# Capítulo 4
# Ingeniería Inversa

**[4. Ingeniería Inversa]**

*"El análisis de un sistema para identificar sus componentes actuales y las dependencias que existen entre ellos, para extraer y crear abstracciones de dicho sistema e información de su diseño"* [Chifofsky, 1990].

*"El proceso de analizar el código, documentación y comportamiento de un sistema para identificar sus componentes actuales y sus dependencias para extraer y crear una abstracción del sistema e información de diseño. El sistema en estudio no es alterado, sino que se produce conocimiento adicional acerca del sistema"* [SEI, 2004].

En este capítulo se trata el proceso de ingeniería inversa, sus variantes, y la realización de la misma sobre la implementación particular de un WFS desarrollado por una iniciativa privada para obtener una abstracción que nos permita generar la documentación necesaria para hacer una implementación propia de ese servicio. Como se mencionó en los alcances y limitaciones de esta tesis, el proceso de ingeniería inversa es muy amplio y debido al tiempo para desarrollar esta tesis, toda la información descrita en la sección 4.2 es para comprender el proceso, más no se realizaron todas las etapas del mismo. La sección 4.3 contiene el trabajo realizado de ingeniería inversa sobre la implementación particular del WFS que se estudió.

## 4.1 Ingeniería Inversa, un proceso de reingeniería

La ingeniería inversa tiene la misión de desentrañar los misterios y secretos de los sistemas en uso. Consiste principalmente en recuperar el diseño de una aplicación a partir del código.

Esto se realiza principalmente mediante herramientas que extraen información de los datos, procedimientos y arquitectura del sistema existente.

Es aplicable a sistemas con las siguientes características:

- Documentación inexistente o totalmente obsoleta.
- Programación en bloque de códigos muy grandes y/o sin estructurar.
- Inexistencia de documentación interna en los programas, o bien ésta es incomprensible o está desfasada.
- La aplicación cubre gran parte de los requisitos y del rendimiento esperado.
- La aplicación está sujeta a cambios frecuentes, que pueden afectar a parte del diseño.
- Se prevé que la aplicación pueda tener aún larga vida.

La ingeniería inversa puede extraer información de diseño del código fuente, pero el nivel de abstracción, la completitud de la documentación, el grado con el cual trabajan al mismo tiempo las herramientas y el analista humano, y la direccionalidad del proceso son sumamente variables [Cass, 1988].

### 4.1.1 Nivel de abstracción

El nivel de abstracción de un proceso de ingeniería inversa y las herramientas que se utilizan para realizarlo aluden a la sofisticación de la información de diseño que se puede extraer del código fuente. El nivel de abstracción ideal deberá ser lo más alto posible, es decir, el proceso de ingeniería inversa debe ser capaz de derivar:

- Sus representaciones de diseño de procedimiento (con un bajo nivel de abstracción).
- La información de las estructuras de datos y de programas (un nivel de abstracción ligeramente más elevado).
- Modelos de flujo de datos y de control (un nivel de abstracción relativamente alto)
- Modelos de entidades y de relaciones (un elevado nivel de abstracción).

A medida que crece el nivel de abstracción se proporciona al ingeniero de software información que le permitirá comprender más fácilmente estos programas [Pressman, 2003].

### 4.1.2 Completitud

La completitud de un proceso de ingeniería inversa alude al nivel de detalle que se proporciona en un determinado nivel de abstracción. En la mayoría de los casos, la completitud decrece a medida que aumenta el nivel de abstracción. Por ejemplo, dado un listado del código fuente, es relativamente sencillo desarrollar una representación de diseño de procedimientos completa. También se pueden derivar representaciones sencillas del flujo de datos, pero es mucho más difícil desarrollar un conjunto completo de diagramas de flujo de datos o un diagrama de transición de datos.

La completitud mejora en proporción directa a la cantidad de análisis efectuado por la persona que está efectuando la ingeniería inversa [Pressman, 2003].

### 4.1.3 Interactividad

La interactividad alude al grado con el cual el ser humano se “integra” con las herramientas automatizadas para crear un proceso de ingeniería inversa efectivo. En la mayoría de los casos, a medida que crece el nivel de abstracción, la interactividad deberá incrementarse, o si no la completitud se verá reducida [Pressman, 2003].

### 4.1.4 Direccionalidad

Si la direccionalidad del proceso de ingeniería inversa es monodireccional, toda la información extraída del código fuente se proporcionará a la ingeniería del software que podrá entonces utilizarla durante la actividad de mantenimiento. Si la direccionalidad es bidireccional, entonces la información se suministrará a una

herramienta de reingeniería que intentará reestructurar o regenerar el viejo programa [Pressman, 2003].

### 4.1.5 El proceso de ingeniería inversa

El proceso de ingeniería se representa en la figura 4.1. Antes de que puedan comenzar las actividades de ingeniería inversa, el código fuente no estructurado ("sucio") se reestructura para que solamente contenga construcciones de programación estructurada[1]. Esto hace que el código fuente sea más fácil de leer, y es lo que proporciona la base para todas las actividades subsiguiente de ingeniería inversa.

