Una aplicación inteligible de validación de servicios INSPIRE

LOPEZ-PELLICER, Francisco J; BARRERA, Jesús; RODRÍGUEZ, Antonio F; ABAD POWER, Paloma; AGUDO MOLINA, José M; ZARAZAGA-SORIA, F Javier; JULIÃO, Rui Pedro

La aplicación de la Directiva INSPIRE requiere verificar la conformidad de un amplio número de Servicios de Red con las Normas de Ejecución de dicha directiva. Verificar la conformidad de esos servicios con INSPIRE es un proceso complejo que requiere el uso de *software* especializado que suele generar informes ininteligibles. Este tipo de *software* debería explicar cómo se ha realizado la verificación e identificar la no conformidad y sus causas de una forma comprensible. Además, esas herramientas suelen requerir para su uso un elevado grado de conocimiento técnico (programadores, administradores de sistemas, expertos en Servicios de Red, etc.) que hace muy difícil a partes interesadas no técnicas (usuarios finales, gestores, expertos en un dominio, etc.) participar de forma efectiva en el proceso de conformidad e incluso comprender las implicaciones reales de los resultados de no conformidad. Esta situación puede llegar a provocar desinterés e incluso causar el desistimiento en la resolución de los problemas de no conformidad.

Con el objetivo de facilitar la participación de todas las partes interesadas, esta comunicación presenta una aproximación para verificar la conformidad de Servicios de Red OGC basada en un proceso de desarrollo de *software* denominado BDD (*Behaviour Driven Development*). Este proceso enfatiza la participación de las partes no técnicas en el diseño de las pruebas. Se caracteriza por describir el comportamiento observable esperado de un proceso utilizando un lenguaje controlado en un documento pensado para ser leído por usuarios humanos y, al mismo tiempo, para ser ejecutado por máquinas.

Un resultado concreto de esa aproximación a la conformidad es una herramienta de conformidad de Servicios de Visualización y Descubrimiento INSPIRE que próximamente estará disponible en el geoportal de la IDEE. Las pruebas genéricas de esa herramienta están especificadas en inglés, español y portugués con la ayuda de *Gherkin*, un lenguaje natural de especificación, y pueden ser ejecutadas para validar la conformidad de un servicio concreto con la herramienta de validación *Cucumber*.

PALABRAS-CLAVE

INSPIRE, Servicios de Red, WMS, CSW, Conformidad, BDD, Gherkin, Cucumber

INTRODUCCIÓN

La aplicación de la Directiva INSPIRE requiere verificar la conformidad de determinados Servicios de Red con las Normas de Ejecución de dicha Directiva. En la practica, esta tarea asume la existencia de herramientas de *software* especializadas que automatizan total o parcialmente dicho proceso. Estas herramientas suelen requerir para su uso un elevado grado de conocimiento técnico (programadores, administradores de sistemas, expertos en Servicios de Red, etc.) que hace muy difícil a partes interesadas no técnicas (usuarios finales, gestores, expertos en un dominio, etc.) participar de forma efectiva en el proceso de conformidad e incluso comprender las implicaciones reales de los resultados de no conformidad identificados por dichas herramientas.

Este artículo desarrolla y concreta una aproximación presentada en las III Jornadas Ibéricas de Datos Espaciales [1] para verificar la conformidad de Servicios de Red. Esta aproximación, que está alineada con la Norma ISO 19105 [2], se basa en buenas prácticas de la Ingeniería de Software para el desarrollo y puesta en práctica de pruebas de aceptación denominadas *Desarrollo Guiado por el*

Comportamiento (*Behavior-driven development* o BDD) [3]. El principio rector seguido es que los intereses de las partes no técnicas involucradas en INSPIRE y la perspicacia técnica de los desarrolladores, ambos, guíen el proceso de verificación de la conformidad.

Este artículo presenta tres contribuciones. La primera contribución es un análisis de las similitudes y las diferencias con la Norma ISO 19105 del lenguaje Gerhkin[1] y de la herramienta Cucumber[2], dos de de las herramientas de software más utilizadas en BDD [4]. La segunda contribución es la descrición de una metodología sencilla para la elaboración de colecciones de pruebas genéricas (ATS) y colecciones de pruebas ejecutables (ETS) multilingües utilizando herramientas BDD. Esta metodología se ha aplicado en la elaboración de ATS y ETS para verficar la conformidad de Servicios de Visualización y de Descubrimiento con las Normas de Ejecución de la Directiva INSPIRE. Finalmente, la tercera contribución es un prototipo de aplicación Web de acceso público[3] capaz de ejecutar dichos ETS para verificar la conformidad de Servicios de Visualización y Descubrimiento concretos con las Normas de Ejecución. Una versión más avanzada se encontrará próximamente disponible en el geoportal de la IDEE.

¿QUÉ ES BDD?

