Behaviour-driven development applied to the conformance testing of INSPIRE Web services

Francisco J. Lopez-Pellicer, Miguel Ángel Latre, Javier Nogueras-Iso, Jesús Barrera and F. Javier Zarazaga-Soria

Abstract The implementation of the INSPIRE directive requires to check the conformity of a large number of network services with the implementing rules of INSPIRE. The evaluation whether a service is fully conformant with INSPIRE is complex and requires the use of specialized testing tools that should report how verification has been made and should identify non-conformances. The use of these tools requires a high degree of technical knowledge. This fact makes very difficult for non-technical stakeholders (end users, managers, domain experts, etc.) to participate effectively in conformance testing, hinders stakeholders understanding of the causes and consequences of non-conformant results and may cause in some stakeholders disinterest in conformance testing. This work explores the suitability of a *behaviour-driven development* (BDD) approach to the conformance testing of OGC Web services in the context of the INSPIRE directive. BDD emphasizes the participation of non-technical parties in the design of acceptance tests by means of automatable abstract tests expressed in a human readable format. Using this idea as base, this work describes a BDD based workflow to derive abstract test suites and executable test suites from INSPIRE implementation requirements that can be written in the language used by non-technical stakeholders. This work also analyses if BDD and popular BDD tools, such as Gherkin and Cucumber, are compatible with ISO 19105:2000 testing methodology. As demonstration, we present an online conformance tool for INSPIRE View and Discovery services that executes BDD test suites.

Francisco J. Lopez-Pellicer
Universidad Zaragoza, Zaragoza, Spain, e-mail: fjlopez@unizar.es

Miguel Ángel Latre
Universidad Zaragoza, Zaragoza, Spain, e-mail: latre@unizar.es

Javier Nogueras-Iso
Universidad Zaragoza, Zaragoza, Spain, e-mail: jnog@unizar.es

Jesús Barrera
GeoSpatiumLab, Zaragoza, Spain, e-mail: jesusb@geoslab.es

F Javier Zarazaga-Soria
Universidad Zaragoza, Zaragoza, Spain, e-mail: javy@unizar.es

1 Introduction

The implementation of the INSPIRE directive must undergo the implementation of a software testing infrastructure to verify the conformance of Web based Geographic Information (GI) services with the implementing rules of INSPIRE on interoperability. Some authors, such as Bertolino (2007), describe software testing as a task "*ad hoc, expensive and unpredictably effective*". In the opinion of Canfora and Di Penta (2009), software testing is even more costly and risky when services are involved. INSPIRE stakeholders are aware that conformance testing tools for INSPIRE Web services are necessary (Bernard et al. 2005). For example, the ACE-GIS testing suite is one of the earliest examples (Esbrí et al. 2004). However, it is really very difficult to ensure an effective participation of non-technical stakeholders (end users, managers, domain experts, etc.) in the conformance testing process due to its inherent complexity. A relevant symptom is that available INPIRE tools that automate total or partially such process (e.g. GDI-DE Test suite (Hogrebe 2012), INSPIRE Metadata Validator (JRC IES/SDI Unit 2011), NeoGeo WMS INSPIRE Tester (Chartier 2011)) are targeted to technically skilled end-users with deep knowledge of UML models and XML processing tools (testers, developers, Web services experts, etc.).

This work focuses on the suitability of a *behaviour-driven development* (BDD) approach to the conformance testing of Web based GI services against the requirements of INSPIRE stakeholders. These requirements are embodied in the documents that define the technical guidance for the implementation of INSPIRE Network Services (European Commission 2013). In Software Engineering, BDD is a lightweight and non-formal model-based software development process in which software developers and domain experts collaborate in developing a human readable model of a system for acceptance tests (North 2007).

The main contributions of this paper are an analysis of the suitability of BDD techniques and tools for INSPIRE conformance testing, and the presentation of an application that implements such approach. To do so, we first discuss in section 2 existing approaches to test the conformance of Web services applicable to Web based GI services. Next, in section 3, we present how BDD can be applied to INSPIRE conformance testing, and, in section 4, we confront the BDD approach against the ISO 19105:2000 testing methodology identifying similarities and differences. Following, we present in section 5 an online test execution application for INSPIRE View and Discovery Services based on BDD. In section 6, we discuss the use of BDD for conformance testing of Web-based GI services. We conclude with some remarks on the use of a BDD approach for conformance testing of Web based GI services.