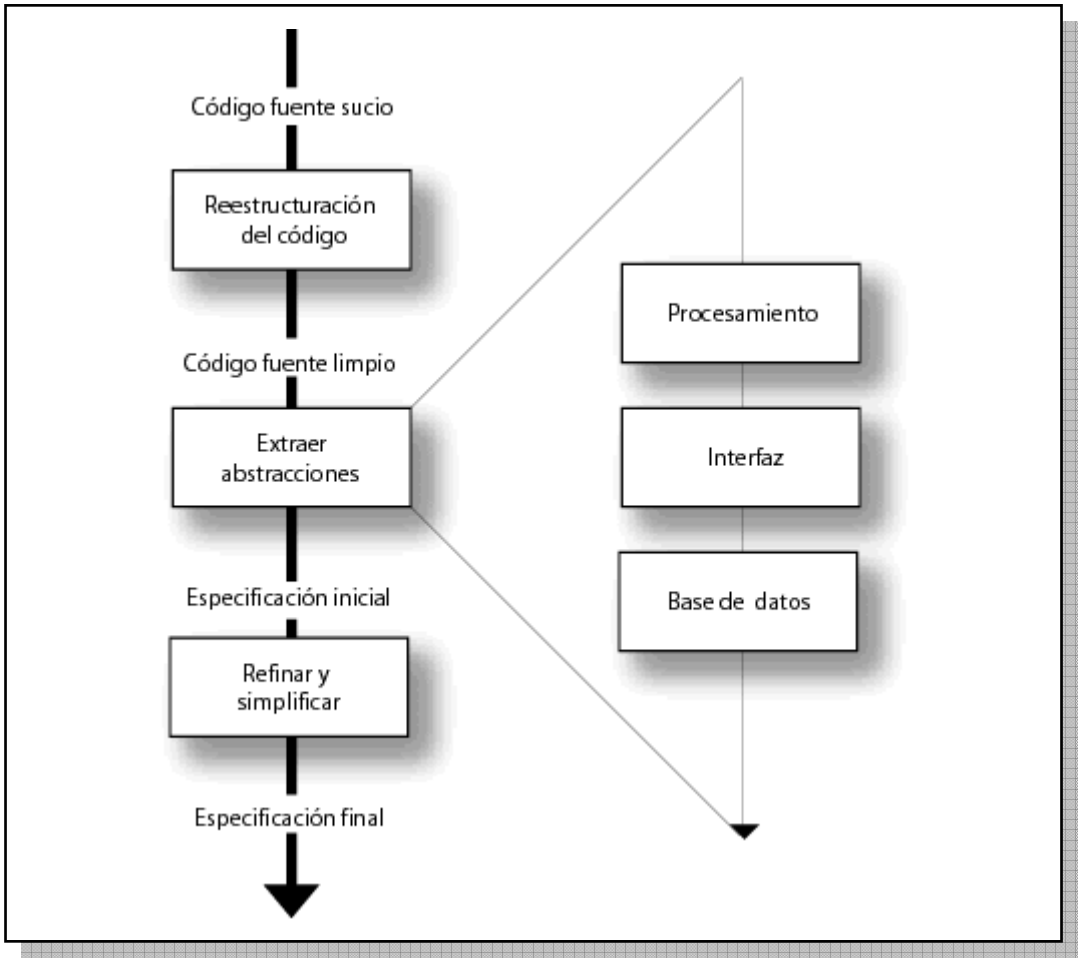



**Figura 4.1 – El proceso de ingeniería inversa**

[1] El código se puede reestructurar automáticamente empleando un "motor de reestructuración", es decir, una herramienta CASE que reestructura el código fuente.

El núcleo de la ingeniería inversa es una actividad denominada *extracción de abstracciones*. El ingeniero tiene que evaluar el viejo programa y a partir del código fuente (que no suele estar documentado) tiene que extraer una especificación significativa del procesamiento que se realizará, la interfaz de usuario que se aplica y las estructuras de datos de programa o de base de datos que se utiliza [Pressman, 2003].

Para el caso específico de esta tesis, se analizó y se llevo a cabo la abstracción para comprender el **procesamiento**, por lo tanto, solo trataremos esa parte del proceso de ingeniería inversa del software.

#### 4.1.5.1 Ingeniería inversa para comprender el procesamiento

La primera actividad real de la ingeniería inversa comienza con un intento de comprender y extraer después abstracciones de procedimientos representadas por el código fuente. Para comprender las abstracciones de procedimientos, se analiza el código en distintos niveles de abstracción: sistema, programa, componente, configuración y sentencia.

La funcionalidad general de todo el sistema de aplicaciones deberá ser algo perfectamente comprendido antes de que tenga lugar un trabajo de ingeniería inversa más detallado. Esto es lo que establece un contexto para un análisis posterior, y se proporciona ideas generales acerca de los problemas de interoperabilidad entre aplicaciones dentro del sistema. Cada uno de los programas de que consta el sistema de aplicaciones representará una abstracción funcional con un elevado nivel de detalle. También se creará un diagrama de bloques como representación de la iteración entre estas abstracciones funcionales. Cada uno de los componentes efectúa una sub-función, y representa una abstracción definida de procedimientos. En cada componente se crea una narrativa de procesamiento. En algunas situaciones ya existen especificaciones de sistema, programa y componente.

Cuando ocurre tal cosa, se revisan las especificaciones para evaluar si se ajustan al código existente[2].

Todo se complica cuando se considera el código que reside en el interior del componente. El ingeniero busca las secciones de código que representan las configuraciones genéricas de procedimientos. En casi todos los componentes, existe una sección de código que prepara los datos para su procesamiento (dentro del componente), una sección diferente de código que efectúa el procesamiento y otra sección de código que prepara los resultados del procesamiento para exportarlos de ese componente. En el interior de cada una de estas secciones, se encuentran configuraciones más pequeñas.

Por ejemplo, suele producirse una verificación de los datos y una comprobación de los límites dentro de la sección de código que prepara los datos para su procesamiento.

Para los sistemas grandes, la ingeniería inversa suele efectuarse mediante el uso de un enfoque semi-automatizado. Las herramientas CASE se utilizan para "analizar" la semántica del código existente. La salida de este proceso se pasa entonces a unas herramientas de reestructuración y de ingeniería directa que completarán el proceso de reingeniería [Pressman, 2003].

### 4.1.6 Reestructuración

*"La transformación desde una forma de representación a otra en el mismo nivel de abstracción, preservando las características externas del sistema (funcionalidad y semántica)"***.** [Chifofsky, 1990]

La reestructuración del software modifica el código fuente y/o los datos en un intento de adecuarlo a futuros cambios. En general, la reestructuración no modifica la arquitectura global del programa. Tiene a centrarse en los detalles de diseño de módulos individuales y en estructuras de datos locales definidas dentro de los

[2] Con frecuencia, las especificaciones escritas en fases centradas de la historia del programa no llegan a actualizarse nunca. A medida que se hacen cambios, el código deja de satisfacer esas especificaciones.

módulos. Si el esfuerzo de la reestructuración se extiende más allá de los límites de los módulos y abarca la arquitectura del software, la reestructuración pasa a ser ingeniería directa (*forward engineering*) [Pressman, 2003].

Arnold [Arnold, 1989] define un cierto número de beneficios que se pueden lograr cuando se reestructura el software:

- Programas de mayor calidad – con mejor documentación y menos complejidad, y ajustados a las prácticas y estándares de la ingeniería del software moderna.
- Reduce la frustración entre ingenieros del software que deban trabajar con el programa, mejorando por tanto la productividad y haciendo más sencillo el aprendizaje.
- Reduce el esfuerzo requerido para llevar a cabo las actividades de mantenimiento.
- Hace que el software sea más sencillo de comprobar y de depurar.

La reestructuración se produce cuando la arquitectura básica de la aplicación es sólida, aun cuando sus interioridades técnicas necesiten un retoque. Comienza cuando existen partes considerables del software que son útiles todavía, y solamente existe un subconjunto de todos los módulos y datos que requieren una extensa modificación[3] [Pressman, 2003].

