Universidad de la República

Facultad de Ingeniería

Instituto de Computación

Informe de Proyecto de Grado

Gobernanza de servicios en la plataforma de interoperabilidad de gobierno electrónico

Autores:
Arian Bessonart
Juan Pablo Lucas
Miguel Renom

Tutores:
MSc. Ing. Laura González
MSc. Ing. Guzmán Llambías

Agosto, 2014

Resumen

La Agencia para el Desarrollo del Gobierno de Gestión Electrónica y la Sociedad de la Información y del Conocimiento (AGESIC) es responsable por el desarrollo y mantenimiento de una plataforma de interoperabilidad (PDI) que brinda la base para la implementación de una arquitectura orientada a servicios *(Service Oriented Architecture, SOA)* a nivel del Estado.

En la actualidad, la PDI experimenta un crecimiento tal que hace necesario un mecanismo más avanzado de gobernanza para la gestión de sus servicios. La gobernanza en SOA provee un marco que da soporte a la formalización de los procesos que conciernen al funcionamiento de este tipo de arquitecturas con el fin de asistir en la toma de decisiones que ocurren a lo largo del ciclo de vida de los servicios.

Este proyecto propone soluciones para la mejora de la gobernanza de servicios en la PDI, enfocadas en el monitoreo de la calidad, el versionado, y el descubrimiento y registro de servicios. Además, se propone un ciclo de vida que cubre las etapas por las que debe transcurrir un servicio en la plataforma, en particular, cubriendo los puntos anteriores.

Las propuestas fueron validadas por un prototipo de software que modela el ciclo de vida propuesto, implementa varias de las métricas del modelo de calidad definido, permite llevar a cabo las estrategias de versionado propuestas y ofrece la visualización de los servicios disponibles a través de una aplicación web.

Palabras clave Gobierno electrónico, Plataforma de interoperabilidad, SOA, Gobernanza en SOA, Ciclo de vida de servicios, Modelo de calidad, Estrategia de versionado, Registro de servicios

Índice general

Capítulo 1

Introducción

AGESIC es una institución del Estado que tiene como misión liderar la estrategia de implementación del *Gobierno Electrónico (GE)* del país, como base de un Estado eficiente y centrado en el ciudadano promoviendo un buen uso de las tecnologías de la información y la comunicación [1] del Estado. AGESIC tuvo a cargo la construcción y puesta en marcha de una *Plataforma de Interoperabilidad* como parte de la *Plataforma de Gobierno Electrónico (PGE)* del país de forma de cumplir con el objetivo de facilitar y promover la implementación de servicios de gobierno electrónico en Uruguay. Para esto, la PDI brinda mecanismos que apuntan a simplificar la integración entre los organismos del Estado y posibilitar un mejor aprovechamiento de sus activos con el objetivo final de implementar una *Arquitectura Orientada a Servicios (Service Oriented Architecture, SOA)* a nivel de Estado.

Por otro lado, cuando una organización decide implantar una SOA en su infraestructura, necesita articular los medios necesarios para controlar la evolución de esa arquitectura. Al mismo tiempo necesita en todo momento identificar las necesidades de negocio que la arquitectura debe satisfacer. Las soluciones basadas en SOA son entornos dinámicos donde intervienen actores a diferentes niveles (ejecutivos, técnicos, profesionales TI y de negocio), y por eso es necesario asegurarse de que crece y cambia en el tiempo respetando ciertas normas establecidas. La gobernanza en SOA es la tarea de administración que formula y hace cumplir estas normas, de modo de gestionar y optimizar el funcionamiento de la arquitectura.

Sin embargo, en la actualidad, la AGESIC no cuenta con un proceso de gestión formal para los servicios de la PDI. Dado que la plataforma ha ido creciendo con el tiempo, se ha vuelto necesario incorporar mecanismos más avanzados de gobernanza sobre el ciclo de vida de los servicios, llevar a cabo una supervisión mejor definida de los acuerdos de nivel de servicio *(Service Level Agreement, SLA)*, y aplicar estrategias de versionado que optimicen costos y reduzcan los riesgos de que los usuarios experimenten interrupciones en el acceso a la plataforma con el despliegue de nuevas versiones.

1.1. Objetivos

El objetivo general del proyecto es proponer soluciones que mejoren el proceso de gobernanza de servicios en la PDI, desarrollando las cuestiones relacionadas al ciclo de vida de los servicios, especialmente en lo referente a la supervisión (monitoreo) y versionado de los mismos. Con este fin, se plantean una serie de objetivos específicos:

1. Proponer procesos de gobernanza en SOA en base a la realidad de AGESIC
2. Proponer una definición del ciclo de vida de un servicio, en especial de las etapas de monitoreo y versionado
3. Definir un modelo de calidad ajustado a la realidad de AGESIC que sirva como base para el monitoreo de servicios
4. Establecer una estrategia de versionado de servicios de acuerdo a la PDI
5. Proponer mejoras para el catálogo de servicios existente
6. Realizar un prototipo de software para el proceso de gobernanza de servicios que permita validar las soluciones propuestas

1.2. Aportes

Los principales aportes del proyecto son:

1. Un proceso de gobernanza en SOA que brinda las guías para la gestión de los servicios en la PDI
2. Definición del ciclo de vida de un servicio
3. Definición de un modelo de calidad de servicios para su monitoreo como parte del proceso de gobernanza
4. Definición de una estrategia de versionado de servicios dentro de la plataforma
5. Propuesta de mejora del catálogo de servicios aprovechando la nueva información relativa a la calidad y el versionado de servicios
6. Prototipo de software que valida las soluciones propuestas

Aportes adicionales

1. Análisis y comparación de las principales herramientas del mercado que dan soporte a la gobernanza en SOA
2. Plantillas XSLT para transformación de solicitudes SOAP XML de modo de hacerlas compatibles con versiones de servicio distintas a aquellas para las cuales están destinadas
3. Aplicación que funciona como catálogo mostrando los servicios gobernados con la herramienta utilizada para el prototipo

1.3. Organización del documento

El contenido del informe comienza con el Capítulo 2 donde se desarrollan los conceptos y definiciones necesarias para comprender el resto del documento. El Capítulo 3 contiene un análisis de la realidad de AGESIC en la que se describe el sistema de gobernanza actual, el descubrimiento de servicios en la PDI, el proceso de versionado utilizado por la plataforma y los procesos relacionados a la calidad de servicios. A partir de este análisis, se elabora en el Capítulo 4 una propuesta sobre gobernanza en SOA aplicada a AGESIC, profundizando los aspectos de monitoreo y versionado dentro del ciclo de vida. En el Capítulo 5 se describe la implementación de un prototipo que valida la solución propuesta, junto con un análisis de las principales herramientas que dan soporte al proceso de gobernanza en SOA, finalizando el capítulo con un caso de estudio. El informe finaliza en el Capítulo 6 con conclusiones, identificación de aspectos a mejorar y posibles extensiones al proyecto.

Capítulo 2

Marco Conceptual

En este Capítulo se describen los principales conceptos que se deben adquirir para el entendimiento del desarrollo del documento. La Sección 2.1 presenta una descripción general de las arquitecturas orientadas a servicios; seguido con una breve introducción sobre las tecnológicas *Middleware* y *ESB* en las Secciones 2.2 y 2.3 respectivamente. Luego se hace un resumen de las principales definiciones de las tecnologías *XML* (Sección 2.4) para poder comprender los conceptos de *Web Services* presentes en la Sección 2.5. La Sección 2.6 se define el proceso de gobernanza en SOA con la descripción de los roles y etapas de un proyecto en SOA; luego la Sección 2.7 se enfoca en los elementos que forman parte de un modelo de calidad y se presentan los principales modelos aplicados a servicios. La Sección 2.8 analiza distintas estrategia de versionado, realizando comparaciones entre ellas. Finalizando el capítulo, las Secciones 2.9 y 2.10 se centran en comprender los beneficios que brinda un catálogo de servicios; y una breve introducción de los componentes que forman parte de la plataforma de interoperabilidad de AGESIC.

2.1. Arquitectura Orientada a Servicio (SOA)

Una *Arquitectura Orientada a Servicios (Service Oriented Architecture, SOA)* es una arquitectura de *Software* que propone la construcción de aplicaciones mediante el ensamblado de bloques reusables, débilmente acoplados y altamente interoperables, cada uno de los cuales es representado como un servicio. Los mismos pueden encontrarse distribuidos y pertenecer potencialmente a diferentes propietarios. Es por ello que dicha arquitectura es utilizada para la integración de aplicaciones empresariales y comercio electrónico entre empresas [2]. Las funcionalidades de negocios son expuestas como servicios independientes basados en interfaces estándares y publicas bien definidas para ser invocados por los clientes [3]. Las soluciones implementadas en SOA son soportadas por infraestructuras de tipo *Enterprise Service Bus* (*ESB*).

2.2. Middleware

El *Middleware* es un software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre la capa de aplicación y las capas inferiores (sistema operativo y red) [4]. Proporciona una API para acceder a una fácil programación y manejo de aplicaciones distribuidas, dado que el *Middleware* brinda una abstracción de la complejidad y heterogeneidad tanto de las redes de comunicación subyacentes, sistemas operativos y lenguajes de programación [5]. Una solución *Middleware* debe permitir conectar entre sí a una variedad de productos procedentes de diversos proveedores. De esta manera, se puede separar la estrategia de sistemas de información de soluciones propietarias de un sólo proveedor. La API que provee ofrece un conjunto de servicios:

- **Servicios de Comunicación** permiten la comunicación de sistemas remotos sin preocuparse de la complejidad existente del ambiente de red.
- **Servicios de Acceso a Datos** permiten ejecutar consultas o distintas actualizaciones tanto a archivos planos como de Bases de Datos, ubicados en uno o más servidores, asegurando la integridad de los datos y la disponibilidad de la aplicación.
- **Servicios de planificación de ejecución** permiten ejecutar múltiples procesos simultáneamente, balancear la carga y priorizar tareas homogéneamente entre distintas plataformas.
- **Servicios de Seguridad** son empleados para conectar sistemas diferentes, en donde cada uno posee su propio sistema de seguridad.
- **Servicios de Directorio** ofrece método para ubicar y administrar recursos en una red, ejemplo LDAP.

Un ejemplo de *Middleware* es un *ESB*.

2.3. Enterprise Service Bus (ESB)

Enterprise Service Bus (ESB) es una plataforma de integración basada en estándares, combina mensajería, *Web Services*, transformación de datos y ruteo, de manera que se pueda conectar y coordinar la interacción entre aplicaciones diversas [6]. Los *ESB* utilizan una SOA, permitiendo una fácil integración de los sistemas existentes independiente de la tecnología, *JMS*, *Web Services*, *JDBC*, *HTTP*. La Figura 2.1 muestra el rol como intermediario de mensajes.

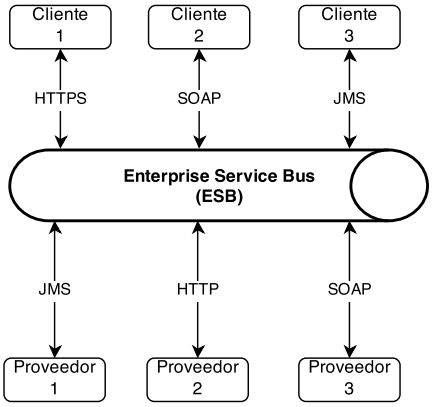

Figura 2.1: Interacción entre proveedores y consumidores a través de un ESB

Las principales funcionalidades de un *ESB* son:

Transparencia de ubicación: desliga al consumidor de la ubicación del servicio del proveedor. El *ESB* provee una plataforma central para comunicarse con cualquier aplicación sin tener que vincular el consumidor con el proveedor del servicio [7].

Transformación de mensajes: permite la transformación de mensajes desde un formato a otro, gracias a *XSLT* y *XPath*. De esta forma aplicaciones que utilizan modelo de datos o distintos formatos podrán comunicarse [7].

Ruteo de mensajes: es la capacidad por la cual el *ESB* determina en tiempo de ejecución, el destino de un mensaje de acuerdo a distintos factores. Existen varios tipos de ruteo: basados en contenido, balance de carga, contexto, itinerario. El ruteo basado en contenido determina el destino en base a su contenido. Para el caso de los mensajes *SOAP*, la lógica del ruteo se basa en el contenido del cuerpo o cabezal del mensaje. El ruteo por balance de carga determina el destino de un mensaje a través de una estrategia de balanceo como *Random*, *Round-Robin* entre otras. Ruteo por contexto determina el destino del mensaje en base a propiedades del contexto de ejecución. Finalmente el ruteo basado en itinerario incluyen en el mensaje y especifica en el *ESB* por donde debe seguir el mismo [8].