2 Related works

Conformance testing is the process to determine the extent to which a product or system conforms to the requirements of a specification with the aid of testing (Gray et al. 2010). It is acknowledged in the GI domain that the availability of conformance tests for data, metadata and services promotes and eases the adoption of interoperability initiatives (Nebert et al. 2007). Conformance testing for data and metadata often focuses on syntactic and semantic validation against schemas and rules. There are available many works about conformance testing for data and metadata in very different scenarios (e.g. domain conformance (Martirano 2013), online validation tool (JRC IES/SDI Unit 2011), metadata edition (Nogueras-Iso et al. 2012)). Service conformance testing is different from data and metadata conformance testing. Service conformance tests are built for verifying if a service behaves as it is supposed to behave according to a specification. Survey papers (e.g. Canfora and Di Penta (2009), Bozkurt et al. (2013)) show that there are a multitude of tools, testing techniques and procedures that have been proposed for testing any kind of Web services. In Europe, thanks to the INSPIRE directive, the need for tools, testing techniques and procedures suitable for Web based GI services has soared across organizations and countries recently. The most outstanding examples are the discussion platform *Persistent Test Bed* (PTB) (Östman 2010) and the testing tools developed by the *Geo Data Infrastructure Germany* (GDI-DE) (Hogrebe 2012) and the European Commission's JRC Institute for Environment and Sustainability (JRC IES/SDI Unit 2011). INSPIRE conformance testing has become also a research area. For example, Horák et al. (2011) show how to analyse performance, capacity and availability of view services. Guiliani et al. (2013) perform a similar analysis for download services. Kliment et al. (2012) and Martirano (2013) are examples of recent efforts towards a methodology for conformance testing of INSPIRE Network Services. The industry, represented by the OGC, has a program named *OGC Compliance and Interoperability Testing and Evaluation* (CITE) (Bermudez and Bacharach 2013) that has developed tools to determine a product implementation of an OGC Web service standard fulfils all mandatory elements. The CITE tools are the *Compliance Test Language* (CTL) and the TEAM Engine tool. The CTL is an XML grammar for documenting and scripting test suites that embeds XML stylesheet transformations (XSLT) and calls to native code. The TEAM Engine is a test execution tool able to run CTL files. In addition, the CITE program has developed test suites for OGC standards following the ISO 19105:2000 testing methodology. Several INSPIRE conformance testing initiatives, such as the GDI-DE Testsuite, are based on these tools and test suites.

3 BDD applied to INSPIRE conformance testing

BDD is an agile software development process in which developers, domain experts, users and stakeholders collaborate to specify in a human readable model written in a ubiquitous language the expected behaviour of a system for acceptance testing purposes. The concept of ubiquitous language describes a language built to be shared and used by developers, domain experts, users and stakeholders to promote a common understanding of the business domain (Evans 2003). This concept is fundamental in BDD. The ubiquitous language used in BDD is often referred as the Gherkin language[2] and typically follows the template for describing the behaviour of a system presented in Figure 1.

```
Feature [title]
   In order to [benefit]
   As [role]
   I want [feature]
   Scenario [title]
     Given [context]
     And [some more contexts]…
     When [event]
     And [some more events]…
     Then [outcome]
     And [some more outcomes]…
   Scenario [title]…
 Feature [title]…
```

Fig. 1. Typical Gherkin template used in BDD for depicting the behaviour of a system

Corriveau and Shi (2013) classify BDD as a *model-based testing* (MBT) tool. MBT is a kind of black-box testing where tests cases are generated from a specification, and then executed (Utting and Legeard 2010). BDD is a special case of MBT because its ubiquitous language is not formal and the automatic derivation of test cases from BDD models only outputs test stubs. As many other MBT tools, BDD is supported by a set of tools able to execute the scenarios found in BDD models. RSpec, JBehave, StoryQ, SpecFlow, Behat and Cucumber are examples of those toolkits (Solis and Wang 2011). Compared with other MBT tools, BDD can be considered too simple. Corriveau and Shi (2013) express this concern when comparing BDD with other tools based on formal modelling languages such as Spec Explorer (Veanes et al. 2008). However, the simplicity of BDD is the most probable cause of its adoption by the industry (Lerner 2010).

[2] Properly speaking, the Gherkin language is the ubiquitous language understood by the Cucumber and Behat test execution tools.