### 4.1.7 Redocumentación

La redocumentación es también una forma de ingeniería inversa. Es el proceso mediante el que se produce documentación retroactivamente desde un sistema existente. Si la redocumentación toma la forma de modificación de comentarios en el código fuente, puede ser considerada una forma suave de reestructuración. Sin embargo, puede ser considerada como una sub-área de la ingeniería inversa porque la documentación reconstruida es usada para ayudar al conocimiento del programa. Se piensa en ella como una transformación desde el código fuente a pseudocódigo

[3] En algunas ocasiones, resulta difícil distinguir entre una reestructuración extensa y un nuevo desarrollo. Ambos son casos de reingeniería.

y/o prosa, esta última considerada como más alto nivel de abstracción que la primera.

Aunque la aparición de multitud de herramientas facilita las labores de la ingeniería inversa, es la labor humana (*humanware*) la decisiva a la hora de completar el estudio del sistema [Tilley, 1995].

### 4.2 Caso particular: WFS Geoserver

El proyecto Geoserver v. 1.1.1 [Geoserver, 2004] consiste en la implementación de un WFS basado en la especificación de OpenGIS [WFS, 2002]. Es gratuito y está disponible bajo la licencia GPL 2.0 en la dirección:

http://sourceforge.net/project/showfiles.php?group_id=25086

Esta implementación se mostró bastante robusta cuando se probó, y haciendo uso de la licencia GPL se decidió que encajaba como solución al problema propuesto. Al estudiarlo más a fondo, descubrimos que carecía de información del sistema, incluso el código tenía una documentación muy escasa, casi nula. Dadas estas circunstancias se decidió realizar un proceso de ingeniería inversa con la cual, mediante abstracciones a partir del código fuente, se pudiera obtener información importante acerca de la implementación, como diseño de implementación, diagramas, paquetes y demás información que sea necesaria para el desarrollador en caso de querer modificar en un futuro la aplicación.

Se entabló comunicación con una de las personas encargadas del proyecto Geoserver y se le comentó la idea de tener una implementación propia de la que desarrollaron ellos. El se mostró interesado, pues al ser aún un proyecto, cualquier tipo de retroalimentación que se les proporcionará sería de gran ayuda. Una de los principales intereses fue la idea de implementar el sistema sobre una arquitectura no probada, como la que tenemos en la UDLA-P. Con esto se logra, no sólo tener una solución robusta a los problemas planteados con esta tesis, sino que además participamos en el desarrollo del proyecto Geoserver.

En la actualidad no hay un documento que exija o que evalúe cuales son los pasos para una ingeniería inversa satisfactoria. Así que a continuación se propone una metodología que sirva como pautas para orientar en el proceso de ingeniería inversa a quien así lo desee.

### 4.2.1 Metodología

Debido a que no hay un documento que especifique como es exactamente un proceso de ingeniería inversa, cada ingeniero de software que desea realizar un proceso de este tipo propone su propia metodología. El instituto de ingeniería de software propone un marco de trabajo para llevar a cabo un proceso de ingeniería inversa [SEI, 2004]. Tomando en cuenta este documento y con la ayuda de un experto en la materia se propuso la metodología de la figura 4.2 [fuente propia] para realizar un proceso de ingeniería inversa. Principalmente lo que incluye la metodología propuesta es:

1. Definición y delimitación del componente de *software*
2. Recolección de funcionalidades existente
    a. Documentación de Casos de Uso
3. Generación automática de diagramas de clase y de paquete mediante una herramienta semi-autimatizada CASE
    a. Diagramas de clase
    b. Diagramas de paquete
4. Análisis de código y relación entre paquetes
5. Mapeo de requerimientos a paquetes y de ser posible a clases

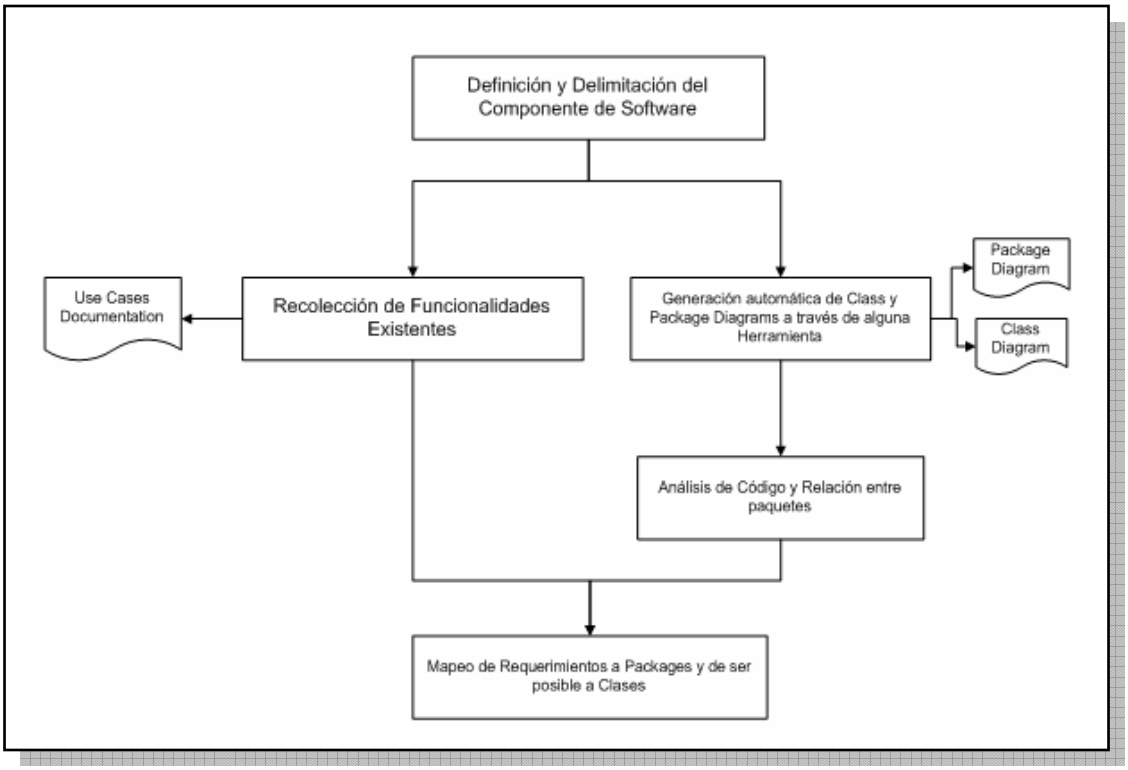



**Figura 4.2 – Metodología propuesta para el proceso de ingeniería inversa**

#### 4.2.1.1 Definición y delimitación

Esta es la primera etapa del proceso de ingeniería inversa y es la que comprende la selección y evaluación del o de los componentes a los que se les aplicará el proceso de ingeniería inversa. Geoserver fue escogido primeramente por ser una implementación pública que cumple con los requisitos que se necesitan para nuestra solución. Algunos de estos requisitos es que está hecho en Java, es *OpenSource*, cumple con la especificación que rige las implementaciones WFS, maneja peticiones por HTTP GET y HTTP POST.