Seguridad: brinda autenticación, autorización y funcionalidad de encriptación, para asegurar los mensajes entrantes. Igualmente estas funcionalidades se aplican a mensajes salientes para satisfacer requerimientos de seguridad del proveedor del servicio a consumir.

Flujos de mediación: permiten especificar una secuencia de operaciones de mediación a ejecutar en los mensajes. Son procesos simples y de

corta duración. Un ejemplo de operación de mediación es una transformación o ruteo. La especificación se realiza generalmente en el *ESB* con herramientas gráficas, archivos de configuración o lenguajes específicos de dominio [8].

Monitoreo y administración: posibilita monitorear la ejecución de mensajes y su flujo dentro del *ESB*. Provee distinta información de monitoreo para los servicios, por ejemplo, tiempo de respuestas y cantidad de mensajes procesados. Dependiendo del producto *ESB* que se utilice, varía las funcionalidades que ofrece, en algunos casos tiene la capacidad de enviar alertas si ciertos valores de esas métricas no son adecuadas [8].

2.4. Tecnologías XML

2.4.1. XML

Extensible Markup Language (XML) es un formato de texto simple, originalmente diseñado para cumplir con los desafíos de la publicación electrónica a gran escala. No es un lenguaje, es un metalenguaje. Permite crear lenguajes, definir etiquetas, atributos, establecer relaciones entre las etiquetas y describir *Web Services* [9, 10].

2.4.2. XSLT

Extensible Stylesheet Language Transformations (XSLT) es un lenguaje de transformación de documentos *XML* en otros documentos, como *XML*, *HTML*. El flujo de una transformación se puede notar en la Figura 2.2

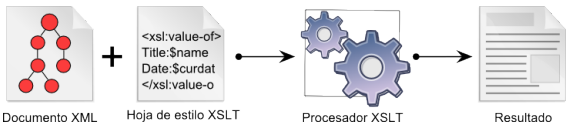

Figura 2.2: Etapas de aplicación de un XSLT

1. Un documento *XML* origen.
2. La hoja de estilo *XSLT* es el documento que contiene el código fuente del programa, es decir, las reglas de transformación que se van a aplicar al documento de origen.
3. El procesador *XSLT* es el programa que aplica al documento de origen, las reglas de transformación definidas en la hoja de estilo *XSLT*.
4. El resultado de la ejecución del programa es un nuevo documento.

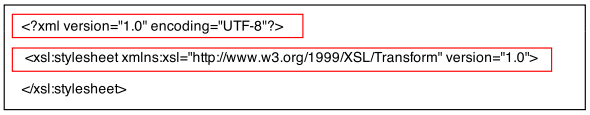

Figura 2.3: Estructura básica *XSLT*

Una hoja de estilo *XSLT* es un documento *XML*, donde se utilizan etiquetas para definir dicha hoja. Tal como muestra la Figura 2.3, las mínimas etiquetas necesarias para la correcta definición son:

1. La declaración *xml* <*?xml*>, propia de cualquier documento *XML*

2. La instrucción <*xsl:stylesheet*> es la etiqueta raíz de la hoja de estilo, sus atributos indican la versión y el espacio de nombres correspondiente

Dentro de la instrucción <*xsl:stylesheet*> se encuentran los llamados elementos de alto nivel y las plantillas [9], como muestra la Figura 2.4 se pueden notar las siguientes etiquetas:

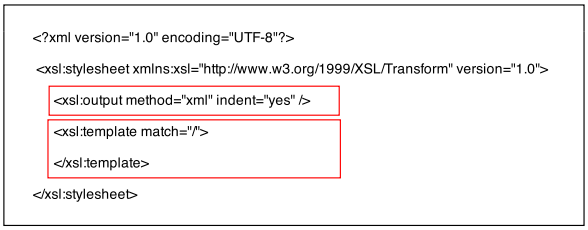

Figura 2.4: Ejemplo básico XSLT

1. El elemento de alto nivel <*xsl:output*> indica el tipo de salida producida

2. La instrucción <*xsl:template*> es una plantilla.

   a) El atributo *match* indica los elementos afectados por la plantilla y contiene una expresión *XPath*

   b) El contenido de la instrucción define la transformación a aplicar (si la instrucción no contiene nada, como en el ejemplo anterior, sustituirá el nodo por nada)

A través de un *XML Schema* se establece la estructura detallada de los mensajes definiendo qué elementos y atributos se permiten, en qué orden deben aparecer y qué tipo de datos pueden contener [8, 10].

2.4.3. XML Path Language(XPath)

Es un lenguaje dedicado a direccionar partes de un documento *XML* a través de una sintaxis de direccionamiento basado en el camino de la estructura o la jerarquía lógica. Resulta como esfuerzo por proporcionar una sintaxis y semántica para la funcionalidad entre *XSLT* y *XPointer* [11]. *XPath* utiliza una sintaxis que no es *XML* para facilitar su utilización sobre URI (Uniform Resource Identifier) y valores de atributos *XML*. *XPath* ve los documentos *XML* como un árbol de nodos, por lo que facilita la navegabilidad sobre los elementos.

2.4.4. XML Schema(XSD)

XML Schema (XSD) es un estándar del *World Wide Web Consortium* (W3C) que permite describir la estructura y el contenido de un documento *XML*. Se trata, por tanto, de una tecnología similar a *Document Type Definition (DTD)*, pero mucho más potente y versátil. En un *XSD* se definen elementos, atributos y tipo de datos, además de condiciones y restricciones sobre éstos [12]. La estructura de un *XSD* se puede ver en la Figura 2.5. El primer elemento corresponde a la etiqueta *xsd:schema*, en éste se incluye una variedad de atributos, como *xmlns* que especifica el espacio de nombres para el esquema, o *targetNamespace* que permite asociar el esquema al espacio de nombres definido.

Existen dos clases de tipos que se pueden definir, los elementos simples (*simpleType*) y los complejos (*complexType*). Los tipos simples son valores como contenido de un elemento o atributo. Este tipo de datos no puede contener elementos ni atributos. Algunos tipos predefinidos son *string*, *integer*, *date*. Mientras que los tipos complejos son aquellos que pueden contener atributos o elementos hijos.

Los atributos son similares a los elementos aunque por lo general son de tipo simple, y además éstos pueden aparecer en cualquier orden y no pueden incluir otros elementos o atributos.

XSD permite definir tipos simples con restricciones, estas son utilizadas para definir los valores aceptables para los elementos o atributos *XML*, un ejemplo de restricción es definir la cantidad de ocurrencias sobre un tipo complejo particular, se utiliza los elementos *minOccurs* y *maxOccurs*, de esta forma pudiendo implementar la opcionalidad o no de un elemento.

Otros elementos que pertenecen a un *XSD* pueden ser *any*, *choice*, *documentation* donde:

- *any*: permite indicar la presencia de cualquier tipo de elemento, no se tiene la obligación de definir un elemento específico.

- *choice*: se puede especificar una lista de elementos alternativos para incluirse en determinado elemento.

- *documentation*: etiqueta dedicada exclusivamente a documentación.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://actioncon.com/schema/gameinfo"
xmlns="http://actioncon.com/schema/gameinfo"
elementFormDefault="qualified">
   <xsd:element name="game" type="GameType"/>

   <xsd:complexType name="GameType">
     <xsd:sequence>
        <xsd:element name="gtin" type="xsd:string"/>
        <xsd:element name="title" type="xsd:string"/>
     </xsd:sequence>

     <xsd:attribute name="productID" type="xsd:string"/>
   </xsd:complexType>
</xsd:schema>
```

Figura 2.5: Ejemplo XML Schema

2.4.5. XML namespaces

Un *namespace* es una única *URI*. Tiene como fin el quitar la ambigüedad a aquellos elementos utilizados en los documentos *XML* que comparten el mismo nombre. [13]). Un *namespace* se declara utilizando una familia de atributos reservados. Este nombre de atributo debe ser *xmlns* o comenzar *xmlns*. Estos atributos, al igual que cualquier otro atributo *XML*, se pueden proporcionar directamente o por defecto. Al nombre de atributo se le puede agregar un prefijo de forma de identificar un *namespace* dentro del documento. La Figura 2.6 muestra un ejemplo de utilización de *namespace* donde se puede notar el recurso del prefijo *xsd*.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://actioncon.com/schema/gameinfo" xmlns="http://actioncon.com/schema/gameinfo"
elementFormDefault="qualified">

   <xsd:element name="game" type="GameType"/>

   <xsd:complexType name="GameType">

     <xsd:sequence>

        <xsd:element name="gtin" type="xsd:string"/>

        <xsd:element name="title" type="xsd:string"/>

     </xsd:sequence>

     <xsd:attribute name="productID" type="xsd:string"/>

   </xsd:complexType>

</xsd:schema>
```

Figura 2.6: Ejemplo XML namespace

2.5. Web Services

Un *Web Service* es una aplicación de software identificada por una URI [14], cuyas interfaces se pueden definir, describir y descubrir mediante documentos *XML*. Los *Web Services* hacen posible la interacción entre agentes de software (aplicaciones) intercambiando mensajes *XML* mediante protocolos de Internet. Actualmente, los *Web Services* son el principal mecanismo para lograr la interoperabilidad entre aplicaciones tecnológicamente heterogéneas y constituyen la tecnología preferida para la implementación de una SOA[14]. La tecnología de *Web Service* está basada en tres estándares fundamentales:

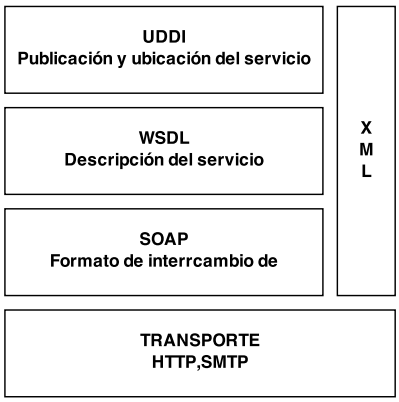

Figura 2.7: Arquitectura básica de protocolos de *Web Services*

Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL) y *Universal Description Discovery and Integration (UDDI)* como se muestra en la Figura 2.7.

2.5.1. SOAP

Simple Object Access Protocol (SOAP) es un formato de mensaje *XML* utilizado en interacciones de *Web Service*. Los mensajes *SOAP* habitualmente se envían sobre *HTTP* o *JMS*, pero se pueden utilizar otros protocolos. *SOAP* no restringe la semántica de los mensajes de intercambio, tampoco encarga de cuestiones de fiabilidad, integridad de los mensajes, transacciones, seguridad, etc [3]. En la Figura 2.8 se puede observar la estructura de

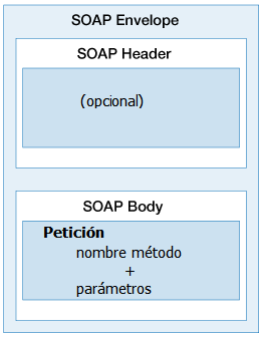

Figura 2.8: estructura de un mensaje *SOAP*

los mensajes *SOAP*. Consiste en un *envelope* que contiene una cabecera *(etiqueta <Header>)* que es un componente opcional que brinda la información sobre el mensaje a usar por la infraestructura del *Web Service*; y el otro componente es el cuerpo *(etiqueta <Body>)*, obligatorio, donde contiene información específica a usar por las aplicaciones que usan o implementan el *Web Service*. Los extremos son responsables de acordar el formato de la información intercambiada y de generar y/o procesar su contenido [3]. La Figura 2.9 es un ejemplo de un mensaje *SOAP* invocando a la operación *busquedaProveedores* con los parámetros *codigoPais, numeroDocumento, nombre y CodigoUsuario*. En el mensaje *soap* se establece el *namespace* que es quien contiene la definición en *XML* del servicio. En el ejemplo el *namespace* está definido en *xmlns:ws="http://ws.rupe.acce.gub.uy/"*. Como resultado de la invocación, se recibe un mensaje como representa la Figura 2.10.

