Quality Assurance in FIWARE

FIWARE team

Table of Contents

1 Introduction

This document explains all the work done and obtained results in the Quality Assurance task of FIWARE. The goal of this task has been analyzing and assessing the quality of the most used FIWARE GEs considering functional and non-functional aspects.

FIWARE is rapidly moving from experimental to production environments in which the platform must scale in reliable and real workload conditions. This fact implies that all FIWARE GEris must work at an adequate quality, reliability and at performance level appropriate for these conditions. The reported task in this deliverable was launched in the framework of the initiative to analyze and assess the level of quality of each GE, providing diverse kind of **reports, labels for GEs and an assessment dashboard**.

The quality is evaluated from different points of view:

- **Curation of GEs documentation** (documentation testing), both inspecting the code and the accompanying documentation (installation manuals, user guidelines, academy courses and similar). The goal of this assessment is to support FIWARE users with high-quality support for installation, configuration and operation of FIWARE technology, thereby improving the FIWARE user experience in general.
- **Verification of the GE specification** (functional testing), developing the appropriate test cases to assess if the GEs implementation corresponds to what is defined in the specification.
- **Assessment of performance, stability and scalability** of GEs in operational environments, like under excessive workload (stress testing). Test scenarios are defined and executed such that limits of a GE under test are identified, and can be compared with reference levels. The goal of this assessment is to favor the applicability of FIWARE in purely commercial scenarios.

The **testing of the documentation and verification** has been done for all GE not deprecated in FIWARE Catalogue (**28 in total**). Three phases have been required to complete the QA functional test process. The first phase verifies for each GE the completeness of documentation, the consistency of artefacts and the soundness of information. The usability of documentation, by example, in case of installation manual is checked installing step by step the GE. In the second phase specific method calls verified the single APIs and the response correctness of each GEs. The last phase consisted of functional verifications based on reference architectures integrating some GEs. As result a live dashboard collects and maintains the assessment information and GE owners are punctually requested to correct the encountered deficiencies. At the end of the task, near 95% of the high priority GEs has passed successfully the documentation and verification tests. The medium and low priority GEs are around 80% of success but they are working on solving the issues.

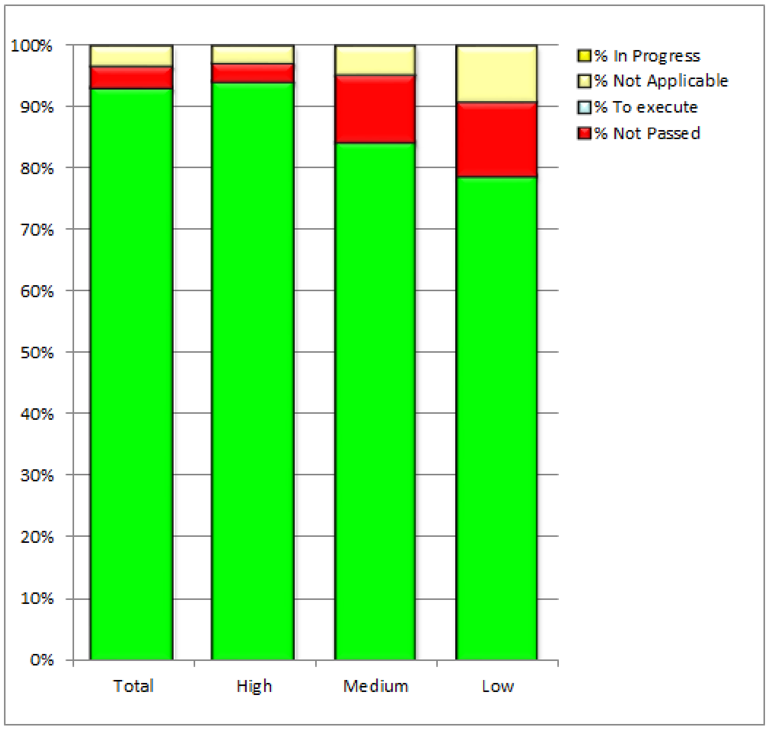


Figure 1: Overall functional testing results

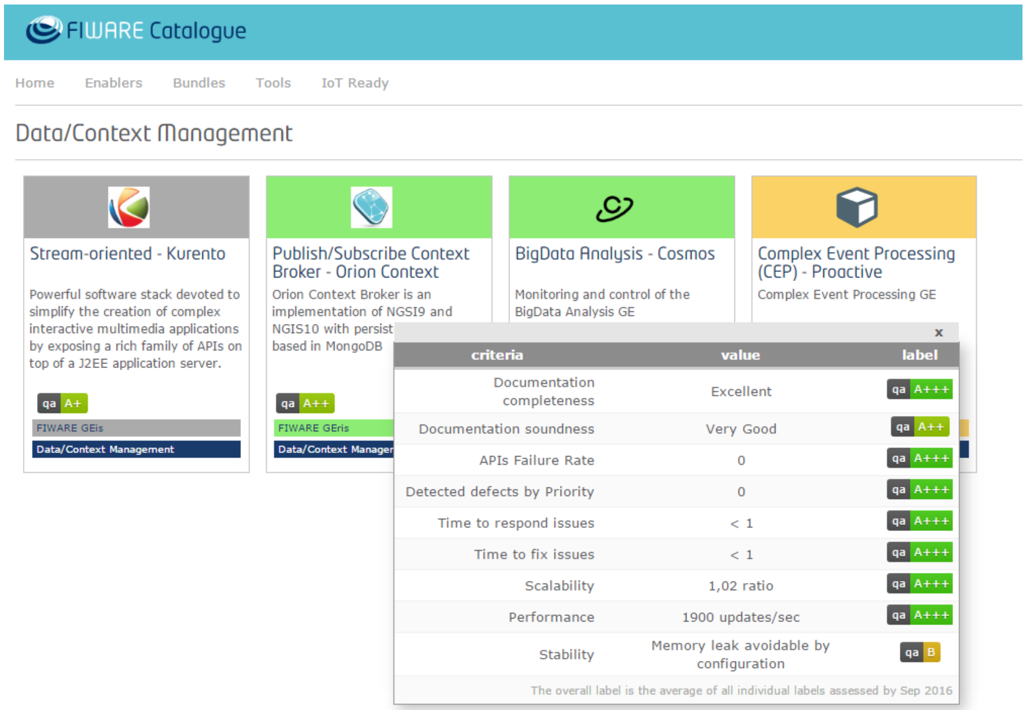
On the other hand, the **stress testing** has been performed only for those GEs most critical in terms of performance in the overall architecture. An iterative process and operative methodology have been put in place, obtaining after each iteration, a complete report with the measures obtained after stress test and analysis of the data. The reports were sent to the GE owners for considering improvements about performance and stability for next release. Three iterations have been achieved until September this year: one took place in February testing 9 GEs (Orion, Proton, IoT Broker, IDAS, Kurento, Wilma, KeyRock, Cepheus, Bosun); the second one in May testing new versions of these GEs; and final one has tested again a new updated version of some of these GEs (Orion, Proton, Bosun, Wilma, KeyRock) plus two more identified (AuthZForce and Cygnus) and more frequent combination of GEs (IDAS+Orion+Cygnus and Wilma+KeyRock+AuthZForce). In summary, **10 GEs and 2 bundles** were tested in stress conditions.

Once the first iteration of stress testing was conducted, a quality assurance expert was consulted for carrying out an **independent assessment** of the followed process and executed tests to produce an assessment of the achieved work. The main conclusions of his assessment were:

- Important performance borders were identified
- Robustness of use within bounds was shown
- Documentation needs to be improved

According to this assessment, FIWARE GEs are fit for being released in a commercial operational environment with some adjustments. A new external independent assessment has been requested at the end of the task. The report was not available already at the time of this report.

As part of the overall testing process and based on the obtained results in the three aspects (documentation, verification and stress) above mentioned, an overall label of quality is granted to each GE. This global label represents the degree of quality of the GE by adopting the energy labelling system used by EU for devices. Specific labels for each analyzed category (usability, reliability, efficiency, scalability, performance and stability) are also granted. Thus, in the Catalogue each GE is labelled with a global label expanded by clicking of detailed labels map.



Data/Context Management

Stream-oriented - Kurento

Powerful software stack devoted to simplify the creation of complex interactive multimedia applications by exposing a rich family of APIs on top of a J2EE application server.

Publish/Subscribe Context Broker - Orion Context

Orion Context Broker is an implementation of NGSI9 and NGSI10 with persist based in MongoDB

BigData Analysis - Cosmos

Monitoring and control of the BigData Analysis GE

Complex Event Processing (CEP) - Proactive

Complex Event Processing GE

criteria	value	label
Documentation completeness	Excellent	qa A+++
Documentation soundness	Very Good	qa A++
APIs Failure Rate	0	qa A+++
Detected defects by Priority	0	qa A+++
Time to respond issues	< 1	qa A+++
Time to fix issues	< 1	qa A+++
Scalability	1,02 ratio	qa A+++
Performance	1900 updates/sec	qa A+++
Stability	Memory leak avoidable by configuration	qa B

The overall label is the average of all individual labels assessed by Sep 2016

Figure 2: Visualization of quality labels in the FIWARE Catalogue

List of online resources those are available:

- Wiki page recording all the tests and results: http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_QA_Chapter
- Code and guidelines for executing the functional tests: https://github.com/Fiware/test.Functional
- Code and guidelines for executing the stress tests: https://github.com/Fiware/test.NonFunctional
- All the reports and other related generated assets at Docman in FIWARE forge under FIWARE Quality Assurance folder.
- Blog post at FIWARE blog: https://www.fiware.org/2016/09/20/assessing-fiware-ges-quality/

2 Functional Testing

The functional test activity verifies the consistency of the documentation and of the software package of each Generic Enabler and its usability both in standalone mode and combined as integrated platform.

2.1 Motivation and approach

The main motivation to perform the functional test activity is to verify the consistency of GEri's release and their usability both in standalone mode and combined as integrated platform.

- The first scope is to check the completeness of documentation, specification, implementation and installation of the GEris.

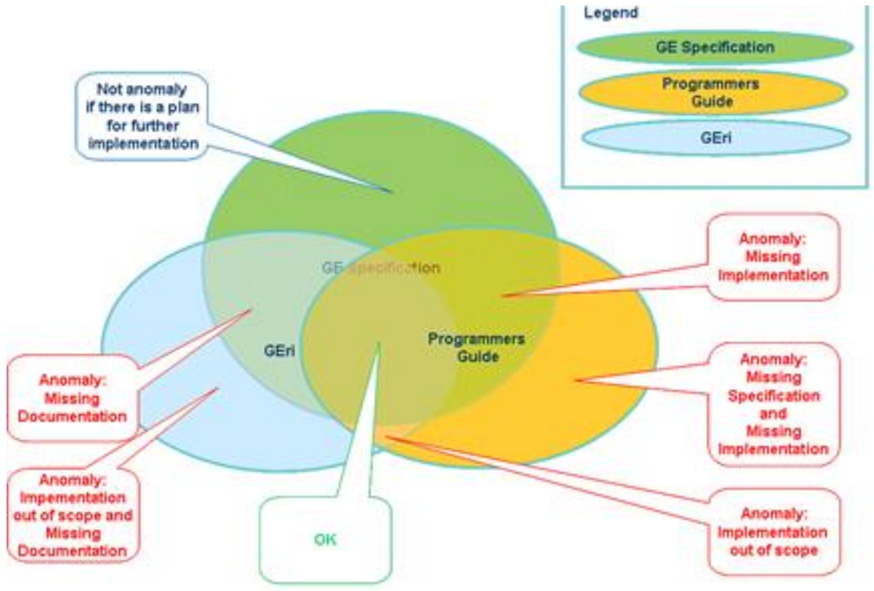

Legend
GE Specification
Programmers Guide
GEri
Not anomaly if there is a plan for further implementation
GE specification
GEri
Programmers Guide
Anomaly: Missing Implementation
Anomaly: Missing Specification and Missing Implementation
Anomaly: Missing Documentation
Anomaly: Impementation out of scope and Missing Documentation
OK
Anomaly: Implementation out of scope

Figure 3: Artefacts completeness & consistency

- Another scope is to validate the APIs of the most used GE, the same ones already object of the not-functional test (performance test).
- The last scope is to perform the functional tests on the main functionalities of a bundle of GEs that identify the reference architecture selected as case study.