Durante la incepción de una herramienta de validación de la conformidad con las Normas de Ejecución de la Directiva INSPIRE para el portal de la IDEE se llegó a la conclusión que la mejor manera de hacer útil e inteligible dicha herramienta a las partes interesadas no técnicas era seguir una aproximación basada en el *Desarrollo Guiado por el Comportamiento* (*Behavior-driven development* o BDD) [3]. BDD es un término de la Ingeniería de Software que identifica a un tipo de proceso de desarrollo de software guiado por pruebas en el que expertos en un dominio y desarrolladores de software colaboran con el objetivo de crear software con valor añadido. Aunque BDD representa la idea de que el desarrollo de software debe involucrar a los expertos en el dominio, en la práctica BDD asume que existe un conjunto de herramientas especializadas que soportan este proceso. Estas herramientas son el elemento clave de BDD y sirven para automatizar especificaciones semi-formales inteligibles que validan el comportamiento del sistema que han sido elaboradas y validadas por las partes interesadas y los desarrolladores. Es decir, si somos capaces de escribir los requisitos de implementación de las Guías Técnicas como especificaciones BDD, las herramientas BDD podrían ejecutar dichas especificaciones. Y no solo eso, si además aplicamos la idea de que el desarrollo de dichas especificaciones debe involucrar a expertos en el dominio no técnicos se conseguría el objetivo propuesto de hacer útil e inteligible la herramienta de validación.

Gherkin es uno de los formatos en el que estas especificaciones en texto plano se escriben (ficheros con extensión `.feature`). Por defecto Gherkin asume que se va a escribir en inglés la especificación y que se va a utilizar términos controlados como `Feature` (requisito), `Scenario` (prueba), `Given` (paso de un método de prueba que lleva el sistema a sus estado inicial), `When` (paso de un método de prueba) y `Then` (paso de un método de prueba que analiza un resultado) para estructurar la especificación. Otros términos como `And` y `But` sirven para dividir un método de prueba complejo en tareas simples.

Gherkin proporciona términos controlados para más de 40 idiomas. Por ejemplo, el Listado 1 muestra un ejemplo de especificación de comportamiento escrita en español en Gherkin. La

Tabla 1 nos proporciona un ejemplo de las facilidades de Gherkin ofrecidas para redactar especificaciones en diversos idiomas.

Listado 1: Una especificación del comportamiento deseado de un visualizador

```
1    # language : es
2    Característica: Visualizar KML
3      Como usuario final
4      Para poder añadir contenido personalizado en el visualizador
5      Quiero cargar ficheros KML
6
7      Escenario: Añadir un KML desde un fichero
8        Dado que he oprimido el botón "Añadir información en formato KML"
9        Y se ha abierto el diálogo "Añadir información en formato KML"
10       Cuando selecciono la opción "Desde local"
```

[1] https://github.com/cucumber/cucumber/wiki/Gherkin
[2] http://cukes.info/
[3] http://martin.cps.unizar.es:8080/servicesValidator/

```
11        Y oprimo el botón "Elegir fichero"
12        Y selecciono el fichero FabricaDeArmas.kml
13        Y oprimo el botón "Cargar"
14        Entonces el visualizador debe mostrarme un pin con el nombre "Campus
      Tecnológico de la Fábrica de Armas" en las coordenadas 39.865, -4.041
15
16      Esquema del escenario: Añadir un KML desde una URL
17        Dado que he oprimido el botón "Añadir información en formato KML"
18        Y se ha abierto el diálogo "Añadir información en formato KML"
19        Cuando selecciono la opción "Desde la Web"
20        Y escribo "<url>" en el campo "URL"
21        Y oprimo el botón "Cargar"
22        Entonces el visualizador debe mostrarme una colección denominada
      "<nombre>" con <numero> <tipo>
23
24      Ejemplos:
25        | url | nombre | numero | tipo |
26        | https://… | Campus de Toledo | 11 | polígonos |
27        | https://… | Universidad de Castilla la Mancha | 6 | puntos |
```

Tabla 1: Palabras claves de Gherkin equivalentes en inglés, español y portugués

Inglés	Español	Portugués
	#Language : es	#Language : pt
Feature	Característica	Funcionalidade | Característica | Caracteristica
Background	Antecedentes	Contexto | Cenário de Fundo | Cenario de Fundo | Fundo
Scenario	Escenario	Cenário | Cenario
Scenario Outline	Esquema del scenario	Esquema do Cenário | Esquema do Cenario| Delineação do Cenário | Delineacao do Cenario
Examples	Ejemplos	Exemplos | Cenários | Cenarios
Given	Dado | Dada | Dados | Dadas	Dado | Dada | Dados | Dadas
When	Cuando	Quando
Then	Entonces	Então | Entao
And	Y	E
But	Pero	Mas

Cucumber es una de las herramientas más populares para automatizar la ejecución de especificaciones Gherkin. Cuando Cucumber ejecuta un paso (`Given`, `When`, `Then`, `And` o `But`) en un Scenario busca la definición correspondiente de dicho paso correspondiente para ejecutarla. Una Definición de Paso (*Step Definition*) es una pieza de código anotada con un patrón o expresión regular [5]. El patrón es utilizado para enlazar dicho código con todos los pasos que dicha expresión capture, y el código será lo que Cucumber ejecute cuando evaule un paso en una especificación Gherkin. El Listado 2 muestra un ejemplo de código Java anotado como Definición de Paso aplicable al Listado 1 gracias a las anotaciones proporcionadas por la versión de Cucumber para lenguajes ejecutables sobre la máquina virtual de Java, Cucumber-JVM[4].

