HORIZONS 2020 PROGRAMME

Research and Innovation Action – FIRE Initiative

Call Identifier:	H2020–ICT–2014–1
Project Number:	643943
Project Acronym:	FIESTA-IoT
Project Title:	Federated Interoperable Semantic IoT/cloud Testbeds and Applications

Certification Suite V1

Document Id:	FIESTA-IoT-D62-170215-Draft
File Name:	FIESTA-IoT-D62-170215-Draft.pdf
Document reference:	Deliverable 6.2
Version:	Draft
Editor:	Mengxuan Zhao
Organisation:	Easy Global Market
Date:	15 / 02 / 2017
Document type:	Other
Dissemination level:	PU

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Mengxuan Zhao Franck Le Gall	EGM	2016/11/01	TOC proposal
V02	Mengxuan Zhao Franck Le Gall	EGM	2016/11/28	Introduction, Certification level, Tool implementation
V03	Paul Grace	ITINNOV	2016/12/12	Model-based Interoperability Testing Tool – all content across sections
V04	Mengxuan Zhao Olivier Tosello	EGM	2016/12/20	Conclusion, implementation of the web portal
V051	Tiago Teixeira	UNPARALLEL	2017/02/03	Technical review
V052	JaeYoung Hwang	KETI	2017/02/04	Technical review
V053	Tarek Elsaleh	UNIS	2017/02/14	Quality review
V06	Mengxuan Zhao, Franck Le-Gall Paul Grace	EGM ITINNOV	2017/02/15	Final version for submission
V07	Martin Serrano	NUIG	2017/02/16	Format Update and Final Checks
V08	Martin Serrano	NUIG	2017/02/16	Circulated for Approval
Draft	Martin Serrano	NUIG	2017/02/20	EC Submitted

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ACRONYMS

API	Application Programming Interface
EaaS	Experimentation-as-a-Service
FIRE	Future Internet Research and Experimentation
IoT	Internet of Things
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
OWL	Ontology Web Language
RA	Reference architecture
RDF	Resource Description Framework
REST	Representational State Transfer
SDK	Software Development Kit
TPS	Testbed Provider Services
UI	User Interface
URI	Uniform Resource Identifier
WP	Work Package

1 INTRODUCTION

1.1 Reminder of the proposed certification program

From the previous work carried out in Work Package 6 together with other work packages in the Fiesta-IoT project [1], we were able to make a first proposal of the global market confidence program. This program aims at certifying and labelling the products or data not only in scope of the Fiesta-IoT project, but also for the IoT market in general. FIESTA-IoT will look at not only the testbeds providers, experimenters, researchers which could interact within the FIRE community, but more important will look at the overall market demand for IoT tests and interoperability. Therefore main objective of the certification and labelling program with best practices and tools is:

1. To assist researchers and experimenters to develop robust and interoperable software that underpins cross-technology IoT experimentation
2. To assist standard delegates to promote interoperability best practice supported by standards
3. Most **important**, to provide the marketplace with tools to check conformity to best practices and standards

The development of this program will be in two phases:

1. **Within the Fiesta-IoT project's timeframe**. In this phase, the program is designed mainly to satisfy the project's needs, for example, to help the project stakeholders to test their semantic interoperability regarding the requirements from the project. These in-house users will give their feedback based on their experience of using the program which will help the program to be improved.
2. **Beyond the Fiesta-IoT project**. In spite of the fact that the program is primarily intended for the Fiesta-IoT project in the first development phase, it is designed in a generic way that it can be easily adapted to the IoT market. In this phase, the achievement from the first phase will be adapted according to the common requirements from the IoT market. The main stakeholders in the program will be any platform or data provider who wish to test and certify their product against reference specifications or standards.

This program consists of four forms:

1. **Online testing and certification tools**. These are the tools that can be accessed through the certification web portal. Following an online certification workflow, user will be guided to use all or part of the tools to get a certificate.
2. **Consulting**. This service consists of providing expertise to help those who wish to make their product pass the certification tests and get a certification.
3. **Training**. This service consists of helping clients to attain the competence for making their product compliant to a specific specification to get a certificate.
4. **Solution design**. This service is an extension of Consulting in case that the client requires a ready-to-use solution for interoperability.

1.2 Certification suite components

The certification suite presented here consists of the first service that we provide in the global market certification program: the online testing and certification tools. Users can use this online certification suite to:

1. Test the semantic interoperability of their product regarding to a specific specification or standard. During the development of a product, the developer needs several tools to check the compliance of their product to a reference specification. If the product does not conform to the specification, some feedback reporting the reason of the unconformity is essential for the developer to improve the product towards the final conformity and interoperability. As this suite integrates several useful tools, it can be used as a testing tool suite.
2. Certify their product. A certification can promote the visibility and confidence of a product, hence the provider has the interest to get their product certified. The certification suite will guide the user to complete the certification process and get the certificate if all the criteria have been satisfied.

This certification suite is composed of the following components:

1. Certification web portal. Online tools are integrated and accessible from the web portal. Related information about the certification suite, such as: the guide to the certification process, and links to useful document such as Fiesta deliverables, are also available on the web portal. A user can review his historical data, such as test reports in the past, a list of all the certifications he/she gets from the current certification suite. Digital certificates are generated through this web portal based on the user's demand and the results of the evaluation tools against the product.
2. Scorecard. This is a score providing the maturity level of a product under test. A user can use this scorecard to evaluate his product by answering the questions. A score is given when the questionnaire is finished, and the product which is the subject of the scorecard is mapped to an interoperability maturity class according to the score.
3. Online tests. Addition to the self-assessment using Scorecard, a user can use the online tests provided as a web service to prove the product's interoperability regarding to the specifications that he/she claimed compliant. The result will add more credibility to the interoperability level of the product (because it is tested by a reputable 3rd party—namely FIESTA-IoT), thus it brings the product to an upper certification level.

These components will be depicted more in details in the following sections.

2 CERTIFICATION LEVEL

The current certification suite proposes two levels of certification. The first one is the self-assessment level. Users can gain certification by answering a questionnaire. The maturity level of interoperability is generated after the questionnaire answers are submitted. The second level of certification is achieved via the web service testing level. A certificate is generated according to the test results on the product using the web service testing tools available on the web portal.

2.1 Self-assessment certification level

A questionnaire was developed according to the target evaluation subject that have been identified within Fiesta-IoT. The objective of this questionnaire is to assess the interoperability level of the target product or data in terms of an overall score. More details about this questionnaire and about the interoperability level definition can be found in section 4.1.

2.2 Web service certification level

The self-assessment scorecard is considered as a preliminary certification level as it's based only on the response of the questionnaire. This means of certification needs to be complemented by other testing tools to increase its level of credibility. Thus, online testing services are provided to support the result obtained from the self-assessment scorecard.

At the time when this report is being written, two web services are considered and integrated into the web portal:

- Ontology validator[1]. This tool checks sample data that come from the product under certification and delivers a report of the conformity of the data regarding the declared reference ontology. Some of the reference ontologies are subject of several questions in the self-assessment scorecard.
- Model-based Interoperability testing tool [1]. This tool checks the interoperability of the product under certification regarding to the APIs defined in declared standards or reference specifications. It can also perform simple performance interoperability tests. These features can cover several questions in the "Interfaces and Services" section and in the "Quality Auditing Aspects" section.

The certification suite will not take the "all-or-nothing" approach. The generated certificate will cover all the evaluation and test results that user performed, and the omitted evaluation or tests will not figure on the certificate.

3 CERTIFICATION PROCESS

3.1 Overview

This section explains the workflow that a user should follow to get his/her product certified. **Figure 1** illustrates the steps in the certification process:

1. An introduction and guide is available for user to get familiar with the certification process and with the tools. The user should also log in or register himself/herself on the certification suite in this step in order to proceed the next steps.
2. The user is invited to fill in the self-assessment scorecard. A report is generated which contains the overall and per section scores, as well as the evaluated interoperability level of the product regarding the Fiesta-IoT Platform.
3. After the scorecard assessment, the user can choose to continue the certification process to achieve a higher certification level. The user can start with either the semantic data validation process, or the interoperability process with his product. These two processes are independent, as illustrated in **Figure 1**, user can come back to semantic data validation from interoperability tests. At the end of each process, a report containing the testing result is generated.
4. After getting a report of the self-assessment scorecard, the user can choose to generate a certificate whenever he/she wants. The certificate contains the results of the scorecard and of the tests already performed.

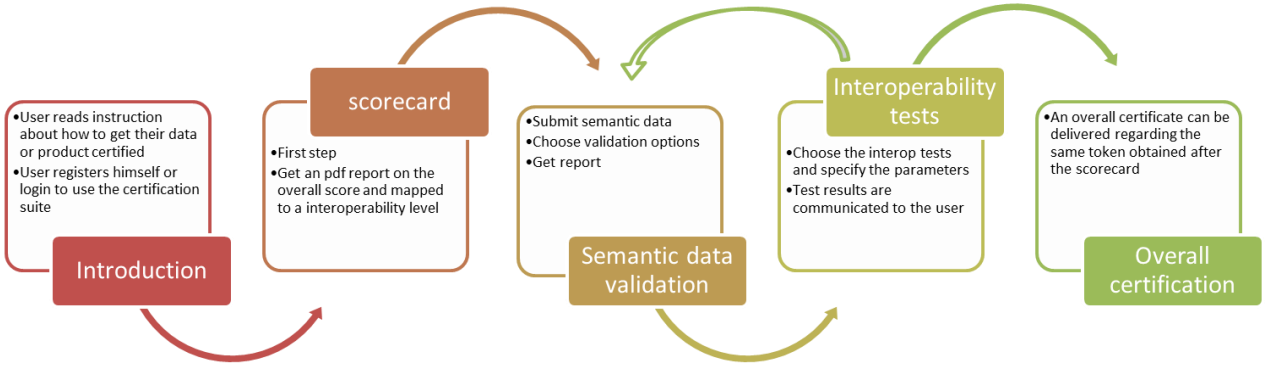


Figure 1 The Global Market Confidence Program certification workflow

3.2 Certification process scenario example

Assumption on the system under certification. The system prepares to join the Fiesta-IoT EaaS platform as an extension, which means data and resource provider. The owner of the system has the semantic data annotator developed and the TPS implemented. The specifications of the annotator and TPS are described in [2].

Certification process:

1. The testbed owner logs into the certification web portal to launch the process
2. The testbed owner answers the questionnaire according to his testbed's capabilities in order to evaluate the interoperability level of the testbed
3. The testbed owner submits a piece of semantic data of the testbed resources or observations that the annotator produced. The validation will be performed against the Fiesta-IoT ontology.
4. The testbed owner chooses the Fiesta-IoT TPS interoperability tests in the interoperability tool, and enter the endpoint IP address of the testbed under certification. The tool will perform the chosen tests automatically.
5. The testbed owner chooses to generate a certificate based on the scorecard and performed tests.

3.3 What need to be prepared for the certification process?

In principle, a certification process is designed in a way that users can use intuitively. However, some information about the product under certification needs to be communicated to the suite in order that the tools can perform the tests correctly. Here a "product" refers to any kind of system, like a device or a testbed, or just a dataset.

- For using the certification suite, you need an email address to register yourself, and this login information is needed for reviewing and retrieving the existing testing results and certificates
- For using the ontology validator testing tool, the semantic data to be validated should be provided, and the RDF serialization format needs to be communicated if the data is directly input in the textbox. (The format will be automatically detected using the suffix of the uploaded file containing the semantic data)
- For using the model-based interoperability tool, the user must input two pieces of information: i) the standard or API specification they wish to perform interoperability tests against (i.e. they select the set of tests they wish to carry out), ii) the endpoint of their product or testbed to be tested—this must be hosted and available via a publically accessible IP address. Such information allows the certification protocol to execute the correct tests on the remote product.

3.4 How to understand the delivered certificate?

The generated certificate is composed of two mains parts: 1) the resume which reports the overall results of the executed evaluation; 2) the reports of each of the evaluation performed.