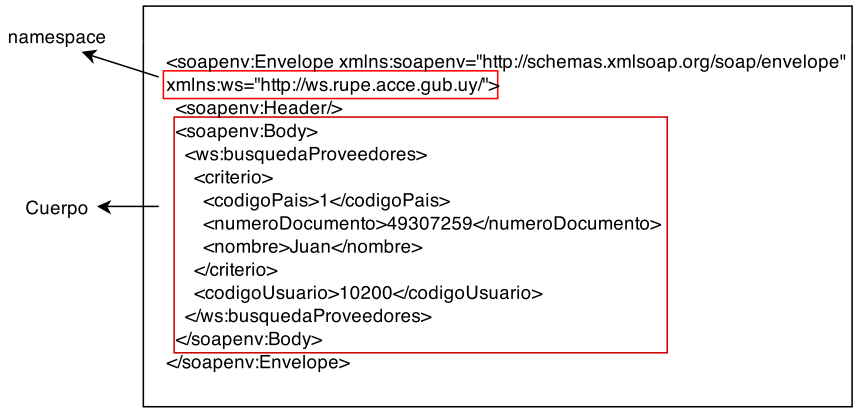

Figura 2.9: Ejemplo de mensaje *SOAP*

```
<soapenv:Envelpe
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://ws.rupe.acce.gub.uy/">
 <soapenv:Header/>
 <soapenv:Body>
  <ws:busquedaProveedoresResponse>
   <listaProveedores>
    <proveedores>
     <idProveedor>19432</idProveedor>
     <version>2.0</version>
     <fechaVersion>2014-02-23</fechaVersion>
    </proveedores>
   </listaProveedores>
  </ws:busquedaProveedoresResponse>
 </soapenv:Body>
</soapenv:Envelpe>
```

Figura 2.10: Ejemplo de respuesta mensaje *SOAP*

2.5.2. WSDL

Web Services Description Language (WSDL) es un lenguaje de definición de interfaces basado en *XML* que permite describir de forma estándar a los *Web Services* [8] y representa una especie de contrato entre el proveedor y el cliente (contrato *WSDL*). A través de este lenguaje se definen y describen distintos métodos que están disponibles en un *Web Service*, con sus parámetros de entrada y salida, tipos de datos. Dentro de la definición del *WSDL* se encuentra el *endpoint* el cual indica la localización específica para acceder al *Web Service* usando un protocolo y formato de datos específico.

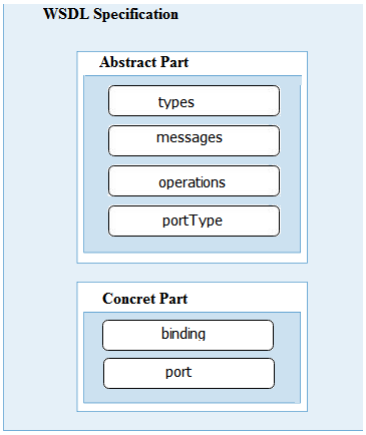

Figura 2.11: Estructura *WSDL*

En la Figura 2.11 se puede observar la estructura definida por un *WSDL*. La *descripción abstracta* contiene [15]

1. *Types* para indicar los tipos de datos que utiliza el *Web Service*
2. *Message* representa una definición abstracta de los datos transmitidos
3. *PortType* es un conjunto de operaciones abstractas. Cada operación refiere a mensajes de salida y entrada

En la *descripción concreta* se encuentran los elementos:

1. *Binding* especifica un protocolo concreto así como el formato de datos para las operaciones, y los mensajes definidos en un *portType* particular. Puede existir cualquier número de *bindings* para un *portType*. Cada operación define un *soapAction* en el *WSDL* de forma de identificar a la operación cuando sea invocada en un mensaje *soap*.
2. *Port* especifica una dirección para un *binding*. Define un *endpoint* particular especificando una dirección única para un *binding*.

2.5.3. UDDI

Universal description, discovery, and integration (UDDI) es la especificación de una plataforma estándar interoperable que permite registrar y buscar *Web Services* en directorios [16]. Provee un catálogo de registro de servicio con documentos *WSDL*, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los *Web Services*. El propósito funcional de un registro *UDDI* es la representación de datos y meta-datos acerca de *Web Service*, tanto para ser usado en una red pública como dentro de una organización. Un registro *UDDI* ofrece un mecanismo basado en estándares para clasificar, catalogar y manejar *Web Services* de forma de que puedan ser descubiertos y consumidos por otras aplicaciones. La información en un registro *UDDI* se almacena en archivos *XML* con una estructura jerárquica. Los elementos de esta estructura son:

1. ***BusinessEntity*** describe la organización que ofrece el servicio (nombre, dirección, etc.).

2. ***BusinessService*** grupo de *Web Services* relacionados ofrecido por una *BusinessEntity*, pero ofrecida en diferentes direcciones, versiones, y tecnologías. Al igual que las *BusinessEntity*, pueden incluir información de clasificación.

3. ***BindingTemplate*** información técnica para utilizar el servicio, (dirección del servicio, referencias documentos (tModels) describiendo la interfaz u otras propiedades, como dar valor a los parámetros y valores por defecto).

4. ***tModel (Technology Model)*** estructura de meta-datos genérica para representar cualquier concepto o construcción (definiciones de protocolos, ficheros *WSDL*, *XML Schemas*, espacios de nombres, esquemas de categorías, etc.).

2.5.4. WS-I

Web Services Interoperability (WS-I) es una organización autorizada para establecer buenas prácticas para la interoperabilidad de *Web Services*. Los perfiles de *WS-I* son directrices y recomendaciones sobre cómo se deben utilizar los estándares [17]. Estos perfiles tienen como fin eliminar las ambigüedades mediante la adición de restricciones sobre las especificaciones correspondientes. Los *WS-I profiles* son reglas de buenas prácticas, el cual describen un conjunto de guías para el uso de un *Web Service* más allá de los protocolos básicos. Estas guías se basan en validaciones a nivel del esquema *XML*, de los mensajes y paquetes *SOAP*. Un *Web Service* compatible con el perfil básico *WS-I* debe utilizar las siguientes características [17]

1. Utilizar *HTTP* o *HTTPS* como vínculo de transporte. *HTTP 1.1* se prefiere a través de *HTTP 1.0*
2. Utilizar la sintaxis de mensaje de fallo más estricta
3. Utilizar la versión de *XML1.0*

Existen varias versiones de *WS-I Basic Profile*. En particular *WS-I Basic Profile 1.1* es una especificación que consta de un conjunto de especificaciones de *Web Service* no propietario junto con aclaraciones, ajustes, interpretaciones y ampliaciones de las especificaciones que promueven la interoperabilidad, como *SOAP* y *WSDL* [18].

2.6. Gobernanza en SOA

El proceso de *gobernanza* en SOA, es la tarea de administración de una SOA: brinda las reglas para la toma de decisiones, establece los procesos necesarios, define los roles que forman parte del sistema administrativo y establece las métricas que determinan el ajuste de la toma de decisiones a las reglas establecidas. El proceso no indica cuándo ni cómo tomar una decisión; determina quién debería hacerlo y establece los límites para esa persona o grupo.

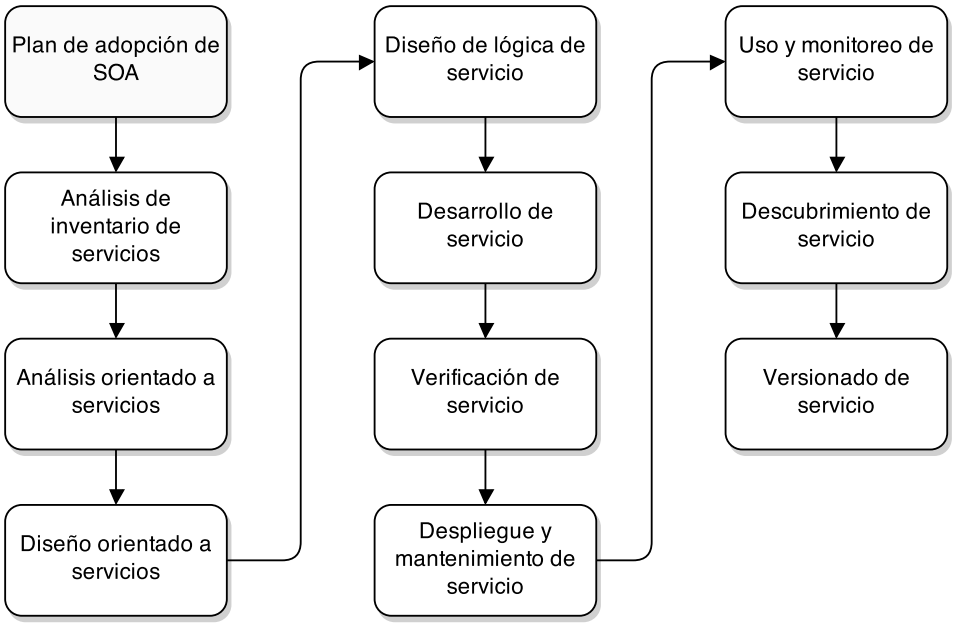

Figura 2.12: Etapas de un proyecto SOA [19]

La bibliografía actual introduce etapas, roles y actividades dentro de un proyecto de gobernanza, que varían de una publicación a otra [20] [21]. Para

la elaboración de este informe se toman como punto de partida los conceptos presentados en [19] en donde las etapas de gobernanza conforman el ciclo de vida de los servicios como se muestra en la Figura 2.12.

Plan de adopción de SOA En esta etapa se toman las decisiones fundamentales para la ejecución del proyecto. Durante el desarrollo se abordan cuestiones de alcance (qué servicios serán abarcados por el proceso de gobernanza y cuál es el objetivo para cada uno de ellos), hitos del proyecto en cuanto al alcance, tiempos de ejecución, forma de financiación, sistema de gobernanza, sistema de gestión, metodología, y gestión del riesgo. Una vez completada, esta etapa no se repite, dando lugar al comienzo de las etapas de análisis de servicios.

Análisis de inventario de servicios Un inventario representa «una colección de servicios independientemente estandarizados, administrados y gobernados», es decir, que es un conjunto de servicios que son propiedad de una misma área de una organización y que tienen una serie de aspectos en común entre ellos, a partir de los cuales se pueden definir nuevos servicios para el mismo. El objetivo de esta etapa es definir el inventario para poder así identificar servicios y determinar las relaciones de funcionamiento entre ellos. El producto de esta etapa son «blueprints» o planos que describen las características comunes que definen a los servicios. Un *blueprint* permite normalizar los servicios de manera que éstos no se superpongan, maximizando la reutilización y la separación de responsabilidades dentro del inventario. Una dedicación mayor al análisis por adelantado, dará como resultado un *blueprint* mejor definido y como consecuencia, inventarios mejor definidos. Este análisis sucede en un ciclo de análisis orientado a servicios sobre cada uno, refinando el *blueprint* en cada iteración.

Análisis orientado a servicio Es la primera etapa del ciclo de desarrollo de servicios. Durante ésta se recolectan datos para modelar distintos tipos de servicios. Es llevado a cabo en forma iterativa como parte del proceso de análisis de inventarios para cada proceso de negocio, lo que implica que se concentra en un inventario particular y aporta al refinamiento de su *blueprint*. El éxito de esta etapa depende de la colaboración de analistas de negocio y arquitectos, ya que los primeros son quienes poseen los conocimientos acerca de la lógica de negocio que sirve de entrada para el análisis.

Diseño orientado a servicio Es la siguiente etapa en el ciclo de desarrollo de servicios, dedicada a la elaboración de los contratos —documentos técnicos sobre el funcionamiento— de los candidatos a servicios como producto de la finalización del proceso de análisis orientado a servicios.

El diseño orientado a servicios toma consideraciones técnicas adicionales, buscando alinearlo con los contratos de otros servicios del mismo inventario. Durante esta etapa los arquitectos deben asegurar que los contratos cumplan los requerimientos del negocio a la vez que representan un contexto funcional adherido a los principios de la orientación a servicios. En el transcurso de esta etapa se elaboran los acuerdos de nivel de servicio.

Diseño de lógica del servicio Una vez establecido el contrato, se abre paso al diseño de la arquitectura y lógica que llevarán a cabo la funcionalidad definida en el mismo. El diseño de la lógica es influenciado por los principios del diseño orientado a servicios y aspectos relacionados al ambiente en donde será desplegado.

Desarrollo del servicio Con la especificación del diseño de la lógica y la arquitectura los desarrolladores del servicio se encuentran en condiciones de codificarlo.