Listado 2: Ejemplo de Definiciones de Paso en Java

```
1   public class ImplementacionPruebas {
2
3     @Dado("que he oprimido el botón \"(.+)\"")
4     @Cuando("oprimo el botón \"(.+)\"")
5     public void oprimirUnBotonVisible(String nombre){…}
6
7     @Dado("se ha abierto el diálogo \"(.+)\"")
8     public void comprobarExisteDialogoAbierto(String nombre){…}
```

[4] https://github.com/cucumber/cucumber-jvm

```
 9
10      @Cuando("selecciono la opción \"(.+)\"")
11     public void seleccionarUnaOpcionEnUnCheckboxVisible(String nombre){…}
12
13      @Cuando("selecciono el fichero \"(.+)\"")
14      public void seleccionarFicheroUtilizandoDialogoDeSistema(String nombre){…}
15
16      @Cuando("escribo "(.+)" en el campo \"(.+)\"")
17      public void escribirEnCampoDeTexto(String texto, String etiquetaDeCampo){…}
18
19      @Entonces("el visualizador debe mostrarme un pin con el nombre \"(.+)\" en
    las coordenadas (-?\\d+\\.\\d+), (-?\\d+\\.\\d+)")
20      public void comprobarExistePinEnVisualizador(String nombre, double latitud,
    double longitud){…}
21
22      @Entonces("el visualizador debe mostrarme una colección denominada \"(.+)\"
    con (\\d+) (.+)")
23      public void comprobarVisualizaColeccion (String nombre, int numero, String
    tipo) {…}
24
25    }
```

Una vez que tenemos un la especificación de un requisito en Gherkin (fichero `.feature`) y las Definiciones de Paso correspondientes podemos utilizar Cucumber para automatizar la ejecución de la prueba del requisito. Cucumber generará un informe basado en la propia especificación del requisito que ofrecerá una traza de la ejecución de la especificación indicando llegado el caso el paso en el que se ha producido algún error (Listado 3).

Listado 3: Informe de ejecución de la especificación

```
1    $ cucumber.sh --modo implicado visualizador.feature
2    Característica: Visualizar KML
3      Como usuario final
4      Para poder añadir contenido personalizado en el visualizador
5      Quiero cargar ficheros KML
6      Escenario: Añadir un KML desde un fichero
7        Dado que he oprimido el botón "Añadir información en formato KML" (PASA)
8        Y se ha abierto el diálogo "Añadir información en formato KML" (PASA)
9        Cuando selecciono la opción "Desde local" (PASA)
10       Y oprimo el botón "Elegir fichero" (PASA)
11       Y selecciono el fichero FabricaDeArmas.kml (ERROR)
12         No se ha encontrado el fichero FabricaDeArmas.kml
13       Y oprimo el botón "Cargar"
14       Entonces el visualizador debe mostrarme un pin con el nombre "Campus Tecnológico de
15   la Fábrica de Armas" en las coordenadas 39.865, -4.041
16
```

CUCUMBER, GHERKIN Y LA NORMA ISO 19105

La Norma ISO 19105 proporciona el armazón conceptual para la verificación de la conformidad dentro del campo de la información geográfica. La definición de pruebas y su implementación para Servicios de Red siguiendo esta Norma es una tarea compleja. La iniciativa denominada Compliance Testing Program (CITE) iniciada por OGC en el año 2003 es un buen ejemplo. CITE ha desarrollado un lenguaje XML que mezcla expresiones XSLT denominado CTL [6] para especificar pruebas genéricas y su implementación. Estas pruebas pueden ser ejecutadas por la herramienta TEAM Engine, también desarrollado por el programa CITE.

TEAM Engine y CTL permiten desarrollar conjuntos de pruebas genericas y conjuntos de pruebas ejecutables en los terminos que exige la Norma ISO 19105. Es razonable plantearnos si Cucumber y Gherkin permiten también desarrollar conjuntos de pruebas conformes con dicha Norma. La Tabla 2 muestra las similitudes que existen entre los conceptos que manejan la Norma ISO 19105, Cucumber y Gherkin cuando se refieren a requisitos, pruebas y métodos. En general hay una correspondencia reconocible entre conceptos relacionados con la prueba de un requisito particular. Sin embargo Cucumber y Gherkin no proporcionan un mecanismo robusto para dar soporte a los conjuntos de pruebas definidos en la Norma. La experiencia muestra que un desarrollador puede sin mucha dificultad organizar físicamente los ficheros `.feature` emulando la estructura jerárquica de un conjunto de pruebas.

Tabla 2: Similitudes entre conceptos

ISO 19105	Definición	Cucumber/Gherkin	Definición
Requisito	Característica deseable	`Feature`	Característica deseable
Prueba Genérica	Prueba genérica para un requisito particular	`Scenario`	Prueba genérica para una característica (*Feature*) determinada
Método de prueba genérica	Método para probar implementaciones independientemente de cualquier procedimiento particular; debe incluir el criterio de veredicto de la prueba	*Step List*	Un escenario se descompone en pasos que describen textualmente acciones o condiciones; da un resultado positivo si en el periodo de pruebas todos los pasos tienen implementación (*Step Definition)* y se ejecutan todos los pasos sin detectarse una condición de error.
Prueba ejecutable	Prueba específica de una implementación para satisfacer requisitos particulares	*Step Definition*	Prueba parametrizable en un lenguaje de implementación concreto. Los parámetros se extraen de la descripción textual del paso. Una expresión regular asociada a la definición se utilizarpara determinar en tiempo de ejecución su vinculación con distintos pasos.
Módulo de pruebas genéricas	Conjunto de pruebas genéricas relacionadas	*Tagged Features*	Conjunto de características anotadas con la misma etiqueta (`@imporante`, `@obligatoria`, etc.). Las etiquetas son un mecanismo para organizar características y escenarios en Gherkin.
Conjunto de pruebas genéricas (ATS)	Módulo de pruebas genéricas que especifican todos los requisitos de conformidad que deben satisfacerse		
Conjunto de pruebas ejecutables (ETS)	Conjunto de pruebas ejecutables		Conjunto formado por una serie de *Features* y una serie de *Step Definitions* que van a ser utilizados por una herramienta para probar un sistema.