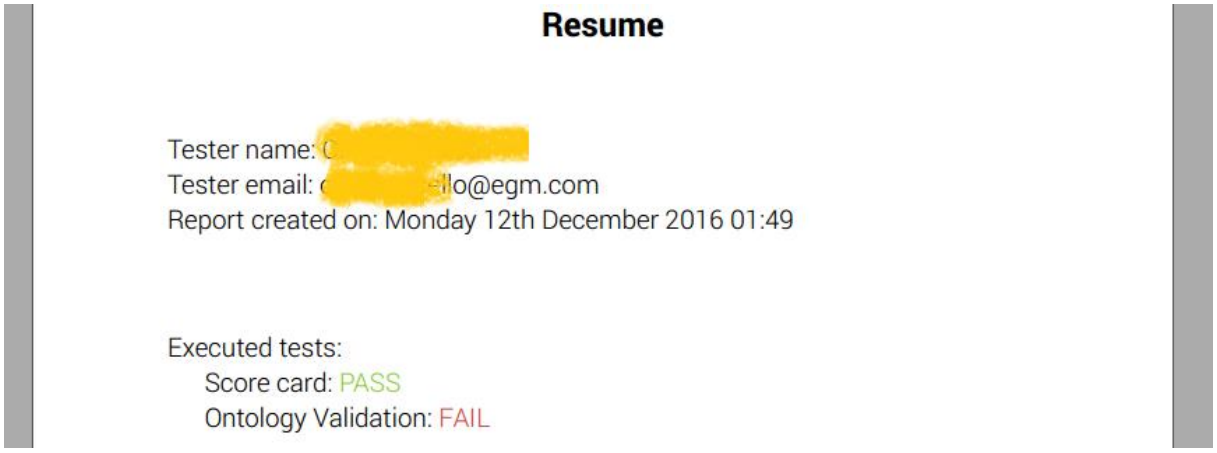


Figure 2 Generated certificate: Resume page

The resume page (

Figure 2) contains the basic information of the certificate: who performed the certification process and when the certificate is generated. It also recaps the overall results of the executed evaluation and tests. If the user needs more details of each tests, he/she can go to the second part of the certificate.

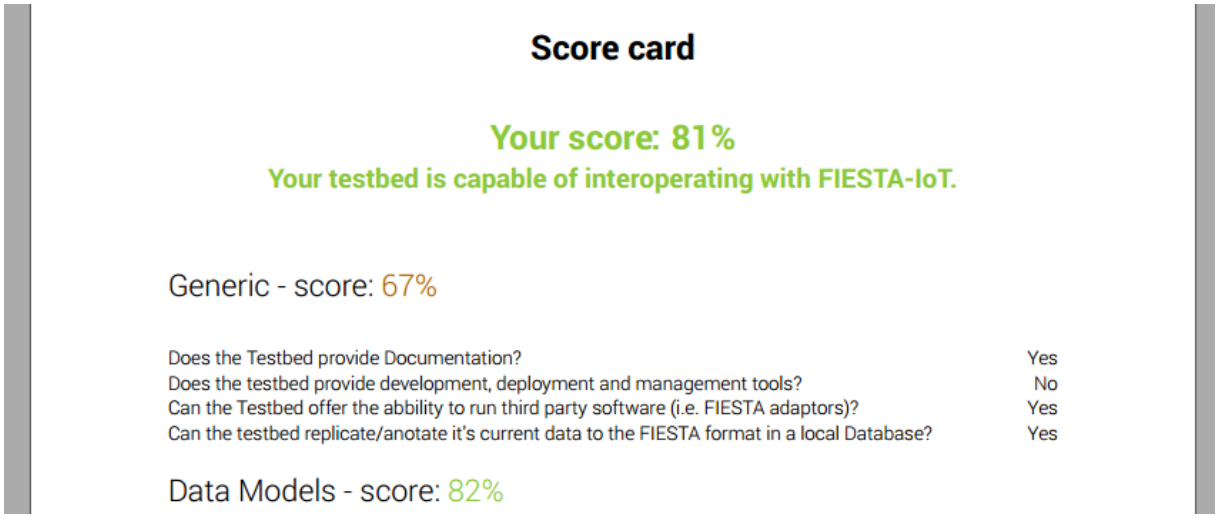


Figure 3 Generated certificate: Scorecard report

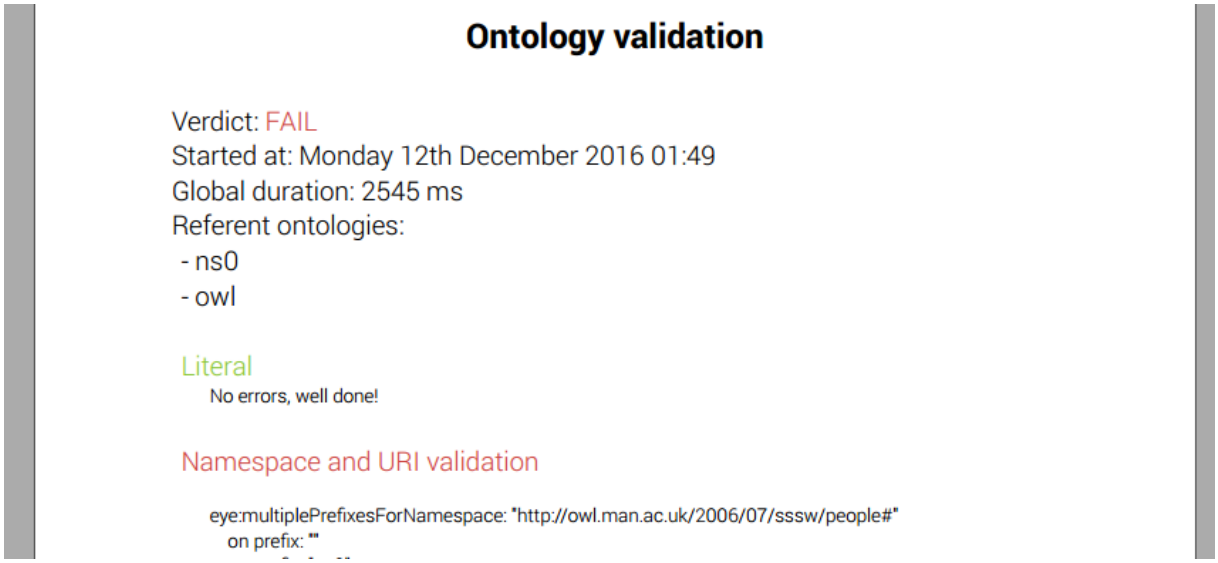


Figure 4 Generated certificate: Ontology validation report

The second part integrates the detailed reports of every tests. In the report of the scorecard (

Figure 3), the answer to every question is recorded and the score of each section is figured (for more details of the scorecard, please refer to section 4.1). In the report of the ontology validation (

Figure 4), the time when the test was performed, the duration of the validation, the reference ontologies used for the validation and the test results of each ontology validation type (more details about the ontology validation are available in section 4.2).

4 TOOL IMPLEMENTATION

4.1 Scorecard

The scorecard implemented on the certification web portal is based on the "scorecard for extensions" developed for the first open call evaluation which is available on the Fiesta-IoT Moodle Platform[1]. The following Table 1 recaps the content of the current scorecard. It has five sections regrouping the questions by theme. For each question, user can answer Yes or No. Answer Yes gives a score of 1 and No gives 0. The column "weight" indicates the coefficient to be multiplied to the score of the answer. The score is calculated as the percentage of the result score over the total score.

Table 1 Self-assessment scorecard design

	Items	Description	Weight
	Data Models		
1.1	SSN Ontology	Does the testbed supports the SSN ontology?	2
1.2	FIESTA Ontology	Does the testbed supports the FIESTA ontology?	2
1.3	IoT Lite Ontology	Does the testbed supports the IoT-lite ontology?	2
1.4	Other languages for data annotation	Does the testbed supports other language to annotate data such as SensorML, SWE?	1
1.5	Proprietary Format	Does the testbed support a proprietary language to represent the sensor data and/or sensor descriptions?	1
1.6	Data Extraction	Does the testbed provides the ability to extract data and/or sensor descriptions in a document format (i.e. CSV, Excel, XML, RDF, JSON, etc.)?	1
1.7	Graph Database	Does the testbed store its data and/or sensor descriptions in a Graph Database?	1
1.8	Document Database	Does the testbed store its data and/or sensor descriptions in a Document Database?	1
1.9	Relational Database	Does the testbed store its data and/or sensor descriptions in a Relational Database?	1
	Interfaces and Services		
2.1	SPARQL End Point	Does the Testbed offer a SPARQL (Graph DB) endpoint?	1
2.2	NGSI Interface	Does the Testbed offer an OMA Next Generation Services (NGSI) Interface?	1
2.3	OCCI Interface	Does the Testbed offer an Open Cloud Computing Interface (OCCI)?	1
2.4	Virtual Entity Endpoint	Does the Testbed offer an Virtual Entity end point	1
2.5	Relational Database End	Does the Testbed offer a Relational DB endpoint?	1

[1] http://moodle.fiesta-iot.eu/course/view.php?id=2

	Point		
2.6	Document DB Endpoint	Does the Testbed offer a document DB endpoint?	1
2.7	IoT Services End Point (1)	Does the Testbed offer Resource Discovery (filtered by location, type, name, etc.)?	2
2.8	IoT Services End Point (2)	Does the Testbed offer direct access to sensors' observations thru services (historical queries and/or event-based subscriptions)?	2
2.9	IoT Services End Point (3)	Does the Testbed offer access to sensors' observations stored on database through services (historical queries)?	2
2.10	IoT Services End Point (4)	Does the Testbed offer actuation through offered services?	2
	Security		
3.1	Data Encryption	Does the Testbed offer secure encrypted communication channel between all testbed interfaces and FIESTA Platform?	1
3.2	Authentication	Can the testbed trust FIESTA to identify and authenticate experimenters on its behalf?	1
3.3	Identity Management	Determine who is using what features of the testbed	1
3.4	Authorization	Is the testbed able to specify access rights to specific resources?	1
3.5	Testbed-based Access Control	Can the testbed choose to perform local access control decisions and enforcement?	1
	Quality Auditing Aspects		
4.1	Response time	Do you control or set a threshold before which your testbed must give a response to the received request?	1
4.2	Processing time	Do you control or set a threshold before which your testbed must finish processing the request in the most complex case?	1
4.3	Computational assets	Does your testbed implement any resource optimizing mechanism?	1
4.4	Service prioritization	Does the testbed support the execution of services with different priorities?	1
4.5	Reliability	Do you define a ratio of failure time/working time that the testbed must respect?	1
	Generic		
5.1	Documentation	Does the Testbed provide Documentation?	1
5.2	Tools	Does the testbed provide development, deployment and management tools?	2
5.3	Adaptors	Can the Testbed offer the ability to run third party software (i.e. FIESTA adaptors)?	2
5.4	Additional DB	Can the testbed replicate/annotate its current data to the FIESTA format in a local Database?	1

A result per section and an overall result are generated after the user has submitted his answers. According to the obtained score, a feedback message is returned, per section and overall, to the user indicating the effort that he/she should commit to make the testbed interoperable with Fiesta-IoT platform. The overall result score and feedback message will be used for generating the certification. The following Table 2 recaps the messages that user can expect according his/her per section and overall score.