#### 4.2.1.2 Casos de Uso

Para el proceso de recolección de funcionalidades existentes se desarrollaron los casos de uso de las tres operaciones que soporta el WFS. La figura 4.3 muestra el

diagrama de caso de uso que incluye a estas tres operaciones y después están los casos de uso de cada operación.

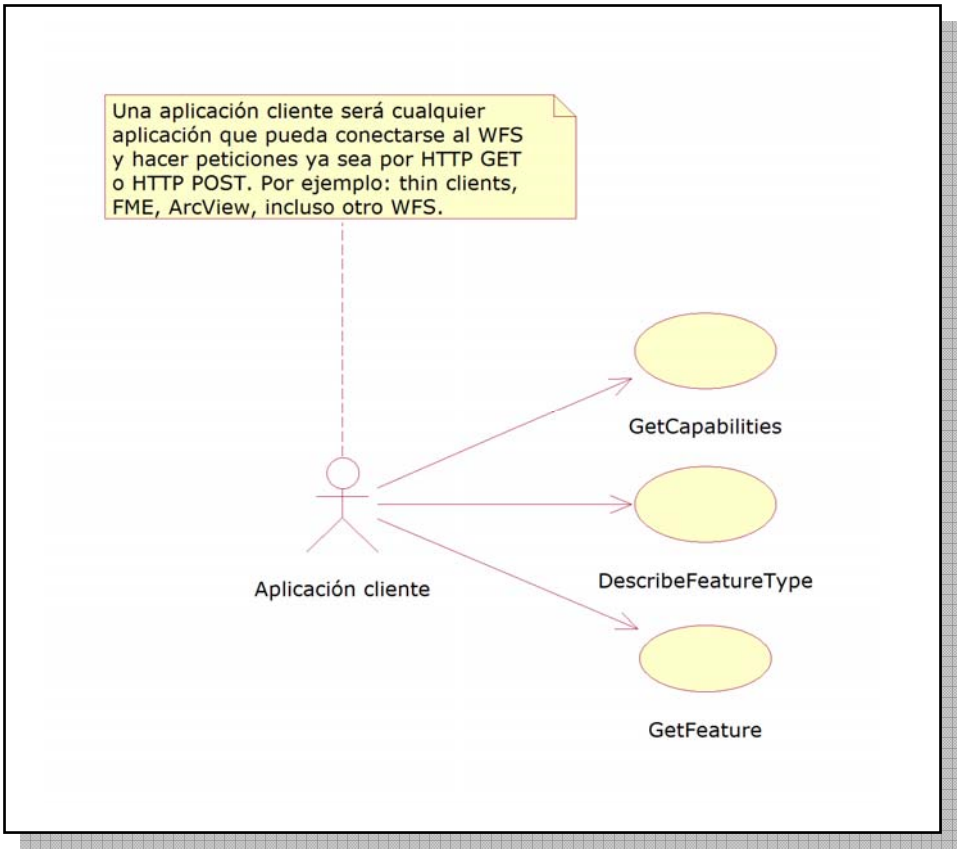



**Figura 4.3 – Diagrama de caso de uso de las tres operaciones soportadas por el WFS**

# Reingeniería para la implementación de un WFS
# Use-Case: Operación GetCapabilities

**Versión 1.0**

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 12/04/04 | Draft_a | Operación GetCapabilities | Abraham A. López Ameneyro |
| 14/04/04 | Draft_b | Correcciones | Abraham A. López Ameneyro |
| 15/04/04 | 1.0 | Versión final | Abraham A. López Ameneyro |

# Use-Case: Operación GetCapabilities

### *1 Propósito*

El propósito de este caso de uso es explicar de manera textual y posteriormente gráfica, el uso de la operación GetCapabilities.

### *1.1 Alcance*

Este caso de uso es la descripción de una de las tres operaciones que soporta el servicio WFS, por lo tanto guarda una relación con la descripción de las restantes dos operaciones (GetFeature y DescribeFeatureType).

### *1.2 Definiciones, Acrónimos y Abreviaciones*

- WFS – Web Feature Service
- OGC – OpenGIS Consortium
- GML – Geography Markup Language
- XML – eXtensible Markup Language
- CQL – Common Query Language
- Feature – Abstracción de un evento del mundo real, y que se define como un conjunto de propiedades, puede ser pensado como una tupla {nombre, tipo, valor}
- Interface – Conjunto de operaciones que caracterizan el comportamiento de una entidad

- Aplicación cliente - Una aplicación cliente será cualquier aplicación que pueda conectarse al servidor WFS y hacer peticiones ya sea por HTTP GET o HTTP POST. Por ejemplo: thin clients, FME, ArcView, incluso otro WFS.

### *1.3 Referencias*

- OpenGIS Web Feature Service Implementation Specification

http://www.opengis.org/docs/02-058.pdf

- OpenGIS Filter Encoding Implementation Specification

http://www.opengis.org/docs/02-059.pdf

- OpenGIS Catalogue Interface Implementation Specification

http://www.opengis.org/docs/02-087r3.pdf

## 2. Operación GetCapabilities

### *2.1 Descripción Breve*

Esta operación se encarga de presentar al usuario un documento, el cual describe todas las capacidades (operaciones y features) del servicio WFS.

## 3. Flujo de eventos

### *3.1 Flujo básico*

*3.1.1 <GetCapabilities>*

El caso de uso inicia cuando una aplicación cliente solicita (E1) el documento que describe las capacidades del servicio. El servidor recibe la petición, genera el documento de capacidades el cual debe estar definido y validado contra el esquema A.5 del apéndice del documento OpenGIS Web Feature Service Implementation Specification [WFS, 2002], y lo envía a la aplicación cliente para su visualización, con esto termina este caso de uso.

### *3.2 Flujos alternativos*

*3.2.1 <GetCapabilities Exception>*

(E1) La aplicación cliente puede formar mal la petición para el documento de capacidades, a lo que el servidor debe responder con una excepción de servicio.

## 4. Categoría

Core

## 5. Requerimientos especiales

- La interface debe estar definida en XML y se define con respecto al esquema A.3 del documento OpenGIS Web Feature Service Implementation Specification [WFS, 2002]
- Las bases de datos deben ser opacas a la aplicación cliente
- La solicitud debe definirse en XML y ser derivada de CQL
- Se debe usar esquemas XML (SCHEMAS) que son documentos GML válidos, para la descripción del servicio

# Reingeniería para la implementación de un WFS Use-Case: Operación DescribeFeatureType

**Versión 1.0**

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 12/04/04 | Draft_a | Operación DescribeFeatureType | Abraham A. López Ameneyro |
| 15/04/04 | Draft_b | Correcciones | Abraham A. López Ameneyro |
| 16/04/04 | 1.0 | Versión final | Abraham A. López Ameneyro |

# Use-Case: Operación DescribeFeatureType

### *1. Propósito*

El propósito de este caso de uso es explicar de manera textual y posteriormente gráfica, el uso de la operación DescribeFeatureType.