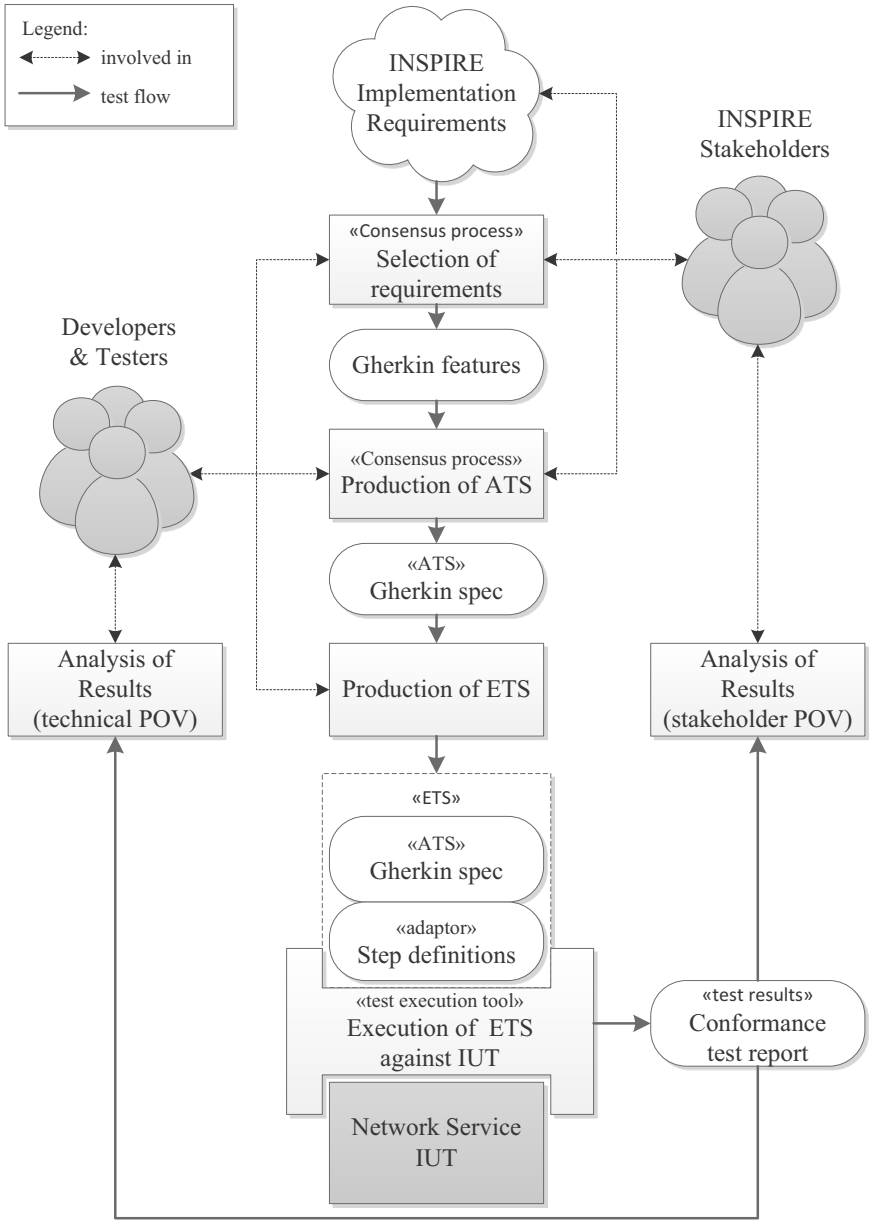

Fig. 2. BDD applied to INSPIRE Network Service conformance testing

The production of a test framework for INSPIRE conformance testing of network services based on BDD should follow the five main steps of MBT (see Figure 2).

1. Selection of requirements.
2. Production of an *abstract test suite* (ATS).
3. Production of an *executable test suite* (ETS).
4. Execution of ETS against an *instance under test* (IUT).
5. Analysis of results.

The first step is the selection of requirements. It is a process based on consensus where stakeholders, developers and testers agree on a subset of INSPIRE implementation requirements for Web based GI services whose conformance must be tested. The outcome of this process should be a high abstraction model of the behaviour of the system expressed as a set of expected features. In this context, the term feature identifies a specific desired behaviour to be tested. In MBT, this model is known as abstract model because it must not refer to specific instances (Utting and Legeard 2010). If the BDD specification language is the Gherkin language, the abstract model will be represented in plain text files where each expected feature is stored in a separate file with the ".feature" filename extension. Each file must contain a line with the keyword "Feature"[3] followed by free indented text that describes a specific behaviour to be tested. The relationship between the feature and the source requirements must be clearly documented in text to support traceability.

The second step is the production of abstract tests from the model by consensus. Since resources for testing are limited and there is an infinite number of possible tests, involved parties should agree first on some criteria to decide which and how many abstract tests should be specified. The output of this step is an ATS. Each abstract test is a sequence of operations or steps related to a behaviour that put an IUT in a state where an expected outcome should happen. In the Gherkin language, each abstract test is encoded as a scenario of the feature associated to the behaviour that the test relates. Every scenario starts with the keyword "Scenario" on a new line after a feature or scenario declaration, and is followed by a free indented text that describes the test. Every scenario consists of an ordered list of steps. Each step must start with one of the following keywords "Given", "When", "Then", "But" or "And", and followed by a free text description of the step. The purpose of the "Given" steps is to put the system in a known state, the purpose of the "When" steps is to describe a key action and finally the purpose of "Then" is to observe outcomes[4]. "But" and "And" are used to increase the readability of the abstract test. Gherkin uses tags to group features and scenarios together.

The third step is to implement the ATS into an ETS. In BDD, this is done by coding some adaptor code that implements each step described in the ATS in terms of the Web service application-programming interface of the IUTs. The

[3] We assume in this section that the behavior model will be written in plain English although the Gherkin language supported by Cucumber and Behat provides keywords for more than 40 languages.