Verificación del servicio Además de la verificación tradicional llevada a cabo en cualquier aplicación, los servicios deben ser sometidos a nuevos métodos de verificación. Por ejemplo, servicios componibles con otros servicios deben ser verificados en forma individual y en conjunto con otros servicios.

Despliegue y mantenimiento del servicio Esta etapa representa la puesta en producción del servicio y todas las tareas de las que depende. El mantenimiento refiere a actualizaciones o cambios necesarios en el entorno, pero no refiere a cambios en la lógica u otros que impliquen modificaciones a los contratos o la generación de nuevas versiones.

Uso y monitoreo del servicio Entran en esta etapa del ciclo todos aquellos servicios que han sido puestos a disposición de potenciales consumidores. El monitoreo se realiza sobre el uso con fines de determinar el mantenimiento necesario para la escalabilidad, robustez, etc., así como también para calcular los costos de uso, ROI, entre otros. Si bien no se trata de una etapa de entrega de servicios, se debe tomar en cuenta ya que es posible ejercer actividades de gobernanza durante la misma (la verificación sobre atributos de calidad es un ejemplo de ello).

Descubrimiento del servicio El objetivo principal de esta etapa es identificar servicios independientes que puedan cumplir los requerimientos de procesos de negocio que deban ser automatizados. La motivación para este proceso es la reutilización de estos servicios. Para ello, el mecanismo principal es la elaboración de un registro que contenga los meta-datos relevantes de servicios existentes y en proceso de ser pu-

blicados, así como también acceso a los contratos y documentación asociada.

Versionado y retiro del servicio Una vez que un servicio ha sido puesto en producción es posible que surja la necesidad de realizar cambios en la lógica o la funcionalidad, lo que implica la introducción de una nueva versión del servicio y/o de su contrato. Con el fin de minimizar el impacto de un cambio como este en los usuarios del servicio, un proceso de versionado formal deberá ser introducido como parte del proceso de gobernanza. Existen distintas estrategias de versionado con sus propios conjuntos de reglas y prioridades, las cuales afectarán la forma en la cual los consumidores y desarrolladores se ven afectados. El retiro de un servicio también debe ser sometido a un proceso formal por su impacto en los consumidores, aunque no aplique una estrategia.

Jurisdicción sobre inventarios En un caso ideal, los inventarios están relacionados directamente con algún *dominio* o línea de negocios de la organización. Es esperable que estos dominios —cuando son más de uno— estén a cargo de un propietario dentro de la organización, quien los gobierna u administra. En grandes proyectos, la cantidad de dominios y propietarios —y la relación existente entre ellos— puede resultar propicia para establecer una jerarquía de gobernanza de los mismos que permita establecer criterios adecuados para cada conjunto, y a la vez, que éstos estén alineados con los objetivos generales de gobernanza del proyecto SOA.

La entidad reguladora de estos dominios dentro de la organización es la *oficina del programa de gobernanza en SOA* (SGPO, por sus siglas en inglés). Una SGPO es un área encargada de uno o más dominios de servicios dentro de la organización, según el modelo de jurisdicción adoptado por el sistema de gobernanza. En una organización dada, múltiples SGPO pueden coexistir en base a los dominios identificados y sus propietarios. En la Tabla 2.1 y Figura 2.13 se describen los distintos modelos de jurisdicción de SGPO básicos aplicables a una organización.

El tipo de jurisdicción seleccionado determinará la forma en que se desarrolla la gestión dentro del conjunto de inventarios de servicios que se encuentren en una organización o conjunto de organizaciones, por lo que forma la base de la toma de decisiones desde el inicio de la concepción de una SOA gobernada.

2.6.1. Roles involucrados

En cada etapa del proyecto participan distintos actores tomando roles particulares para el desarrollo de las etapas. Si bien es posible, no es necesario establecer una relación uno a uno entre roles y personas; es válido asignar más de un rol a un individuo así como también más de un individuo puede

Tipo	Descripción
SGPO empresarial centralizada	Única SGPO encargada de un único dominio de servicios de la organización
SGPO con dominios centralizados	Distintos dominios estandarizados son abarcados por un único sistema de gobernanza dirigido por una SGPO central.
SGPO con dominios federados	Varias SGPO responsables de un dominio cada una donde llevan a cabo su propio sistema de gobernanza, el cual debe cumplir con lineamientos introducidos por una SGPO central
SGPO con dominios independientes	SGPO individuales responsables de un dominio cada una donde aplican el sistema de gobernanza en forma independiente.

Cuadro 2.1: Modelos de jurisdicción de SGPO

ejercer el mismo rol durante el proyecto. No se establece un criterio para la toma de estas decisiones.

Analista de servicios (Service Analyst) Se especializa en el análisis orientado a servicios y modelado, en donde brinda la experiencia necesaria para definir los distintos servicios dentro de la organización.

Arquitecto de servicios (Service Architect) Su participación está asociada con la construcción de los servicios por lo que se lo identifica con las etapas de diseño de éstos y su lógica, así como también están se involucra en la etapa de análisis de servicios. A lo largo de un proyecto participa en la definición de servicios a la vez que se encarga de la especificación del diseño de forma tal que cumpla tanto con los requerimientos del negocio como con los estándares de diseño y principios de orientación a servicios.

Desarrollador de servicios (Service Developer) Debe ser un programador capacitado para el desarrollo de los servicios aplicando los estándares de diseño utilizados en la organización.

Custodio de servicios (Service Custodian) Es el encargado de la gestión y gobernanza de uno o más servicios específicos. Su tarea se destaca en la protección de la integridad de los servicios que no están asociados a un único requerimiento (servicios agnósticos), asegurándose de que ninguna de las especificaciones de éstos sea modificada con la intención de favorecer un uso en particular.

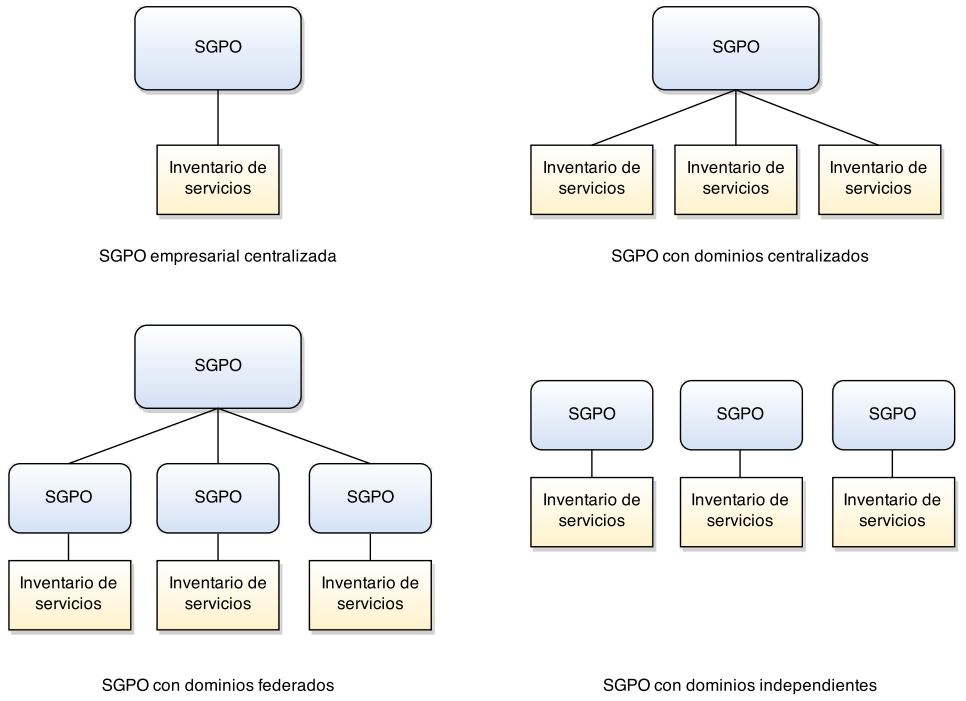

Figura 2.13: Representación gráfica de modelos de jurisdicción de SGPO [19]

Propietario de servicios de la nube (Cloud Service Owner) Trata de la persona u organización que legalmente es propietaria del servicio alojado en la nube. Los dos escenarios posibles son que el propietario sea quien aloja al servicio (el caso en el cual la organización posee su propia infraestructura) o que sea quien consume el servicio (un tercero provee la infraestructura). Este es un rol que no está directamente asociado a las actividades de gobernanza sino que sirve al ámbito legal.

Administrador de servicios (Service Administrator) Administra el funcionamiento de uno o más servicios, asegurando que todas las dependencias y recursos compartidos funcionen correctamente de modo de mantener el cumplimiento de los acuerdos de nivel de servicio establecidos para estos. Este rol existe en infraestructuras administradas por la propia organización, ya que para los casos de despliegue en la nube se tiene el rol de *Administrador de recursos de la nube*.

Administrador de recursos de la nube (Cloud Resource Administrator) Además de asegurar el correcto funcionamiento de un servicio desplegado en la nube, debe controlar mecanismos como los de pago por uso y elasticidad que brinden los proveedores de la infraestructura, ya que es-

tos pueden representar costos adicionales para la organización. Se distingue del *Administrador de servicios* en que puede encontrarse administrando un servicio desplegado en una nube de acceso público que luego la organización consumirá, o puede hacerlo en una infraestructura propia de la organización con un servicio de tipo *Software como servicio (Software as a Service, SaaS)* que esta provee.

Custodio de esquemas (Schema Custodian) Es responsable por la estandarización y posicionamiento de esquemas de contratos de servicios como partes centrales de los inventarios de servicios, es decir, que cualquier servicio que deba ser agregado a un inventario, disponga de un esquema actualizado del mismo.

Custodio de políticas (Policy Custodian) Mantiene las políticas técnicas y de lenguaje humano que son aplicadas a los servicios en general. Usualmente se aseguran de que estas estén alineadas con las políticas del negocio que da origen a los servicios.

Custodio de registro de servicios (Service Registry Custodian) Se encarga de la administración del registro de uno o más servicios. Su tarea no sólo abarca la instalación y el mantenimiento de la aplicación que da soporte al registro (catálogo) sino que también involucra al aseguramiento de la calidad de los datos disponibles en este sobre cada servicio. Su tarea está enfocada casi exclusivamente en la etapa de descubrimiento.

Especialista en comunicación técnica (Technical Communications Specialist) Su tarea consiste en expresar información sobre descubrimiento utilizando vocabulario estándar de modo que un amplio espectro de los miembros del equipo pueda comprender los contratos y perfiles de los servicios con la finalidad de reducir el riesgo de mala interpretación sobre sus características.

Arquitecto de negocio (Enterprise Architect) Este rol participa de prácticamente todas las etapas, contribuyendo con la de adopción, la creación de estándares de diseño para la organización, el desarrollo de servicios, la determinación de la infraestructura necesaria para estos, las consideraciones de seguridad e incluso puede actuar como propietario de un *blueprint*. En organizaciones de gran tamaño, pueden existir varios arquitectos de negocio encargados de dominios particulares de servicios.

Custodio de estándares de diseño de negocio (Enterprise Design Standards Custodian) Se encarga del alineamiento de los estándares introducidos a la organización a medida que éstos evolucionan así como tam-

bién de asegurar su aplicación a los procesos de diseño. Es un rol que implica también la realización de auditorías, por lo que quien lo lleve a cabo debe contar con la autoridad necesaria para hacer cumplir dichos estándares.

Especialista en aseguramiento de la calidad en SOA (SOA Quality Assurance Specialist) Asociado con la etapa de verificación, quien cumple este rol debe asegurar la calidad de los servicios, en particular la de aquellos que pueden ser reutilizados para distintas áreas de negocio ya que son los que forman parte de más conjuntos de dependencias dentro de la organización.

Especialista en seguridad en SOA (SOA Security Specialist) Debe asegurar que todos los servicios están correctamente protegidos, y que aquellos que son agnósticos (y por tanto utilizados con múltiples fines), sean lo suficientemente flexibles como para ser incorporados a otras arquitecturas de seguridad.

Especialista en gobernanza en SOA (SOA Governance Specialist) Es un experto en los procesos de gobernanza en SOA así como también en el uso de las tecnologías relacionadas. Forma parte de todas las etapas de gobernanza.