La Tabla 3 analiza las similitudes entre el proceso de evaluación propuesto en la Norma y el proceso que se sigue al utilizar Cucumber y Gherkin. Es bastante evidente que Cucumber y Gherkin poseen algunas características interesantes para su uso durante un proceso de evaluación como es el hecho de automatizar la ejecución del ETS y la generación del informe de la prueba de conformidad. Sin embargo, Cucumber incoporta en su código un determinado criterio de evaluación de resultados.

Tabla 3: Similitudes en el proceso de evaluación

ISO 19105	Descripción	Cucumber/Gherkin	Descripción
Preparación para la prueba	Elaboración de las ATS identificación de los requisitos relevantes, creación de las pruebas ejecutables, configurar parámetros de ejecución para el sistema bajo prueba.	**Preparación para la prueba**	Elaboración de las *Features*, sus *Scenarios* y *Steps*, identificar las *Features* relevantes mediante *Tags*, crear las definiciones de los pasos, configurar parámetros de ejecución del sistema bajo prueba.
Período de pruebas	Ejecución de un ETS para un sistema bajo prueba.	**Ejecución de la prueba**	En tiempo de ejecución Cucumber recorre las *Features* seleccionadas iterando la *Step List* de cada *Scenario*. Para cada paso se ejecuta la definición de paso que le corresponde determinando el éxito de la prueba. Si falla lanzando una excepción se considera un resultado negativo. La correspondencia se descubre en tiempo de ejecución buscando aquella definición de paso cuya expresión regular capture la descripción del paso.
Análisis de resultados	Evaluación de los resultados de pruebas. Se puede solapar con el período de pruebas.		
Informe de la prueba de conformidad	Documentación de los resultados de la prueba de conformidad.		La herramienta genera automáticamente informes de la prueba en los formatos deseados.

Sin embargo, aun cuando ISO 19105, Cucumber y Gherkin tienen aspectos similares, hoy por hoy, no poseen la misma capacidad expresiva que ISO 19105. Por ejemplo, podemos señalar que no tienen un buen soporte para expresar requisitos de conformidad condicionales, no disponen de una lógica ternaria en los veredictos, no pueden organizar las pruebas en jerarquías y niveles, y no hay un mecanismo para señalar la dependencia entre pruebas (Tabla 4).

Tabla 4: Limitaciones de Cucumber y Gherkin

ISO 19105	Descripción	Limitación	Implicaciones
Requisitos de conformidad condicionales	Los requisitos de conformidad condicionales deben ser observados si las condiciones establecidas en la especificación se aplican.	Los pasos *Given* no tienen el rol de guarda condicional del escenario correspondiente.	No es compatible con esta semántica aquellos requisitos condicionales que sólo se puede comprobar si se cumple o no la condición una vez comenzado el periodo de pruebas.
Veredictos no concluyentes	Un veredicto no concluyente significa que la prueba no origina ni un veredicto positivo ni negativo.	No hay soporte para veredictos no concluyentes.	Los resultados tienen que ser analizados en detalle para identificar si hay falsos veredictos positivos o negativos que en realidad son veredictos no conluyentes.
Estructura jerárquica de los módulos de pruebas genéricas	Los módulos de pruebas genéricas pueden contener a su vez otros módulos de pruebas genéricas.	Los *Tags* no estan pensados para organizar jerárquicamente las pruebas	Dificulta replicar la estructuración de un conjunto de pruebas muy jerarquizado.
Niveles de conformidad	Un nivel de conformidad es un tipo de conformidad en la que los requisitos de un nivel superior contienen todos los requisitos de niveles inferiores.	Los *Tags* no están relacionados entre sí.	Los *Tags* pueden utilizarse para identificar la pertenencia a clases y niveles de conformidad, sin embargo hay que declarar explícitamente todos los niveles de conformidad a los que una especificación pertenece.
Dependencia entre métodos de pruebas	Un método de prueba puede dependen de lel resultado de las comprobaciones de otros métodos de prueba.	No hay soporte.	Complica la elaboración de las especificaciones al producir especificaciones que se solapan.

METODOLOGÍA PARA LA ELABORACIÓN DE PRUEBAS ABSTRACTAS Y EJECUTABLES

En esta sección presentamos la metodología seguida basada en BDD para la elaboración de pruebas abstractas y de pruebas ejecutables para validar la conformidad de servicios de red de INSPIRE. Se compone de tres partes diferenciadas:

- **Elaboración de pruebas abstractas de referencia** redactadas en *Gherkin* a partir de los requisitos de implementación en el idioma original por parte de expertos en el dominio y desarrolladores.
- **Elaboración de pruebas ejecutables de referencia** compuestas por Definiciones de Paso (*Step Definitions*) en un lenguaje de programación concreto por parte de expertos en el dominio.
- **Traducción de las pruebas abstractas y las pruebas ejecutables a otro idioma** (y/o lenguaje de programación)

En una primera etapa, a partir del texto original de cada uno de los requisitos de implementación (por ejemplo el Listado 4), expertos en el dominio del problema (la parte implicada) y desarrolladores (la parte técnica) analizan cómo abordar su validación descomponiendo cada requisito en una serie de verificaciones independientes o escenarios. Cada escenario debe proporcionar un valor a la parte implicada en relación con el requisito. Este valor debe poder ser medido de forma objetiva. Los términos utilizados en la descripción del escenario, así como su método de prueba, deben ser, en la medida de lo posible, los términos que los expertos utlizarían. La redacción final de los métodos de prueba abstracta se realiza siguiendo el esquema *Given-When-*

Then propuesto en *Gherkin*. El resultado es un documento `.feature` con una `Feature` que contiene uno o varios escenarios que a su vez se descomponen en varios pasos (por ejemplo el Listado 5).