Table 2 Interoperability level based on the self-assessment scorecard

Result score	Feedback message
75%-100%	Your product is capable of interoperating with FIESTA-IoT
50%-75%	Your product needs some effort to be interoperable with FIESTA-IoT
0%-50%	Your product have some shortcomings that will make it harder to integrate with the FIESTA-IoT platform

The scorecard has been developed directly in the certification portal rather than using external components (such as survey monkey or lime survey) to offer a more integrated experience to the user. (

Figure 5)

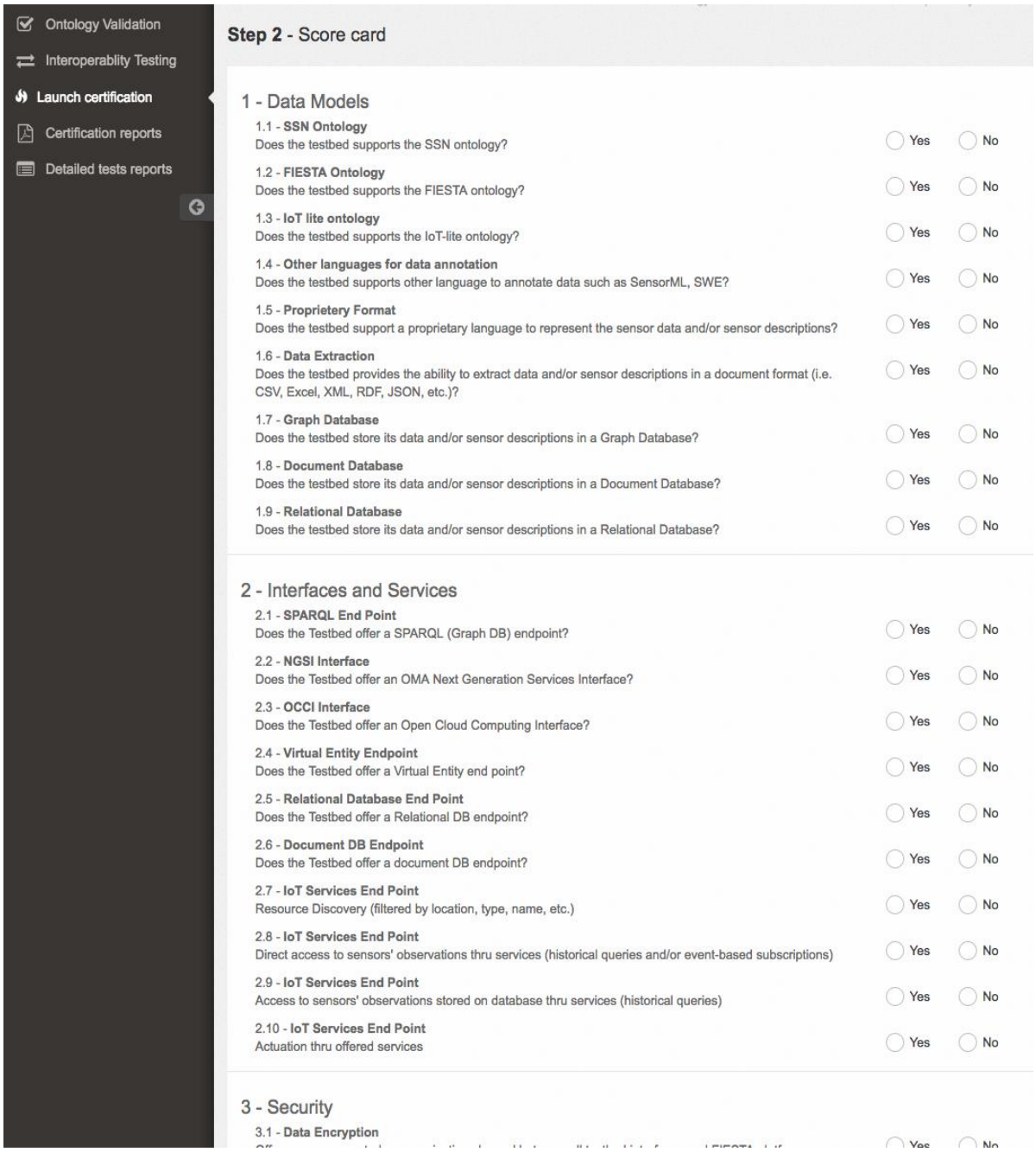


Figure 5 Scorecard implementation

4.2 Ontology validator

The ontology validator has been described in detail in [2]. Here is only a brief summary of its features:

- Through the UI, user can upload his/her semantic data to be validated as a file or through a simple text input field. (Figure 6)

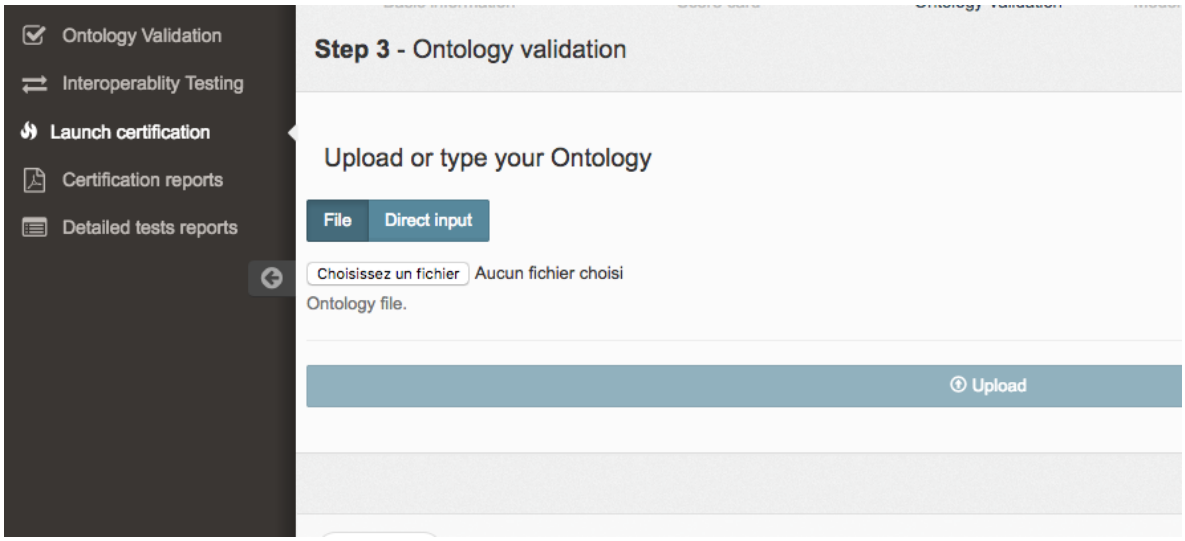


Figure 6 Upload a file or input text for validation

- Popular semantic data serialization formats are supported, such as jsonld, rdf/xml; owl/xml.
- User can choose the reference ontology that his/her semantic data need to be validated against (Figure 7)
- User can choose the type of errors to be detected in their semantic data. If there are other errors but belonging to another error category, they will be omitted in the report. (Figure 7)

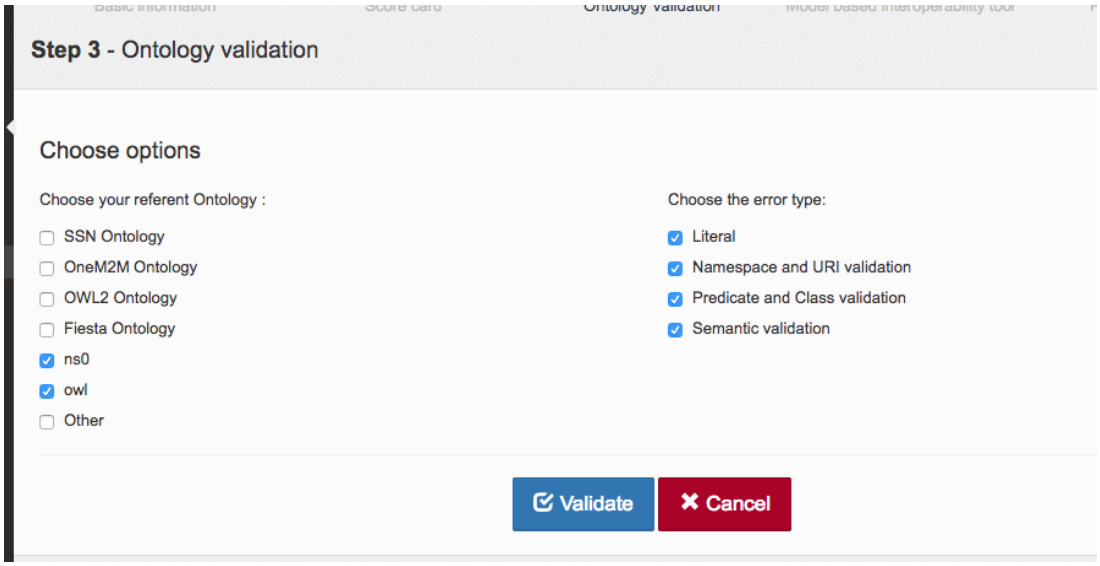


Figure 7 Choose reference ontologies and error types

User gets a complete report declaring the types of errors that have been looked for in the data to be validated, the validation result and the errors detected if applicable. (

- Figure 8)

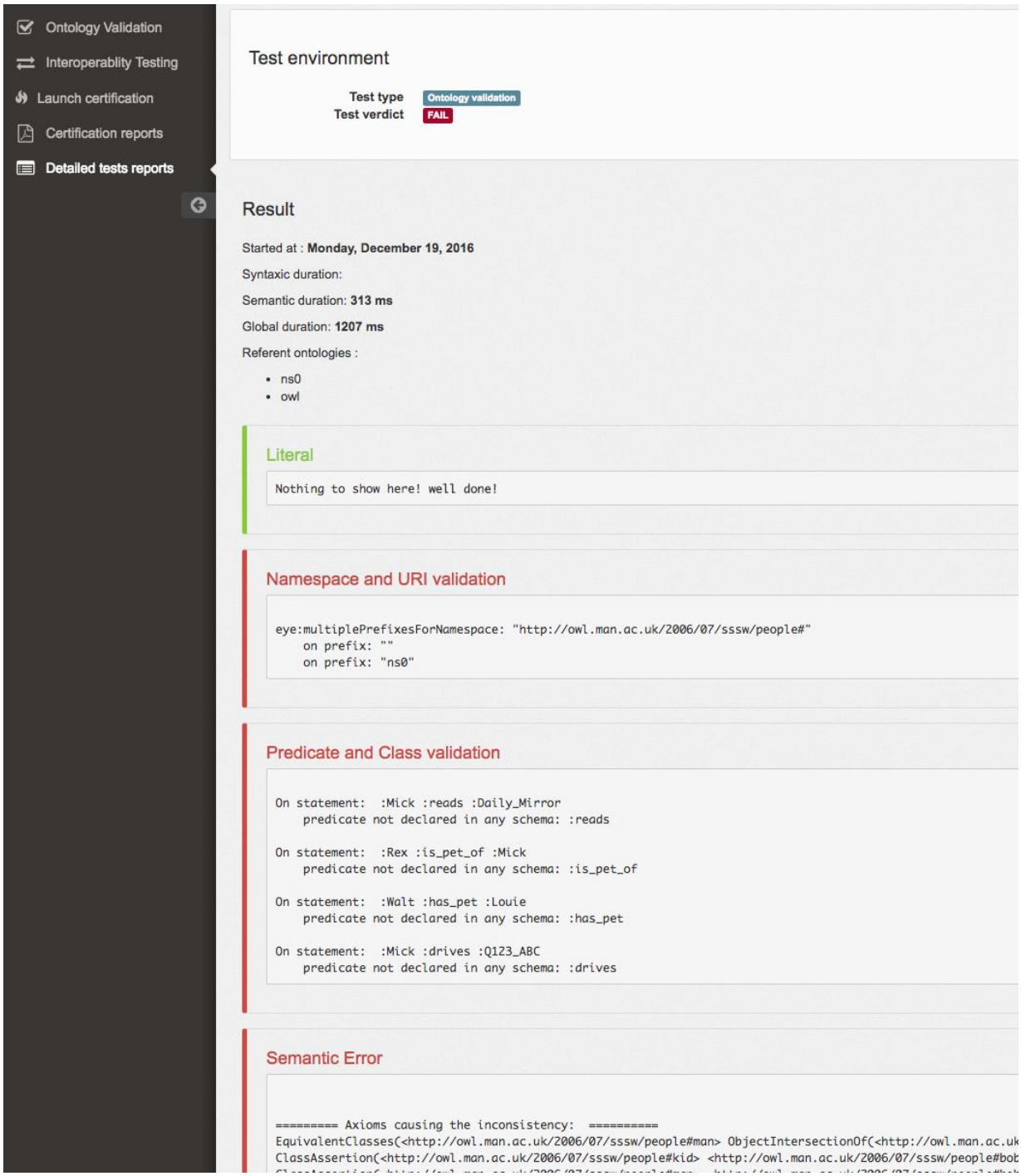


Figure 8 Ontology validation report

The UI features are integrated natively into the certification web portal. The validation itself is a service that can be invoked by the web portal using HTTP methods towards the API endpoint provided. Using this validator, the answers to question 1.1 to 1.4 can be supported by testing results.

The following Table 3 shows an example of using the ontology validator API endpoint. The keywords surrounded by `{{}}` are variables that take either the user's input or an attributed value from the web portal. In this example, we can observe that all the error types have been chosen for the validation (the '`ErrType`' keyword). If

the report to this current validation contains no error, the semantic data is considered as compliant with the Fiesta-IoT ontology.

Table 3 Ontology validation API

Description	Example: Execute a validation of semantic data regarding the Fiesta-IoT ontology
HTTP method	POST
URL	/ontologyValidator/ontology/test
Header	Content-Type = application/json
Payload	{ "format": "{{format}}", "filename": "{{filename}}", "ErrType": ["Namespace and URI validation", "Literal", "Predicate and Class validation", "Semantic validation", "Complete", "Other"], "file_id": "{{file_id}}", "ref_ontologies":[{"OntoPath":"defaut", "OntoName":"Fiesta Ontology"}] }

The ontology validator will only generate a validation report which is stored under the user's account. If the test has been performed against all the types of errors and no error is detected, a certificate that declares "compliant with reference ontology" can be delivered. Otherwise, the semantic data is considered not compliant with the reference ontology.