### *1.1 Alcance*

Este caso de uso es la descripción de una de las tres operaciones que soporta el servicio WFS, por lo tanto guarda una relación con la descripción de las restantes dos operaciones (GetFeature y GetCapabilities).

### *1.2 Definiciones, Acrónimos y Abreviaciones*

- WFS – Web Feature Service
- OGC – OpenGIS Consortium
- GML – Geography Markup Language
- XML – eXtensible Markup Language
- CQL – Common Query Language
- Feature – Abstracción de un evento del mundo real, y que se define como un conjunto de propiedades, puede ser pensado como una tupla {nombre, tipo, valor}
- Interface – Conjunto de operaciones que caracterizan el comportamiento de una entidad

- Aplicación cliente - Una aplicación cliente será cualquier aplicación que pueda conectarse al servidor WFS y hacer peticiones ya sea por HTTP GET o HTTP POST. Por ejemplo: thin clients, FME, ArcView, incluso otro WFS.

### *1.3 Referencias*

- OpenGIS Web Feature Service Implementation Specification

http://www.opengis.org/docs/02-058.pdf

- OpenGIS Filter Encoding Implementation Specification

http://www.opengis.org/docs/02-059.pdf

- OpenGIS Catalogue Interface Implementation Specification

http://www.opengis.org/docs/02-087r3.pdf

## 2. Operación DescribeFeatureType

### *2.1 Descripción breve*

Esta operación se encarga de presentar al usuario un documento en el cual describe como el servicio WFS espera que se codifiquen los features en las consultas, y como será la salida generada por el servicio WFS.

## 3. Flujo de eventos

### *3.1 Flujo básico*

*3.1.1 <DescribeFeatureType>*

El caso de uso inicia cuando una aplicación cliente solicita (E1) el documento que describe como se codificarán los features tanto en la entrada como en la salida. El servidor recibe la petición y genera el esquema que describe como se encuentra estructurado el feature solicitado el cual es un GML válido. Finalmente lo envía a la aplicación cliente y con esto termina este caso de uso.

### *3.2 Flujos alternativos*

*3.2.1 <DescribeFeatureType Exception>*

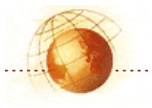
(E1) La aplicación cliente puede formar mal la petición del documento de descripción de features, a lo que el servidor debe responder con una excepción de servicio.

## 4. Categoría

Core

## 5. Requerimientos especiales

- La interface debe estar definida en XML y se define con respecto al esquema A.3 del documento OpenGIS Web Feature Service Implementation Specification [WFS, 2002]
- La solicitud debe definirse en XML y ser derivada de CQL
- Se deben usar esquemas XML (SCHEMAS) que son documentos GML válidos, para la descripción de los features



# Reingeniería para la implementación de un WFS
# Use-Case: Operación GetFeature

**Versión 1.0**

## Historia de revisiones

| Fecha | Versión | Descripción | Autor |
|---|---|---|---|
| 12/04/04 | Draft_a | Operación GetFeature | Abraham A. López Ameneyro |
| 15/04/04 | Draft_b | Correcciones | Abraham A. López Ameneyro |
| 15/04/04 | 1.0 | Versión final | Abraham A. López Ameneyro |

## Use-Case: Operación GetFeature

### *1. Propósito*

El propósito de este caso de uso es explicar de manera textual y posteriormente gráfica, el uso de la operación GetFeature.

### *1.1 Alcance*

Este caso de uso es la descripción de una de las tres operaciones que soporta el servicio WFS, por lo tanto guarda una relación con la descripción de las restantes dos operaciones (GetCapabilities y DescribeFeatureType).

### *1.2 Definiciones, Acrónimos y Abreviaciones*

- WFS – Web Feature Service
- OGC – OpenGIS Consortium
- GML – Geography Markup Language
- CQL – Common Query Language
- XML – eXtensible Markup Language
- Feature – Abstracción de un evento del mundo real, y que se define como un conjunto de propiedades, puede ser pensado como una tupla {nombre, tipo, valor}
- Interface – Conjunto de operaciones que caracterizan el comportamiento de una entidad
- Aplicación cliente - Una aplicación cliente será cualquier aplicación que pueda conectarse al servidor WFS y hacer peticiones ya sea por HTTP GET o HTTP POST. Por ejemplo: thin clients, FME, ArcView, incluso otro WFS.

### *1.3 Referencias*

- OpenGIS Web Feature Service Implementation Specification

http://www.opengis.org/docs/02-058.pdf

- OpenGIS Filter Encoding Implementation Specification

http://www.opengis.org/docs/02-059.pdf

- OpenGIS Catalogue Interface Implementation Specification

http://www.opengis.org/docs/02-087r3.pdf

## 2. Operación GetFeature

### *2.1 Descripción breve*

Esta operación se encarga de hacer la consulta y recuperación de los features disponibles del servicio WFS.

## 3. Flujo de eventos

### *3.1 Flujo básico*

*3.1.1 <GetFeature>*

El caso de uso inicia cuando una aplicación cliente genera una consulta (E1) y la envía al servidor. El servidor recibe la petición, la interpreta y la resuelve, generando un documento con el resultado de la consulta el cual es un GML válido y debe también ser válido contra el esquema generado por la operación DescribeFeatureType. Finalmente se envía a la aplicación cliente. Con esto termina este caso de uso

### *3.2 Flujos alternativos*

*3.2.1 <GetFeature Exception>*

(E1) La aplicación cliente puede formar mal la consulta, a lo que el servidor debe responder con una excepción de servicio.

## 4. Categoría

Core

## 5. Requerimientos especiales

- La interface debe estar definida en XML y se define con respecto al esquema A.3 del documento OpenGIS Web Feature Service Implementation Specification [WFS, 2002]
- La consulta o filtro debe definirse en XML y ser derivada de CQL
- Se debe usar GML para expresar los features dentro de la interface

#### 4.2.1.3 Generación automática de diagramas de clase

Esta parte del proceso de ingeniería inversa comprende la generación de los diagramas de clase y/o de paquetes usando una herramienta CASE semi-automatizada, como por ejemplo JBuilder 9.0, Enterprise Architect o Rational Rose. En nuestro proceso de ingeniería inversa utilizamos JBuilder para generar los diagramas de clase. Debido al gran número de diagramas generados, sólo se muestra uno representativo en este documento, sin embargo todos están disponibles en línea en este mismo proyecto Gisweb. La figura 4.4 es un ejemplo de diagrama de clase generado por una herramienta CASE semi-automatizada, en este caso JBuilder. Este describe la clase WFService del paquete org.vfny.geoserver.servlets y muestra sus relaciones de dependencia y extensiones. Más detalladamente muestra que hereda de la clase *AbstractService* y que depende de las clases *ExceptionHandler* y *WfsExceptionHandler*. Esta clase es muy importante, el paquete org.geoserver.servlets.wfs que contiene los *servlets* principales de este servicio, hereda de esta clase.