2.2 Methodology

Thus, the functional tests are structured into three phases:

1. Documentation Testing
2. APIs Testing
3. Bundle Integration Testing

After these three phases, there is the activity of **test and bug management** to make sure that each test activity performed is traced on the FIWARE Backlog and linked to Jira platform as a work-item ticket.

Each bug is reported on the Jira project corresponding to the Technical Chapter/Component under verification and linked to the work-item (test activity). When the test phase finishes, the work-item is closed.

The functional test activity includes also the evaluation of the **training efficiency** of the courses published and available on the FIWARE Academy (edu.fiware.org).

The training material created and published to the FIWARE Academy was evaluated on the basis of identified criteria corresponding to specific recommendations to be implemented for granting an efficient training offering. These recommendations (listed in the FIWARE wiki page “Working with the FIWARE Academy”) are both formal (minimum set of requirements to make the course comprehensible to users) and QA (set of requirements to address in order to comply with a good quality). The verification of formal requirements has been always an on-going activity within the sustainability task, in charge of the training content development and organization.

Thus, the evaluation activity included two different tasks performed by different teams that cooperated together and worked in parallel both in the definition of evaluation criteria and in the review phase:

- the evaluation of the QA requirements, resulting in the consequent QA label;
- the verification of formal requirements related to the structure of the course and to its efficient output.

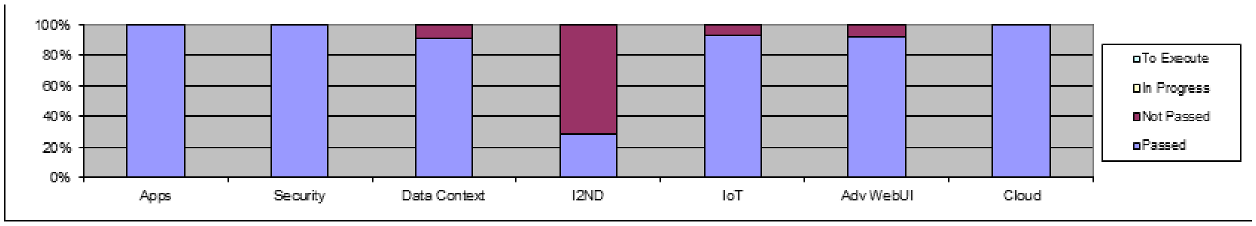
2.3 Performed tests

Documentation Testing

In the first phase it will be verified for each GEris, the completeness of documentation, the consistency of artifacts and the soundness of information, based on a specific user profile.

Moreover it will be checked the usability of installation manual by installing step by step the GEri, performing the sanity check operations and invoking the main APIs.

Here below the summary of the documentation tests results. It can be observed that almost all chapters are above 90% of completeness and soundness, except the ISND chapter which is below the 40%.



Documentation Testing Validation criteria
The designed tests aim to checks the completeness, consistency, soundness and usability of GEs.

Some checks are subjective because they are based on the evaluator profile. Therefore two evaluation levels (decision makers and developers) have been identified.

Checking completeness means to verify that each released artefact is complete in all its parts.
The consistency check intends to verify that the release contains all the expected artefacts and that they are consistent between them.

The soundness' verification ensures that each artefact has proper contents to its purpose and suitable to the profile of those who uses them. In fact, the content of a document might be enough to a manager who must decide whether to adopt a solution based on FIWARE but not enough to the developer that must implement and vice versa.

Finally, the usability check intends to verify that a document or a package is easily usable, for example, that an installation manual allows to properly installing a released package.

Some verifications of "Completeness" are made on the web catalogue and they aim to ensure that information are complete, updated and the linked contents are really accessible. A detailed checklist, containing all verifications to execute, is linked to these tests. This information is most useful to high-level user profiles that need an overview to determine if an application is useful to their goals.

Another verification of completeness, useful to both profiles, is to check the Programmer's Guide and Open Specification in order to verify the Programmer's Guide covers fully the Open Specification.

Regarding Usability:

- for a decision maker it is required:
    - all basic information are available and easily attainable
    - the catalogue is easily navigable
    - training / online courses are available
- for a developer it is required:
    - get simple and fast installation methods such as docker, script
    - make a step by step GEri installation using the installation manual

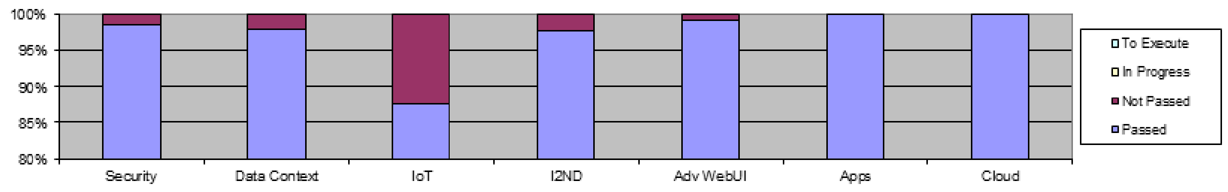
- execute the Sanity Check Procedures
- invoke easily the exposed APIs.

APIs Testing

In the second phase, it will be verified all APIs of the GEs, through the specific method call and verifying the response correctness.

Each Application Programming Interface has been verified as atomic procedure, without combining business logic. Each procedure has been tested only in the positive case in order to verify the correct execution of the functionality. The purpose is to verify that the software package actually contains all the APIs declared in the programmer guides without paying special attention to the negative case or wrong input of each specific interface.

Here below the summary of APIs tests summary. It can be observed that most of chapters are around 100% of correctness, except the IoT chapter which is between 85-90%, a very good figure as well.



Bundle Integration Testing

The functional bundle integration testing activity performs verifications based on functional scenarios combining some of the main GE's interfaces to highlight the interaction among the GEs composing the bundle. The functional bundle integration tests simulate a real use case.

The proposed functional scenario is a simple Parking management, involving sensors to detect when a new car enter into the parking, another sensor when a car exits out of the parking and the last sensor to detect the CO2 level in the parking. The users can manage and verify the parking information according to specific policy of access control.

The GEs that compose the integrated FIWARE bundle where the functional integration tests are performed are:

Chapter	GE
Security	Identity Management - KeyRock
	PEP Proxy – Wilma
	Authorization PDP – AuthZForce

Internet of Things Services Enablement	Backend Device Management – IDAS IoT Agent Ultraligth2.0/HTTP
Data/Context Management	Publish/Subscribe Context Broker - Orion
	Complex Event Processing (CEP) - Proactive Technology Online (Proton)
Applications/Services and Data Delivery	Data Visualization - SpagoBI

The functional scenarios use some of the main GE interfaces to highlight the interaction among the GEs of the platform. The integration tests have been performed using a Client Application in order to simulate a real application. Here follows the list of the scenarios grouped by functional area contents.

Test Suite	ID TestSuite_TestCase	Test case
Security setup	**¡Error! No se encuentra el origen de la referencia.**	Register User
	¡Error! No se encuentra el origen de la referencia.	Register Application
	¡Error! No se encuentra el origen de la referencia.	Register new PEP-PROXY
Access Control	**¡Error! No se encuentra el origen de la referencia.**	Login as Guest
	¡Error! No se encuentra el origen de la referencia.	Login as Operator
	¡Error! No se encuentra el origen de la referencia.	Login as Administrator
Entity Management	**¡Error! No se encuentra el origen de la referencia.**	Create Entity Parking
	¡Error! No se encuentra el origen de la referencia.	Remove Entity Parking
	¡Error! No se encuentra el origen de la referencia.	Modify Entity Parking - Access Permit
	¡Error! No se	Modify Entity Parking - Access Denied

	encuentra el origen de la referencia.	
	¡Error! No se encuentra el origen de la referencia.	Get Entity Parking
Service Registration	**¡Error! No se encuentra el origen de la referencia.**	Create service Parking
Device Registration	**¡Error! No se encuentra el origen de la referencia.**	Create device: Sensor sCarEntry
	¡Error! No se encuentra el origen de la referencia.	Create device: Sensor sCarExit
	¡Error! No se encuentra el origen de la referencia.	Create device: Sensor sCO2
Observation Measurement	**¡Error! No se encuentra el origen de la referencia.**	Sensor sCarEntry detects a car entry (parking not full)
	¡Error! No se encuentra el origen de la referencia.	Sensor sCarEntry detects a car entry (parking full)
	¡Error! No se encuentra el origen de la referencia.	Sensor sCarExit detects a car exit
	¡Error! No se encuentra el origen de la referencia.	Sensor sCO2 measures the CO2 level
Data visualization	**¡Error! No se encuentra el origen de la referencia.**	Dynamic Report - Parking data
	¡Error! No se encuentra el origen de la referencia.	Static Report – Parking data
	¡Error! No se encuentra el origen de la referencia.	Static Report – Parking Statistics

Table 1: Functional scenarios of integration tests

The integration tests have been performed imagining to simulate a real application with a simple business logic. The test suite is a group of test cases or scenarios related to the same functional content. Each Integration Test Scenario is a test case containing a list of steps explaining the flow of the communication through the GEs and for each test case is reported the log of the main process steps.

The detailed results are reported in the ANNEX 2 of the current document

The attached image describes spreadsheets collecting the designed test suite.