[4] Popular BDD tools, such as Cucumber, do not distinguish semantically among these steps. This behavior has practical, strong implications discussed in next sections.

main advantage of this approach is the isolation of the ATS from the implementation details. The only requirement for reusing the ATS in a different test execution environment is to code an appropriate adaptor code. For example, if the test execution environment is Java based, the Cucumber-JVM tool provides the required Java artefacts for implementing the adaptor code; if the execution environment is .Net based, the SpecFlow tool can be used instead (Solis and Wang 2011). In addition, BDD assumes that test execution tools will automate the execution of the lists of steps found in the ATS. These tools will look up the implementation of a step in the adaptor code by some matching procedure at runtime. For example, the Cucumber tool will look for a step definition annotated with a keyword, string or regular expression that matches the text of a Gherkin step and extract from the matching text parameters for invoking the code. That is, in BDD, an ETS for a specific test execution environment is a bundle composed by an ATS and an adaptor code for such environment.

The fourth step is to execute the ETS against an IUT with an appropriate test execution tool. In BDD, the reports of the test executions are generated from the ATS bundled in the ETS. That is, the reports are expressed in human readable terms that were agreed and written by one of the final recipients of these reports: the INSPIRE stakeholders. For example, this paper presents a Web based testing tool able to execute ETS for OGC Web services. In this tool, users can select the ETS to be executed and the location of the capabilities XML of the OGC Web Service that they want to test. Moreover, each ETS is multilingual, that is, each bundle consists of an ATS written in English, an ATS written in Spanish and a shared adaptor code. Hence, the user can select which ATS drives the tests, and the INSPIRE conformance report produced by the tool will be written in the corresponding language.

Finally, the fifth step requires that involved parties analyse the human readable results of the ETS executions from their point of view. For example, when an IUT fails to pass, the main cause of the failure or error should be determined. Technical parties may use the reports to find faults in the IUT, in the testing execution tool, in the adaptor code, and even in the ATS. Non-technical parties may use reports to improve the communication with technical parties while the fault is fixed, to discover faults in the ATS that technical parties may not be aware of and, perhaps, to discover that a flawed implementation requirement is the main cause of the failure.

4 BDD and ISO 19105:2000 testing methodology

The ISO 19105:2000 testing methodology (ISO/TC 211 2000), which is based on testing methodology for software, is the conceptual framework for conformance testing in the domain of geographic information (Kresse and Fadaie 2004). Any testing framework intended to be used in the geographic information domain should be aligned to ISO 19105:2000 in order to detect its strengths and weak-

nesses. Table 1 maps key ISO 19105:2000 concepts to BDD concepts presented in the above section. In general, there is a recognizable correspondence between ISO 19105:2000 and BDD concepts. The mapping also reveals that BDD does not provide a robust mechanism for defining modules and suites yet.

Table 1. Mapping between ISO 19105:2000 concepts and BDD concepts.

ISO 19105:2000	Definition	BDD	Definition
Abstract test case	Generalized test for a particular requirement.	**Scenario**	A sequence of operations necessary to test for a particular feature
Abstract test method	Method for testing implementation independent of any particular test procedure.	**Step list**	The sequence of steps that define a scenario.
Abstract test module	Set of related abstract test cases.	**Set of tagged scenarios**	Set of scenarios or features annotated with the same tag.
Abstract test suite (ATS)	Abstract test module specifying all the requirements to be satisfied for conformance.	**Feature suite**	All the scenarios specifying all the features to be satisfied for acceptance.
Executable test case	Specific test of an implementation to meet particular requirements.	**Scenario and step definitions (adaptor code)**	A sequence of operations necessary to test for a particular feature along with its adaptor code for a particular test execution tool.
Executable test suite (ETS)	Set of executable test cases.	**Feature suite and step definitions (adaptor code)**	All the scenarios specifying all the features to be satisfied for acceptance along with their adaptor code for a particular test execution tool.

The conformance assessment process in ISO 19105:2000 involves four phases: *preparation for testing*, *test campaign*, *analysis of results* and *conformance test report*. The first three phases of BDD applied to INSPIRE Network Service conformance testing (*selection of requirements*, *production of ATS*, *production of ETS*) fall within the scope of the *preparation for testing* phase. The *execution of ETS against an IUT* phase is equivalent to the *test campaign* phase as both are the process of executing the ETS against an IUT and recording in a log the observed test outcome and any other relevant information. The shared *analysis of results* phase presents a subtle difference. In ISO 19105:2000, it refers to the evaluation of the observed test outcome against the pass and fail criteria prescribed by the abstract test case. This analysis may overlap in time with the test campaign. In BDD, an automated execution tool computes during the execution of the ETS a pass or fail test verdict automatically. Hence, the evaluation also involves confirming or overturning the computed verdict. Finally, ISO 19105:2000 identifies a *conformance test report* phase where the results of the conformance assessment process are documented in a proforma conformance test report. This phase does not exist

explicitly in the BDD approach because execution tools can generate automatically proforma test reports based on the ATS.

Although ISO 19105:2000 and BDD have similarities, BDD tools cannot be considered as mature tools yet. For example, the most popular BDD tools the Gherkin language and the Cucumber tool (Wynne and Hellesøy 2012) do not provide in its present state a complete support to the ISO 19105:2000 testing methodology. We can point out that the Gherkin language (Table 2) and the Cucumber tool (Table 3) do not support conditional requirements, inconclusive verdicts, hierarchical ATS, conformance levels and dependence between abstract tests methods. Such features can be emulated producing more complex ATS (pervasive use of tags and duplicate steps) and requires a careful analysis of results in some scenarios (risk of wrong computed verdicts).