2.7. Calidad en Servicios

La *calidad de los servicios (*Quality of Service*, QoS)* es un elemento importante que permite determinar el rendimiento, utilidad y facilidad de uso de los servicios, entre otros factores. Se utiliza normalmente para describir las características no funcionales de los *Web Services* y como criterio para la evaluación de los diferentes servicios.
Cada servicio debe cumplir con un contrato escrito entre el proveedor del mismo y el cliente, donde se define un *acuerdo de nivel de servicio (Service Level Agreement, SLA)* en el cual el proveedor debe respetar todos los criterios de calidad que se compromete a cumplir. [22]

2.7.1. Metamodelo de calidad

Un meta-modelo de calidad se define como un conjunto de niveles que permite evaluar criterios de calidad en un servicio. *Quality Abstract Model* representa una estructura básica de distintos niveles con diferentes perspectivas de análisis [23]. El mismo se encuentra organizado en:

1. Dimensión: Es el aspecto de la calidad correspondiente a la perspectiva que brinda el mayor nivel de abstracción de la misma.

2. Factor: Se ubica en el nivel inmediato inferior a la dimensión por lo que corresponde a un aspecto particular de la dimensión de la calidad. Un factor pude ser más adecuado que otro para algún tipo de problema o aplicación que se esté analizando.

3. Métrica: La métrica corresponde a un instrumento que define una forma de medir un factor de calidad. Puede haber diferentes métricas para el mismo factor, las cuales miden el factor desde diferentes puntos de vista. Consta de tres elementos que tienen que estar bien definidos

   a) Semántica (define cómo se mide la métrica, le da significado)

   b) Unidad de medida (es la cantidad en términos de magnitud o porcentaje)

   c) Granularidad de la medida (especificidad a la que se define un nivel de detalle)

4. Método: El método corresponde al proceso por el cual se implementa la métrica. Puede haber más de un método que implemente la misma métrica. La implementación del método depende del dominio de la aplicación en concreto y de aspectos de bajo nivel como pueden ser la estructura de datos que es objeto de estudio.

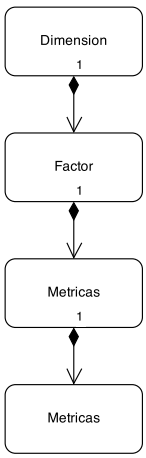

Figura 2.14: Metamodelo abstracto de calidad

La Figura 2.14 muestra la jerarquía sobre cada nivel de análisis del metamodelo.

2.7.2. Modelos de Calidad

Un *modelo de calidad* es el resultado de proponer un conjunto de elementos por lo cual se quiere analizar. Permite homogeneizar características

de los servicios ya que son evaluados bajo los mismos criterios; y asegura el cumplimiento de ciertos requerimientos y un ordenamiento de los servicios. Existen varios enfoques interesantes de modelos. Particularmente se hace énfasis en el modelo de calidad para *Web Services* de OASIS, el modelo de calidad en base al proyecto S-Cube y los principales factores de calidad que son requeridos para apoyar la QoS por IBM.

Modelo OASIS

OASIS (Organization for the Advancement of Structured Information Standards) es una organización sin fines de lucro que conduce el desarrollo, unificación y adopción de estándares para la sociedad de información global. Ha producido muchos de los estándares hoy usados para *Web Services* [24]. La organización presenta un modelo conceptual que define la interacción entre las partes involucradas *Actores de calidad (Quality Associates)*, *Actividades de calidad (Quality Activities)* y *Factores de calidad (Quality Factors)* para habilitar la calidad de servicio.

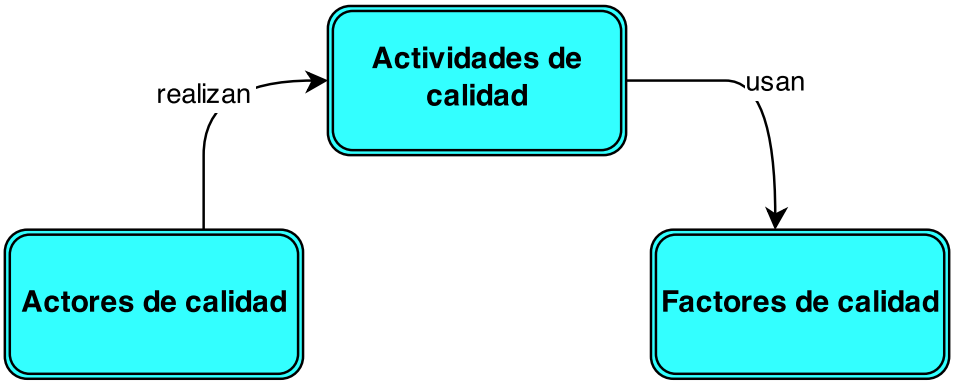

Figura 2.15: Modelo conceptual de calidad para *Web Services* OASIS

En la Figura 2.15 se muestra el modelo conceptual de OASIS. Por ejemplo, un consumidor y un proveedor de *Web Services (actores de calidad)* pueden establecer un SLA *(actividad de calidad)*, que involucre el tiempo de respuesta y la disponibilidad de los servicios *(factores de calidad)* [8]. Al realizar un mapeo con el meta-modelo abstracto de calidad visto en la Sección 2.7.2, se muerta en la Figura 2.14 la agrupación del modelo propuesto por OASIS en la estructura Dimensión - Factor - Métrica [25]. En el Apéndice A.1 se describen cada una de las métricas del modelo. Algunos factores de ejemplo que forman parte del modelo son la tasa de transferencia, tiempo de respuesta, y disponibilidad que permite evaluar la *Performance* del servicio.

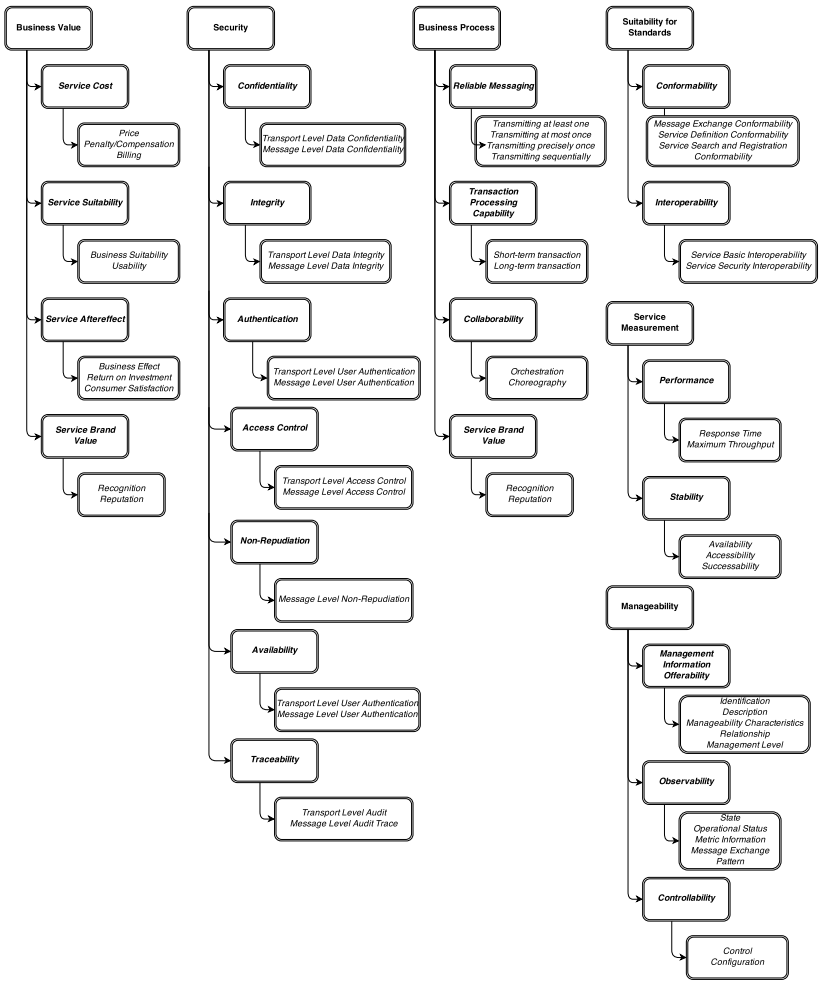

Figura 2.16: Metamodelo de calidad propuesto por OASIS

Modelo S-Cube

S-Cube es un proyecto Europeo que define principios, técnicas y métodos para monitorear la calidad de los servicios. Diseñó un modelo de calidad para sistemas basados en servicios seleccionado distintos modelos de diferentes disciplinas, seleccionando los atributos de calidad relevante para los asegurar la calidad en los servicios [26]. Una vez concluida la elección de atributos, se agrupan en categorías y forman el modelo presentado en la Figura 2.17. [25]. En el Apéndice A.3 se describen cada una de las métricas del modelo, como por ejemplo factores de *Usability* que permite evaluar los aspectos de calidad según la opinión de los usuarios.

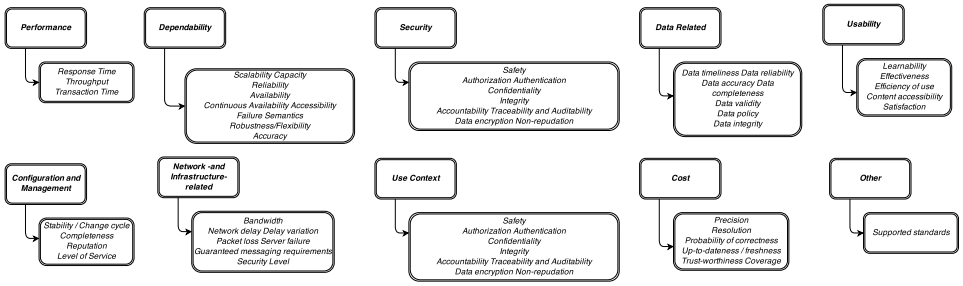

Figura 2.17: Modelo de calidad S-Cube

Modelo IBM

International Business Machines (IBM) es una empresa multinacional estadounidense de tecnología y consultoría. IBM fabrica y comercializa hardware y software para computadoras, y ofrece servicios de infraestructura, alojamiento de Internet, y consultoría en una amplia gama de áreas relacionadas con la informática. Con la evolución de los *Web Services*, IBM analiza los principales requisitos de calidad que se deben monitorear en los servicios [27]. En la Figura 2.18 se muestran los factores de calidad que propone IBM para un modelo de calidad. En el Apéndice A.2 se describen cada una de las métricas del modelo, por ejemplo *Accessibility* que es el aspecto de la calidad de un servicio que representa el grado en que es capaz de servir a una solicitud de *Web Service*.

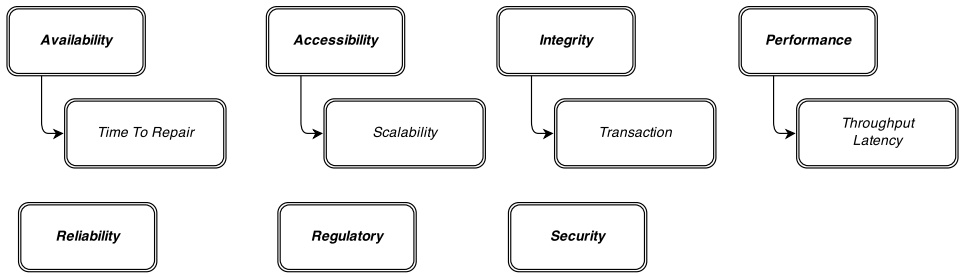

Figura 2.18: Modelo de calidad IBM

2.8. Versionado de servicios

Cuando un servicio sufre algún cambio, entra en juego la importancia de decidir crear una nueva versión. En esta sección se introducen distintos conceptos relacionados al versionado de servicios para comenzar a comprender esta práctica. De ahora más se considera como contrato de servicio al contrato *WSDL* descrito en la Sección 2.5.2.

2.8.1. Compatibilidad

Hacia atrás (Backward Compatibility): La compatibilidad hacia atrás enfocada a *Web Service* significa que ante modificaciones en el contrato de servicio la nueva versión continúa soportando las aplicaciones diseñadas para trabajar con su versión anterior.

Hacia adelante (Forward Compatibility): Es una propiedad que presentan aquellos contratos de servicio que son diseñados de manera que soportan a futuros consumidores sin la necesidad de algún cambio.

2.8.2. Los cambios

Cambio compatible

Son aquellos cambios sobre el contrato de servicio que no afectan negativamente a los consumidores que se encuentran utilizando el servicio. Un ejemplo simple es el agregado de un parámetro opcional como se puede notar en la Figura 2.19 se agrega el parámetro *available* en donde se define con la propiedad *minOccurs* en 0, esto implica que es opcional. Nuevos consumidores podrán utilizar este nuevo parámetro como no, consumidores actuales acostumbrados a no utilizar este nuevo parámetro podrán seguir del mismo modo [28].