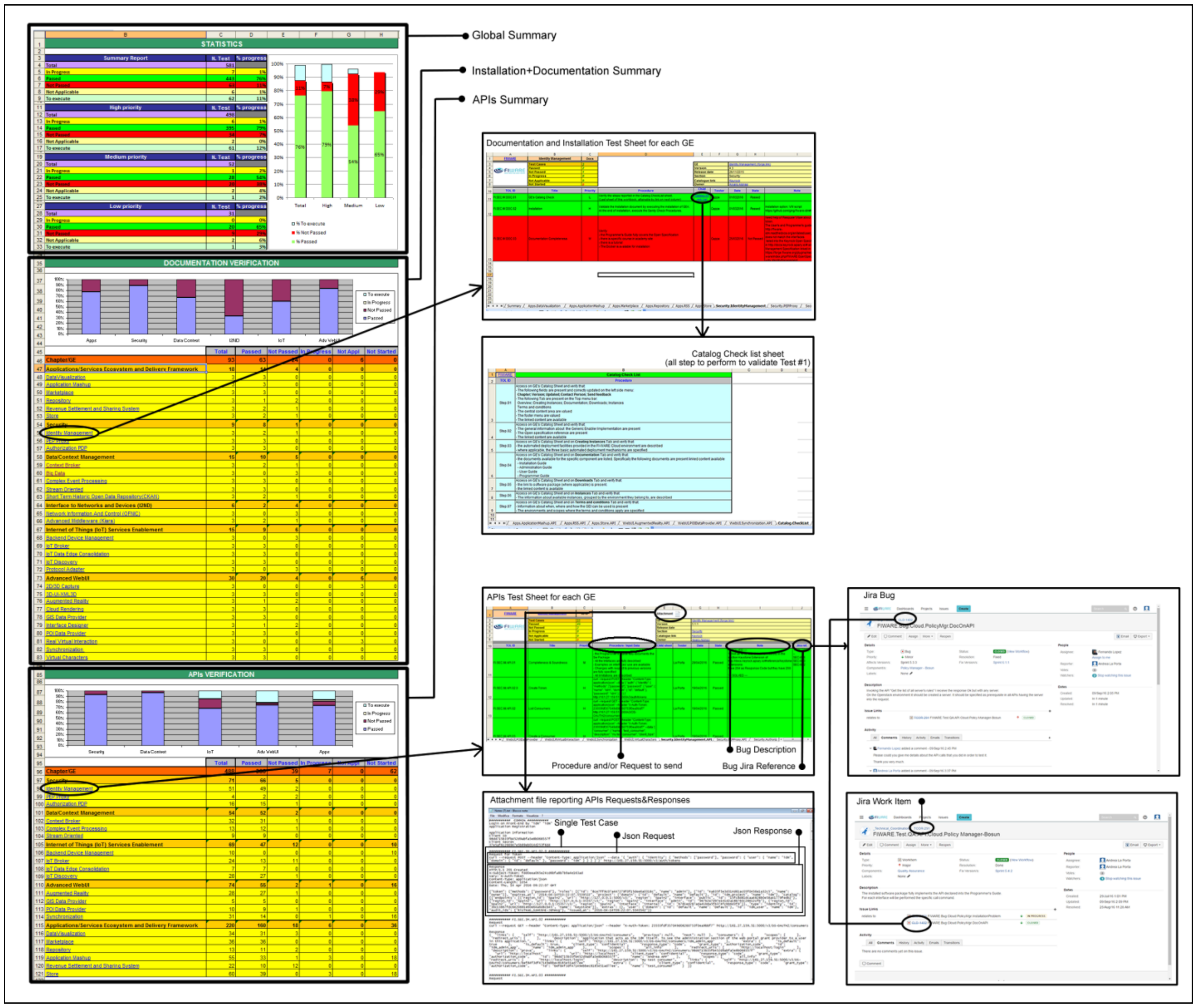

Global Summary
Installation+Documentation Summary
APIs Summary
Documentation and Installation Test Sheet for each GE
Catalog Check list sheet
(all step to perform to validate Test #1)
APIs Test Sheet for each GE
Procedure and/or Request to send
Bug Description
Bug Jira Reference
Jira Bug
Attachment file reporting APIs Requests&Responses
Single Test Case
Json Request
Json Response
Jira Work Item
STATISTICS
DOCUMENTATION VERIFICATION
APIs VERIFICATION

Figure 4: Spreadsheets collection with functional tests results

It has a summary sheet that reports the global overview on activity progress and detailed sheets for each GE.

These sheets show all planned tests, permit to trace the execution information and the Jira reference in case of failed result.

Each sheet also reports the general information (owner, version, etc.) about GE or Platform to test and the global pre-requirements in order to perform the tests.

For some tests, in addition to the execution procedure and the validation criteria, a “child” sheet that represents a checklist is linked.

It reports, based on the test scenario, the items list to verify, the steps to execute or the APIs to invoke.

Academy courses testing

The testing activity of the FIWARE Academy courses is the task performed to evaluate the efficiency of training.

The courses are publicly available at edu.fiware.org, and any user interested in using FIWARE can easily understand whether a given course does fit the user's need or not.

The training material created and published to the FIWARE Academy was evaluated on the basis of the criteria listed in the guidelines to be followed by the editor of the course for granting an efficient training offering. The guidelines are publicly available in the FIWARE wiki page "working with the FIWARE Academy# Course Evaluation for Efficient Training"[1] , and the GE Owners are asked to follow those guidelines, reported below.

The following table shows the list of the academy courses evaluated and the relative results.

Chapter	Course ID	Status	Score
Security			
Identity Management	79	Completed	Good
PEP Proxy	131	Completed	Good
Authorization PDP	144	Completed	To Improve
Access Control (OAUTH-API-AZ)	57	Completed	To Improve
Applications/Services			
Apps and Services Overview	52	Completed	Sufficient
DataVisualization	141	Completed	Sufficient
Application Mashup	53	Completed	Good
Marketplace	21	Completed	Sufficient
Repository	127	Completed	Sufficient
Revenue Settlement and Sharing System	117	Completed	To Improve

[1] https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Working_with_the_FIWARE_Academy#Course_Evaluation_for_Efficient_Training

Store	104	Completed	Sufficient
Data/Context Management			
Context Broker	132	Completed	Good
Context Broker (2)	44	Completed	Sufficient
Context Broker (5)	149	Completed	Sufficient
Big Data	69	Completed	Sufficient
Complex Event Processing	58	Completed	Good
Stream Oriented	62	Completed	Good
Short Term Historic Open Data Repository (CKAN)	145	Completed	To Improve
Interface to Networks and Devices (I2ND)			
Network Information And Control (OFNIC)	72	Completed	To Improve
Advanced Middleware (Kiara)	140	Completed	To Improve
Internet of Things (IoT) Services Enablement			
Backend Device Management (IDAS)	128	Completed	Sufficient
IoT Broker	33	Completed	Sufficient
IoT Data Edge Consolidation	36	Completed	Sufficient
IoT Discovery	40	Completed	Sufficient
Advanced WebUI			
3D-UI-XML3D	97	Completed	To Improve
Cloud Rendering	92	Completed	Sufficient
GIS Data Provider	88	Completed	To Improve
Interface Designer	91	Completed	Sufficient
POI Data Provider	96	Completed	Sufficient
Synchronization	111	Completed	Sufficient

Virtual Characters	112	Completed	Sufficient
Cloud Hosting			
Policy Manager	119	Completed	Sufficient

Table 2: Results of academy courses testing

2.4 Summary of obtained results

The quality assurance functional testing activities have been performed by a team with technical competences corresponding to the target teams who will use the FIWARE platform.

Also the human factor is an important value in the propagation of innovative platforms even if not perfect and experimentally because they are easily accepted, fitted and implemented by open mind teams.

The activities have achieved positive results and in case of insufficiencies, using the Jira tool, bugs have been reported to the owners of the generic enables to improve their products. The Jira tool has been very useful to track the problems and also to require information, in most cases a successful cooperation has been reached between the QA team and the owner of the GEs.

Few simple comments to summarize the functional test activities:

- **Documentation testing**
  The GE manual allows installing the components but the documentation is not always clear, readily available from the many links.
- **APIs testing**
  The installed software package implements the API declared into Open Specification.
  The main failures concern the missing information on documentation.
- **Bundle Integration tests**
  This activity is the most interesting, because it combines more generic enablers going to compose a platform. They are test on how they work together by simulating functional scenarios.
- **Academy courses testing**
  The training is efficient if it fits the user's need and grants to acquire a quick understanding of the product. Overall the academy courses are sufficient; also in this case many tips have been reported on Jira tool to help the owner of the GEs to improve the training.

3 Non-functional (stress) testing

The non-functional test activity assesses the behavior of each GE in limit conditions of loading and stress.

3.1 Motivation and approach

FIWARE is approaching real life and production environments in which the platform must behave in reliable and real workload conditions. This fact implies that all FIWARE GEs must work at an adequate quality, reliability and performance level for these conditions. In previous platform stages, testing at component level has been performed by GE owners, but now, both functional and stress testing need to be put in place, helping GE owners to improve the quality of their GEs. The present section states why stress (or non-functional) testing is convenient at this stage, under which assumptions this kind of testing are conducted and who are involved in such task.

Main motivation for having a task like this in the project is to evaluate the performance and stability of FIWARE GEs in similar conditions to a production and real environment by stressing the GEs up to their maximum capacity and reaching load peaks. Time responses, usage of memory, response error rate among other parameters are measured by professional testing units of FIWARE partners (Atos and ENG) who are performing the evaluation.

In order to be practical and operative, a light and practical method commonly agreed by involved partners has been followed (see Section 3.2). The section 3.3 shows all the performed stress tests and the results obtained. The tests have been reported homogeneously following a common template (as it is included as annexes of this document).

Finally, an external third party has elaborated an assessment of the testing process and obtained results to verify that tests are being done properly under standard method. The assessment has been conducted twice, one at the end of first wave of tests; and a second one at the end of all tests. The recommendations provided by the first assessment were taken into consideration for the further iterations of tests. The two reports are annexed to this deliverable.

Due to the objective itself of the task, it was planned very operative and driven by prone results. This fact implied to establish a set of principles that allowed us to perform the task in optimal conditions adapted to these external factors. Thus, following principles can be enumerated:

- The non-functional testing will first focused on performance and stability, extended to scalability when possible. Orion GE has been the most deeply tested about scalability (horizontal and vertical).
- The approach will be very practical, but some method is need to be followed defining how to test and how to produce the results, as describe in section 3.2.

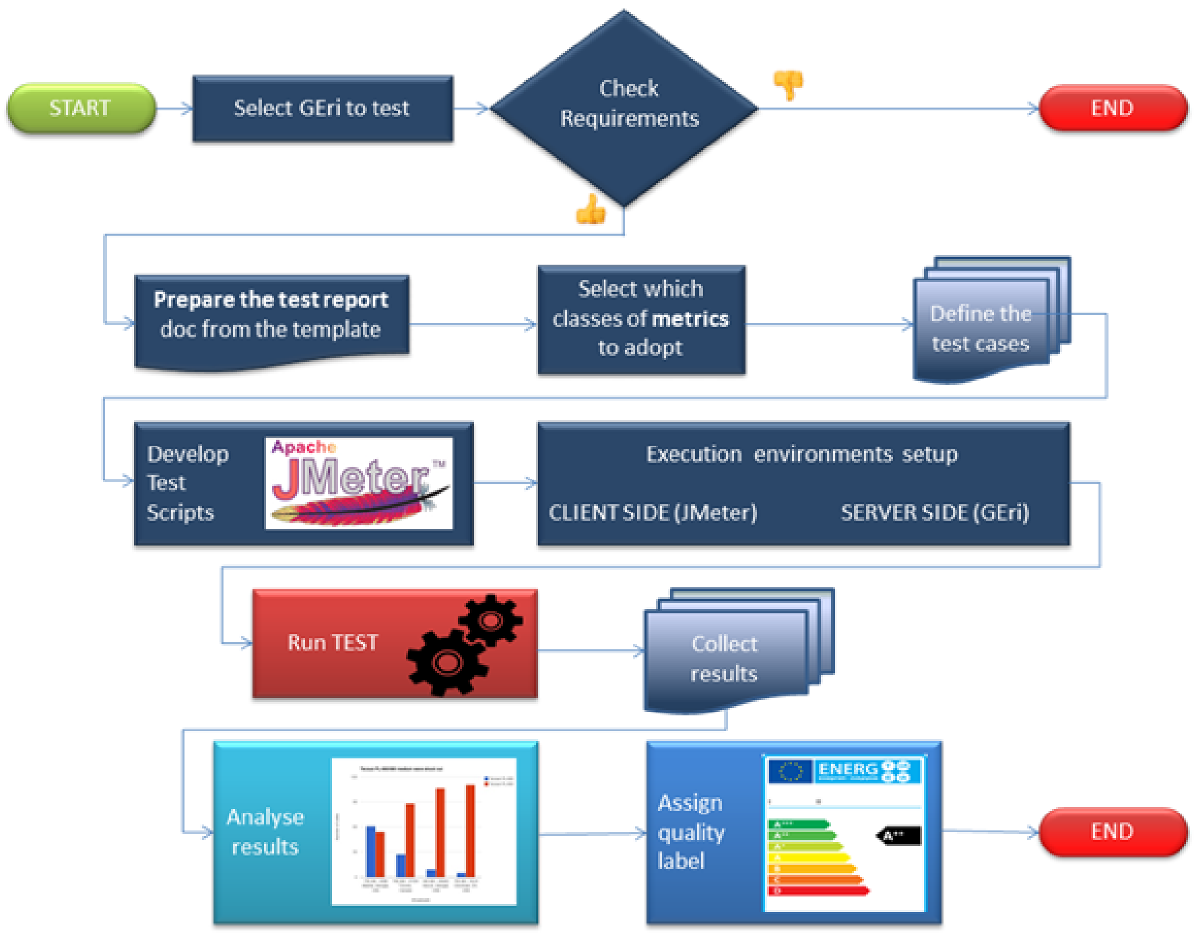
- The method will be non intrusive with GE owners' testing assurance procedures, but providing them recommendations about their GE's behavior to improve in future GE versions
- When possible, the task will re-use some existing infrastructure for performing the tests; otherwise ad-hoc environment might be created
- The testing execution and results are updated with every release of GEs, to keep constant the performance level of GEs or even improve it ideally. In the future, the tests will be automated when possible to be launched for each new release.
- Automation is a nice to have but not a must, since it is not trivial due to the diverse nature and typology of the GEs. Each GE requires a completely different infrastructure and method of testing.
- All the tests scripts and methods are published to allow anyone to replicate the test when wished. A dedicated project in FIWARE GitHub is available for such purpose (named as test.nonfunctional).