Listado 4: Texto original de un requisito de implementación

```
1  Implementation Requirement 46: Style shall be mapped to the <wms:Style> element. The
2  humanreadable name shall be mapped to the <wms:Title> element and the Unique Identifier shall be
3  mapped to the <wms:Name> element.
```

Listado 5: Interpretación consensuada entre los expertos y los desarrolladores

```
1   Feature: Requirement 46
2     Style shall be mapped to the <wms:Style> element. The humanreadable name
3     shall be mapped to the <wms:Title> element and the Unique Identifier shall
4     be mapped to the <wms:Name> element.
5
6      Scenario: Check if every Style has a Title
7        Given the service's capabilities document
8        And prefix wms is http://www.opengis.net/wms
9        Then there is a wms:Name node in each wms:Style section
10       And there is a wms:Title node in each wms:Style section
```

A partir de este momento los desarrolladores comenzarán a desarrollar las pruebas ejecutables de forma incremental mediante el desarrollo de las Definiciones de Paso. El primer paso de dicho desarrollo es la identificación de los patrones que anotan las Definiciones de Paso. Este patrón será utilizado en tiempo de ejecución por la herramienta BDD seleccionada para ejecutar dicho código al ver que el patrón captura el texto asociado a un Paso. A modo de ejemplo, la Tabla 5 muestra patrones que se pueden utilizar en las Definiciones de Paso que implementan Pasos del Listado 5.

Tabla 5: Ejemplos de patrones usados para anotar las Definiciones de Paso

Línea	Expresión regular	Acción asociada
7	`the service's capabilities document`	Si no hay una copia cacheada del documento Capabilities XML del servicio bajo test lo obtiene mediante una petición GetCapabilities
8	`prefix ([^\s]+) is ([^\s]+)`	Comprueba que en la copia cacheada del documento Capabilities XML existe un determinado prefijo (*primer grupo capturado*) y tiene asociado un determinado namespace (*segundo grupo capturado*). De la línea 8 extraerá "wms" y "http://www.opengis.net/wms"
9 y 10	`there is a ([^\s]+) node in each ([^\s]+) section`	Comprueba que en la copia cacheada del documento Capabilities XML se cumple que todos los nodos con un determinado nombre (*segundo grupo capturado*) contienen un nodo determinado (*primer grupo capturado*). De la línea 9 extraerá "wms:Name" y "wms:Style". De la línea 10 extraerá "wms:Title" y "wms:Style"

El segundo paso es la implementación de las Definiciones de Paso. Es un proceso muy dependiente del lenguaje de programación y de la herramienta BDD seleccionada. Por ejemplo si nuestro lenguaje de programación es Java y la herramienta BDD seleccionada es *Cucumber-JVM* utilizaremos las anotaciones Java `@Given`, `@Then` y `@When` para indicar qué métodos deben ser considerados como Definiciones de Paso y asociarles su correspondiente patrón. A modo de ejemplo, el Listado 6 muestra cómo se anotaría el código con los patrones identificados en la Tabla 5.

Listado 6: Anotaciones Given-Then-When en acción

```
1   public class ImplementacionPruebasWMS {
2      …
3      @Given("the service's capabilities document")
4      public void cachearCapabilities() {…}
5
```

```
6       @Given("prefix ([^\\s]+) is ([^\\s]+)")
7       public void ligarNamespace(String prefix, String namespace) {…}
8
9       @Then("there is a ([^\\s]+) node in each ([^\\s]+) section")
10      public void comprobarExistencia(String nodoHijo, String nodoPadre) {…}
11      …
12  }
13
```

Durante el periodo de prueba, el código que contiene las Definiciones de Paso debe estar situado en una localización conocida por la herramienta BDD seleccionada para que pueda ser ejecutado el código correspondiente al Paso que esté examinando la herramienta.

Todo el esquema descrito anteriormente puede ser adaptado a situaciones donde se requiera una solución multilingüe. La aproximación más natural es tomar como Implementación de Referencia (IR) ATS y ETS ya desarrollados en un idioma y traducirlos al resto de idiomas (y/o lenguajes de programación). En ese caso los pasos a seguir son:

- **Traducir pruebas abstractas**. Cada prueba abstracta de la IR expresada en *Gherkin* es traducida al nuevo idioma. Como vimos anteriormente, *Gherkin* proporciona un juego de palabras claves equivalentes en más de 40 idiomas y obliga, cuando no está en inglés, a añadir una anotación al documento para identificar el idioma en el que estará escrito (Listado 7).
- **Extraer patrones**. Se extraen nuevos patrones para anotar las Definiciones de Paso correspondientes. Si la traducción de la IR se ha hecho con cuidado es posible que cada patrón utilizado en la IR se corresponda con un único patrón en el nuevo idioma (Tabla 6).
- **Reutilizar pruebas ejecutables**. Se implementan las Definiciones de Paso reutilizando, en la medida de lo posible, las Definiciones de Paso de la IR. Este proceso es muy dependiente de la herramienta BDD seleccionada. Por ejemplo, el Listado 8 muestra que es trivial en *Cucumber-JVM* asociar un método *Java* con patrones en diferentes idiomas.