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://actioncon.com/schema/po"
xmlns="http://actioncon.com.schema/po">
    <xsd:element name="LineItem" type="LineItemType"/>
    <xsd:complexType name="LineItemType">
      <xsd:sequence>
        <xsd:element name="productID" type="xsd:string"/>
        <xsd:element name="productName" type="xsd:string" minOccurs="0"/>
        <xsd:element name="available" type="xsd:boolean" minOccurs="0"/>
      </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figura 2.19: Cambio compatible

Cambio incompatible

Son cambios que se efectúan sobre un contrato de servicio de manera que impactan negativamente sobre los consumidores existentes. A través de la Figura 2.20 se puede notar un ejemplo sobre este tipo de cambios, se agrega

un nuevo parámetro obligatorio a un método dado. El impacto de este cambio sobre los consumidores existentes es alto ya que éstos desconocen del nuevo parámetro obligatorio por lo que no lo utilizan para invocar al método. Existe una serie de cambios incompatibles por los cuales mediante mecanismos, como la modificación de los mensajes de intercambio, se puede neutralizar el resultado adverso.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://actioncon.com/schema/po"
xmlns="http://actioncon.com.schema/po">
    <xsd:element name="LineItem" type="LineItemType"/>
    <xsd:complexType name="LineItemType">
        <xsd:sequence>
            <xsd:element name="productID" type="xsd:string"/>
            <xsd:element name="productName" type="xsd:string" minOccurs="0"/>
            <xsd:element name="available" type="xsd:boolean" minOccurs="2"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Figura 2.20: Cambio incompatible

2.8.3. Identificación de versiones

La identificación de versiones es esencial para la simplificación sobre la gestión de liberaciones de nuevas versiones de servicio. La estrategia debe facilitar el reconocimiento de múltiples versiones de un mismo servicio. Existen distintas opciones de identificación. Se puede utilizar una convención de v#Mayor.#Menor. Donde *Mayor* corresponde a un cambio de versión mayor, y *Menor* a un cambio de versión menor. Usualmente un cambio de versión mayor corresponde a un cambio incompatible mientras que uno menor a un cambio compatible. Otra manera de identificación es utilizar fechas como parte del nombre como se puede ver en la Figura 2.21. La fecha ayuda a determinar la secuencia de la versión [29].

```
<xsd:schema targetNamespace = http://<dominio>/path/2014/04/02>
```

Figura 2.21: Identificación de versiones utilizando fecha

2.8.4. Estrategia de versionado

Una estrategia de versionado de servicios se basa en un conjunto de reglas y normas a seguir para asegurar que cada servicio dentro de un catálogo se versione correctamente. Principalmente se basan en cómo los servicios pueden ser versionados en respuesta a cambios compatibles e incompatibles y qué hacer ante dichos cambios [19]. A continuación se describen tres estrategias:

Estricta (Strict): Este enfoque es el más simple y seguro, su falta de flexibilidad nunca presentará alguna sorpresa al momento de un cambio de contrato de servicio ya que los cambios posibles sean compatibles o incompatibles tendrán como resultado una nueva versión mayor del contrato de servicio. Esta orientación no soporta ningún tipo de compatibilidad (hacia atrás o hacia adelante), debido a que cada nueva versión mayor de contrato quiebra con la compatibilidad. La principal ventaja es que se posee un control total en la evolución del contrato de servicio, además de que baja la preocupación por el impacto de cualquier cambio, el análisis es simple ya que todo cambio genera una nueva versión mayor. Sin embargo dará lugar a una cantidad de versiones de contrato tanto como evoluciona este, aumentando la carga de la gestión y creando la necesidad de incorporación de estrategias de transición entre versiones [28].

Flexible: A diferencia de la estrategia anterior, en este caso se toman diferentes medidas de acuerdo a distintos tipos de cambios. Para los casos en que el cambio es incompatible se genera una nueva versión mayor del contrato de servicio, mientras que para un cambio compatible se genera una nueva versión menor manteniendo la compatibilidad con los consumidores actuales. Esta estrategia posee compatibilidad hacia atrás sobre los contratos de servicio aunque no hacia adelante. Gracias a este enfoque se da lugar a una gran variedad de cambios que se tratan como cambios compatibles, sin quebrar la compatibilidad y sin afectar a consumidores actuales. Como principal desventaja se destaca que la inclusión de cambios compatibles, fuerzan a que éstos mismos sean permanentes y la única forma de revertirlos es a través de un cambio incompatible: un ejemplo de ello es que en la primer etapa se adiciona un nuevo parámetro opcional manteniendo la compatibilidad, se obtienen nuevos consumidores que hace uso del nuevo parámetro por lo que en la segunda etapa se quisiera remover el parámetro. Esto haría que los nuevos consumidores ya no puedan utilizarlo al parámetro lo que desencadena un cambio incompatible, por lo que es necesario tener un proceso de gobernanza para evaluar cada propuesta de cambio.

Relajada (Loose): Al igual que en la estrategia *flexible*, los cambios incompatibles generan una nueva versión mayor del contrato de servicio

y los cambios compatibles una nueva versión menor. Esta estrategia tiene como principal objetivo el diseño de un contrato de servicio extensible que soporte modificaciones a futuro, buscando que cambios futuros no quiebren con la compatibilidad de los consumidores actuales. Este último requerimiento se logra a través de *wildcards* (parámetro *anyType* o *any*). La desventaja principal se presenta en la definición del contrato de servicio, se debe definir desde un punto abstracto por lo que no presenta ni detalles ni restricciones, esto desencadena la necesidad de validaciones sobre el tipo o sobre la cantidad de ocurrencias de los parámetros, debiendo el proveedor agregar controles por medio de implementaciones [28].

Comparación entre estrategias

En la Tabla 2.2 se muestra la comparación entre estrategias de acuerdo de los siguientes parámetros: rigidez, impacto en gobernanza, complejidad [28].

- Rigidez: El enfoque estricto es claramente el más rígido ya que todo cambio genera una nueva versión mayor del contrato, en su opuesto se encuentra la estrategia *relajada* que brinda la maleabilidad ante nuevos posibles modificaciones, debiendo utilizar el recurso de *wildcards*.

- Impacto en Gobernanza: Las estrategias *estricta* y *relajada* tienen una valoración alta. El enfoque estricto requiere la emisión de nuevas versiones de contrato, lo que afecta a los consumidores y a la infraestructura, mientras que el enfoque *relajado* introduce conceptos de mensajes desconocidos, lo que exige un control por parte del proveedor para regular el formato y tipos de datos de los mensajes.

- Complejidad: Hace referencia a las dificultades de utilización de cada una de las estrategias. Este caso es opuesto sobre la comparación sobre la rigidez, la estrategia que presenta la mayor simpleza es la *estricta*, mientras que la *relajada* es la más complicada de aplicar debido al uso de *wildcards* adiciona necesidades de validaciones extra.

	Estricta	**Flexible**	**Relajada**
Rigidez	Alta	Media	Baja
Impacto en Gobernanza	Alta	Media	Alta
Complejidad	Baja	Media	Alta

Cuadro 2.2: Comparación entre estrategias de versionado

2.9. Catálogo de servicios

Un catálogo de servicios, es una base de datos o documento estructurado con información acerca de todos los servicios activos en la organización. Este contiene información acerca de dos tipos de servicio: los que son visibles como servicios del negocio, y los que son requeridos para componer a uno. El propósito es proveer información consistente sobre todos los servicios acordados a todas las personas autorizadas. [30]

El catálogo pertenece a la cartera de servicios (o *Service Portfolio*), y se diferencia de esta en que sólo contiene a los servicios disponibles o en proceso de estar disponibles, mientras que la cartera contiene además a la especificación de los servicios a desarrollar.

Son utilizados tanto por clientes como por el departamento de TI de la organización. En ellos se documentan detalles como el estado, interfaces de consumo y dependencias. Esta información se hace disponible a personas autorizadas en formatos adecuados para su utilización (electrónica o en papel). Debe contener a todos los servicios disponibles en producción, así como también a aquellos que ya están planificados para ser puestos en producción. En forma opcional puede contener servicios que han sido solicitados.

ITIL propone un proceso de gestión [1] cuyo objetivo es asegurar la creación y el mantenimiento del catálogo y que toda la información disponible para las partes sea precisa y actualizada para proveer los servicios.

El catálogo es una herramienta que asiste a la gestión de servicios de la organización y que puede ser utilizada en análisis de impacto y otros procesos que requieran de conocer las interacciones entre servicios de la organización y sus consumidores.

2.10. Plataforma de interoperabilidad del Estado

La Plataforma de interoperabilidad (PDI) provee a los organismos del Estado de una infraestructura que facilita la implementación de servicios al público así como el acceso a servicios de otros organismos del Estado [31].

A nivel tecnológico, la PDI permite que los organismos provean sus funcionalidades de negocio a través de servicios, y de esta forma lograr la implementación de una SOA a nivel del Estado, la cual se apoya fuertemente en la tecnología de *Web Service*. Esto facilita la integración entre los organismos, promoviendo la reutilización y aprovechamiento de los recursos de información y tecnológicos con los que cuentan. De esta forma reduciéndose los tiempos de implementación de nuevos requerimientos.

[1] *Service Catalogue Management*

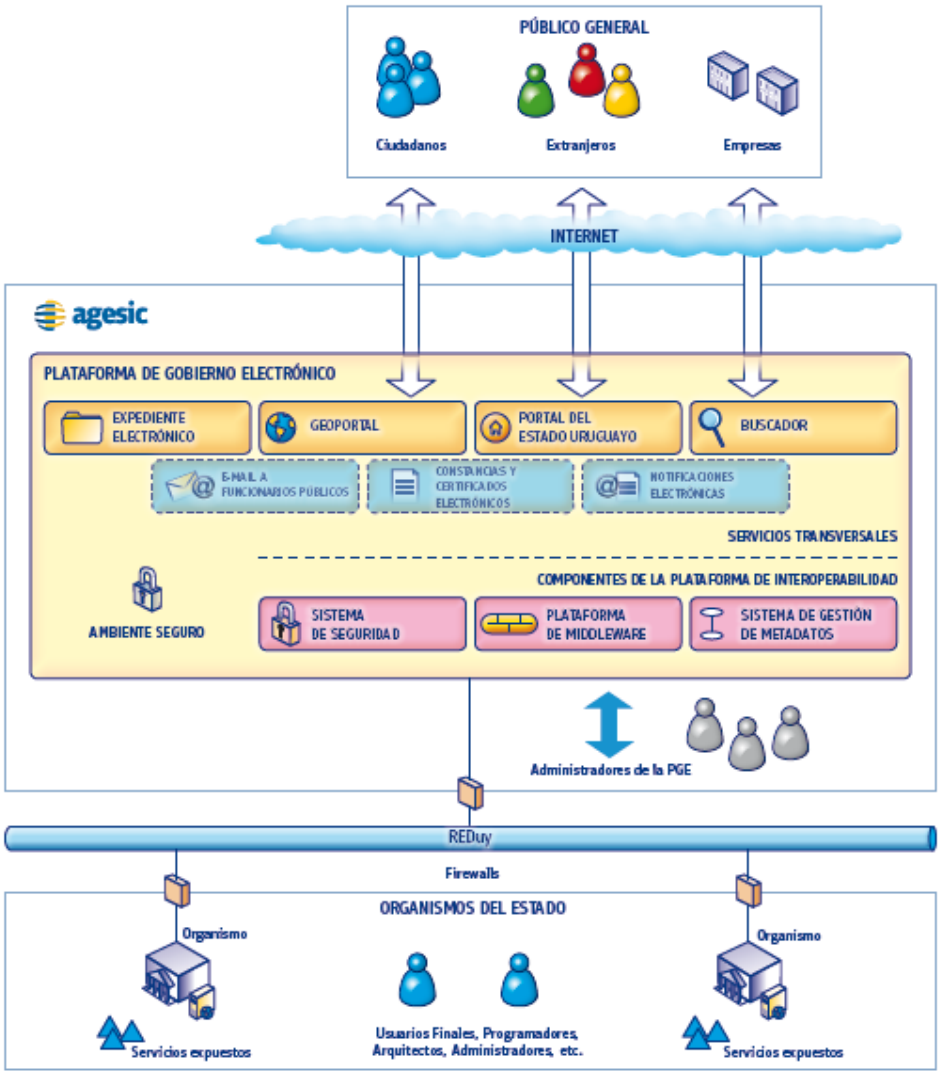

Figura 2.22: Plataforma de gobierno electrónico del Estado

En la Figura 2.22 se muestra un diagrama del funcionamiento forman parte de la PDI, donde se describen algunos de sus componentes:

- **Plataforma de Middleware** provee mecanismos que facilitan el desarrollo, despliegue e integración de servicios y aplicaciones. Además, cuenta con los componentes necesarios para la implementación de la SOA a nivel del Estado.
- **Sistema de Seguridad** provee servicios de seguridad al resto de los componentes. Brinda mecanismos que permiten realizar auditorías de seguridad en la PGE, aplicar políticas de acceso asociadas a los servicios publicados en la plataforma, y facilitar el acceso seguro de los

organismos a la PGE.