3.2 Methodology

As already mentioned, a light and practical method was chosen for carrying out the stress testing. Each Generic Enabler (GE) was tested by following, as much as possible, a "black box" approach, namely soliciting its APIs and examining some non-functional parameters such as, for example, stability and performance, without any knowledge of the internal implementation.

The purpose of this process was in fact to determine the point at which extremes of scalability or performance lead to unstable execution thus assessing the limits/breakpoints of each GE implementation.

In the following chart is represented the flow adopted in order to test each Generic Enabler reference implementation (GEri).

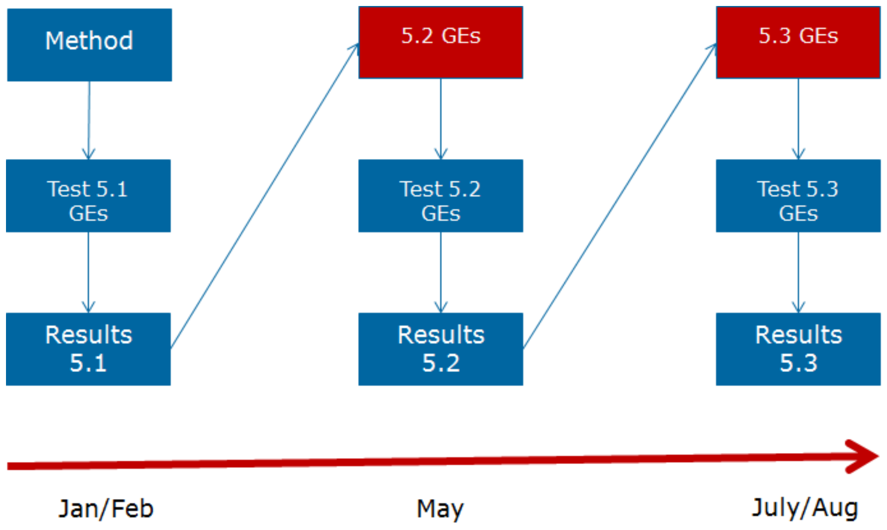


This flow was realized with the execution of ten steps, which are listed as following and then briefly described:

1. Select the GE implementation to be tested
2. Check requirements
3. Prepare the Test report doc from the template
4. Select which classes of metrics to adopt
5. Define the Test cases
6. Develop Test scripts
7. Run Tests
8. Collect results
9. Analyze results
10. Assign quality labels (shared with Functional testing)

3.3 Performed tests

Three waves of tests were done along the task. The first one was executed by beginning of 2016 considering the release 5.1 of the nine selected GEs (Orion, Wilma, KeyRock, Cepheus, Bosun, Proton, IDAS, Kurento, IoT Broker). The results were provided to the GEs who were able to consider them in some cases for the next release, 5.2, which was tested by May 2016. In this occasion, only six GEs out of the initially ten selected were tested (Orion, KeyRock, Wilma, Kurento, Cepheus and Bosun). The others did not produce significant changes in their versions to consider a new relevant testing. Once again the results were sent to the GE owners for improving next version. And finally, version 5.3 of all the GEs (the ten in total) was tested during this summer. All the reports corresponding to each wave of tests can be found in the annexes of this document. This section is summarizing the performed tests to each GE and the obtained results in the last wave of tests, that is, for version 5.3 in most of the cases.



The summary of the three waves of testing can be seen at a glance in following tables:

GE name	Reliability (errors rate)	Performance in stress condition (num requests, response time)	Stability (crashes)
Orion	Very good	Good (4000 subscriptions/sec)	Good
Proton	Very good	Very good (500 requests/sec; more tan 900 creating definitions/sec)	Average (no closing threads)

IDAS	Very good	Very good (200 simultaneous threads, 120 devices/sec)	Average (no releasing memory)
KeyRock	Very good	Good (300 concurrent threads)	Very good
Wilma	Very good	Good (30.000 authentications in 6 min)	Very good
Kurento	Good	Good (but only for less than 50 simultaneuos users and low quality video)	Good
IoT Broker	Average	Not really good (more than 1 sec/request)	Good
Cepheus	Very good	Good (100 requests/sec, but not stable)	Average (no releasing memory)
Bosun	Good	Average (150 simultaneous threads max, 5 HTTP responses/sec)	Good

Table 3: Results of release 5.1 stress tests

GE name	Reliability (errors rate)	Performance in stress condition (num requests, response time)	Stability (crashes)
Orion	Very good	Very Good (2000 attributes updated/sec)	Good
KeyRock	Very good	No good (300 concurrent threads)	Very good
Wilma	Very good	Good (30.000 authentications in 6 min)	Very good
Kurento	Good	Average (for less than 50 simultaneuos users and low quality video)	Good
Cepheus	Very good	Good (100 requests/sec, but not stable)	Average (no releasing memory)
Bosun	Good	Good (185 simultaneous threads max, 26 HTTP responses/sec)	Good

Table 4: Results of release 5.2 stress tests

GE name	Reliability (errors rate)	Performance in stress condition (num requests, response time)	Stability (crashes)
IDAS	Very Good	Very Good (140 updates /second, 200 threads)	Very Good

IoT Broker	Good for two out of three APIs tested (0% for UpdateContext and QueryContext; 44% forSubscribeContext	NOT GOOD: With 16 concurrent users (threads) which sent 3.241 requests in 2' with AV RT < 0,5 s. Above that threshold, AV RT is around 4 s. Results on the average are not yet satisfactory, even after excluding, where viable, the bottleneck represented by IoTDiscovery implementation (NEConfman).	No crash (but many exceptions found in the SubscirbeContext test log file)
AuthZForce	VERY GOOD 0%	VERY GOOD: AV RT around 11 ms; load average value around 4376 requests per second	No crash
KeyRock	Very Good (0 % Error)	AVERAGE authorisation RT registered a much better average value (164 ms and 211 ms) than authentication RT which could register values <1 s only with a number of threads lower than 20; in fact over this threshold, response times increased significantly up to almost 4 s. Authorisation max load with AV RT<1s= 232 requests/s Authentication max load with AV RT<1s=23,30 requests/s No crash	Very Good (0 % Error)
Wilma	Very Good (0 % Error)	VERY GOOD: a low average response time (219 ms in the first test and 83 ms in the second one) allowed the GEri to guarantee, on average, up to 839 requests/s.)	No crash

Table 5: Results of release 5.3 stress tests

3.5 External assessment

In order to ensure the quality itself of the carried out methods and tests in non-functional testing to assess the GEs quality, an external expert was in charge of assessing them to state the validity of the performed work.

The main statement of the first report was "*In general, the process for non-functional testing is adequate and the preliminary results obtained by the non-functional tests are satisfactory*". However, some recommendations were provided to be considered:

- *The baseline (minimal expected load) and targets (maximal load reasonably to be expected) for the non-functional testing should be defined and justified before the actual testing takes place*.

These values were requested to GE owners and most of them claimed that they were not aware of such values. In those cases where the GE owner provided the reference values, they were used for establishing the limit of the GE. When values were not available, reference values were obtained from the accelerators usage of GEs.

- *Potential bottlenecks which could occur in the application of (combinations of) GEris should be better analysed*.

  Initially the task wanted to isolate the GEs in order to know the properties and values of each GE, but combination of GEs has been considered and two bundles have been tested: the one formed by Orion, IDAS and Cygnus; and the one composed of Wilma, KeyRock and AuthZForce.

- *Testing results should be better interpreted, and alternative routes and variants which may occur as a result of non-functional testing should be explicated*.

  An effort to homogenise the tests results has been done. The two partners involved in the task have worked on common templates for reports and agreed on common criteria and measures to align the testing process across all GEs. It has been also making an effort in explaining the rationale behind the obtained results, especially to allow GE owners to understand the reason of got values and how they could improve them.

- *In mid-term, non-functional testing should be linked to other software-development activities*.

  This is a nice to have feature that in the context of present project has been started although not completed. It is planned to enhance and fulfil this objective in near future. Anyway, it is not clear how much automation and integration of stress testing with software development process will be able to be reached. The main concern is the variety of testing scenarios and environments are required for each GE, as they are from diverse type of application and way of usage.

Full report can be consulted at Non-Functional Assessment Report.

4 Labelling model and application

Once the GEs are tested, a way to make evident the result of the test was thought as needed. A model of labels to state the quality of each GE by using a graphical visualization was decided. This section explains the selected approach, the method used for calculating the labels and the application of the model to ten selected GEs.

4.1 Motivation and approach

You cannot control what you cannot measure" is a well-known old motto by Tom Demarco [3]. But you cannot measure what you are not able to properly define and you cannot define what you do not properly know. Thus, the first step is properly to know and define the entities of interests (EoI) for our analysis. Second, measure and establish a measurement protocol for reducing and minimizing the probability of a bad measurement. Third, to store historical data for determining trends, useful to refine forecasts and estimations and, in our case, better determine the effort needed to use the 'building blocks' (GE – Generic Enablers) from this project. Each GE needs to have a 'value rating' expressing its goodness, hybridizing quantitative (*measurement*) and qualitative (*evaluation*) viewpoints into a unique, consolidated view: this involves also non-technical stakeholders by proper (visual) communication mechanisms, possibly with analogies from the common daily life: the simpler, the better. An example is therefore to use the 'labeling' concept, as well as done for the consumption of energy. A rating is not a measurement, but the result of an evaluation. In this case better to use an even scale in order to avoid choosing the central value in such distribution. 'Adopt and adapting' such concept can reduce in the target users their possible psychological resistance to change and easily being accepted and understood. That's why it has been set up the seven (7) values (from A+++ to D), as well as in the energy ratings for home appliances also for rating GE's.