This service is deployed using a dockerized platform so that the service can easily be adapted and run in other platforms.

4.3 Model-based Interoperability Testing Tool

4.3.1 Introduction

Interoperability is a measure of the ability of two or more systems to connect, understand and exchange data with one another. It remains one of the fundamental challenges of distributed systems, due to the ever increasing complexity and heterogeneity of individual systems. The Internet of Things exacerbates this problem: pervasive sensors, embedded devices, PCs, mobile phones, and cloud services are

connected using a range of networking solutions, network protocols, middleware protocols, and application protocols and data types. Such heterogeneity builds barriers that software developers must address to achieve interoperability:

- **Data Heterogeneity.** Different systems represent data in different ways; this manifested at two levels. *Syntactic* where different systems use different formats to express the same information e.g. XML versus JSON versus YAML. Further, there is a greater problem with the "meaning" of data messages. Where components use the same syntax and schema (e.g. in XML) there can be inconsistencies e.g. <temperature>10</temperature> could be understood as 10 Celsius or 10 Fahrenheit.
- **Middleware Heterogeneity.** Developers can choose to implement their distributed applications and services upon a wide range of middleware solutions. In particular, these consist of heterogeneous *discovery* protocols (e.g. the Hypercat platform for discovering IoT services and devices[2], or a plug and play platform such as Bonjour[3]) which are used to locate or advertise the services in use, and *heterogeneous interaction* protocols which perform the direct communication between the services (e.g. MQTT, HTTP REST or COAP).
- **Application Heterogeneity.** Devices and services typically expose an application API. The sequence usage of such APIs may introduce interoperability barriers, and hence, protocol usage must be aligned. For example, one IoT service may provide a `GetObservations`() method that returns all sensor values (e.g. temperature and humidity), whereas a different IoT service may implement the API with `GetTemperature`() and `GetHumidity`() methods.
- **Non-Functional Heterogeneity.** Distributed systems have non-functional properties that must also be considered if interoperability is to be achieved. That is, two systems may be able to overcome all of the three prior barriers and functionally interoperate, but if the solution does not satisfy the non-functional requirements of each of the endpoints then it cannot be considered to have achieved full interoperability. For example, peers may have different requirements for the latency of message delivery; if the client requires that messages be delivered within 5ms and the server can only achieve delivery in 10ms then interoperability is not satisfying the solution. Similarly, two systems may employ different security protocols; the interoperability solution must ensure that the security requirements of both systems are maintained.

Where such interoperability barriers exist, how can systems be certified to understand each other and interact? Standardisation and middleware are two established methods to achieve interoperability. However, given the diversity of the Internet of Things it is impossible to rely on global standards or a single common middleware platform employed by all. Instead, interoperability is typically tackled in a more ad-hoc manor, relying on the system developers to create, test and validate interoperable solutions. Such development may include the following activities or behaviour:

[2] HyperCat, http://www.hypercat.io/

[3] https://support.apple.com/bonjour

- Executing per system/protocol compliance tests, e.g. interoperability events for MQTT[4] where developers test one platform against another to verify that they interoperate. Often such system testing is performed at the same geographic location.
- Implementing interoperability with a published API information for a developer to follow (e.g the HyperCat catalogue). This leaves the developer to understand the API and test the interoperability for themselves—it may be that their misunderstanding biases the interoperability testing.

While these solutions help in overcoming interoperability problems, there remains a significant burden on developers to understand and identify problems and then implement and test solutions accordingly.

Therefore there is a growing need for *interoperability testing* to become a fundamental step in the development of IoT systems. But how can such tests be easily designed and executed? *Model-driven software development* offers a principled approach to the engineering of interoperable solutions through: the capture of shared domain knowledge between independent developers; and the automated generation and testing of software. Therefore, raising the level of abstraction increases the understanding about how to achieve and test interoperability because such information is captured in the model, which is a shareable software asset.

In this section we present a *model-driven engineering tool* to simplify the engineering and testing of interoperable systems. This tool allows the developer to create, use, and re-use models of interoperability to reduce development complexity in line with the following requirements to ensure interoperability:

- *Specification compliance*. To check that systems comply with particular specifications, e.g. an IoT sensor produces event data according to the NSGI specification [5] , a sensor generates sensing data based on oneM2M specifications, or streamed data content complies with a data format specification uploaded to the HyperCAT catalogue.
- *Interoperability testing*. This monitors the interaction between multiple systems to test whether they interoperate with one another, identifying the specific issues to be resolved where there is failure.
- Interoperability Certification. To execute and pass a sequence of tests that certifies whether a system complies with a specification e.g. it receives a compliance certificate for NGSI or oneM2M. Or to execute and pass a sequence of interoperability that certify that the system interoperates with another system; for example, to certify that a testbed fully interoperates with the FIESTA-IoT platform.

[4] http://iot.eclipse.org/documents/2014-04-08-MQTT-Interop-test-day-report.html

[5]http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/ngsi-v1-0

4.3.2 Use Cases

Deliverable D6.1 defined the set of stakeholders that leverage the assets of the Global Market Confidence programme. Based on these stakeholders, we define a set of use cases in terms of how these roles utilize the interoperability testing tool.

IoT Infrastructure Providers and/or Testbed Owners will use the interoperability testing tool for purposes relating to the development of service and technology implementations that conform and interoperate with recognised and de-facto IoT standards.

Use Case ID	*IoT-Infra – UC1*
Stakeholders	FIESTA-IoT Testbed developer, IoT service developer
Description	A developer for the infrastructure provider develops the API that matches a given standard X. This could be a FIESTA API or oneM2M API for a testbed owner, or an NGSI API for an IoT service developer. During the development process they test the API implementation conforms to the standard using the testing tool. The testing tool informs the developer of where there are issues with the interoperability implementation (e.g. where a method implementation from the API is wrong) – identifying the issue to be corrected). The tool should be used by the stakeholder in the local environment: i) to protect confidentiality during the testing procedure, and ii) to reduce the risk of distributed systems error affecting the testing procedure.

Use Case ID	*IoT-Infra – UC2*
Stakeholders	FIESTA-IoT Testbed developer, IoT service developer
Description	A developer for the infrastructure wants to test that their system interoperates with another system. An example, is that a testbed developer wishes to test that their testbed fully interoperates with the FIESTA-IoT platform. During the development process they test the implementation of their system interoperates with an identified system using the testing tool. The testing tool informs the developer of where there are issues with the interoperability implementation (e.g. where interoperability has failed– identifying the issue to be corrected). The tool should be used by the stakeholder in the local testing environment: to protect confidentiality during the testing procedure.

Use Case ID	*IoT-Infra – UC3*
Stakeholders	FIESTA-IoT Testbed Owner, IoT service developer
Description	After *IoT-Infra – UC1 and/or UC2* have been carried out successfully, a testbed or infrastructure owner wants to receive a certificate from FIESTA-IoT that states that the

	implementation interoperates correctly. The stakeholder request informs the certification portal which then uses the testing tool to produce a certification report for the system being evaluated.

Experimenters/Researchers and IoT application developers

These developers are interested in tools and services provided by the FIESTA-IoT federation to develop and deploy their applications easily and efficiently; while they may be less interesting in certification. They may wish to advertise that their application interoperates with given IoT standards e.g. this application interoperates with other OneM2M applications.

Use Case ID	*IoT-AppDev – UC4*
Stakeholders	IoT application developer, FIESTA-IoT Experimenter
Description	A developer of an IoT application wishes to test if their application interoperates with a given IoT service or technology. An example, is that a FIESTA-IoT experimenter has written a client application to interact with the FIESTA-IoT APIs. During the development process they test the implementation of their system interoperates with the identified API using the testing tool. The testing tool informs the developer of where there are issues with the interoperability implementation (e.g. where interoperability has failed– identifying the issue to be corrected). The tool should be used by the stakeholder in the local testing environment: to protect confidentiality during the testing procedure.

Use Case ID	*IoT- AppDev – UC5*
Stakeholders	IoT application developer
Description	After *IoT- AppDev – UC4* has been carried out successfully, an application developer wants to receive a certificate from FIESTA-IoT that states that the implementation interoperates correctly. The stakeholder request informs the certification portal which then uses the testing tool to produce a certification report for the tested application.

Standardization bodies. Their main activities are developing, coordinating, promulgating, revising, amending, reissuing, interpreting, or otherwise producing technical standards that are intended to address the needs of some relatively wide base of affected adopters. They may wish to develop test suites and share knowledge for testing purposes and reduce their effort in supporting compliance and conformance activities.

Use Case ID	*Standards – UC6*

Stakeholders	Standards body
Description	A standards body uses the testing tool to produce a model of a test suite that can be used by other stakeholders to perform interoperability testing. They wish to pass interoperability testing knowledge to IoT software developers/

From these use cases, the following four key requirements for an interoperability testing tool are identified:

- **Req-1: A tool to model standards and APIs**

Standards or API developers can use a tool to model their APIs, and from this model interoperability test suites can be created by IoT developers to execute compliance and interoperability tests. *Essentially this is a requirement for ability to create and share knowledge between stakeholders* to reduce development errors (and costs) caused by a lack of shared understanding.

- **Req-2: A development testing tool to test specification implementation compliance**

During software development of the implementation of a standard or API, a developer uses the testing tool to help identify interoperability errors and bugs to be fixed.

- **Req-3: A development testing tool to test interoperability between two or more systems**

During software development of the implementation of a distributed system where two or more systems interact, a developer uses the testing tool to help identify interoperability errors and bugs to be fixed when they interact with one another.

- **Req-4: An online service for performing interoperability certification tests**

A web service for running interoperability tests whose results inform a certification report that can then be used to generate interoperability certificates.

4.3.3 Interoperability Models

Distributed services are typically modelled using interface description languages, e.g. WSDL, WADL and IDL, to both describe the operations available and how to execute them (e.g. using a SOAP or IIOP message). These can then be complemented with workflow (e.g. BPEL) and choreography languages to explain the correct sequence of events to achieve particular behaviour. With these models it is then possible to automate the interoperability testing processes and better support service composition. However, these approaches are often tied to a specific technology type e.g. Web Services and CORBA being clear technology silos, and hence the approach is not well suited to loosely-coupled IoT and cloud services that employ a wide range of technologies and communication protocols.

Furthermore, the models themselves are typically complex to write, use, and maintain which in turn means they are not widely deployed; this can already be seen in the

Internet Services domain where RESTful APIs (e.g. Twitter, Facebook, and others) provide documentation and SDKs to help developers interoperate without the need for separately maintained IDLs.

Our approach explores models that focus solely on interoperability; that is, the specification of the exchanges between IoT services with rules defining the required behaviour for interoperability to be guaranteed. There are two types of model: i) the interoperability model used by application developers and interoperability testers; and ii) specification models and compliance testers.

Finite state machine: General Principles

An interoperability model is specified as a finite state machine; the general format is illustrated in Figure 9. Each circle is a state, that represents the state of a distributed application (not an individual service) waiting to observe an event. A *transition* (arrow between states) represents a change in state based upon an observed event matching a set of rules regarding the required behaviour. Hence, the model represents the series of states that a distributed application proceeds through in reaction to discrete events (e.g. a message exchange, a user input, etc.). If the state machine proceeds such that there is a complete trace from a start state to an end state then we can conclude that software within the distributed system interoperate correctly.

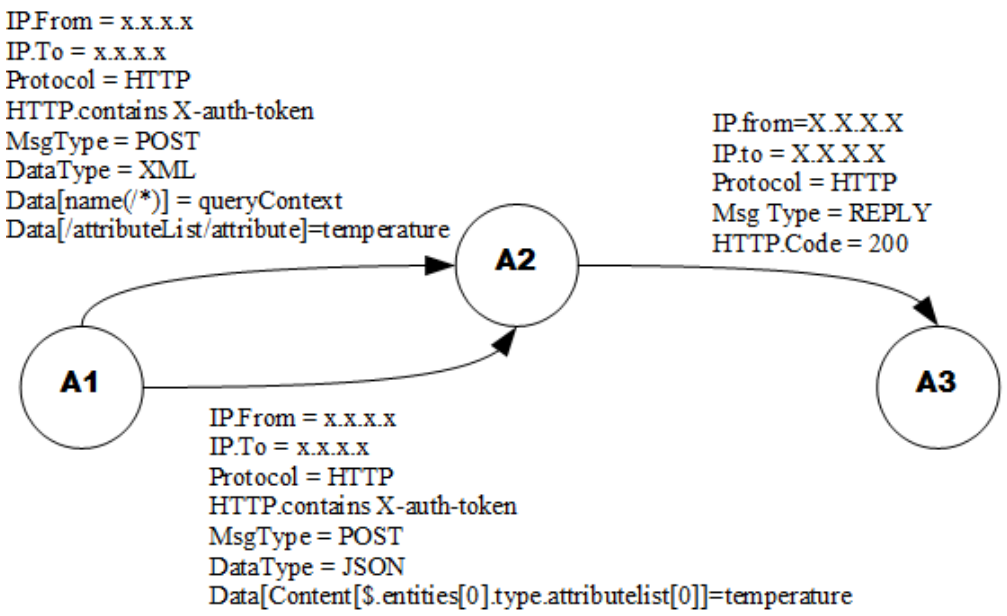


Figure 9: Finite State Machine Model of Interoperability

If an event occurs and no transition can be made (because the event does not fulfil the rules), then the interoperability model identifies a failing condition. Aligned with knowledge regarding why this rule failed, the tool can provide preliminary information for correcting the error. Discrete events are captured messages (e.g. a HTTP message in Figure 9), which are evaluated against the model specification, i.e. transition rules can be evaluated. Where all rules evaluate to true, the state machine transitions to the corresponding labelled state (e.g. from state A1 to A2 in the diagram).