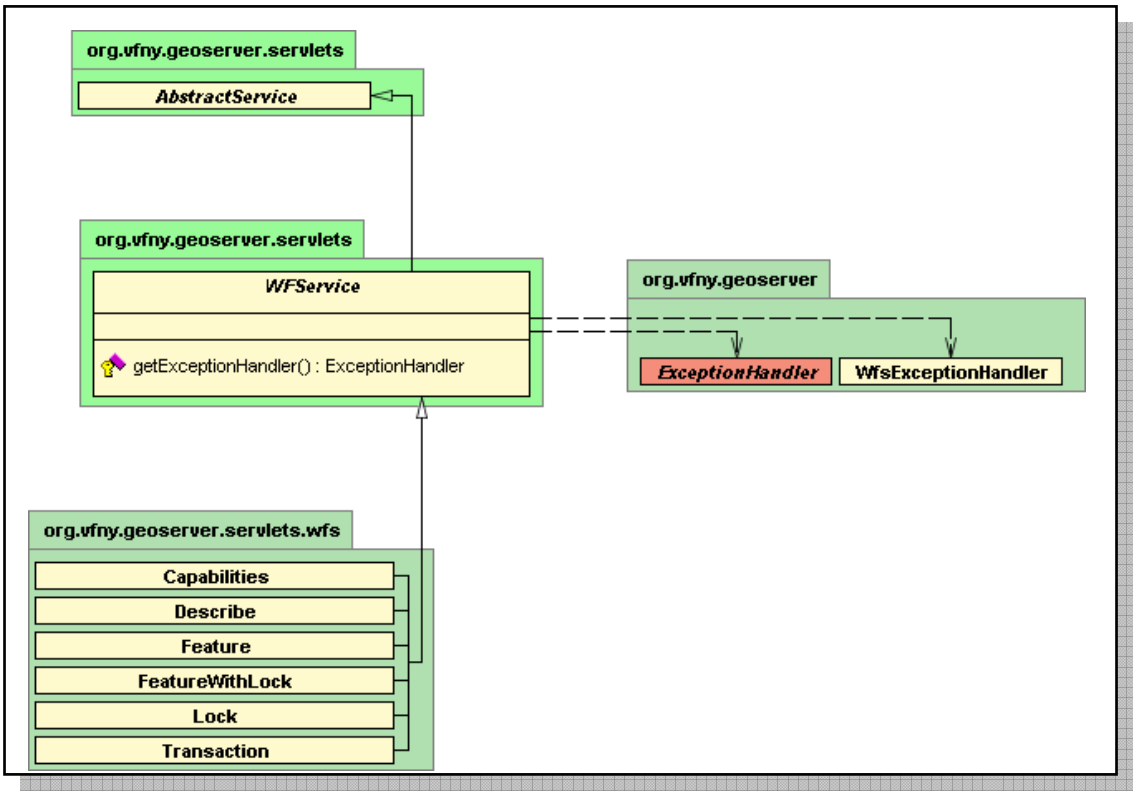



**Figura 4.4 – Ejemplo de diagrama de clase**

#### 4.2.1.4 Análisis del sistema

Esta parte del proceso de ingeniería inversa consiste en el estudio por parte del analista para descubrir con ayuda de la información obtenida hasta el momento el funcionamiento del sistema, pero a un nivel más detallado. Este es tal vez la parte de la metodología más pesada porque entra la pericia del analista para descubrir como funciona el sistema a partir de la poca o nula información que posee. Para ayudarnos y seguir contribuyendo a la documentación de esta implementación en el proceso de análisis se desarrollaron los siguientes diagramas de secuencia.

El primer diagrama de secuencia mostrado en la figura 4.5 pertenece al caso de uso de la operación *GetCapabilities* y es la comunicación de mensajes a alto nivel para responder la petición hecha por la aplicación cliente. El diagrama de la figura 4.6 pertenece al caso de uso de la operación *DescribeFeatureType* y al igual que la figura anterior, es el proceso de comunicación de mensajes entre clases para responder una petición hecha por la aplicación cliente. Finalmente el diagrama mostrado en la figura 4.7 pertenece al caso de uso de la operación *GetFeature* y describe el mismo proceso que las dos anteriores.