Figure 5: EU labelling system

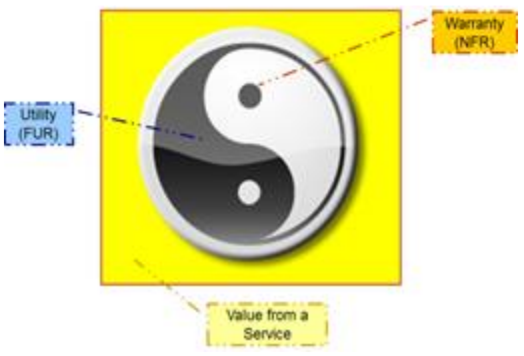
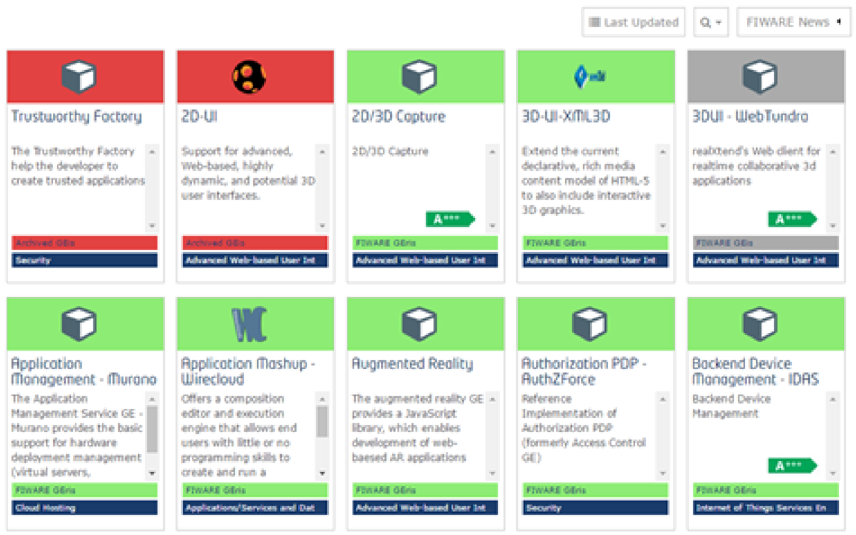


Figure 6: A (service) value representation

Such label will express the 'value' of such component, where such term – as well as in the Service Management domain (e.g. in ITIL, ISO 20000, etc.) – means the logical summation of '*utility*' (what a 'service' should do, according to the customer viewpoint) and '*warranty*' (how and how much a 'service' should operate, according to the users' viewpoint). Only putting together at least such two views, a provider can bring back a 'quality' product/service. In the system/software engineering domain, 'utility' matches with *FURs* (Functional User Requirements) and 'warranty' with *NFRs* (Non-Functional

Requirements). Referring to the standard ISO glossary[2], a product is the 'component' for delivering a service and – from a provider perspective, as the FIWARE consortium is – the product 'is' the 'service' for the European Commission. Here a visual example of such complementarity between the two aspects:

A well-known and logical best practice is to 'adopt and adapt' a well-known concept from other (physical) domains and (personal) experiences reducing the possible resistance to change and be easily accept by the target users. Thus, looking to the FIWARE GE Catalogue (http://catalogue.fiware.org/enablers), here it follows a preview about how it shall appear.



Trustworthy Factory
The Trustworthy Factory help the developer to create trusted applications
Archived GEis
Security

2D-UI
Support for advanced, Web-based, highly dynamic, and potential 3D user interfaces.
Archived GEis
Advanced Web-based User Int

2D/3D Capture
2D/3D Capture
A***
FIWARE GEris
Advanced Web-based User Int

3D-UI-XML3D
Extend the current declarative, rich media content model of HTML-5 to also include interactive 3D graphics.
FIWARE GEris
Advanced Web-based User Int

3DUI - WebTundra
realXtend's Web client for realtime collaborative 3d applications
A***
FIWARE GEis
Advanced Web-based User Int

Application Management - Murano
The Application Management Service GE - Murano provides the basic support for hardware deployment management (virtual servers,
FIWARE GEris
Cloud Hosting

Application Mashup - Wirecloud
Offers a composition editor and execution engine that allows end users with little or no programming skills to create and run a
FIWARE GEris
Applications/Services and Dat

Augmented Reality
The augmented reality GE provides a JavaScript library, which enables development of web-baesed AR applications
FIWARE GEris
Advanced Web-based User Int

Authorization PDP - AuthZForce
Reference Implementation of Authorization PDP (formerly Access Control GE)
FIWARE GEris
Security

Backend Device Management - IDAS
Backend Device Management
A***
FIWARE GEris
Internet of Things Services En

Figure 7: A preview of GE quality labelling in FIWARE Catalogue

Thus, the hybrid measurement ('labeling') for GEs matching functional and non-functional attributes will follow the consolidated assessment criteria in process management in Maturity & Capability Models (MCM), that's an ordinal 4-scale rating (**N/P/L/F**; Not/Partially/Largely/Fully achieved). You cannot achieve the next level if the previous level is not (at least) a L/F evaluation [3]. Using a grid-like approach and a set of criteria for determining the overall quality of a GE, considering both functionalities (*utility*) and non-functionalities (*warranty*), we can determine a three-dimensional '(GE) labeling cube', where each GE shall be described by a set of attributes (as described in the following sections). The criteria for evaluating a product shall arise from the ISO/IEC 25010:2011 quality model applicable both to products and services and for each criteria (attribute) one or more measures shall be applied, obtaining a hierarchy with three levels.

[2] www.computer.org/sevocab

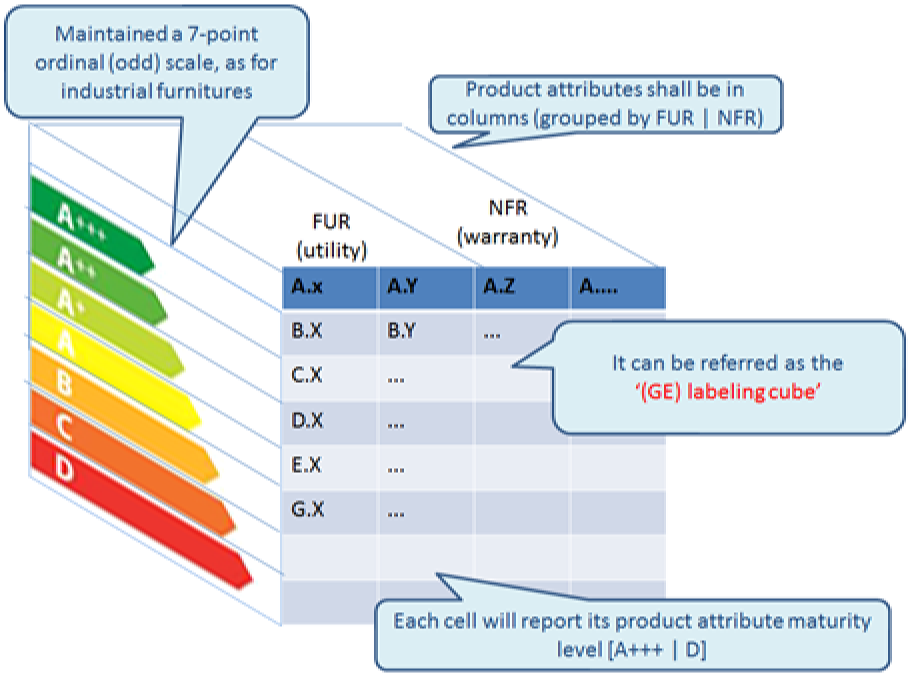

Maintained a 7-point ordinal (odd) scale, as for industrial furnitures
Product attributes shall be in columns (grouped by FUR | NFR)
FUR (utility)
NFR (warranty)
A+++
A++
A+
A
B
C
D
A.x
A.Y
A.Z
A....
B.X
B.Y
...
C.X
D.X
E.X
G.X
It can be referred as the '(GE) labeling cube'
Each cell will report its product attribute maturity level [A+++ | D]

Figure 8: The GE labelling cube

In this way each GE shall have few possible product attributes describing it, using a 7-point ordinal scale as the final rating (labeling), from 'A+++' to 'D'. 'Adopting and adapting' such approach to rate process (aggregating NPLF values for determining Capability Levels) to GE's (products), each defined measure (described in the next sections) will establish thresholds for determining the quality levels and the final overall GE value for determining its label will be average level achieved by ALL the evaluation criteria adopted, as in the following figure.

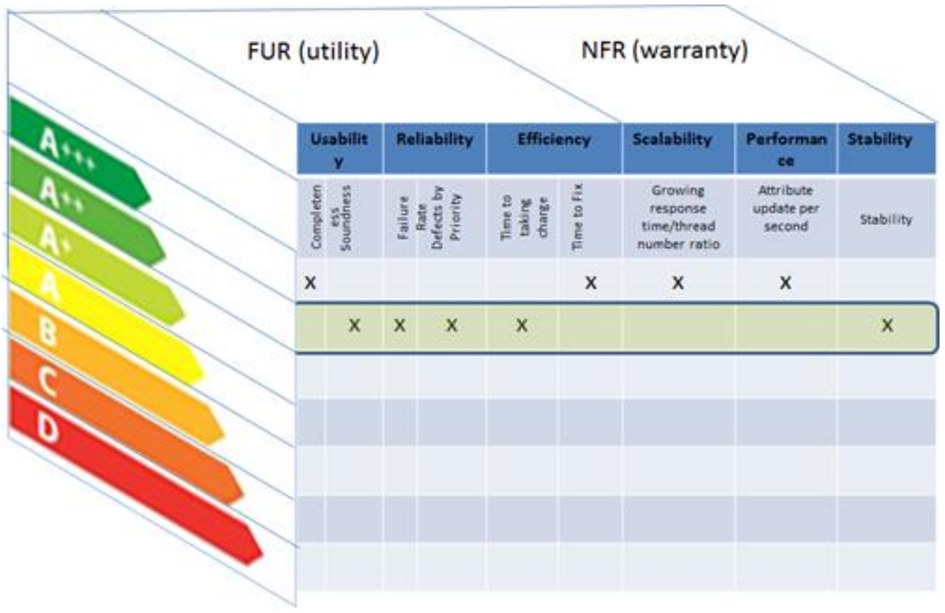


Figure 9: An example of rating (detailed) results

The overall rating for such GE would be a Level 'A++'; (2) the detailed result from the GE Labeling cube will be visualized clicking on the label value from the FIWARE GE catalogue, allowing interested stakeholders to that specific GE to properly understand where such component could be reworked and/or eventually improved first. That's what it is needed to provide them.

See Section 4.6 for the results obtained for the set of GE's.

4.2 Labelling model representation

Once established the labels (detailed and overall) for all GEs, the model must be represented and visualized somehow the obtained results can be shown to the GE's users in the Catalogue and automatically updated as soon as new measurements are taking place.