Finite state machine: States

So far we have introduced a state as observing event where the model waits for a discrete event to occur. However, the model provides a rich set of events to model a broader range of distributed systems and interoperability tests. These are listed as follows (and highlighted in Figure 10):

- A ***start*** state. Represented by a double-lined circle. This specifies where the behaviour begins in the model, and where testing should start. Note, there can be only one start state in each individual testing model.
- A ***normal*** state. Represented as a single lined circle. This is the standard event waiting to observe an event and then making a decision as to how to proceed based on the content of the event. This is the behaviour previously described by Figure 9. It has one or more incoming transitions, and one or more outgoing transitions – all outgoing transitions are labelled with a set of interoperability rules to evaluate the discrete event against.
- An ***end*** state. Represented by a filled circle. This specifies where the model terminates. There is one or more end states in a model.
- A ***trigger*** state. Represented with an arrow added to the state. This is an active state, so rather than waiting to observe an event, it injects an event into the system. For example, to test an API the model can invoke the API by creating an invocation event and observing the response received. There must be one and only one outgoing event from a trigger state. The transition is labelled with the message content that forms the event to inject.

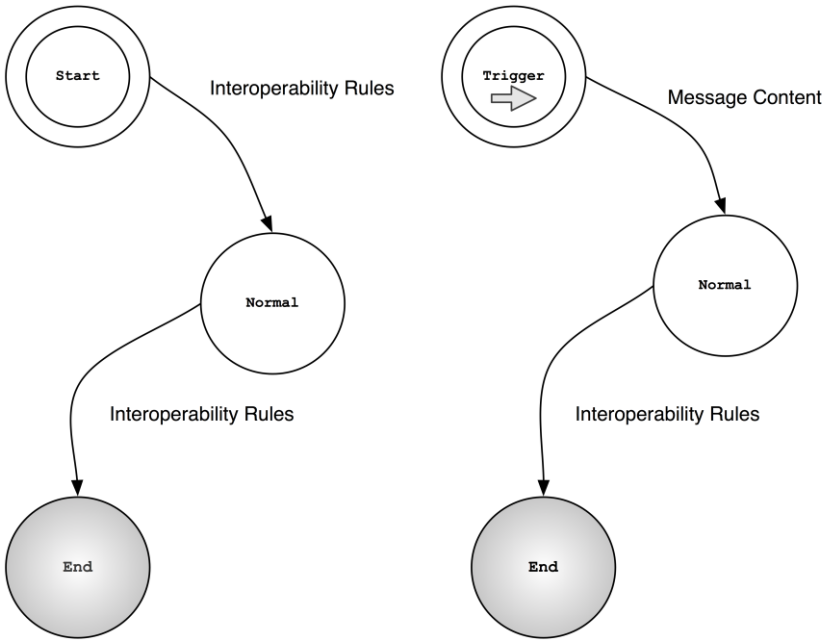


Figure 10: Interoperability Model States

Finite state machine: Interoperability Testing Transitions using examples

In Figure 9 we present a very simple example to illustrate how a model is used in practice to test interoperability i.e. two separate systems communicate with one another. Here, we model a distributed system where a client requests the temperature of a room sensor (transition from state A1 to A2), and a context service

providing the sensor data replies (transition from A2 to A3). They interact with each other to complete a single request-response type operation.

The correct transitions in the behaviour are modelled as follows. The client must send a REST HTTP post operation which can contain either XML or JSON (two alternative transition paths). A number of rules are presented to illustrate how interoperability rules are attached to transitions; each transition can specify one or more rules concerning different characteristics of events. These fall into *protocol* specific or *data* specific rules:

- *Protocol specific* rules. Evaluate events according to the structure and content of an observed protocol message (not the application/data content). For example, check the IP address of sender of the message to verify which services are interacting with each other. Further, evaluating the protocol type (HTTP, IIOP, AMQP, etc.), and the protocol message type (HTTP GET, HTTP POST or an IIOP request) to ensure that the correct protocol specification is followed. Finally, checking protocol fields (e.g. a HTTP header field exists or contains a required value) to ensure that the message contains the valid protocol content required to interoperate. Currently the tool evaluates HTTP protocol rules.
- *Application and data* specific rules. Evaluate the data content of protocol messages to ensure that services interoperate in terms of their application usage. For example, the data content is of a particular type (e.g. XML or JSON), corresponds to a particular format or schema, contains a particular field unit (e.g. temperature), etc. Furthermore, rules can make constraints on the application message, e.g. ensuring the operations required are performed in order (e.g. A sends a subscribe message to B, before C sends a publish message to B).

Data rules are evaluated using data-specific expression languages; for example we leverage XPATH[6] and JSONPATH[7] tools to extract data fields and evaluate whether a given expression is true (e.g. a rule in the XPATH format: Data[name(/*)] = queryContext requires that the XML data contains a root tag labelled queryContext).

Finite state machine: Specification Compliance Testing Transitions using examples

A compliance model is specified as a finite state machine using the same elements as the interoperability model in Figure 9 but also including the trigger states and message transitions discussed earlier.

[6] http://www.w3.org/TR/xpath20

[7] https://code.google.com/p/json-path

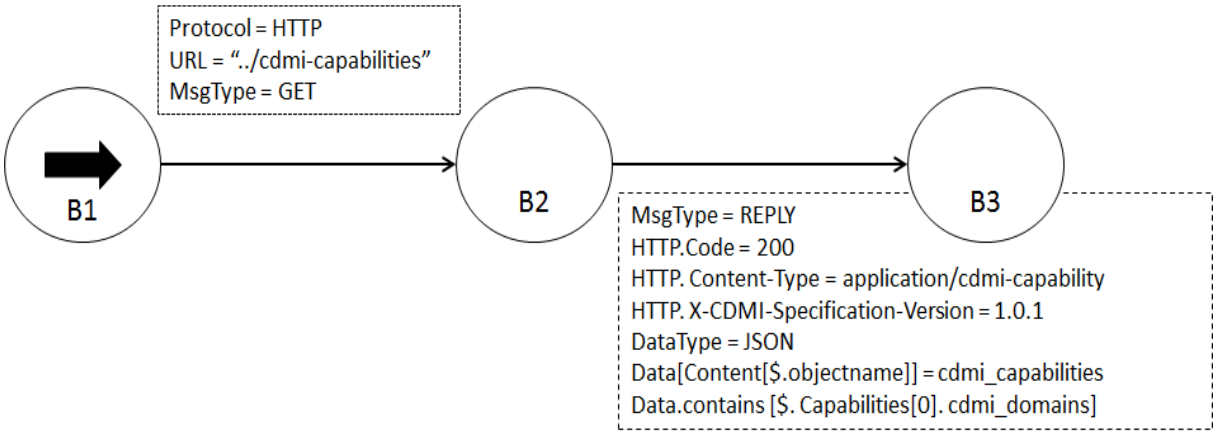


Figure 11: Compliance Model with Trigger States

A simple compliance model is illustrated in Figure 11. Note, a trigger state has an arrow in the circle to distinguish it from a normal observing state. This is a simple model of part of the Cloud Data Management Interface[8] (CDMI API) specification for cloud storage. Here, we are testing if the service correctly implements the *discoverCapabilities* operations to view the technical capabilities and installed features of a CDMI deployment. The first state is a trigger state that sends a HTTP GET message to the `\\cdmi\_capabilities` URL. The second state transition then evaluates a rule set against the received response to ensure that the data in the HTTP response matches the required data format of the api. Again we see rules to test the structure of the HTTP message and that the data has fields *equal to* specific values and *contains* the required fields.

4.3.4 Testing Tool

REQ-1, REQ-2, and REQ-3 state the need for a software development tool in order to underpin interoperability testing in general. Such a testing tool provides the following key behaviour:

- The ability to graphically create an interoperability testing model.
- The ability to use a model to perform both specification compliance tests, and interoperability tests.
- To receive testing reports identifying where the tests have found interoperability problems.

The Model-based Interoperability Testing Tool is illustrated by the screen-shot in Figure 12; it has two core elements:

1. The graphical editor providing drag and drop functionality to create the interoperability models described in the previous section.
2. The monitoring and testing framework that evaluates running distributed applications against the model visualised in the editor and evaluates them for correct interoperability. When the test command is selected in the editor - the models are converted to XML (this also allows them to be permanently stored) and they are input to the **Model Evaluation Engine**.

[8] http://www.snia.org/cdmi

The Model Evaluation Engine is the core back-end functionality that executes testing—it also forms the back-end of the certification process described in the next section.

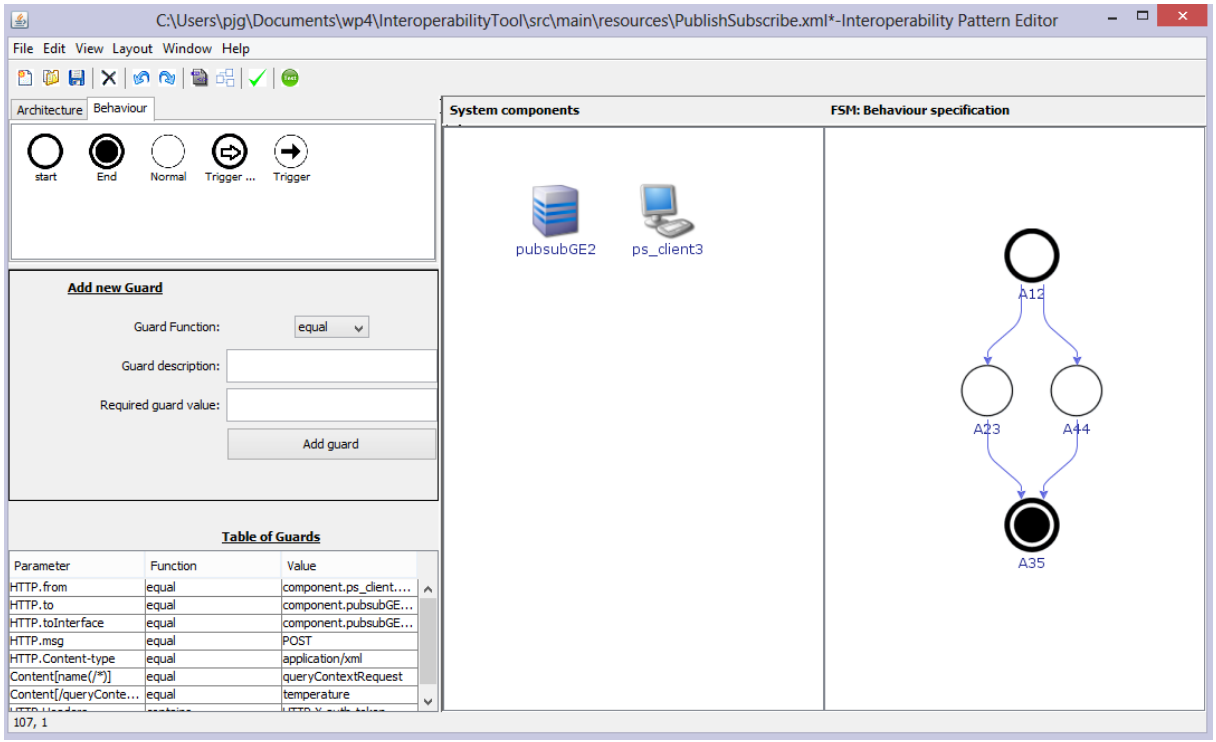


Figure 12: Model-based Interoperability Testing Tool

Without going into significant unnecessary implementation details we can explain the operation of the model evaluation engine in terms of two functions:

- *Monitoring deployment*; the framework takes an interoperability model as input and generates a set of proxy elements that capture discrete events. For example, where the tested systems are HTTP-based, REST events are captured at all interface points in the application). Hence, if we observe that a service receives events at a particular URL; we generate a proxy to capture those events—the proxy simply reads the information before redirecting the request to the actual service. Hence, it does not interfere with the operation of the tested system.
- *Model evaluator*; receives events and evaluates them against the rules specified in the transitions. The evaluator is protocol independent (per protocol plug-ins map concrete messages to the format of the model rules); hence, at present the framework parses HTTP messages, but is extensible to other data protocols. The evaluator creates a report to identify success or failure to the developer; and where a failure occurs, the framework performs simple reasoning to pinpoint the source of the error.