- **Sistema de Gestión de Metadatos** provee una especificación de alto nivel de los conceptos relativos a servicios públicos, de forma de evitar, o eventualmente resolver, ambigüedades en el manejo de estos conceptos por parte de los organismos.

En un informe reciente sobre el índice global de gobierno electrónico elaborado por la Organización de las Naciones Unidas (ONU), posiciona a Uruguay en el primer lugar entre los países de América Latina y el Caribe. El reporte mundial de E-Gobierno creado por la ONU da cuenta de la situación de los países miembros en materia de gobierno electrónico. AGESIC retribuye el logro en la mejora del *raking* a distintos factores como posibilitar trámites disponibles en línea, información y trámites en un solo lugar, gobierno abierto, buen uso de las tecnologías de información y comunicación destacando a la plataforma de interoperabilidad con 81 servicios disponibles para el intercambio de información entre organismos, políticas de seguridad de la información aprobadas y sistema de gestión en los organismos públicos [**?**].

Capítulo 3
Análisis

AGESIC es la agencia encargada de gestionar la plataforma de gobierno electrónico del país. Entre los proyectos que componen los servicios electrónicos administrados, se encuentran el "Portal del Estado Uruguayo", un sistema de expedientes electrónicos y la "Plataforma de Interoperabilidad", siendo esta última el objeto de este estudio.

La PDI consta de tres componentes de software principales: la plataforma de middleware, el sistema de seguridad y el sistema de gestión de meta-datos [32]. La plataforma de middleware tiene el propósito de servir como punto intermedio en la comunicación entre organismos. Todos los servicios registrados por AGESIC son accedidos a través de esta plataforma. El sistema de seguridad permite verificar credenciales y autorizar/desautorizar el acceso a servicios desplegados sobre la plataforma de middleware, así como también la realización de auditorías sobre el acceso. El sistema de gestión de meta-datos es una herramienta que almacena información que describe a los servicios en un lenguaje de alto nivel con el fin de aumentar la comprensión sobre la o las funcionalidades provistas por cada servicio.

En la actualidad, no todos los servicios provistos por entes de gobierno forman parte de esta PDI ya que otros han desarrollado sus propios ejemplos de esto con anterioridad a la gestión que se realiza hoy en día, aunque gradualmente se está realizando una integración de las mismas. Tal es el caso del Banco de Previsión Social (BPS), ente que con anterioridad a la implantación de la PDI, había desarrollado su propia versión de una plataforma de servicios, intercambiando datos con la Dirección General Impositiva (DGI); en la actualidad, sus servicios se están incorporando progresivamente a la centralización a través de la PDI.

3.1. Realidad de AGESIC

Este análisis provee una vista sobre los procesos que asisten y forman parte de la gobernanza en SOA ejecutada en la actualidad. Interesa abarcar los aspectos de gobierno que ocurren bajo jurisdicción de AGESIC ya que las responsabilidades en el desarrollo y mantenimiento de los servicios están

desperdigadas entre los organismos proveedores y los administradores de la PDI. Por este motivo, no todas las etapas de un proceso de gobernanza en SOA son controlables bajo el contexto actual, ya que existen límites en la jurisdicción de la AGESIC.

3.1.1. Proceso de consumo y publicación de servicios

Dentro de la plataforma se puede participar tanto publicando servicios como consumiendo (también es posible participar en servicios multi-institucionales [32]).

Una definición de proceso para comenzar con la participación en la PDI se encuentra en [32] siguiendo un esquema en pasos como los que siguen:

1. La AGESIC entrega información sobre pautas de uso al ente que pretende publicar su servicio o consumir un servicio existente en la PDI.
2. Un responsable en el ente contacta con la AGESIC para planificar la ejecución del proyecto de integración.
3. La AGESIC, en conjunto con el ente, desarrolla actividades de formación y pruebas.
4. El ente define un plan de uso del servicio en cuestión en conjunto con la AGESIC.
5. La AGESIC realiza un seguimiento de las pautas establecidas en el primer punto una vez en producción (sea para un servicio nuevo o una aplicación que consume un servicio existente).

Solicitud para consumir un servicio

Un representante de una entidad de gobierno rellena un formulario disponible con múltiples datos acerca de los motivos de la solicitud [33]. El solicitante debe conocer previamente acerca del servicio que requiere. En el documento se deben especificar campos como el nombre del servicio y la entidad responsable. En caso de que se trate de la solicitud de un nuevo servicio, AGESIC actúa como intermediario en una solicitud a la entidad considerada responsable de proveer tal servicio para su desarrollo. Si en cambio, se trata de un servicio existente, se toma la información en base al catálogo disponible –tomando en cuenta aspectos técnicos relevantes para el desarrollo de un cliente que consuma dicho servicio– y se procede a negociar un acuerdo de nivel de servicios entre el organismo consumidor, y AGESIC y el organismo proveedor en conjunto, además de cualquier otro requerimiento legal de intercambio de información que sea necesario.

Como parte de dicho acuerdo de intercambio, se establece un acuerdo de uso por parte del solicitante, que normalmente implica lo que se conoce como SLA (Service Level Agreement) o “Acuerdo de nivel de servicio” .

Acto seguido, los equipos de desarrollo de la entidad solicitante proceden a trabajar en la producción del software cliente, para lo cual AGESIC brinda un soporte en los aspectos de conectividad, pero no participa de ninguna otra forma más que en asistencia; es una tarea enteramente adjudicada a quien desarrolla las aplicaciones que consumen servicios.

Luego de que la pieza de software se encuentra terminada, la entidad puede comenzar a hacer uso de la misma en base a los acuerdos de uso establecidos.

Solicitud para publicar un servicio

Cuando una entidad de gobierno desea publicar un servicio, ya sea por voluntad propia o ante una necesidad de otro organismo que presentó una solicitud ante AGESIC (el caso más común), el proceso comienza rellenando un formulario para la publicación de servicios en el cual se deben detallar aspectos tanto técnicos como de gestión: personas de contacto, soporte técnico, entre otros. [34] AGESIC no realiza un control acerca del propósito del servicio con motivo de categorizarlos independientemente de la entidad encargada, sino que maneja una categorización por organismo: los servicios se agrupan por proveedor y se asume que un proveedor no intentará publicar servicios que cubran aspectos abarcados por otro proveedor.

Los servicios deben ser desarrollados en base a un documento de pautas estipuladas por AGESIC. Dicho documento contiene información sobre aspectos técnicos a tener en cuenta para el correcto funcionamiento de un servicio dentro de la plataforma [35]. Al momento de la publicación del servicio se realiza la verificación de cumplimiento de estas pautas. Para la ejecución de pruebas se considera que el servicio ha sido pasado por los procesos de verificación de calidad adecuados, y que todo está en funcionamiento del lado de la entidad proveedora.

El despliegue en la plataforma consiste en configurar una política de autorización que implica hacer disponible al servicio para ser vinculado con otros entes para establecer los intercambios de información. Esto no sucede hasta que una entidad solicita el uso del servicio que se quiere publicar. Por otra parte, se crea un *proxy* del servicio dentro de la plataforma de interoperabilidad el cual estará configurado apropiadamente para comunicar a los consumidores con los servidores proveedores del servicio dispuestos por la entidad publicadora. Como parte de este proceso, se realizan pruebas de integración que verifiquen el correcto funcionamiento de la comunicación con el servicio desde la plataforma, aspectos de seguridad de acceso y aspectos de comunicación hacia los servidores que mantienen al servicio funcionando. Existe un procedimiento definido para realizar dichas pruebas.

3.1.2. Mantenimiento y seguimiento de la plataforma

AGESIC no participa en el mantenimiento de los aspectos individuales de los servicios, sino que se encarga de las tareas de mantenimiento general de la plataforma de interoperabilidad. Existen procedimientos de comunicación con las entidades consumidoras de servicios a través de la plataforma, quienes son notificados acerca de mantenimientos programados. Normalmente éstos se realizan en los horarios en los cuales se dan los picos de uso más bajos, de modo de afectar la operativa de la menor forma posible. Estos horarios se definen en base a la información provista por los solicitantes en los formularios de solicitud, así como también a reportes obtenidos a través de una aplicación encargada de monitorear el uso de los servicios de modo de determinar los horarios más convenientes para las tareas.

En los casos en que las tareas de mantenimiento son llevadas a cabo por los propios entes encargados de los servicios, éstos se lo comunican a AGESIC, quien se encargará de notificar a los clientes del servicio en cuestión acerca del mantenimiento programado.

3.1.3. Actores en el proceso

Los encargados de los servicios del lado de los organismos, toman el rol de *Service Custodian* (Custodio de servicio). Si bien éstos no forman parte del equipo activo en las instalaciones, forman parte del proceso por ser los responsables de uno o más servicios ofrecidos por el organismo.

AGESIC actúa en el rol de *Cloud Service Owner* (Propietario del servicio en la nube) y *Cloud Resource Administrator* (Administrador de recursos en la nube), ya que es quien presta parte de la infraestructura necesaria para establecer la comunicación con los servicios; específicamente, se encarga de la infraestructura de la plataforma. Los organismos disponen de sus actores en estos roles para encargarse de la infraestructura propia.

Participa el rol de *Service Administrator* (Administrador de servicio) encargándose del funcionamiento de los mismos sobre la plataforma. Este rol es compartido con los organismos ya que aquellos tienen sus propios administradores trabajando en sus instalaciones.

Un *Schema Custodian* (Custodio de esquema) en AGESIC se encarga de definir esquemas normalizados de meta-datos para los servicios, aunque éstos existen en carácter de recomendación para los organismos. Estos son los datos requeridos en los formularios de inscripción de servicios.

Existe un *Policy Custodian* (Custodio de política) también, quien se encarga de mantener las políticas de seguridad de acceso a la PDI. En este caso son políticas definidas por autoridades de AGESIC en base a las necesidades de seguridad.

Para la recolección de datos acerca de los servicios con motivo de ser utilizados en el descubrimiento (en el catálogo), AGESIC dispone de actores

en el rol de *Service Registry Custodian* (Custodio del registro de servicio), siendo las actividades involucradas compartidas con los organismos ya que éstos proveen la mayor parte de la información.

Encargándose de la arquitectura de la PDI y los estándares de seguridad de acceso y publicación en la plataforma, encontramos actores en los roles de *Enterprise Architect* (Arquitecto empresarial) (quien define la arquitectura de la SOA) y *Enterprise Design Standards Custodian (Custodio de estándares de diseño empresarial)* (quien verifica el seguimiento de los estándares definidos por las políticas que gobiernan la SOA) respectivamente.

Por último, actuando en los aspectos más directamente relacionados con la gobernanza de arquitecturas orientadas a servicios, existen hoy en día un *SOA Security Specialist* (Especialista en seguridad en SOA) encargado de la seguridad de acceso a los servicios en forma específica, y un *SOA Governance Specialist* (Especialista en gobernanza en SOA), encargado de dirigir las tareas de gobernanza en SOA en toda de la institución. Este último rol está presente en todas las etapas de toma de decisión.

Como se puede ver, varios de los roles que se consideran dentro del proceso de gobernanza en SOA no están contemplados dentro de AGESIC ya que no se tiene participación en todas las actividades relacionadas al diseño y desarrollo de los servicios.

3.1.4. Descubrimiento de servicios

En lo que refiere al descubrimiento, no se cuentan con procesos definidos para la recolección de meta-datos sobre los servicios. La catalogación se hace en base a la información básica provista por parte de los organismos al momento del registro. Dado que la información es provista en formularios completados en forma física, es posible que se realicen controles sobre la obligatoriedad de ciertos campos, pero se desconoce la existencia de un proceso estandarizado para completar los campos, más allá de lo indicado en el propio formulario.