By proposed design, labels should not be static images, but an output of a labeling service. In the backend, measurement data as described in following sections is aggregated and by applying a set of business rules the service returns a label bound to an Enabler. This service can be contextually called from the Catalogue. While the automation of labels visualization is completed, a static visualization of the obtained labels so far is shown in the Catalogue. A set of icons have been generated for representing each of labels value:

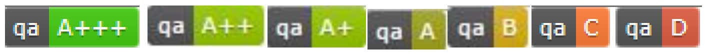
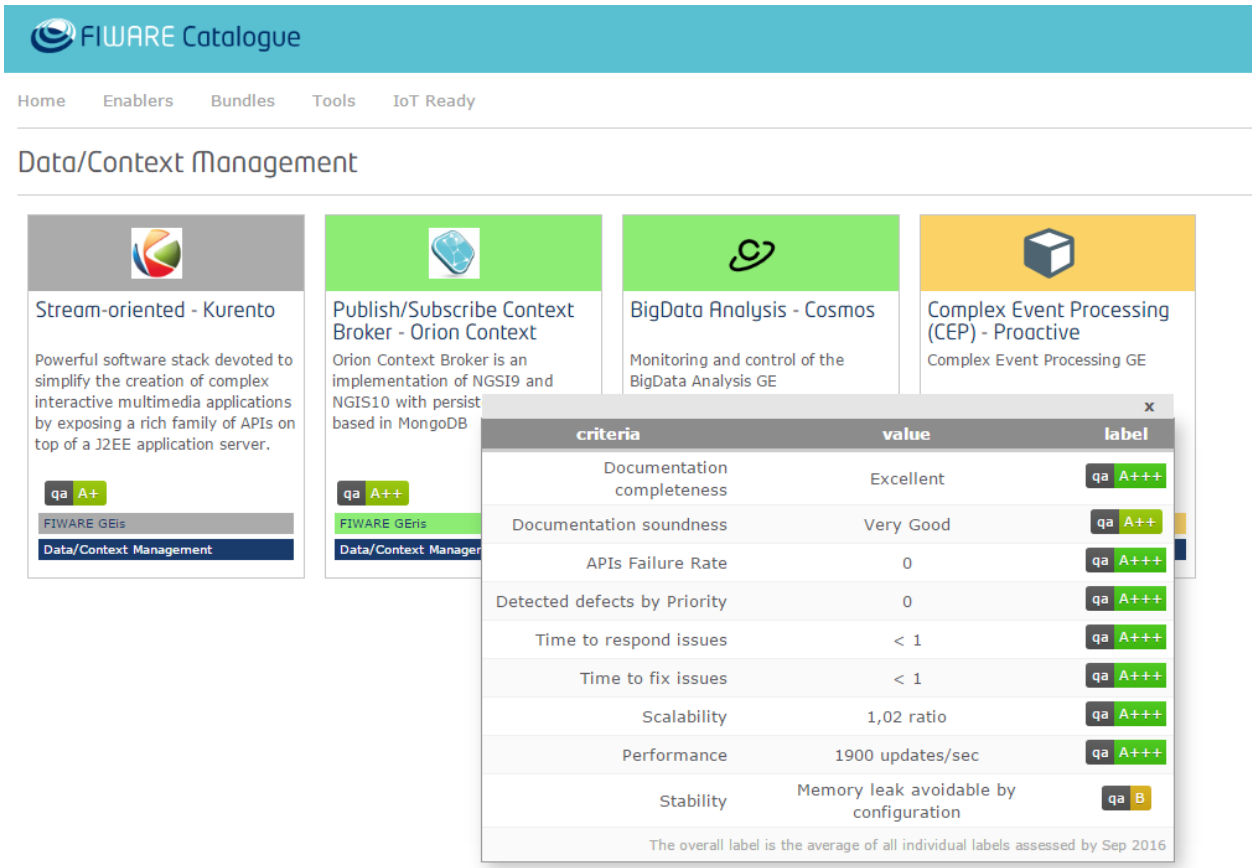


Figure 10: Labels icons

Then, the corresponding overall label has been embedded into corresponding GE box in the Catalogue, as for example can be seen below for Orion. By clicking on the label icon, a pop up window shows the detailed labels along the obtained values and meaning of each evaluated criteria.



Data/Context Management

Stream-oriented - Kurento

Powerful software stack devoted to simplify the creation of complex interactive multimedia applications by exposing a rich family of APIs on top of a J2EE application server.

qa A+

FIWARE GEIs

Data/Context Management

Publish/Subscribe Context Broker - Orion Context

Orion Context Broker is an implementation of NGSI9 and NGIS10 with persist based in MongoDB

qa A++

FIWARE GEris

Data/Context Manager

BigData Analysis - Cosmos

Monitoring and control of the BigData Analysis GE

Complex Event Processing (CEP) - Proactive

Complex Event Processing GE

criteria	value	label
Documentation completeness	Excellent	qa A+++
Documentation soundness	Very Good	qa A++
APIs Failure Rate	0	qa A+++
Detected defects by Priority	0	qa A+++
Time to respond issues	< 1	qa A+++
Time to fix issues	< 1	qa A+++
Scalability	1,02 ratio	qa A+++
Performance	1900 updates/sec	qa A+++
Stability	Memory leak avoidable by configuration	qa B

The overall label is the average of all individual labels assessed by Sep 2016

Figure 11: Visualization of Orion labels in FIWARE Catalogue

4.3 Labelling functional aspects

In order to label GEs according to *functional* attributes, a table as the following one was filled, with an evaluation of three main product attributes: Usability, Reliability and Efficiency. Each one will be split in a set of related measures and the average of them would be considered.

The proposed measures were introduced in the following table and later discussed in detail:

Prod. Attribute	Derived Measure	Base Measures	Formula
Usability	Completeness	Nominal value based on document verification	
	Soundness	Nominal value based on document verification	
Reliability	Failure Rate	Total number of Test failed (TF) Total number of Test cases executed	TF/TE

		(TE)	
	Defects by Priority	Total number of Blocking/Critical level Defects (DB) Total number of Major level Defects (DM) Total number of Minor/Trivial level Defects (DT) Total number of Defects (D)	(DB*w1+DM*w2+ DT*w3)/D <u>Note</u>: *Defect Blocking/Critical weight (w1=3)* *Defect Major weight (w2=2)* *Defect Minor/Trivial weight (w3=1)*
Efficiency	Time to Taking charge	Date of taking charge of each defect (TD) Date open for each defect (OD) Total number of Closed defects (D)	∑(TD-OD)/D
	Time to Fix	Date of fix for each defect (FD) Date of taking charge of each defect (TD) Total number of Closed defects (D)	∑(FD-TD)/D

Table 6: Calculation criteria for functional labels

1. Usability measures

This set of measures evaluates the ease of use of documentation, therefore the ability to find “the right information at the right time”, verifying that a document is complete, findable and consistent.

1.1. Completeness measure

This is a nominal measure based on the documentation availability and its completeness to ensure the document usability.

The analyzed items, sorted by relevance, are:

- Open Specification
- Installation guide
- Docker for installation
- Course or Tutorial in Academy site

Completeness			
Label	**Value**	**Base Measures**	**Formula**
A+++	Excellent	Nominal value based on document verification	Each expected document is available. The information are exhaustive, easily accessible and easy to use. There are examples, comments or other utilities that improve the reading/comprehension.

A++	Very good		Each expected document is available. The information are exhaustive, easily accessible and easy to use.
A+	Good		Each expected document is available. The information are fully exhaustive.
A	Fair		Each expected document is available and enough exhaustive
B	Poor		Each expected document is available but the information are not always exhaustive, easily accessible and easy to use.
C	Very poor		Some documents are missing. Those available are not always exhaustive, easily accessible and easy to use.
D	Bad		All or many documents are missing

Table 7: Measurement criteria for Completeness evaluation

1.2. Soundness measure

This is a nominal measure based on the documentation soundness to ensure the correctness of APIs usability.

The analyzed items (sorted by relevance) are:

- Programmer's Guide
- Software Package
- APIs Specification

Soundness			
Label	**Value**	**Base Measures**	**Formula**
A+++	Excellent	Nominal value based on document verification	The information is exhaustive, easily accessible and easy to use. There are examples, comments or other utilities that improve the reading/comprehension.
A++	Very good		The information are exhaustive, easily accessible and easy to use.
A+	Good		The information are fully exhaustive.
A	Fair		The information is enough exhaustive
B	Poor		The information are not always exhaustive, easily accessible and easy to use.

C	Very poor		Some information are missing. Those available are not always exhaustive, easily accessible and easy to use.
D	Bad		Core information are missing

Table 8: Measurement criteria for Soundness evaluation

2. Reliability measures

This second set of measures is about the capability of a GE to be reliable as much as possible in order to maximize its MTBF (Mean Time Between Failures). Even if this would be a product attributes in ISO/IEC 25010:2011, (thus as a non-functional attributes), here it should be seen as a set of attributes (and related measures) for showing how much it should be fixed for obtaining a proper 'uptime' for the observed GE.

2.1. Failure Rate measure

It determines the % of Failures in the executed test cases.
This measure is calculated from data gathered by Functional Test Reports.

Failure Rate			
Label	**Value**	**Base Measures**	**Formula**
A+++	< 0,05	Total number of Test failed (TF) Total number of Test cases executed (TE)	TF/TE
A++	0,05 - 0,16		
A+	0,17 - 0,27		
A	0,28 - 0,38		
B	0,39 - 0,49		
C	0,50 - 0,6		
D	> 0,6		

Table 9: Measurement criteria for Failure Rate evaluation

2.2. Defects by Priority metric

It determines the number of defects based on "Blocking/Critical" - "Major" – "Minor/Trivial" Priority (Service Desk's viewpoint)
This measure is calculated from data gathered by a Jira installation (https://www.atlassian.com/software/jira)

Priority	Description

Blocking/Critical	This has to be fixed immediately. This generally occurs in cases when an entire functionality is blocked and no testing can proceed as a result of this or, in certain other cases, if there are significant memory leaks
Major	Normally when a feature is not usable as it's supposed to be, due to a program defect, or that a new code has to be written or sometimes even because some environmental problem has to be handled through the code
Minor/Trivial	A defect with low priority indicates that there is definitely an issue, but it doesn't have to be fixed to match the "exit" criteria. However this must be fixed before the delivery. Sometimes it could be even a cosmetic error.

Defects by Priority			
Label	**Value**	**Base Measures**	**Formula**
A+++	< 1,5	Total number of Blocking/Critical level Defects (DB) Total number of Major level Defects (DM) Total number of Minor/Trivial level Defects (DT) Total number of Defects (D)	(DB*w1+DM*w2+DT*w3)/D Note: Defect Blocking/Critical weight (w1=3) Defect Major weight (w2=2) Defect Minor/Trivial weight (w3=1)
A++	1,5 - 1,7		
A+	1,71 - 1,9		
A	1,91 - 2,1		
B	2,11 - 2,3		
C	2,31 - 2,5		
D	> 2,5		

Table 10: Measurement criteria for Defects by Priority evaluation

3. Efficiency Metrics

This third set of measures is about the capability of a Service Desk (SD) to solve incidents related to GEs and manage them for maximizing the customer/user satisfaction.

3.1. Time to Taking charge measure

It determines the number of average working days valuated from the opening date to the taking charge date of the defect.

This measure is calculated from data gathered by a Jira installation (https://www.atlassian.com/software/jira)

Time to Taking charge			
Label	**Value**	**Base Measures**	**Formula**
A+++	< 1	Date of taking charge of each	∑(TD-OD)/D

A++	1 - 3,7	defect (TD) Date open for each defect (OD) Total number of Closed defects (D)	
A+	3,8 - 6,5		
A	6,6 - 9,3		
B	9,4 - 12,1		
C	12,2 - 15		
D	> 15		

Table 11: Measurement criteria for Time to Taking evaluation

3.2. Time to Fix measure

It determines the number of average working days valuated from the taking charge date to the fix date of the defect.

