#55	System software environmental	Architecture	GiD/IFX, AccessLib,	Nagios	TBC*

	adaptability		QueryManager, StorageModule, HBase, Flume and Data Injector		
M.AR#56	Ease of installation	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#57	Ease of setup retry	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#58	Available coexistence	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#59	Continued use of data	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*
M.AR#60	Function inclusiveness	Architecture	GiD/IFX, AccessLib, QueryManager, StorageModule, HBase, Flume and Data Injector	Nagios	TBC*

Table 4. Data collection Table for FEM Use Case.

5 Interpretation of Data Collected

Data collected on chapter before provides information about basic functionalities aspects of VELaSCCo platform developed so far. At this stage of the project, the more advanced modules developed are those related with database storage in a proper data model, simulation data ingestion and client visualization tools. The main challenge so far has been the proper interaction among these services to allow complete workflow execution successfully.

In this context, metrics presented above display satisfactory results in terms of functionalities achieved, because both Use Cases have been finally assessed and platform environment has been able to manage them successfully. To fulfil this objective, Use Cases have been implemented by using small simulations examples, in order to prioritize functionalities and modules compatibility over performance metrics. According to this, it is observed how metrics calculated are those related with services main requirements (M.EU#1-M.EU#8) whereas hardware-related metrics (M.AR#N) have been postponed to next evaluation iterations. Some of the most interesting metrics are those related to data ingestion and access: for instance, M.EU#2 and M.EU#4 indicates the number of events generated during data ingestion process and the time taken on finishing such ingestion process respectively. Both metrics measure empirically the performance of data ingestion which can provide an idea about size limit for VELaSCCo platform supported simulations. Hence, scalability for future iterations in terms of simulations size is directly related to these metrics.

Besides this, it is important to remark than VQueries are under development phase, so currently not all of them have been available to be exhaustively evaluated. This is the reason why main metrics related to Client visualization (GiD/iFX) are M#EU.11 and M#EU.22, related to user login and logout process. According to this, once that module interoperability is assured and visualization client can load simulation models properly, next evaluation should be able to measure Open Model times as well as execute several VQueries over the models, like GetListOfAnalyses, GetListOfTimeSteps, GetListOfMeshes, GetListOfMeshes, GetListOfAnalysis, GetListOfTimeSteps and GetListOfResults.

6 Conclusions

Initially the deliverable reports on the current status of the application of GQM methodology to End User Functionalities and Architecture Dimensions. In addition to that an overview of the study set-up is provided including a detailed description of Acuario Cluster, Testing tools and Use case scenarios decomposed in tasks.

Subsequently the Measurement Plan associated to the two dimensions considered is defined extending to the GQM Metrics table reported in D5.1 [1] . As a prelude to the specification of the extended tables we provide a description about how to read and use the metrics that appear in the GQM Metrics Extended Table, providing information such as description, purpose, formula, interpretation of measured value, metric scale type, measure type, tool and technique for each metric of the table.

From the conceptual point of view the metrics related to End User Functionalities Dimension have been grouped into the following groups: Injection Metrics, Simulation Configuration Metrics, Security Metrics and Performance Metrics. Respect to Architecture Dimension the metrics defined have been grouped into the following groups: Reliability Metrics (Maturity Metrics, Fault Tolerance Metrics, Recoverability Metrics), Efficiency Metrics (Time Behavior Metrics, Resource Utilization Metrics), Maintainability Metrics (Analyzability Metrics, Testability Metrics), and Portability Metrics (Adaptability Metrics, Installability Metrics, Co-existence Metrics, Replaceability Metrics).

Once Measurement Plan was defined the Data collection phase started. TestComplete, Nagios and VELaSCCo System Logs were the mechanisms for data collection in order to cover the different metrics proposed collecting and validating the data.

Due to the limited availability of features developed in visualization clients GiD and IFX at 30/11/2015 only a minimum subset of the total metrics has been calculated. Most VQueries needed to evaluate all assessment steps described in section 2.2.3 were not available.

Analysing the data to assess conformance to the goals we can say that only a minimal set of objectives has been achieved. In concrete only the goals related to Data injection and the goals related to Connect and Disconnect from the Platform.

7 References

[1] VELaSCCo D.5.1. Evaluation Methodology.
[2] VELaSCCo D1.1. End-users requirements and Users panel.
[3] VELaSCCo D1.5. Definition of criteria and methodology for system evaluation
[4] VELaSCCo D3.1. Query framework implementation in the project database system & report.
[5] VELaSCCo D4.2. VELaSCCo First Prototype Of High Performance Visualization Client
[6] VELaSCCo D3.4. Engine able to perform first-time visualizations and simple queries of the last results and over the unmodified domain or the transformations performed in D3.2 (EDM) & report.
[7] VELaSCCo D4.1. Specifications of the GPU-driven representations and architecture of the GPU-based scientific visualization pipeline.
[8] VELaSCCo D2.4. Design a petabyte sized engineering data solution.
[9] ISO/IEC 9126-2: Software engineering. Product quality. Part 2: External metrics. http://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-2%20Standard.doc