The tool is available for download from Github (https://github.com/pjgrace/connect-iot). It is installed on the developer's machine, and therefore meets the requirements to be used for interoperability testing before requesting certification.

Performing Interoperability Testing

The following provides a brief description of how the tool is used to carry out interoperability testing, and how the output of the tool can be used to identify interoperability issues. Appendix A describes the sequence of steps to execute an interoperability test; at the end an Interoperability Report is returned – it contains the following information:

```
Test started - run the application
---------------------------------
Starting trace at Node:A1
Invoked action - moving to state: A2
     Guard test succeeded: http.from is fiesta-iot.eu
     Guard test succeeded: http.msg is reply
     Guard test failed: http.code is 201 which !=200
Fail: no transition possible
Interoperability Error: Fail: no transition possible
```

This is report of a testing trace through the model as events occur between systems. It is a set of reports for each transition between states. The above shows the transition report between states A1 and A2. The following details can then be observed in the report:

- *Transition Success/Fail* – if every transition succeeds without fail interoperability is tested as correct.
- *Guard tests* – this is the test on each guard. It explains where an interoperability concern is observed. For example, this test on a system states that a 200 response is required, but the observed system returns 200 so is an interoperability error. If all guard tests pass then a transition succeeds.

4.3.5 Certification

In the previous sections, we have discussed interoperability testing in general. Here, we define how such interoperability tests can be applied to certify interoperability. There are two types of certification that can be carried out, which we define as follows:

- *Interoperability Certification*. The successful completion of a set of interoperability tests (defined by a model) that then certify that two or more systems correctly interoperate with one another.
- *Compliance Certification*. The successful completion of a set of compliance tests (defined by a model) that then certify that a system correctly implements an API or standard.

These certifications are defined by the different types of interoperability testing models; as described in the prior sections.

Interoperability and Compliance certification is provided as a service from the FIESTA-IoT Certification Web Portal. To explain the design and implementation of this service, we first present the architecture of the platform, the services API, and then describe in turn the flow of certification.

Architecture

Figure 13 illustrates the key elements of the interoperability certification functionality. It is provided as a web-based service—with a front-end, back-end style. Beginning with the back-end Interoperability Testing Module:

- The models describing the interoperability and compliance tests for specific certification processes are stored in the Model Data Service.
- The Model Execution Engine operates as described previously, but in this case there is an extra function: interoperability certification, which given the results of the interoperability tests returns a certification result that is input to the certification process.

The Interoperability Testing Service REST API provides an interface to perform the underlying actions. The operations of this interface as specified in

- Table 4.

There are two stakeholders shown in the diagram: to the right, the user wishing to carry out certification; to the left a FIESTA-IoT administrator uploading a new certification model to the system. For this purpose, FIESTA-IoT creates a new model using the interoperability testing tool and generates the XML version to be uploaded to the service. Once complete they POST the model via the REST API shown in

Table 4. These are operations to change the certification models (that are used as part of the certification tests).

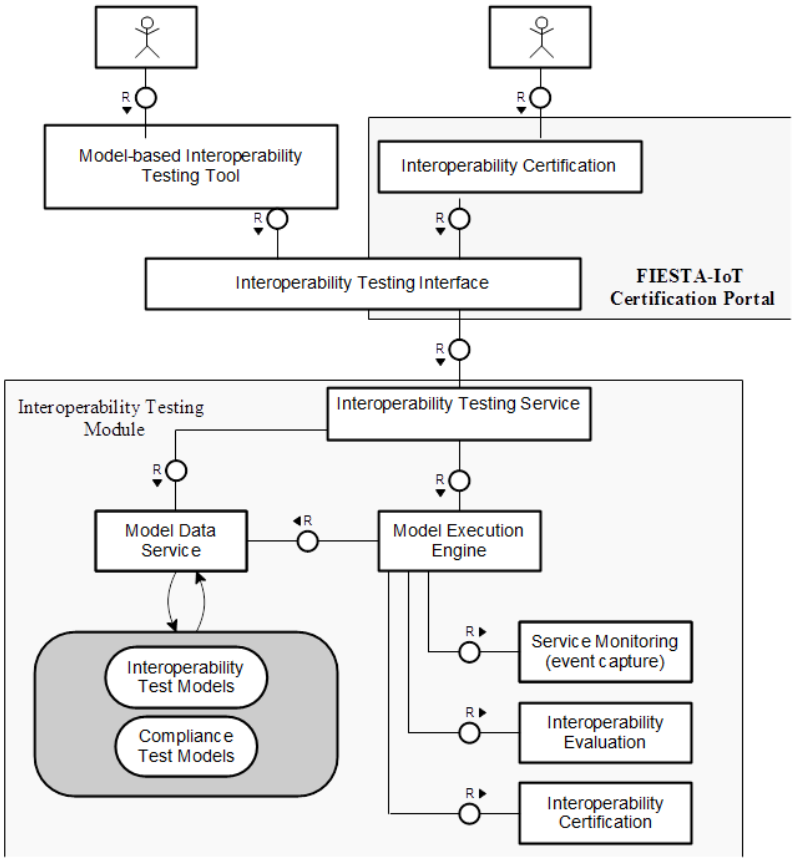


Figure 13: FIESTA-IoT Interoperability Certification Architecture

Table 4: Interoperability Testing Module REST API operations

Verb	URI	Description	Input/Output
GET	/Interop/models	Gets a list of all registered models.	Out – XML testing model
POST	/ Interop /models	Registers a new model	In – XML testing model
GET	/Interop/models/{id}	Get a specific model	Out – XML testing model
DELETE	/ Interop/models/{id}	Deletes a model	
PUT	/ Interop/models/{id}	Updates model	In – XML testing model
POST	/ Interop/models/{id}/certify	Execute a given model, evaluate the certification of the system and produce a certificate	In – XML testing model Out – Certification report

The important API method is /**certify** which is called in order to execute a certificate test and produce a certification report. This has the following input/outputs:

- Input - The interoperability model (in XML) this contains the filled in information about the service being tested (e.g. IP and API URL).
- Output – A simple status report in XML. This contains three fields: i) the certification success; a Boolean indication if certification is awarded; ii) the level of certification; an integer between 0 and 3 to allow different levels to be awarded: 0 is no certification, 1 is basic certification (bronze), 2 is good certification (silver), and 3 is full certification (gold).

It is up to the certificate designer how to award a certificate, based on an interoperability test. We give the following example from FIESTA-IoT – these are operations available from the TPS API [2]:

```
POST /getLastObservations
POST /getObservations
POST /pushLastObservations
POST /stopPushofObservations
GET /pushSingleObservation
```

If a testbed implements **one** of these operations correctly then they successfully interoperate with FIESTA-IoT. Hence, there is only one level of certification – if the test shows that the testbed interoperates with one of these, then a level 3 (gold) is awarded.

Interoperability Certification Front-end

The front-end GUI accessible via the portal, provides the endpoint for a developer to request certification of a technology. A design of the front-end screen within the certification portal is shown in

Figure 14. The developer selects the standard/specification they wish to certify against (e.g. a FIESTA specification) – this loads up the necessary input boxes for that specification (e.g. the URL endpoint of the API to perform a compliance test of). After entering the information about their service to be certified – the developer clicks certify. This in turn calls the /certify operation of the web service to create a certification report –which is displayed on the screen. The user can then select ‘save to account’ to persist this certification request i.e. if they are not happy with the result they can try again later.

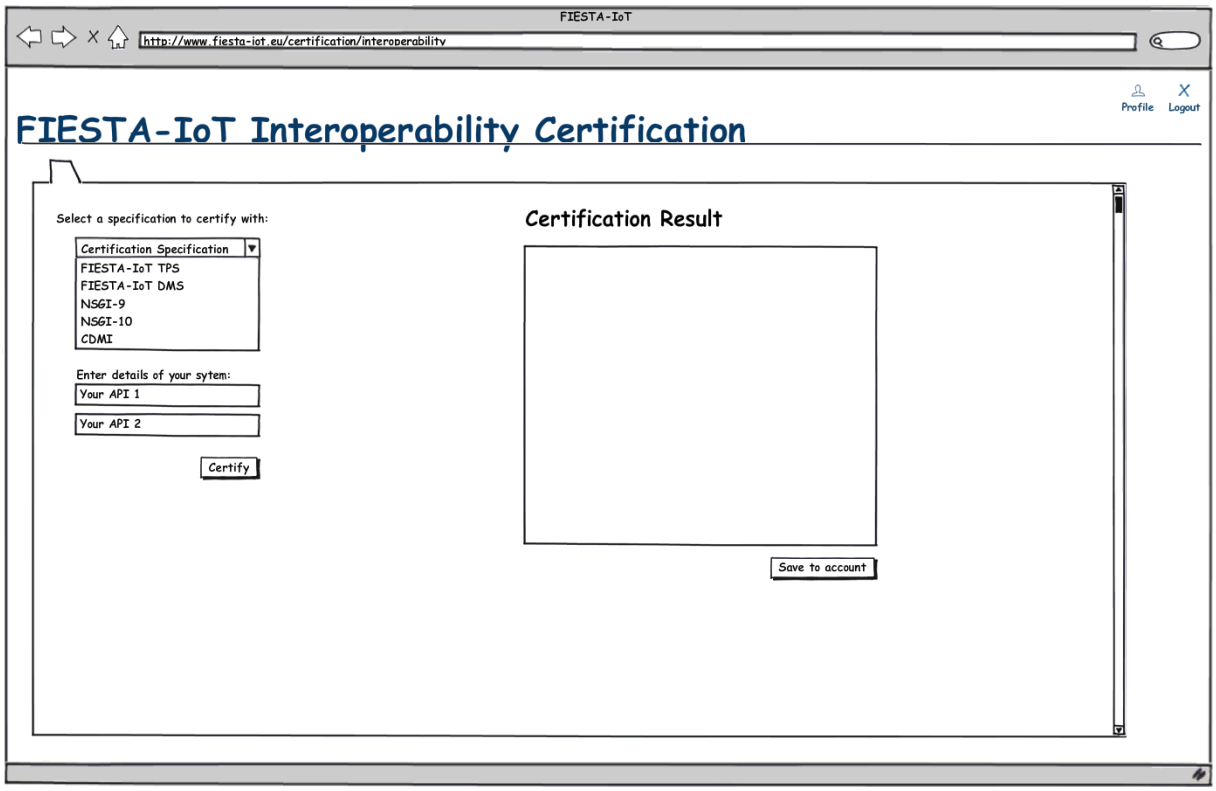


Figure 14: Interoperability Certification Front-end design

5 CERTIFICATION WEB PORTAL IMPLEMENTATION

5.1 Overview and technical architecture

The portal, known as the Certification portal, aims to provide users with a unique entry point in order to get their product certified regarding the FIESTA-IoT project with practical tools. It allows users to create accounts, use the provided tools to test their product, review the results of the certification process and generate a digital certificate. Additionally, the portal can be used as a tool suite without requiring final certification; such functionality will help the developer to find unconformity in their product in an early stage and address any interoperability issues before interoperability certification is requested.