Listado 7: Equivalencia en español de la prueba abstracta

```
1   #Language:es
2   Característica: Requisito 46
3     Los estilos se encuentran emparejados en el elemento <wms:Style>.
4     El nombre legible para humanos se encuentra en el elemento <wms:Title> y
5     el identificador único se encuentra en el element

6       Escenario: Comprobar si cada estilo tiene un título
7         Dado el documento de capabilities del servicio
8         Y la URI para el prefijo wms es http://www.opengis.net/wms
9         Entonces existe un nodo wms:Name en cada sección wms:Style
10        Y existe un nodo wms:Title en cada sección wms:Style
```

Tabla 6: Equivalencia en español de los patrones usados para anotar las Definiciones de Paso

Expresión regular en Inglés	Expresión regular en Español	Cambios
the service's capabilities document	el documento de capabilities del servicio	Ninguno
prefix ([^\s]+) is ([^\s]+)	la URI para el prefijo ([^\s]+) es ([^\s]+)	Ninguno
there is a ([^\s]+) node in each ([^\s]+) section	existe un nodo ([^\s]+) en cada sección ([^\s]+)	Ninguno

Listado 8: Anotaciones Dado-Cuando-Entonces en acción

```
1   public class ImplementacionPruebasWMS {
2       …
3       @Given("the service's capabilities document")
4       @Dado("el document de capabilities del servicio ")
5       public void cachearCapabilities() {…}
6
```

```
7       @Given("prefix ([^\\s]+) is ([^\\s]+)")
8       @Dado("la URI para el prefijo ([^\\s]+) es ([^\\s]+)")
9       public void ligarNamespace(String prefix, String namespace) {…}
10
11      @Then("there is a ([^\\s]+) node in each ([^\\s]+) section")
12      @Entonces ("existe un nodo ([^\\s]+) en cada sección ([^\\s]+)")
13      public void comprobarExistencia(String nodoHijo, String nodoPadre) {…}
14      …
15  }
```

APLICACIÓN DE VALIDACIÓN

Todo lo descrito en las secciones anteriores ha sido materializado en una aplicación Web prototipo capaz de validar la conformidad de Servicios de Visualización y Descubrimiento OGC con las Normas de Ejecución de la directiva INSPIRE[5]. Una versión más avanzada y completa estará próximamente disponible en el geoportal de la IDEE. La versión en español de este prototipo, con las especificaciones en español codificadas en Gherkin, se encuentra en la dirección:

`http://martin.cps.unizar.es:8080/servicesValidator/`

La aplicación ha sido diseñada para ser multilingüe. Para acceder a una versión en otro idioma hay que añadir el parámetro `lang` con el identificador del idioma correspondiente:

`http://martin.cps.unizar.es:8080/servicesValidator/?lang=en`

Las especificaciones está localizadas, luego un cambio en el idioma del interfaz implica utilizar las especificaciones redactadas en dicho idioma. Actualmente sólo se dispone de especificaciones redactadas en español e inglés. Está en desarrollo una versión en portugués.

La aplicación ha sido desarrollada con el *framework open source* de desarrollo de aplicaciones Web Grails[6]. La arquitectura de la aplicación está descrita en la Figura 1. La herramienta BDD que la aplicación utiliza es la librería Cucumber-JVM[7]. Esta librería es una implementación de Cucumber para lenguajes que se ejecutan sobre la máquina virtual de Java (JVM). Con el objeto de aumentar la compatibilidad con ISO 19105, la librería Cucumber-JVM utilizada ha sido extendida por el componente Intérprete de especificaciones para soportar requisitos de conformidad condicionales, veredictos no concluyentes y dependencias entre métodos de prueba.

La aplicación expone tres servicios:

- **Constructor de informes**. Orquesta las pruebas necesarias para realizar un informe de conformidad. Las pruebas ejecutables se construyen a partir de una colección seleccionada de especificaciones expresadas en Gherkin programándose su ejecución como trabajos concurrentes mediante la librería Quartz Scheduler[8]. El constructor de informes crea, asigna un identificador único y da persistencia a un borrador del informe de conformidad en el que todas las pruebas ejecutables están pendientes de ejecución.
- **Ejecutor de pruebas**. Utiliza la librería *Intérprete de Especificaciones* para ejecutar una prueba ejecutable que está pendiente de ejecución a partir de la especificación Gherkin correspondiente. Al finalizar la prueba ejecutable, este servicio actualiza el estado de dicha prueba almacenando el veredicto y todos los detalles de la traza de ejecución.
- **Acceso a informes**. Proporciona acceso mediante URL únicas a los informes de conformidad y a los veredictos y detalles de ejecución de pruebas ejecutables. También es el responsable de notificar cambios en tiempo real al usuario que en ese momento esté visualizando un informe de conformidad.