Table 2. Issues found in the Gherkin language

ISO 19105:2000	Description	Issue	Consequences
Hierarchical abstract test modules	Abstract test modules may be nested in a hierarchical way	Lack of semantic relationship between tags.	*Tag explosion*: Nested modules can be implemented by tagging each feature or scenario belonging to these modules with tags that identify the respective container modules.
Conformance clauses with levels	A conformance level is a special class of conformance class in which requirements of a higher level contain all the requirements of the lower levels.	Lack of semantic relationship between tags.	*Tag explosion*: Lower conformance levels can be implemented by tagging each feature or scenario belonging to these levels with tags that identify the respective higher conformance levels.
Dependence among abstract test methods	An abstract test method may depend on the outcome of other abstract test methods.	No supported by the language. The sequence of operations is specific to each scenario.	*Step explosion*: Increases the complexity of the production of ATS due to the risk of an explosion of duplicate sequences of operations.

Table 3. Issues found in the Cucumber tool

ISO 19105:2000	Description	Issue	Consequences
Conditional requirements	Conformance requirements that shall be observed if the conditions set out in the specification apply	The tool does not distinguish semantically steps (e.g. Given steps do not have guard semantics).	*Wrong verdicts*: Check for wrong verdicts in conditional requirements whose guard depends on an observable value known during the execution of the steps.
Inconclusive verdict	Test verdict when neither a pass verdict nor a fail verdict apply.	The tool only supports pass or fail verdicts.	*Wrong verdicts*: Check for false pass or fail verdicts.

5 Test execution tool for INSPIRE Network Services

The approach presented in the previous section has been applied to develop a Web application able to perform an assessment on the conformity of both INSPIRE View and Discovery services[5]. The application is based on two of the most popular BDD software tools: the Gherkin language and the Cucumber-JVM test execution tool. The test execution tool was patched to solve the issues detected in the above section. Next, we describe how an end-user can interact with the application, its architecture, and the production of ATS and ETS.

This application has a landing page where the user fills in a form with the information related to an IUT, that is, a view or download OGC Web service under test: the online location of the capabilities XML document of an OGC WMS 1.3.0 or an OGC CSW 2.0.2 service, and the corresponding ETS. After the user sends the form, the application returns to the browser a master view of the conformance report labelled "in progress". In parallel, each executable test case is running or scheduled to run on the server. The application notifies the user in real time of each of the verdicts produced by the test cases and computes an overall verdict for the service (Figure 3). The user can also request for a detailed view of each executable test cases. Each executable test view displays the abstract test case, the execution trace, the execution outcome and the test verdict (Figure 4).

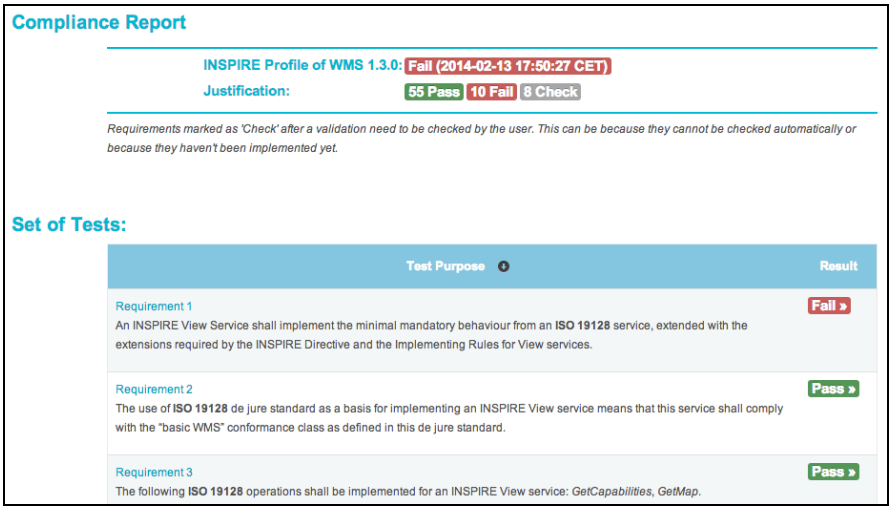

Fig. 3. Conformity test report

[5] This system is planned to be publicly available at IDEE, the SDI of Spain. At the moment of the writing, the access to the development version is restricted. Readers can request the corresponding author access to the service.