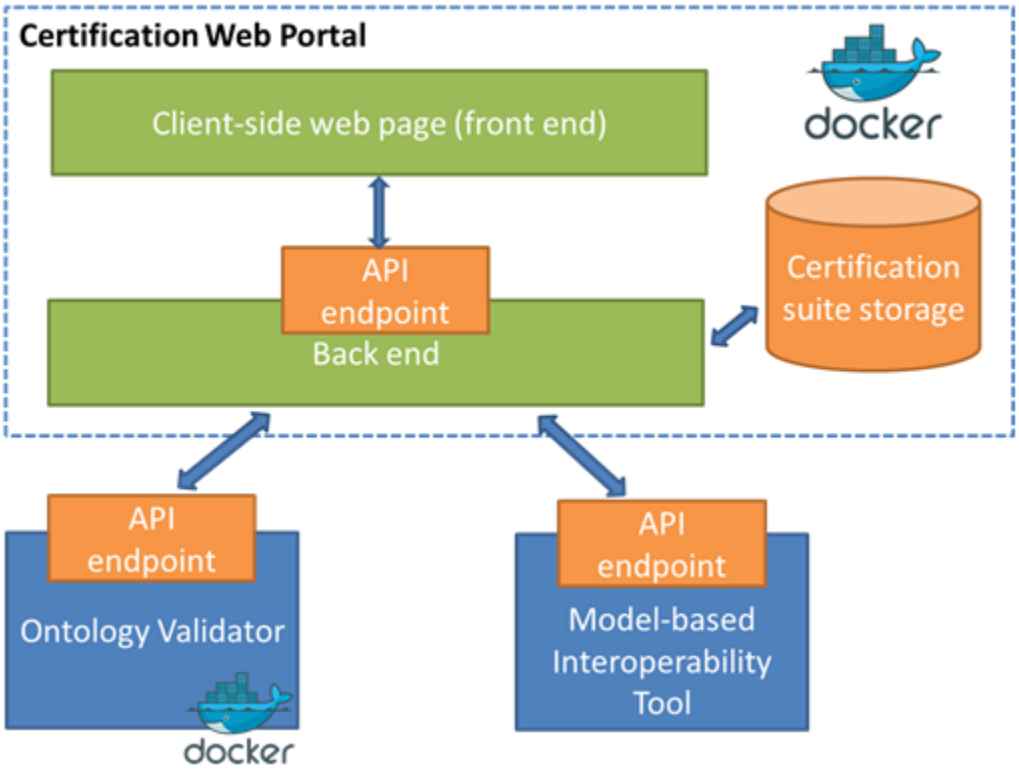


Figure 15 Certification Web Portal architecture

The complete Certification Web Portal resides in the Fiesta-IoT Utilities server. Illustrated in Figure 15, it is composed of: 1) a client-side web page that allows user to access the provided functionalities and tools; 2) a backend module which implements the business logic including the scorecard, the access to the certification suite storage and the communication with the other two web service tools; 3) the certification suite storage where the user account information and the related testing results and certifications are stored. To achieve this architecture, the front-end client-side web page is built on top of the AngularJS[9] framework. This framework provides developers with tools to create multi-pages JavaScript applications; consequently, the portal uses traditional web technologies such as HTML and CSS (and

[9] https://angularjs.org/

JavaScript). The backend is built using NodeJS [10] capable of asynchronous Input/Output in order to optimize throughput and scalability in Web applications with many input/output operations. The storage is a MongoDB[11] database. The Web Portal is deployed using the docker platform. Each of the three components is hosted in a docker container, and the APIs have been designed to allow them to communicate with each other.

5.2 Main features

The main features provided by the portal are the account management, the certification process management, the report generation and the certificate generation.

- **User account management**. If a user wants to get his/her product certified, an account to use the web portal is needed. This account will be used to generate a unique id to link the results from three independent certification tools integrated in the portal (scorecard, ontology validator and interoperability tool) in order to enable to generate unique certification. It also allows the user to review the historical test results and certifications issued under the same account (Figure 16). The information and data related to the account management are stored in the certification suite storage. A native authentication mechanism is available to protect the users' data.
- **Certification process management**. According to the defined certification process described in section 3, this feature controls the workflow and guides the user to get a certificate, e.g. if the user does not complete the scorecard section, he/she is not able to use the other two tools to proceed in the certification process (Figure 17). Note: if the portal is used as a tool suite for development, the user can omit the scorecard part and go directly to the other two tools, however, it is not possible to generate any certification according to the results.
- **Report generation**. A report is generated at the end of each independent certification tool. It is stored under the account and can be reviewed later.
- **Certification generation**. User can explicitly require a certificate with the condition that he/she finishes at least the scorecard. If second level certification tests have been performed after the scorecard, the certificate will include also the results. All the certificates will be stored under the user's account, except that the user deletes it explicitly, and can be reviewed later.

[10] https://nodejs.org/en/

[11] https://www.mongodb.com/community

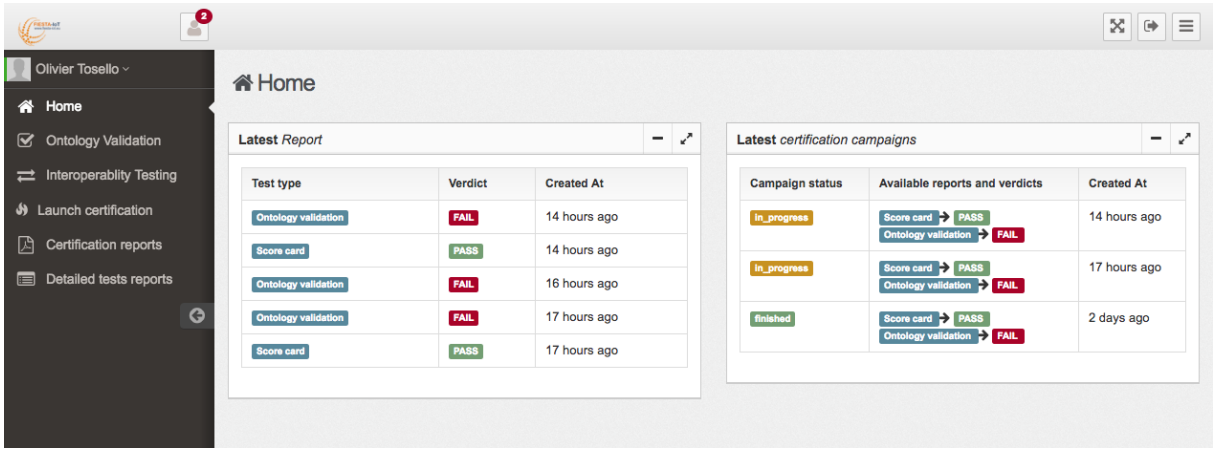


Figure 16 User's home page with historical reports and certificates

Figure 17 Certification process steps

5.3 Web portal as a tool suite for development

In the certification web portal, a section is also planned to allow developers to test their products during the development using the two web service tools (Figure 18). At the end of test, a report is generated including the test results. However, no certificated can be delivered as to obtain a certificate, the certification process needs to be respected.

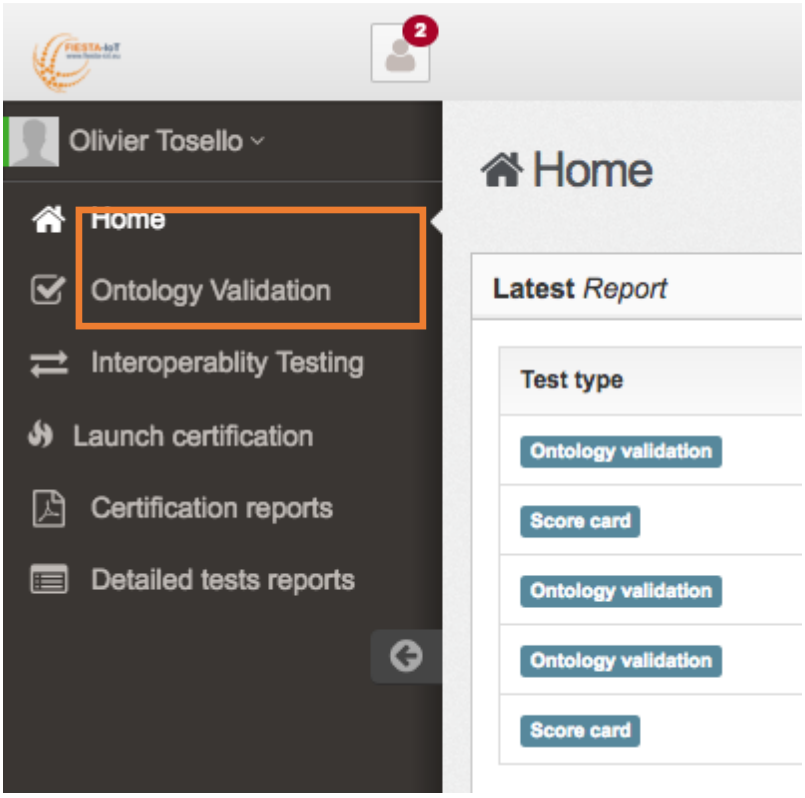


Figure 18 Tools used outside of certification process (in orange rectangle)

6 CONCLUSION

In this deliverable, we firstly introduced the certification process that we want to deliver with the Fiesta-IoT certification framework. The process consists of two main steps, which correspond respectively to two levels of certification: self-assessment certification and web service certification. The former consists of inviting user to answering a questionnaire in order to get an interoperability level score based on his answers. The latter consists of testing the product under certification using tools accessible online. A certificate is generated at the end of the certification process based on the result from the two steps, knowing that the second step is optional but recommended.

The implementation of tools enabling the certification process is described in the second part of the deliverable. All the tools, including the scorecard, the semantic data validator and model-based interoperability tool, are integrated in a certification web portal which is the unique entry for user who want to certify their product within Fiesta-IoT. The web portal manages the certification process, the user account and can also be used as a tool suite for semantic product developer during the development phase.

The next step of the current work is to invite in-house semantic data providers, namely the testbeds, to come to the certification web portal and get their data certified following the certification process. Feedback are expected from partners to make this certification suite improved for the open call participants.

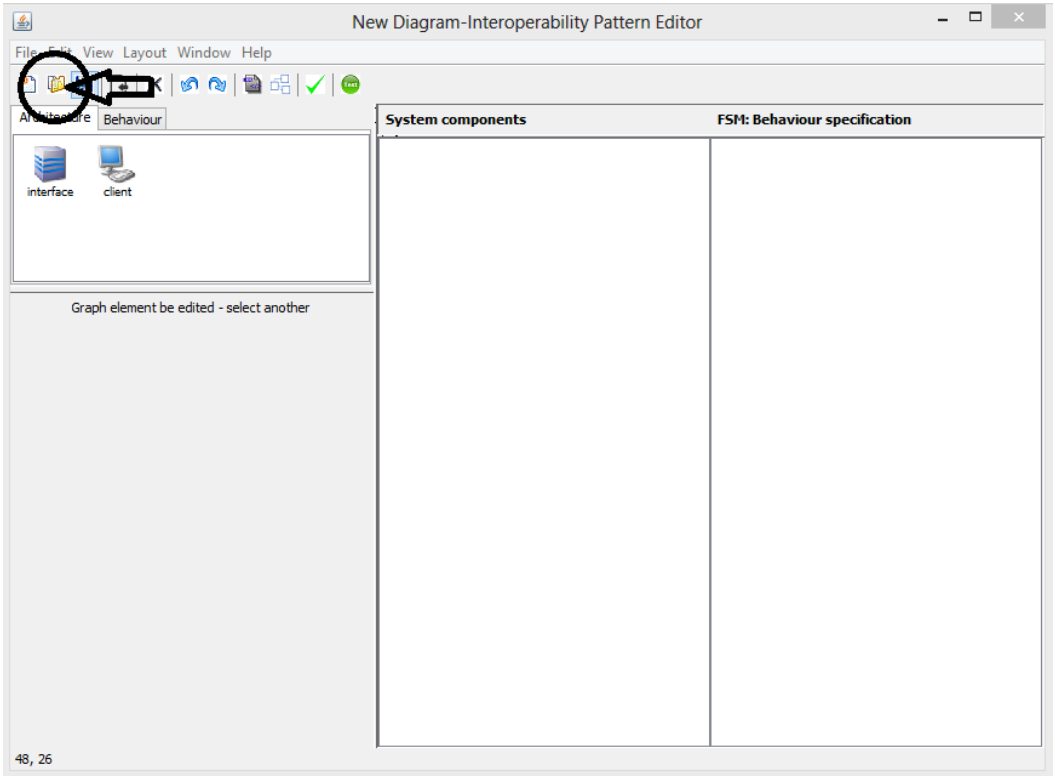
7 REFERENCES

[1] Fiesta-IoT, «Delivrable 3.2.1: Specification and implementation of common testbed interfaces,» 2016.