[5] `http://inspire.jrc.ec.europa.eu/index.cfm/pageid/5`
[6] http://grails.org/
[7] https://github.com/cucumber/cucumber-jvm
[8] http://quartz-scheduler.org/

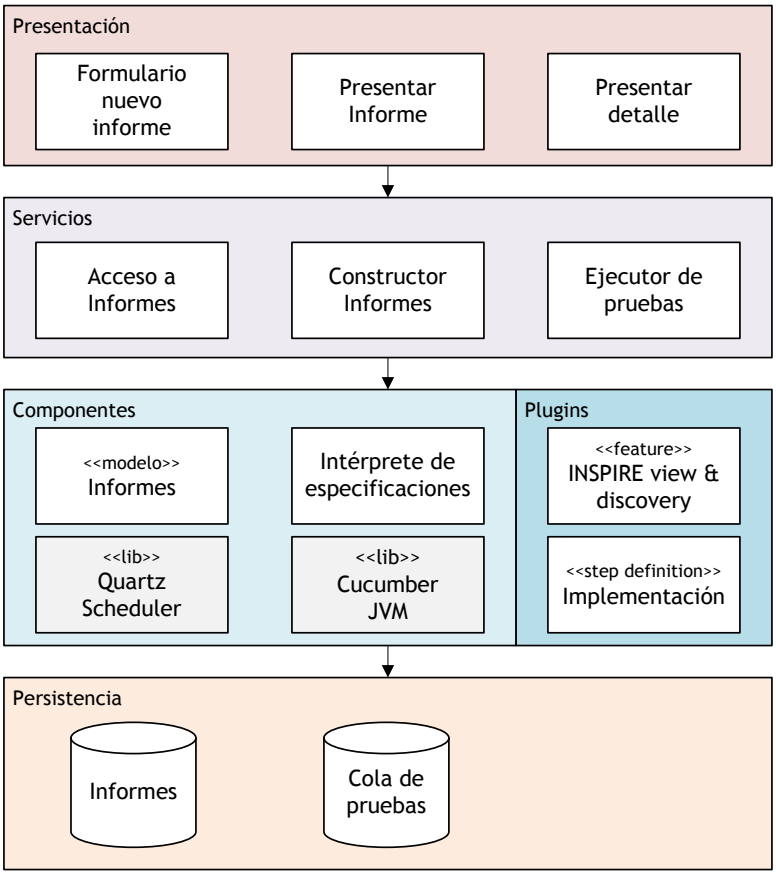

Figura 1: Diagrama de Arquitectura

A continuación se describe cómo se interactúa con esta aplicación. Primero, el usuario accede a la página principal (Figura 2) donde se encuentra con un formulario que le solicita la introducción de la localización del documento Capabilities XML de un servicio de mapas OGC WMS 1.3.0 o de un servicio de catálogo OGC CSW 2.0.2 conforme con el perfil de aplicación ISO. También le solicita que indique contra qué guía técnica INSPIRE va a comprobar la conformidad de dicho servicio.

Figura 2: Página de llegada

Con la información introducida anteriormente, la aplicación llama al servicio *Constructor de Informes* para crear un borrador de informe de conformidad con un identificador único. Este

borrador de informe es inmediatamente presentado al usuario en su navegador Web por el servicio *Acceso a Informes* (Figura 3).

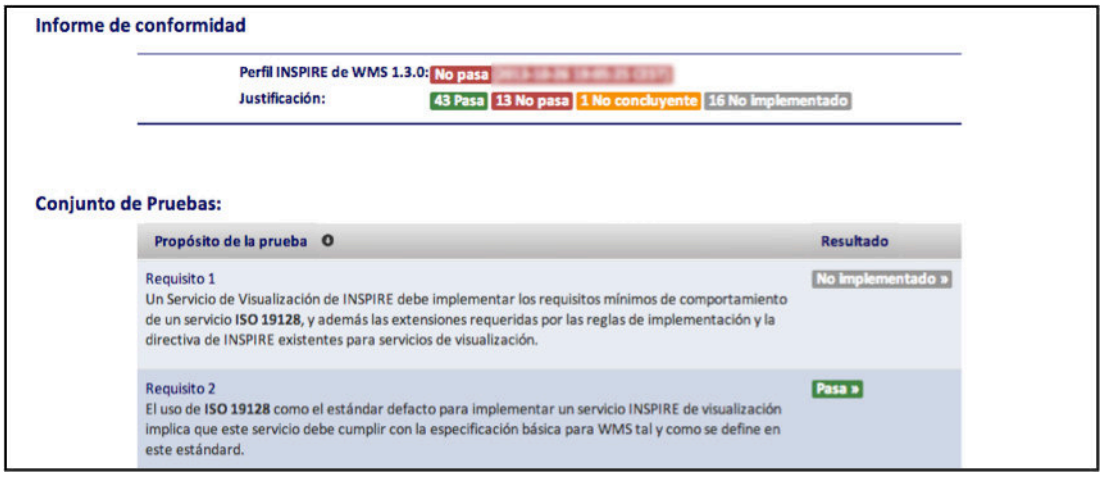

Figura 3: Informe de conformidad

Simultáneamente el servicio *Ejecutor de Pruebas* va ejecutando en un segundo plano cada una de las pruebas ejecutables asociadas al informe de conformidad. Al finalizar la ejecución de una prueba ejecutable se produce un veredicto que puede ser:

- **Pasa** (*veredicto positivo* en ISO 19105), que señala que han ejecutado correctamente todos los pasos de todos los escenarios de una especificación.
- **No concluyente** (*veredicto no concluyente* en ISO 19105), que señala habitualmente que se ha producido un problema al evaluar no achacable al sistema bajo prueba.
- **No pasa** (*veredicto negativo* en ISO 19105), que señala que algún paso no se ha pasado.
- **No implementado**, que señala que algún paso no tiene implementación.