This measure is calculated from data gathered by a Jira installation (https://www.atlassian.com/software/jira)

Time to Fix			
Label	**Value**	**Base Measures**	**Formula**
A+++	< 1	Date of fix for each defect (FD) Date of taking charge of each defect (TD) Total number of Closed defects (D)	∑(FD-TD)/D
A++	1 - 4,7		
A+	4,8 - 8,5		
A	8,6 - 12,3		
B	12,4 - 16,1		
C	16,2 - 20		
D	> 20		

Table 12: Measurement criteria for Time to Fix evaluation

The GEs have been labelled according to the functional measures and for each one are reported the measures. The GEs labelled are the same labelled for the non-functional aspects. See section 4.6 for details.

4.5 Labelling non-functional aspects

In order to label GEs according to non-functional attributes, a table as below was filled, with an evaluation of three measures: Scalability, Stability and Performance. In the next pages, the way to fill these categories is explained.

1. Scalability measure

In this section, the behavior of the GEri when the load increased was studied.
In order to assess the Stability, we have to choose a fixed interval where the system has already reached the maximum requests per second data, and the system is not failing yet. For that fixed time interval, we calculate the ratio between final/starting response time and the ratio between final/starting thread number. Then, we calculate growing response time ratio / growing thread number ratio and check the result in the next table:

Growing Response Time ratio/ Growing thread number ratio	
Label	**Value**
A+++	< 1.05
A++	1.22 - 1.05
A+	1.42 - 1.21
B	1.74 - 1.43
C	2.15 - 1.75
D	2.6 - 2.16
E	> 2.6

Table 13: Value ranges for scalability evaluation

2. Stability measure

This is the evolution of the system along the time. To label the stability, the results in a stability scenario have to be checked. Then, it would be labeled depending on the existence of leaks:

Stability[rz3] [CLH4]	
Label	**Value**
A+++	Nor CPU nor memory increases in the whole test (when the load is stable)
A++	Memory usage is a little higher at the end of the test than at the beginning (probably due to data generated).

A+	Memory and CPU usages are a little higher at the end of the test
A	Memory and/or CPU are significantly higher at the end of the test, but doesn't seem to exist a leak
B	Memory leak avoidable with configuration or load limitation
C	Memory leak not avoidable. The system crashes after a few hours.
D	High leak. System crashes before half an hour

Table 14: Value ranges for stability evaluation

3. Performance measure

To evaluate performance, the maximum request handling per second would be considered. Because the requests of different GE's can have different complexity, a study of the functionality must be done before. The most common functionalities would be considered in the next points.

3.1 GEs with NGSI attribute updates

These are the GEs which are capable of create/update NGSI entities and attributes. In this category is included, for example, the Context Broker GE.

- Update attribute measuring

For this measure, is needed to find the most production point at first (the first point where the maximum responses per second rate is reached) for request that updates attributes. Then, at this point, response times and responses per second have to be measured.

- Updated attributes per second

In order to do an equitable comparison, it has to be considered the number of the attributes which are updated in each request. The reason is that the processing effort is higher if is needed to do more update transactions in database, and some other operations. If this is a random number (for example, each request updates a random number of attributes, between 1 and 3), then the average number can be used (in this example, two attribute updates would be considered). When that number is found, it has to get the number of attribute updates per second. For example, if in the maximum production number, the responses per second rate is 200, and (as seen before), the average attributes updates per requests is 2, then the number of attribute updates per request is 400. In our example, it would be labelled with a 'B' according to the table below.

Updated attributes/second	
Label	**Value**
A+++	> 591
A++	541 – 590

A+	481 – 540
A	391 – 480
B	271 – 390
C	141 – 270
D	<140

Table 15: Value ranges for performance (updated att/sec) evaluation

3.2 GEs with event receiving

These are the GEs which are capable to handle events, for example, the Proton GE. For evaluating them, it is consider their performance in responding to HTTP requests containing events.

- Event receiving measuring

As before, it is needed to find the most production point at first (the first point where the maximum responses per second rate is reached) for requests that update attributes. Then, at this point, response times and responses per second have to be measured.

- Events per second

Despite the updated attributes case, usually only an event for each HTTP request is sent. The next table shows how to label the event processing.

Events/second	
Label	**Value**
A+++	> 492
A++	451 – 492
A+	401 – 491
A	313 – 400
B	195 – 312
C	94 – 192
D	< 94

Table 16: Value ranges for performance (events/sec) evaluation

Again using our previous example, it would be labelled with a ‘B’ according to the table.

3.3. WebRTC based GEs

GEs based on WebRTC protocol fits into this category. For these GEs, standard metrics for HTTP protocol based GEs cannot be used. Instead of that, following metrics are needed.

- Bit Rate metric

In this case, the most meaning value in order to assess the performance is the Total Bit Rate that the system can serve. For that, it is needed to multiply the Bit Rate by the number of users.

- Bit rate and users number

The next table shows how to label in function of the Bit Rate and users number.

(Bit Rate * users number) ratio	
Label	**Value**
A+++	> 4'5 Mbps
A++	2'01 - 4'5 Mbps
A+	1'01 - 2 Mbps
A	0'501 Mbps - 1 Mbps
B	251 Kbps - 500 Kbps
C	100 Kbps - 250 Kbps
D	< 100 Kbps

Table 17: Value ranges for performance (bit rate/user) evaluation

In our example, it would be labelled with an 'A++' according to the table.

3.4. GEs handling authentication/authorisation request

Some GEs provide the most used features handling HTTP requests that imply operations such as the user authentication and/or authorisation. These operations can be directly invoked by a hypothetical IdM GE as well as indirectly by another GE that strictly depends on the first one. In order to evaluate both operations, their performances in response to HTTP requests have to be taken separately and then merged in a unique value through the process described below.

- Completed request measuring

Firstly, starting from the output of the non-functional tests, the already measured response times and the number of responses per second have to be taken into account in order to find the most production point (the first point where the maximum responses per second rate is reached) for completed transactions; secondly, since authorization and authentication operations have different costs, they are weighted in order to make them comparable to each other and finally their average is calculated. The resulting value is then checked with the table provided below in order to find the range which fits with this value and thus the corresponding label can be identified. The operation cost to be taken into account, as suggested for the case of Generic GE, is meant in terms of DB operations -of any type- originated by each type of request. For example, on the one hand, an “authentication” request, in order to be fulfilled, involves 30 DB operations; on the other hand, the corresponding “authorization” involves only 2 DB operations. The ratio between the potential cost of the two operations is thus equal to 30/2=15. A half point of weight is then assigned for each unit of the resulting ratio (0,5 X 15) and the result used as multiplying factor of the “authentication” most production point calculated earlier. Finally, the two values of the most production point are averaged and the resulting value compared with the ranges provided by the following table.

Authentication/Authorization requests/second	
Label	**Value**
A+++	> 591
A++	541 – 590
A+	481 – 540
A	391 – 480
B	271 – 390
C	141 – 270
D	<140

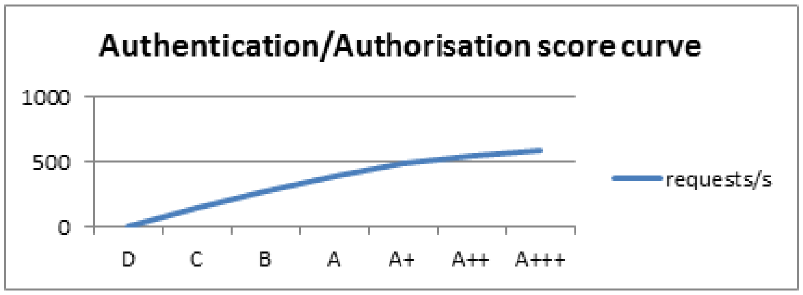


Table 18: Value ranges for performance (auth reqs/sec) evaluation

3.5. Generic GE

These are the GEs which not fit into the other categories. For labeling other GEs which do not fit in the ones mentioned before, a ponderation should be done, comparing the functionality of the GE with one reference functionality (for example, attribute updates). In order to do this, it must be considered:

- Number of operations in the database per request
- Connections with other systems
- Functionality complexity

For example, an NGSI attribute update makes 2 database operations (1 select and 1 update queries). If a functionality makes 4 database operations, the functionality complexity appears to be similar, and it makes no connections with other systems, then it would be considered like a 2 NGSI attributes update query.

- Generic GEs not involving DB operations

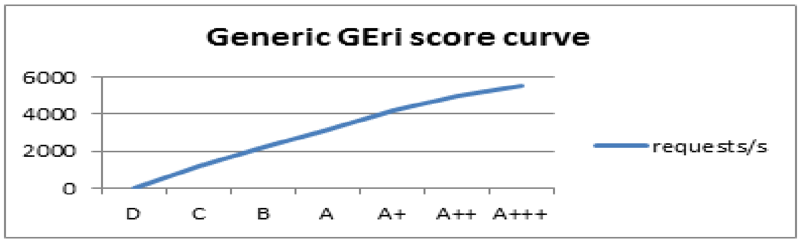
This category specializes the aforementioned “Generic GE” category to those components that on the one hand do not fit with any dedicated category, but on the other hand they have in common the characteristic of not using any DB system. Therefore this type of GE could be tested in a completely isolated way, namely with no other systems connected.

- Completed request measuring

Firstly, starting from the output of the non-functional tests, the already measured response times and the number of responses per second have to be taken into account in order to find the most production point (the first point where the maximum responses per second rate is reached) for completed transactions. This value is then directly compared with the table provided below in order to find the range which fits with it and thus the label to be assigned.

Requests/second	
Label	**Value**
A+++	> 7200
A++	6701 – 7200
A+	5901 – 6700
A	4701 – 5900
B	3201 – 4700
C	1600 – 3200
D	<1600

Table 19: Value ranges for performance (requests/sec) evaluation



3.6 Generic GEs applied to PEP Proxy GE

This category defines a range of values for classifying the results coming out from the test carried out on any implementation of the PEP Proxy GE.

- Completed request measuring

Likewise the originating category that this one specializes, firstly, the already measured response times and the number of responses per second have to be taken into account in order to find the most production point (the first point where the maximum responses per second rate is reached) for completed transactions. This value is then directly compared with the table provided below in order to find the range which fits with it and thus with the label to be assigned.

Requests/second	
Label	**Value**
A+++	> 1340
A++	1221 – 1340
A+	1061 – 1220
A	821 – 1060
B	561 – 820
C	281 – 560
D	<280

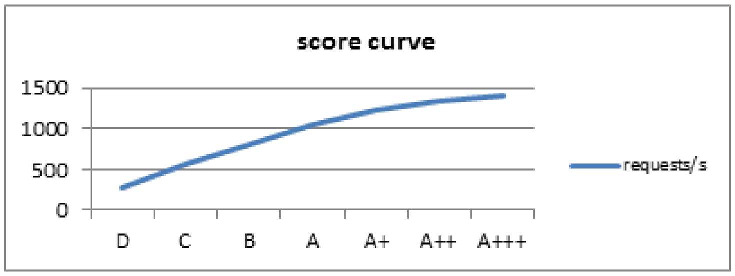


Table 20: Value ranges for performance (requests/sec) evaluation (PEP Proxy)

4.5.1. Benchmarking for adjusting the non-functional results

Due to the possible hardware differences between different GEs' performance tests, it is necessary to adjust the metrics depending on the results of a benchmark.