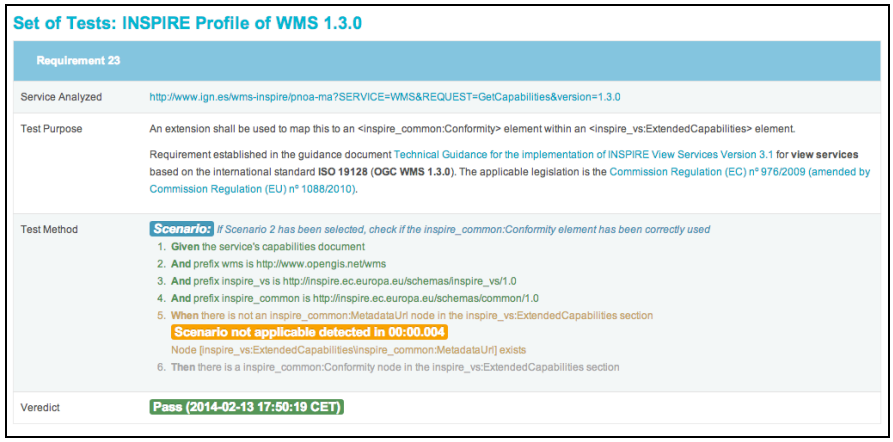

Fig. 4. Test case report

Figure 5 presents the architecture of this Web-based multilayer application. The presentation layer offers a landing page and master and detail views of live conformance test reports. The presentation layer depends on three services:

- **Conformance test builder**. Given the provided information related to the IUT, this service instantiates the appropriate ETS to be executed against the selected instance, schedules jobs to run its executable test cases (test jobs), and creates an empty conformance test report. The unique identifier of this report is returned to the user.
- **Test executor**. This service is invoked when a scheduler fires a test job. It instructs the Cucumber component to run an instantiated executable test case, records in a log its trace, its observed outcome and its verdict (pass, fail or inconclusive), and notifies the verdict to the user. However, if the test case depends on the finalization of other test cases, this service reschedules it.
- **Conformance test report**. This service provides access through unique identifiers to the conformance test reports that consist in an overall verdict and the log and the computed verdict of each test case.

As ETSs are decoupled from their adaptors, it is possible that the test executor discovers at runtime that a step is not implemented or that a feature has no scenarios. In such a case, the test case is ignored for the overall verdict and the user is notified that the test case requires human verification. The overall verdict is computed as follows:

- **Pass verdict**: a minimum number of tests are implemented and all return pass verdicts.
- **Fail verdict**: at least one implemented test returns a fail verdict.
- **Inconclusive verdict**: none of the above verdicts are met.

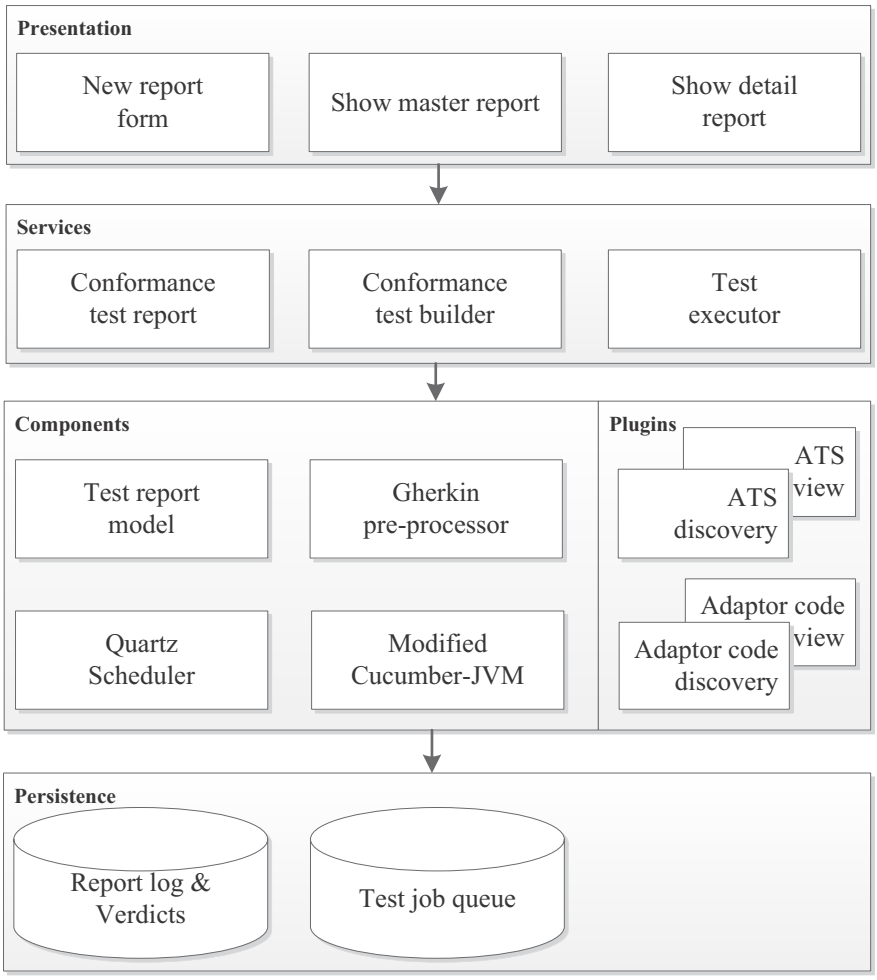

Fig. 5. Architecture diagram of a test execution tool for INSPIRE Network Services

These services use the components exposed by the component layer. The core components of the application are a Gherkin pre-processor that detects explicit dependences between test cases marked with tags (i.e. test modules), a Cucumber-JVM testing that has been modified to support conditional "Given" and "When" clauses and adaptor code that throws inconclusive verdicts, and, as plugins, multi-lingual ETS bundles with shared adaptor code written in Java. Logs, computed verdicts and the test job queue are stored in the persistence layer in a relational database.