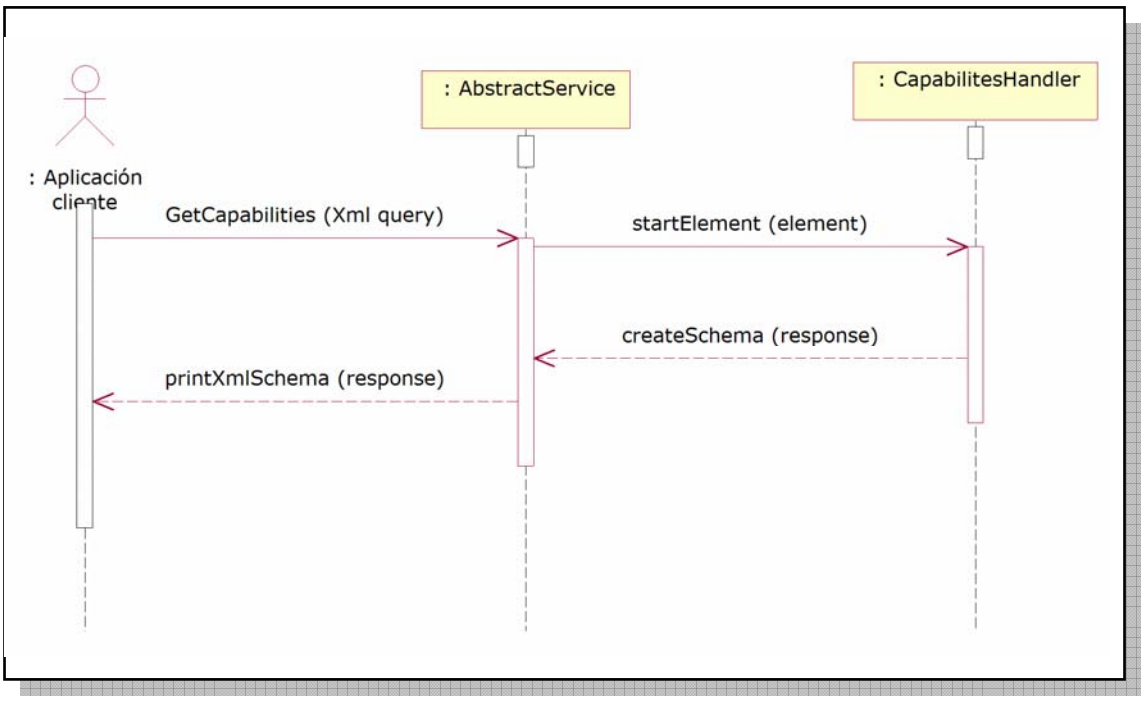



**Figura 4.5 Diagrama de secuencia: GetCapabilities**

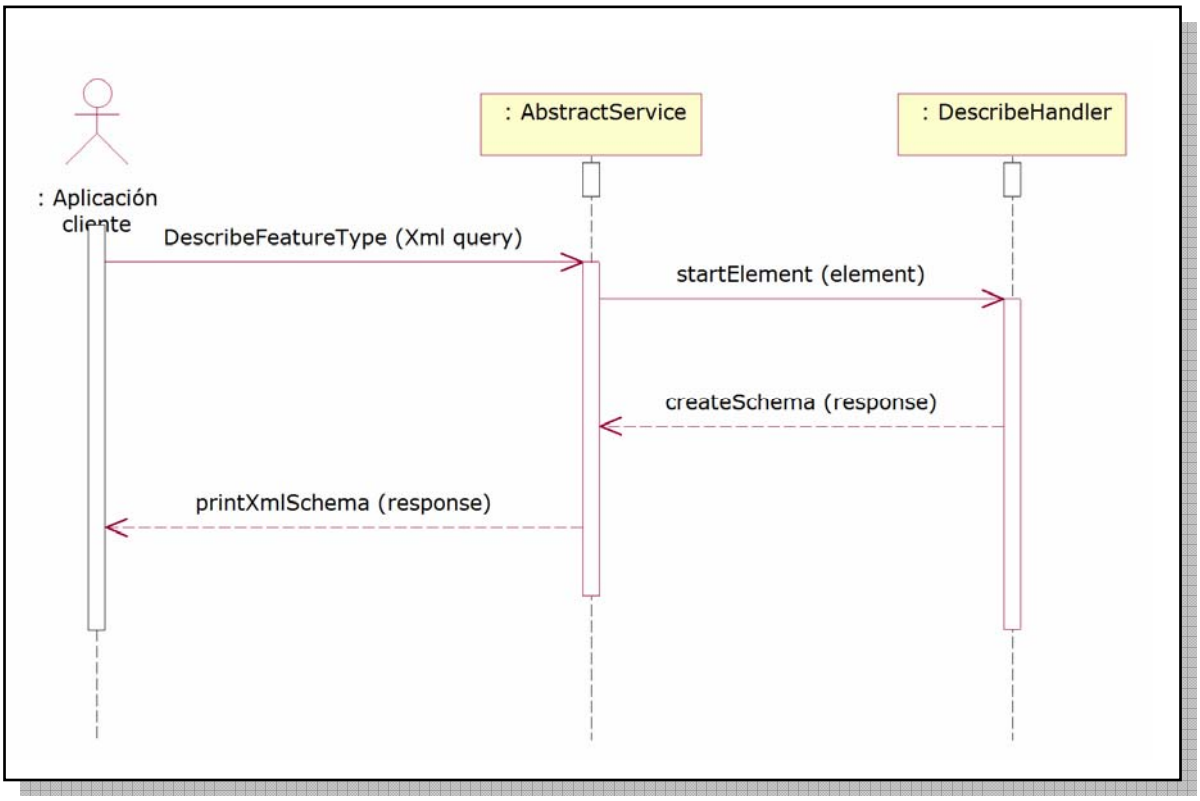



**Figura 4.6 Diagrama de secuencia: DescribeFeatureType**

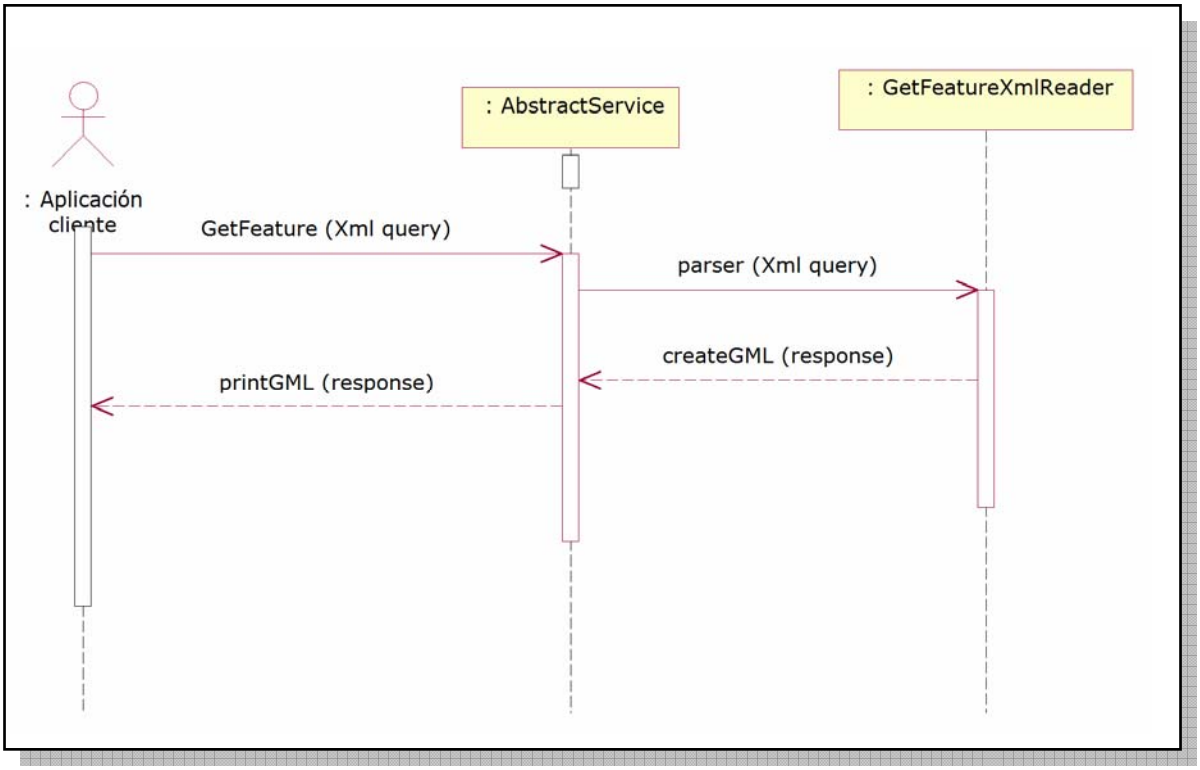



**Figura 4.7 Diagrama de secuencia: GetFeature**

#### 4.2.1.5 Mapeo de requerimientos a paquetes

Esta parte del proceso es una aportación importante a la documentación de un *software* o componente pues facilita mucho identificar que ocurre cuando deseamos modificar una parte del código. Consiste en una tabla donde se ponen los requerimientos del sistema, los casos de uso involucrados, y las clases o paquetes afectados y establece la relación entre todos estos elementos de la tabla. Una vez que se tenga la tabla (la cual puede llegar al nivel de detalle que se necesite) si en un futuro se quiere modificar parte del código la tabla indique que casos de uso se verán afectados y por consiguiente que requerimientos también estarán involucrados. Esto es de gran ayuda en sistemas grandes porque no tenemos que esperar a las pruebas de regresión para descubrir que fue lo que se alteró. La tabla 4.1 muestra el mapeo de requerimientos a paquetes y clases de Gisweb pero de alto nivel, por lo tanto es de los requerimientos funcionales básicos y algunos especiales (estos requerimientos se encuentran en la sección 4.3.1.2 de este documento de tesis). En el siguiente capítulo se implementa este código en una *Webapp* para poder tenerlo en línea con toda la información pertinente.

| Requerimientos | Caso de uso | Clases |
|---|---|---|
| Medio por el cual el WFS proporciona al cliente la opción de preguntar como se estructura y cuales son las características de un *feature* | DescribeFeatureType | org.vfny.geoserver.requests.wfs.DescribeHandler<br>org.vfny.geoserver.requests.readers.wfs.DescribeXmlReader<br>org.vfny.geoserver.requests.wfs.DescribeRequest<br>org.vfny.geoserver.servlets.AbstractService<br>org.vfny.geoserver.responses.wfs.DescribeResponse<br>org.vfny.geoserver.config.CatalogConfig<br>org.geotools.gml.producer.FeatureTypeTransformer$FeatureTypeTranslator |
| Medio por el cual el WFS proporciona al cliente la opción de preguntar por sus propias capacidades, fearures y operaciones disponibles sobre cada *feature*. | GetCapabilities | org.vfny.geoserver.servlets.AbstractService<br>org.vfny.geoserver.requests.CapabilitiesHandler<br>org.vfny.geoserver.requests.readers.wfs.CapabilitiesXmlReader |