For the adjustment mentioned before, the benchmark "Phoronix Test Suite" has been used. Inside Phoronix, benchmark "pts/apache-1.6.1" was used. From this benchmark, a score is obtained that is needed to get a weighting rate. Annex 3 proposes the steps to get this score.

To get this, we take as reference the value "**16733.24**", which is the score obtained for a server reference. With this reference value "x", and the score "y" obtained by the benchmark, the ponderation rate "z" is obtained with the formula z = y/x. For example, for a score of 22013.47, the weighting rate would be 22013.47 /16733.24 = 1.315

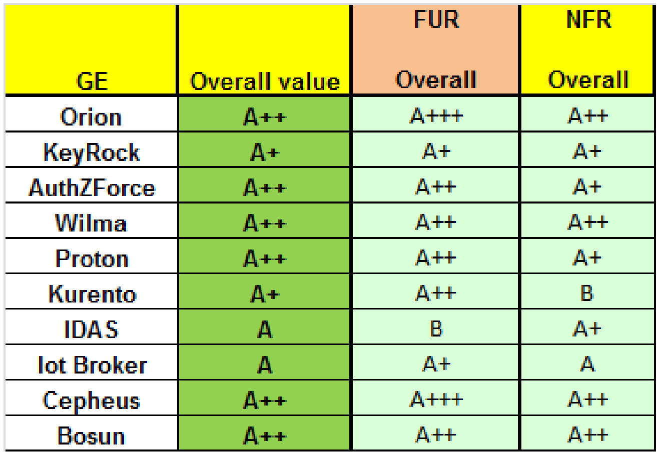
Once the weighted rate has been found, it is easy to get the weight for a measure to be multiplied by the value by the weighting rate when the measure is "lower is better", and subdividing when the measure is "higher is better".

For example, if the weighting rate is 1.315, and we have a performance value of 600 NGSI attribute request per second (higher is better), then we have to divide 600 by 1.315, which result is 456,27, labelled with a 'B'.

4.6 Labelling FIWARE GEs

In order to validate the labelling model, a set of GEs were selected to test the model and obtained their labels. The selected GEs (10 in total) have been tested both from functional and non-functional aspects, so they are a subset of the overall tested GEs, but enough for demonstrating whether the model is valid or it must be adjusted.

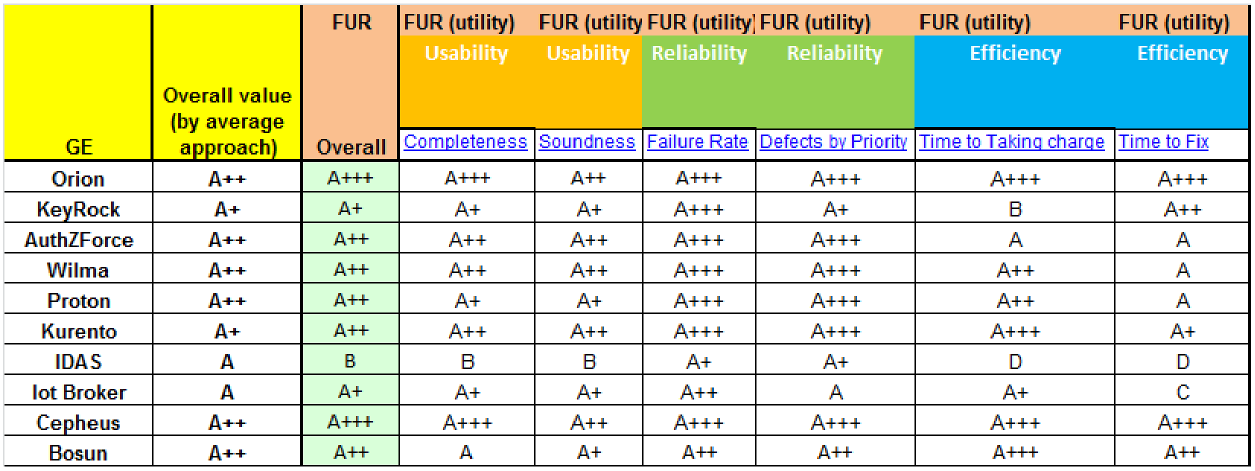
After gathering values for the defined FUR/NFR measures from all the observed GEs, here the summary:



GE	Overall value	FUR Overall	NFR Overall
Orion	A++	A+++	A++
KeyRock	A+	A+	A+
AuthZForce	A++	A++	A+
Wilma	A++	A++	A++
Proton	A++	A++	A+
Kurento	A+	A++	B
IDAS	A	B	A+
Iot Broker	A	A+	A
Cepheus	A++	A+++	A++
Bosun	A++	A++	A++

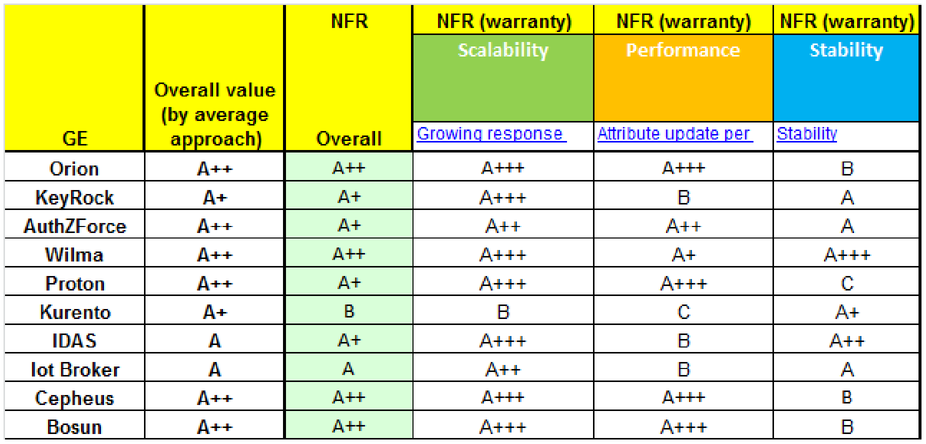
Figure 12: Consolidated labelling (overall values) for 10 observed GEs

For instance, looking at Orion, the overall value is 'A++', with a higher value from functional-related attributes (A+++) more than from its non-functional-related ones (A++). But in order to better understand which attribute could be the one on which improving the rating with a further implementation with priority, here in the following figures there is the detailed for the 'two sides of the story' (FUR and NFR).



GE	Overall value (by average approach)	FUR Overall	FUR (utility) Usability Completeness	FUR (utility Usability Soundness	FUR (utility Reliability Failure Rate	FUR (utility) Reliability Defects by Priority	FUR (utility) Efficiency Time to Taking charge	FUR (utility) Efficiency Time to Fix
Orion	A++	A+++	A+++	A++	A+++	A+++	A+++	A+++
KeyRock	A+	A+	A+	A+	A+++	A+	B	A++
AuthZForce	A++	A++	A++	A++	A+++	A+++	A	A
Wilma	A++	A++	A++	A++	A+++	A+++	A++	A
Proton	A++	A++	A+	A+	A+++	A+++	A++	A
Kurento	A+	A++	A++	A++	A+++	A+++	A+++	A+
IDAS	A	B	B	B	A+	A+	D	D
Iot Broker	A	A+	A+	A+	A++	A	A+	C
Cepheus	A++	A+++	A+++	A++	A+++	A+++	A+++	A+++
Bosun	A++	A++	A	A+	A++	A++	A+++	A++

Figure 13: Consolidated functional labelling (detailed values) for 10 observed GEs



		NFR	NFR (warranty)	NFR (warranty)	NFR (warranty)
			Scalability	Performance	Stability
GE	Overall value (by average approach)	Overall	Growing response	Attribute update per	Stability
Orion	A++	A++	A+++	A+++	B
KeyRock	A+	A+	A+++	B	A
AuthZForce	A++	A+	A++	A++	A
Wilma	A++	A++	A+++	A+	A+++
Proton	A++	A+	A+++	A+++	C
Kurento	A+	B	B	C	A+
IDAS	A	A+	A+++	B	A++
Iot Broker	A	A	A++	B	A
Cepheus	A++	A++	A+++	A+++	B
Bosun	A++	A++	A+++	A+++	B

Figure 14: Consolidated non-functional labelling (detailed values) for 10 observed GEs

The spreadsheet with the detailed labelling by GE and category is available at https://forge.fiware.org/docman/view.php/7/5716/Labelling_Results_Func_Measures_v08.xlsx.

5 Conclusions

This task was conceived with the goal to be an instrument to demonstrate and improve the FIWARE GEs quality in order to make them more reliable and trusty. Along the process, the task has become very useful for the GEs developers to understand better the behavior of their components and the limits they had, allowing them to correct defects and improve documentation, performance, to make them more stable or to provide better training courses.

The starting was not easy as there were many things to test, from different sources, having to test GEs from very different nature and technologies, and overcoming the initial reluctance of collaboration since the task was seen as an overhead and intrusive for some GE owners. But, slowly these drawbacks were overpassed and the testing process became fluent and continuous.

Now, after several phases in the process some overall conclusions can be stated. There exists a significant heterogeneity in the GEs quality, having more mature GEs and ready for market than others. The stated labels are proof of that, having diversity of quality levels. Although there is always a room for improvement in documentation and support for most of the GEs, both of them have improved significantly during last year of the task. It can be also stated significant improvements in performance from first iteration to the second and third one, due to the addressing of recommendations through the iterative testing reports by the GE responsible, which is also a demonstration of the value this activity can bring to FIWARE.

In near future, the main focus will be to enlarge number and type of tests and to automate the tests as much as possible, including them as part of the software development process. Also the visualization of the labels in the Catalogue will be updated automatically as soon as a new label was changed due to new value in tested criteria. In the meantime a set of guidelines have been created in order to be able to replicate all the conducted tests by anyone.

ANNEX 1: Non-functional (stress) testing report

A1.1 Non-functional testing reports (for version 5.1)

Orion stress testing report (release 5.1)
Bosun stress testing report (release 5.1)
Cepheus stress testing report (release 5.1)
IDAS stress testing report (release 5.1)
KeyRock stress testing report (release 5.1)
Kurento stress testing report (release 5.1)
Proton stress testing report (release 5.1)
Wilma stress testing report (release 5.1)
IoTBroker stress testing report (release 5.1)

A1.2 Non-functional testing reports (for version 5.2)

Bosun stress testing report (release 5.2)
Cepheus stress testing report (release 5.2)
KeyRock stress testing report (release 5.2)
Kurento stress testing report (release 5.2)
Orion stress testing report (release 5.2)
Wilma stress testing report (release 5.2)

A1.3 Non-functional testing reports (for version 5.3)

Bundle Orion-IDAS-Cygnus stress testing report
Bundle Wilma-KeyRock-AuthZForce stress testing report
IoTBroker stress testing report (release 5.3)
AuthZForce stress testing report (release 5.3)
IDAS stress testing report (release 5.3)
KeyRock stress testing report (release 5.3)
Orion stress testing report (release 5.3)
Orion scalability testing report (release 5.3)
Wilma stress testing report (release 5.3)
Proton stress testing report (release 5.3)

ANNEX 2: Functional testing report

The functional testing activity reports the results in different spreadsheets listed below:

1. The spreadsheet containing the Test Cases list and the Test execution information used and reported by the functional testing action can be downloaded here.

2. The spreadsheet containing the Academy Courses verifications and results can be downloaded here.