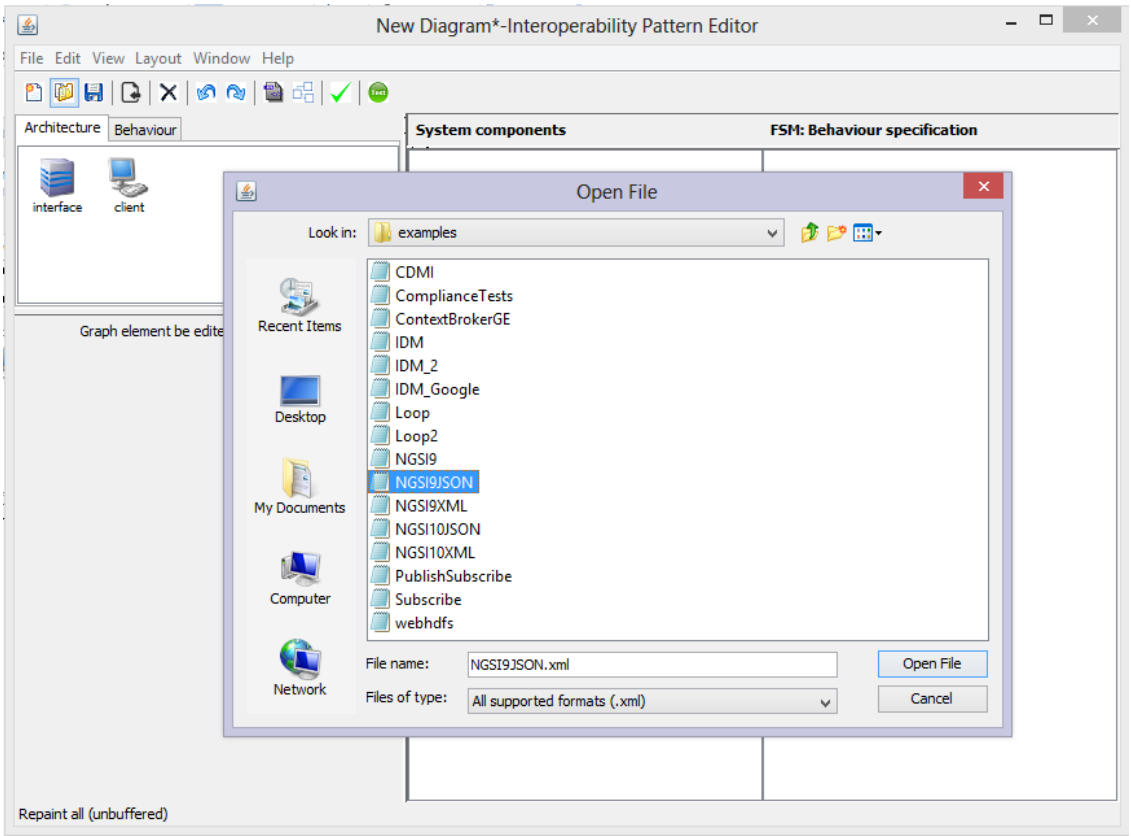
[2] Fiesta-IoT, «Deliverable 6.1 Design of global market confidence programme on IoT interoperability,» 2016.

APPENDIX I – PERFORM INTEROPERABILITY TEST

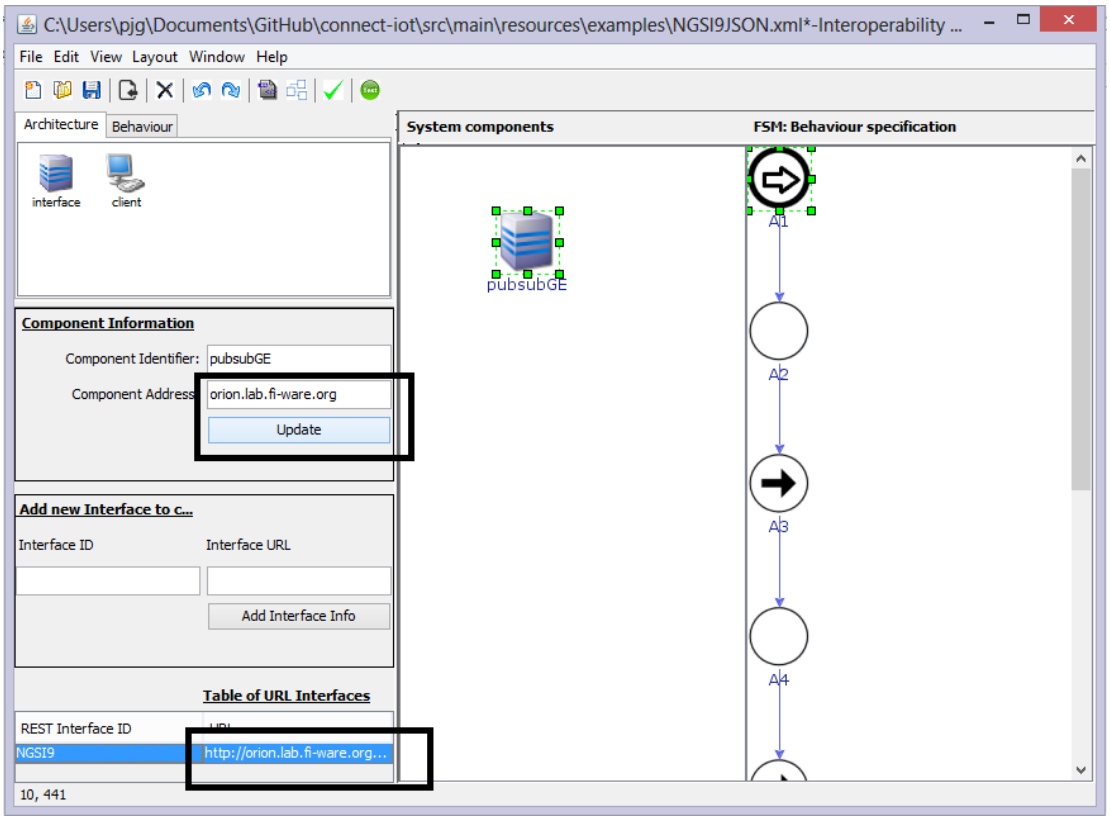
The first step is to load an interoperability test into the tool. Select the open file button:



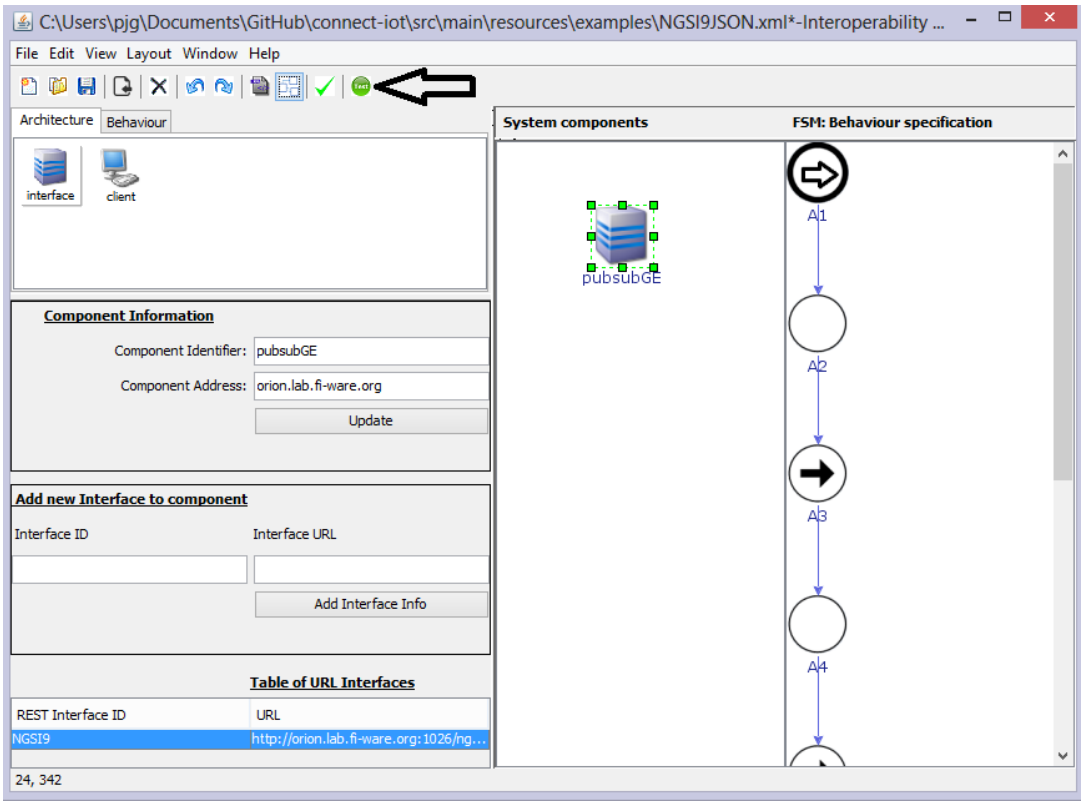
From the file dialogue box – select the interoperability test specification (model):



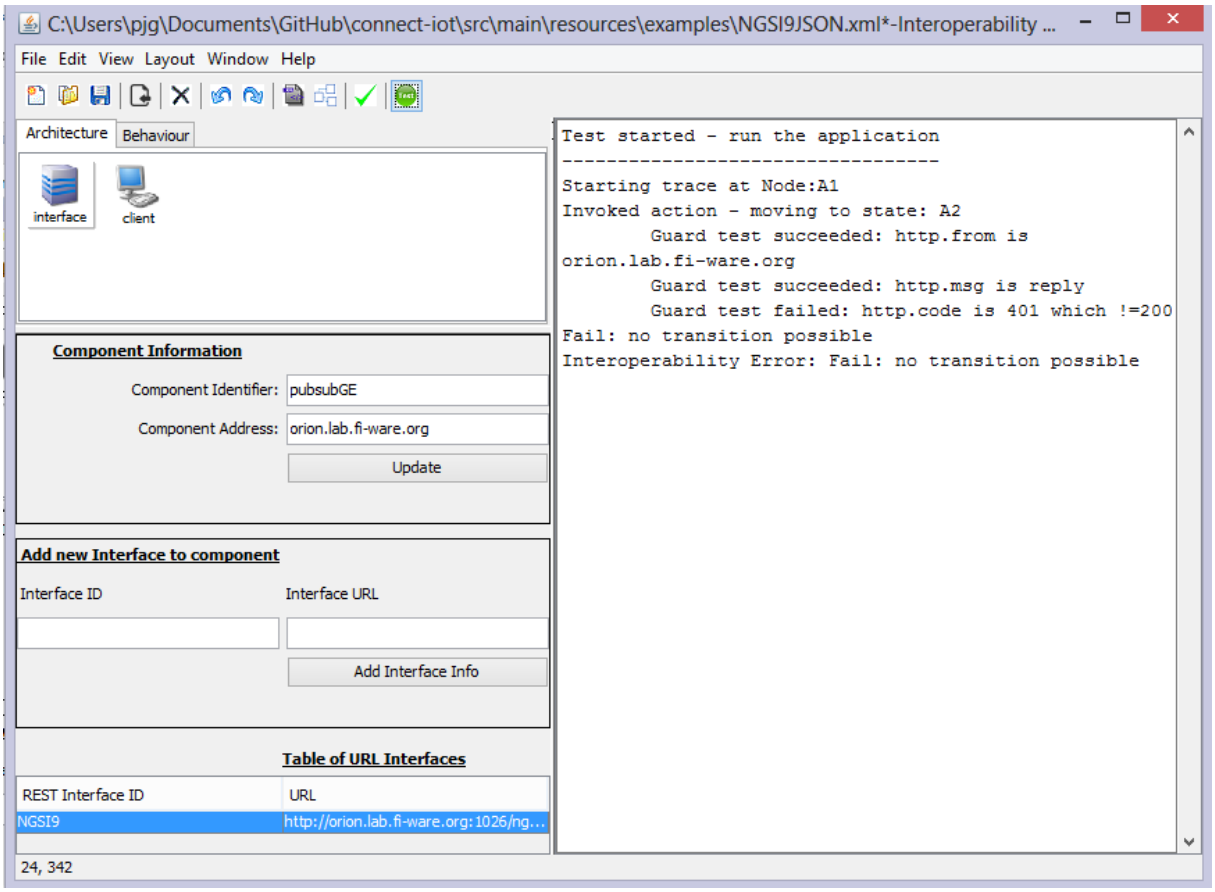
The next step is to define the system component attributes (i.e. the information about the system being tested). Select the component and then enter the IP address and API url in the attribute boxes:



The next step is to execute the interoperability test. To do this select the test button in the menu bar:



The debugging report is displayed on the screen. This highlights exactly which transition has failed in the model and the exact reason.



2016 FIESTA-IoT