| Medio por el cual el WFS proporciona al cliente la opción de hacer consultas espaciales y decriptvas sobre uno o más *features* | GetFeature | org.geotools.filter.LogicSAXParser<br>org.geotools.gml.GMLFilterGeometry<br>org.vfny.geoserver.requests.readers.wfs.GetFeatureXmlReader<br>org.xml.sax.helpers.AttributesImpl org.geotools.filter.FilterFilter<br>org.vfny.geoserver.requests.wfs.FeatureHandler<br>org.vfny.geoserver.servlets.AbstractService<br>org.vfny.geoserver.responses.wfs.FeatureResponse<br>org.vfny.geoserver.config.CatalogConfig<br>org.vfny.geoserver.config.DataStoreConfig<br>org.geotools.factory.FactoryFinder<br>org.geotools.gml.producer.FeatureTransformer$FeatureTranslator |
| Las interfaces deben estar definidas en XML | GetCapabilities<br>DescribeFeatureType<br>GetFeature | org.vfny.geoserver.servlets.wfs.Capabilities<br>org.vfny.geoserver.servlets.wfs.Describe<br>org.vfny.geoserver.servlets.wfs.Feature |
| Usar GML para representar *features* | GetFeature | package org.geotools.feature |
| La petición debe hacerse en XML y ser derivada de CQL | GetCapabilities<br>DescribeFeatureType<br>GetFeature | org.vfny.geoserver.requests.readers.wfs.DescribeXmlReader<br>org.vfny.geoserver.requests.readers.wfs.CapabilitiesXmlReader<br>org.vfny.geoserver.requests.readers.wfs.GetFeatureXmlReader |
| Bases de datos | GetCapabilities<br>DescribeFeatureType<br>GetFeature | org.vfny.geoserver.config.CatalogConfig<br>org.vfny.geoserver.config.DataStoreConfig |

**Tabla 4.1 - Mapeo de requerimientos y clases**

En este capítulo vimos el proceso de ingeniería inversa a la implementación pública de Geoserver de su WFS y como obtuvimos los diagramas de diseño del funcionamiento global del sistema. La ingeniería inversa aquí presentada fue muy general debido a la complejidad del proyecto en cuestión, pero se puede hacer tan detallada como se desee. Se puede repetir este ciclo a niveles cada vez más profundos hasta llegar al análisis de líneas de código. La importancia de la ingeniería inversa es que con el resultado de este proceso, es fácil entender el funcionamiento de un sistema, y saber qué hacer y en dónde cuando ocurra un problema o simplemente cuando sea tiempo de darle mantenimiento al sistema. Se propuso una metodología en base a comentarios de un experto en el área que sirva de guía en futuros procesos de ingeniería inversa. La figura 4.8 muestra que parte de este proyecto de tesis representa este capítulo y el trabajo restante para completar nuestro sistema Gisweb. En el siguiente capítulo se trata la implementación del WFS como *webapp* al integrarse con una interfaz de usuario.

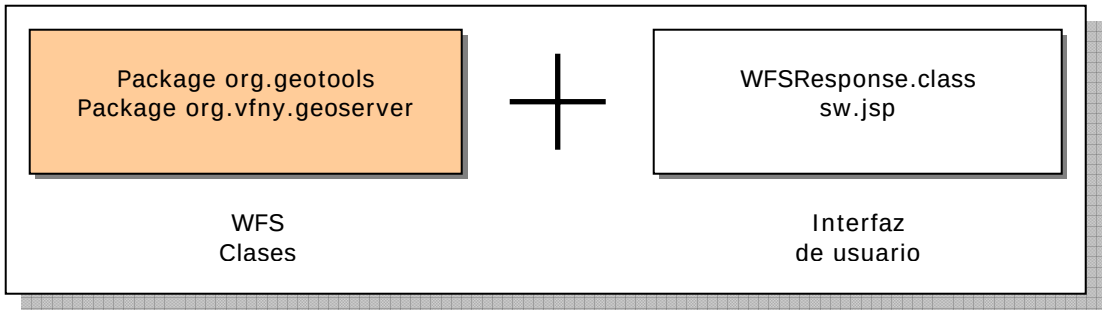



**Figura 4.8 – Diagrama *webapp* Gisweb**