El servicio *Acceso a Informes* mantiene informado al usuario en tiempo real de los veredictos obtenidos actualizando el borrador de informe de conformidad en su navegador. También es el responsable de calcular una valoración global de conformidad teniendo en cuenta sólo los veredictos. El criterio utilizado está recogido en la Tabla 7 y tiene tres veredictos posibles (*Pasa*, *No concluyente* y *No pasa*).

Tabla 7: Criterio de veredicto del informe de conformidad

	Pruebas ejecutables			
	Pasan	No concluyentes	No pasan	No implementadas
Pasa	Al menos una	Ninguna	Ninguna	Se ignora
No concluyente	Se ignora	Al menos una	Ninguna	Se ignora
No pasa	Se ignora	Se ignora	Al menos una	Se ignora

El servicio *Acceso a Informes* es también el responsable de que el usuario pueda acceder desde el borrador del informe de conformidad a los detalles de cada una de las pruebas ejecutables que lo componen. El detalle de una prueba ejecutable explica, utilizando la especificación escrita en Gherkin, su propósito, su método de prueba, su traza de su ejecución y el veredicto resultante (Figura 4).

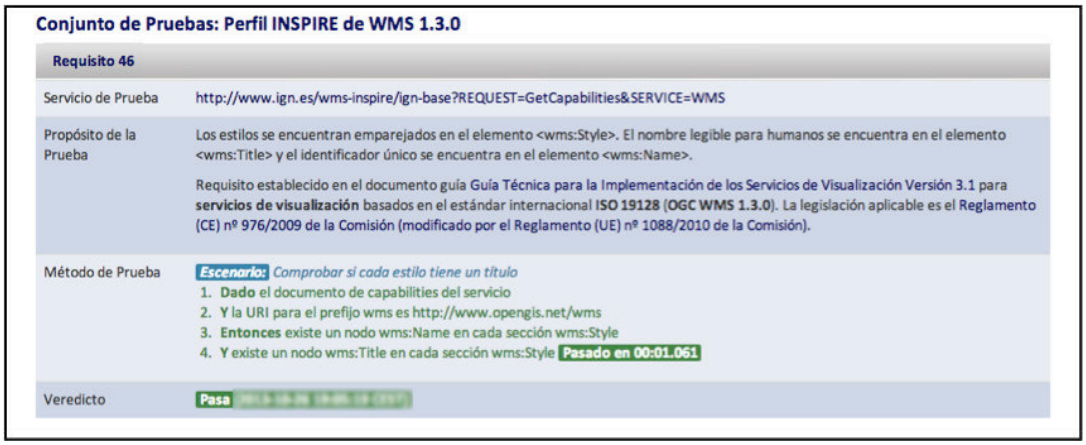

Figura 4: Detalle de prueba

CONCLUSIONES

Hemos presentado los avances realizados en la investigación de procedimientos inteligibles para la verificación de la conformidad de Servicios de Red con las Normas de Ejecución de la Directiva INSPIRE. Los resultados presentados muestran que es factible tecnológicamente aplicar una aproximación BDD a la verificación de conformidad, que es suficientemente compatible con el marco conceptual para la verificación de la conformidad establecido en la Norma ISO 19105 y, sobre todo, que esta aproximación facilita la participación de todas las partes interesadas al estar basada en la elaboración de especificaciones inteligibles.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por el Gobierno de España a través del proyecto TIN2012-37826-C02-01, del Instituto Geográfico Nacional (IGN) y de GeoSpatiumLab S.L.

REFERENCIAS

[1] F. J. Lopez-Pellicer, J. Barrera, P. Abad Power, A. Sánchez, E. López, y P. R. Muro-Medrano, "Una aproximación ágil al problema de la conformidad de servicios con INSPIRE," presentado en Actas de las III Jornadas Ibericas de Infraestructuras de Datos Espaciales JIIDE, Madrid, 2012.

[2] "ISO 19105:2000 Geographic information -- Conformance and testing," International Organization Standardization, 2000.

[3] C. Solís and X. Wang, "A Study of the Characteristics of Behaviour Driven Development," presentado en 37th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), 2011, pp. 383–387.

[4] M. Wynne y A. Hellesøy, *The cucumber book : behaviour-driven development for testers and developers*. Dallas, USA: Pragmatic Bookshelf, 2012.

[5] "Regular expression," en Wikipedia: The Free Encyclopedia. Wikimedia Foundation. 2013, http://en.wikipedia.org/wiki/Regular_expression.

[6] C. Morris, "Compliance Test Language (CTL) Best Practice," Open Geospatial Consortium Inc., OGC 06-126r2, 2009.

AUTORES

Francisco J LOPEZ-PELLICER
fjlopez@unizar.es
Universidad Zaragoza
IAAA

Jesús BARRERA
jesusb@geoslab.com
GeoSpatiumLab S.L.

Antonio F RODRÍGUEZ
afrodriguez@fomento.es
Instituto Geográfico Nacional

Paloma ABAD POWER
pabad@fomento.es
Instituto Geográfico Nacional

José M. AGUDO MOLINA
joselilo@unizar.es
Universidad Zaragoza
IAAA

F Javier ZARAZAGA-SORIA
javy@unizar.es
Universidad Zaragoza
IAAA

Rui Pedro JULIÃO
rpj@fcsh.unl.pt
Universidade Nova de Lisboa
Departamento Geografia e Planeamento Regional