The approach described in section 3 has been followed to produce the ETS from the most recent technical guidance documents for the implementation of INSPIRE view and discovery services. In a first stage, domain experts and test ex-

ecution tool developers decided to select all implementation requirements and create a feature per implementation requirement (73 for view services and 32 for discovery services). Next, they started the process of the production of an ATS for view services and an ATS for discovery services. Both ATS were written in English. The outcome of this process was an ATS for view services with 60 well-defined features (7 test modules, 53 test cases with test methods), 61 scenarios and 270 steps, and an ATS for download services with 25 well-defined features (6 test modules, 19 test cases with test methods), 23 scenarios and 158 steps. During this procedure, it was detected that was non-feasible to devise fully automatable scenarios for some features (13 for view services and 7 for discovery services). They were kept as part of the ATS for documentation purposes although they were not automatable. The steps from both ATS were considered for the production of the adaptor code. As many steps were duplicated or matched by the same regular expression, the 428 steps were mapped to 72 operations implemented as annotated Java methods. Once the adaptor code was ready, each ATS was translated to Spanish and the adaptor code was updated to match also the description of the steps in Spanish. Finally, all the ATS produced along with the shared adaptor code were deployed in the application. Table 4 presents a detailed summary of the testing artefacts produced.

Table 4. Testing artefacts produced

Artefact	View services	Discovery services	Total
Implementation requirements	73	32	105
Features (test modules)	7	6	13
Features (test cases with test methods)	53	19	72
Features (human verification required)	13	7	20
Scenarios	61	23	84
Steps	270	158	428
Operations (annotated Java methods)	-	-	72

6 Discussion

In this section, we discuss the use of BDD for conformance testing of Web-based GI services. Software testing intrinsically faces a lot of challenges but Web service testing faces additional issues that makes it a task of outstanding complexity. For example, Canfora and Di Penta (2009) highlight as key issues lack of observability of service code, lack of test data, complex or not fully specified input/output types, testing costs and side effects of testing. The use of a BDD-approach does not avoid dealing with such issues. For example, 20 requirements

could not be implemented because it was no agreement on a suitable sequence of operations for the identified scenarios, or because such sequence was perceived as not automatable. Similar issues can be found in other conformance testing systems for INSPIRE and, although the technical guidelines are available, it is acknowledged that there are issues that have not been addressed yet (JRC IES/SDI Unit 2011).

Other aspect to analyse is if the use of a ubiquitous language helps a better understanding of the standards, the specifications and the test methods. BDD tests suites are written in a language that have no syntactic noise and is more readable. BDD practitioners claim this feature not only improves the understanding but also ease the participation of stakeholders. There is little empirical evidence available in the literature that supports this claim. Future research needs to evaluate to which extent BDD test suites are perceived as more understandable than test suites produced by alternative approaches.

Traceability helps to understand the test case and its execution, and thus to increase the confidence of stakeholders. Traceability is the ability to relate different items involved in testing, such as requirements and tests. BDD tools provide a quite simple and straightforward support for traceability between tested requirements (*features*), abstract test cases (*scenarios*), and test implementations (adaptor code) that interact with an IUT. Similar support can be found in CITE-based tools. However, an effective development environment for conformance testing needs to support not only traceability but also the debugging of test cases. Nowadays, integrated development editors (IDE) offer an extensive support to run and debug BDD specifications and the respective adaptor code by means of plugins (Chelimsky et al. 2010). The CTL, for example, lacks of such wide support.

7 Conclusions

We have presented the progress made in the investigation of novel procedures for INSPIRE conformance testing of Web based GI services. The use of BDD for conformance testing of Web based GI services is new in this domain. As other MBT approaches, it has as advantage that authoring ATS is truly independent of the implementation of the adaptor code. In addition, non-technical stakeholders can participate in authoring ATS and could gain insights on conformance process. This work also shows that BDD is partially compatible with the ISO 19105:2000 testing methodology and has desirable qualities such as traceability and readability. Therefore, in the INSPIRE context, the adoption of BDD could facilitate a wider participation of stakeholders in the development of ATS and ensure the effective understanding of INSPIRE implementation requirements and their consequences by both technical and non-technical INSPIRE stakeholders.

Acknowledgments This work has been partially supported by the Spanish Government (project TIN2012-37826-C02-01), the National Geographic Institute (IGN) of Spain and GeoSpatiumLab S.L.

References

Bermudez L, Bacharach S (2013) Compliance Testing Program Policies & Procedures. Open Geospatial Consortium, Wayland

Bernard L, Kanellopoulos I, Annoni A, Smits P (2005) The European geoportal—one step towards the establishment of a European Spatial Data Infrastructure. Comput Environ Urban 29:15–31. doi: 10.1016/j.compenvurbsys.2004.05.009

Bertolino A (2007) Software Testing Research: Achievements, Challenges, Dreams. Future of Software Enginnering (FOSE'07), Minneapolis, 23-25 May 2007. doi: 10.1109/FOSE.2007.25

Bozkurt M, Harman M, Hassoun Y (2013) Testing and verification in service-oriented architecture: a survey. Softw Test Verif Reliab 23:261–313. doi: 10.1002/stvr.1470

Canfora G, Di Penta M (2009) Service-Oriented Architectures Testing: A Survey. In: De Lucia A, Ferrucci F (eds) Software Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 78–105

Chartier B (2011) Vos services WMS sont-ils INSPIREd? In: Neogeo Technologies. http://www.neogeo-online.net/blog/archives/1331/. Accessed 3 Dec 2013

Chelimsky D, Astels D, Dennis Z, Helmkamp B, Hellesøy A, North D (2010) The RSpec Book. The Pragmatic Bookshelf, Dallas

Corriveau J-P, Shi W (2013) On Acceptance Testing. International Conference on Software Engineering Research and Practice (SERP 2013), Las Vegas, 22-25 July 2013

Esbrí MÁ, Gould M, López ML (2004) Conformance Test Engines for quality assurance of INSPIRE Services. 10th EC-GI&GIS Workshop, Warsaw, 23-25 June 2004

European Commission (2013) Guidance Documents. In: Network Services: Legislation. http://inspire.jrc.ec.europa.eu/index.cfm/pageid/5. Accessed 4 Dec 2013

Evans E (2003) Domain-Driven Design. Addison-Wesley Professional, Boston

Giuliani G, Dubois A, Lacroix P (2013) Testing OGC Web Feature and Coverage Service performance: towards an efficient access to geospatial data. J Spat Inf Sci (In press). doi: 10.5311/JOSIS.2013.7.112

Gray M, Goldfine A, Rosenthal L, Carnahan L (2010) Conformance Testing. In: Information Technology Laboratory, NIST. http://www.nist.gov/itl/ssd/is/conformancetesting.cfm. Accessed 4 Dec 2013

Hogrebe D (2012) GDI-DE Testsuite. Improving interoperability. INSPIRE Conference, Istanbul, 23-27 June 2012

Horák J, Ardielli J, Růžička J (2011) Performance Testing of Web Map Services. In: Nguyen N, Trawiński B, Jung J (eds) New Challenges for Intelligent Information and Database Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 257–266

ISO/TC 211 (2000) ISO 19105:2000 - Geographic information - Conformance and testing. Geneva, Switzerland

JRC IES/SDI Unit (2011) INSPIRE Geoportal Metadata Validator. In: INSPIRE Geoportal. http://inspire-geoportal.ec.europa.eu/validator2/. Accessed 4 Apr 2013

Kliment T, Tuchyňa M, Kliment M (2012) Methodology for conformance testing of spatial data infrastructure components including an example of its implementation in Slovakia. Slovak Journal of Civil Engineering XX:10–20. doi: 10.2478/v10189-012-0002-y

Kresse W, Fadaie K (2004) ISO Standards for Geographic Information. Springer, Berlin

Lerner RM (2010) At the forge: Cucumber. Linux Journal 2010:7.

Martirano G (2013) The eENVplus approach for data harmonization and validation. eENVplus Workshop, INSPIRE Conference, Florence, 24 Jun 2013

Nebert D, Reed C, Wagner RM (2007) Proposal for a spatial data infrastructure standards suite: SDI 1.0. In: Onsrud H (ed) Research and Theory in Advancing Spatial Data Infrastructure Concepts. ESRI Press, Redlands, pp 147–159

Nogueras-Iso J, Latre MA, Béjar R, Muro-Medrano PR, Zarazaga-Soria FJ (2012) A model driven approach for the development of metadata editors, applicability to the annotation of geographic information resources. Data Knowl Eng 81-82:118–139. doi: 10.1016/j.datak.2012.09.001

North D (2007) Introducing Behaviour Driven Development. In: Dan North & Associates. http://dannorth.net/introducing-bdd/. Accessed 25 Nov 2013

Östman A (2010) Network for testing GI services. GIS Ostrava, Ostrava, 24-27 Jan 2010

Solis C, Wang X (2011) A Study of the Characteristics of Behaviour Driven Development. 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), Oulu, 20 Aug-2 Sept 2011. doi: 10.1109/SEAA.2011.76

Utting M, Legeard B (2010) Practical Model-Based Testing. Morgan Kaufmann, San Francisco

Veanes M, Campbell C, Grieskamp W, Schulte W, Tillmann N, Nachmanson L (2008) Model-Based Testing of Object-Oriented Reactive Systems with Spec Explorer. In: Hierons RM, Bowen JP, Harman M (eds) Formal Methods and Testing. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 39–76

Wynne M, Hellesøy A (2012) The Cucumber book : behaviour-driven development for testers and developers. The Pragmatic Bookshelf, Dallas