3.1.5. Calidad de servicios

En cuanto al monitoreo de servicios, AGESIC se encarga de verificar el estado activo del servicio mediante una consulta tipo "ping" (invocaciones al *endpoint* del proveedor) a la dirección de ruteo establecida en la plataforma de interoperabilidad. Dicho factor no se distingue en ningún modelo de calidad ya que actualmente no existe ningún modelo donde se basen como guía para monitoreo los servicios que publican en la PDI. Otro factor que se analiza es la cantidad de invocaciones sobre los servicios.

El cumplimiento de los acuerdos de nivel de servicio, así como también sobre facturación por uso y demás, es responsabilidad de cada entidad. La

identificación del cliente que consume un servicio se realiza a través de los mecanismos de autenticación de la plataforma para estos fines.

3.1.6. Proceso de versionado

Comienza con la necesidad de parte de una entidad proveedora de exponer una nueva versión de servicio. Esta necesidad se realiza mediante un pedido formal a AGESIC, quién tiene el trabajo de hacer alcanzable la nueva versión del servicio hacia los consumidores a través de la plataforma de interoperabilidad. El proceso se compone de dos etapas, las actividades llevadas a cabo por la entidad y las que lleva AGESIC una vez terminadas las primeras. En el primer caso la entidad lleva a cabo las tareas de implementación, testeo y accesibilidad. Una vez concluida la primera etapa, AGESIC crea una nueva entrada en la PDI (*proxy*) para su acceso, y en el rol de administrador de la plataforma se debe habilitar el tráfico[1] sobre el nuevo servicio.

A modo de ejemplo se nota en la Figura 3.1 que la PDI se encuentra publicando servicios de dos entidades, A y B, cada uno de ellos presenta una versión. Luego de un tiempo razonable las entidades generan una nueva versión del servicio propio, por lo que luego de terminadas las labores de éstas, AGESIC crea dos *proxies* en los puertos 60002 y 60004 para las entidades A y B, respectivamente. A través de la url *http://<agesic_dominio>/<path>_V1.0* los consumidores pueden acceder a los servicios publicados. En este caso el consumidor A invoca el *Web Service* WS V1.0 perteneciente a la entidad A a través de la PDI, utilizando el *Web Service* alojado en el puerto 60001.

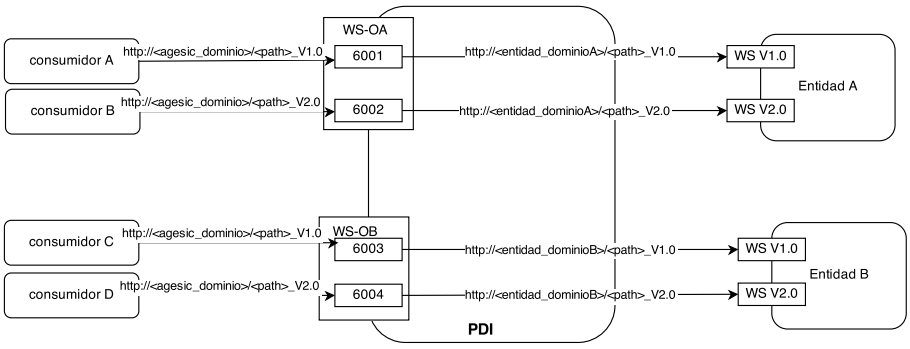

Figura 3.1: Publicación de *proxy* para una nueva versión.

[1] AGESIC utiliza *Firewall XML* que está implementado por el producto IBM Websphere Datapower Xi50, permitiendo o negando el acceso al servicio invocado

3.1.7. Catálogo de servicios

AGESIC presenta un catálogo de servicios cuyo mantenimiento es realizado por la propia agencia, y a su vez cuenta con los servicios de una mesa de ayuda dedicada a atender requerimientos provenientes de las instituciones públicas usuarias del servicio. Es una motivación tener esta información a disposición de cualquier interesado. Es de interés para empresas proveedoras del estado o cualquier institución, pública o privada, que desee beneficiarse con el mismo [36].

El mismo es accedido a través de una interfaz web disponible bajo el dominio de AGESIC[2] como se ve en la Figura 3.2: Cada *Web Service* cuenta

Figura 3.2: Catálogo de servicio de AGESIC

con un conjunto de metadatos sobre el proveedor de servicio: descripción, *WSDL*, entre otros, y niveles de calidad que se comprometen a cumplir: disponibilidad, tiempos de respuesta, volumen de carga máxima. En la Figura 3.3 se muestra un ejemplo de la vista de detalle de un servicio que ofrece el catálogo, asistiendo al descubrimiento de los servicios publicados. En dicha página no es posible ver información con respecto a otras versiones existentes, así como no muestran datos actualizados sobre sus atributos de calidad.

[2] `http://www.agesic.gub.uy/innovaportal/v/1602/1/agesic/catalogo_de_servicios.html`

Figura 3.3: Servicio del BPS en el catálogo de servicio de AGESIC

3.2. Requerimientos del proyecto

En base al estudio de la realidad presentado, es posible identificar los requerimientos que dan origen a los objetivos del proyecto.

El análisis sobre los procesos que ocurren en la actualidad, da cuenta de la necesidad de mejorar los procesos de gobernanza de servicios, estableciendo las bases para la ejecución de actividades y toma de decisiones a lo largo de todo el ciclo de vida. Los procesos de inscripción y publicación requieren de una actualización hacia la gestión electrónica de trámites, permitiendo optimizar los procesos en base a la automatización de tareas comunes (validación de datos, recepción, etc.), aumentando las posibilidades de mejora en la gestión con funcionalidades que permitan hacer el seguimiento por parte tanto de funcionarios de AGESIC como de los entes que pretenden publicar y consumir servicios. Esta migración uniría a AGESIC y sus trámites que involucran el uso de papel con la tendencia hacia un gobierno electrónico del cual pretende ser parte central. Pero el proceso de gobernanza es visto como un aspecto global de los requerimientos identificados; dentro de áreas más específicas se encuentra también utilidad en la introducción de un modelo de

calidad y un proceso de versionado como parte de las actividades que conforman al ciclo de vida, y un catálogo de servicios que afecte especialmente al área de descubrimiento de servicios.

3.2.1. Modelo de calidad

A partir de la descripción de tres propuestas de modelo de calidad desarrollado en el Capítulo 2, se comparan los modelos a través de los principales atributos de calidad haciendo especial énfasis en aspectos de rendimiento y seguridad de servicios, como se puede notar en la Tabla 3.1. Dado que cada modelo presentaba una gran cantidad de factores, se selecciona un conjunto de atributos basados en el interés del cliente.

Atributos	**OASIS**	**S-Cube**	**IBM**
Response Time	✓	✓	
Scalability	✓	✓	✓
Throughput	✓	✓	✓
Latency	✓	✓	✓
Reliability	✓	✓	✓
Regulatory	✓	✓	✓
Availability	✓	✓	✓
Accessibility	✓	✓	
Successability	✓	✓	
Time To Repair			✓
Transaction Time		✓	
Service Cost	✓	✓	
Usability	✓	✓	
Satisfaction	✓	✓	
Content accessibility		✓	
Confidentiality	✓	✓	✓
Integrity	✓	✓	✓
Authentication	✓	✓	✓
Access Control	✓	✓	✓
Non-Repudiation	✓	✓	✓
Traceability	✓	✓	✓
Data validity		✓	
Data policy		✓	
Data integrity		✓	
Data encryption Non-repudation		✓	
WS Interoperability	✓	✓	

Cuadro 3.1: Comparación de atributos de calidad con los modelos OASIS, S-Cube y IBM

Si bien se analizaron varios modelos de calidad que de acuerdo a reuniones con el cliente incluyen factores relevantes para AGESIC (como se muestra en la tabla) es necesario definir un modelo de calidad que se adapte a la agencia con la finalidad de estandarizar o guiar el uso de las funcionalidades que ofrece la tecnología disponible para las plataformas de interoperabilidad[3], brindando información que permita analizar los rendimientos de los servicios y controlar los criterios de calidad con los que se comprometen los proveedores[4]. Teniendo en cuenta las características de cada modelo, es posible construir un modelo de calidad específico que integre la mayor cantidad de factores comunes de cada modelo y resulten adecuados y/o relevantes para los requerimientos de los procesos de calidad a definir.

3.2.2. Proceso de versionado

De la definición de estrategias de la Sección 2.8.4, es posible identificar aquellas que son aplicables a los procesos de versionado de servicios de AGESIC.

En el primer caso, la estrategia *estricta* es una de dos estrategias que exige mucha participación por parte del gestor, dado a que se maneja un número mayor de versiones *mayores* con respecto a las demás estrategias, y cada liberación de estas versiones desata muchas etapas en paralelo. Es deseable que algunos cambios compatibles puedan ser manejados como cambios de versión mayor, es decir, que se publiquen en otro *proxy*, por lo que la estrategia *flexible* no cuadra con lo buscado. Por último la estrategia *relajada* al igual que la *estricta* presenta mucha gestión, en este caso se da debido a los contratos tan abstractos obligando a la implementación de distintas validaciones.

Es necesario establecer una metodología para el versionado de servicios, con el fin de poder organizar una plataforma que va creciendo día a día. Además de esta elección hay que acondicionar esta estrategia para las necesidades propias, definir cuándo es necesario el pasaje a una versión mayor y cuándo no lo es, entre otros aspectos necesarios para la convivencia de múltiples versiones.

3.2.3. Catálogo de servicios

El catálogo de servicios actual es una forma básica de listado que presenta dificultades a la hora de realizar búsquedas sobre la base de datos que lo respalda. Se considera un conjunto de características deseables en un sistema de catálogo de servicios para la PDI:

[3]La tecnología ESB es un ejemplo de ello ya que permite desplegar aplicaciones y ejecutar tareas de monitoreo sobre la actividad de estas.

[4]A través de acuerdos de nivel de servicio

Presentación del modelo de calidad Cada servicio debe detallar los valores de las medidas de calidad del modelo aplicado. Esto permitirá a quien navegue por el catálogo, visualizar las características de rendimiento y efectividad del servicio y evaluar su utilización en su implementación con información basada en mediciones reales.

Listado de versiones de un servicio Las distintas versiones existentes de un servicio deben ser accesibles desde los perfiles, permitiendo a un usuario ver las características propias de cada versión. Cada una debe ser tratada como un servicio independiente con sus propios meta-datos y valores de métricas de calidad. En los casos de servicios o versiones que han sido retirados de la plataforma, los mismos deben ser excluidos de los resultados de búsqueda, pero permanecer en la base de datos como referencia, siempre indicando su estado de retiro.

Integración automática con la plataforma La publicación de servicios en la plataforma debe funcionar en forma sincronizada con el catálogo de servicios de forma tal que al crearse un punto de acceso con sus meta-datos ingresados, una nueva entidad sea creada en el catálogo mostrando el servicio recientemente publicado, sus meta-datos, estado y mediciones de calidad disponibles, reduciendo así la intermediación administrativa y la probabilidad de que existan diferencias entre los datos públicos y los del servicio real.

Acceso público El acceso al catálogo debe incluir un nivel de seguridad de acceso público con el cual se pueda acceder a información básica sobre los servicios disponibles sin necesidad de iniciar una sesión con usuario y contraseña. Se considera básica toda aquella información que no revele detalles técnicos que pongan en riesgo la seguridad e integridad de los servicios y su punto de acceso en la plataforma.

Interfaz de uso intuitiva El sitio debe permitir una navegación de forma tal que usuarios no expertos o con escasa experiencia en la utilización del sitio, puedan llegar a la información que buscan de forma intuitiva. El diseño de experiencias de usuario (UX) se encarga de diagramar sitios que se adelanten a las actitudes de los usuarios frente a la interfaz.

Calificación e integración con características de red social Los usuarios deben poder aportar información acerca de sus experiencias con los servicios a través de calificaciones o puntajes asignados. El catálogo debe mostrar la media de calificaciones para cada servicio, permitiendo al usuario evaluar su utilización al igual que con otras métricas de calidad que son medidas sobre el funcionamiento lógico. Esta característica (al igual que el etiquetado) abre la posibilidad de desarrollar una función social, extendiendo el sistema de puntajes hacia un modelo

de comentarios que puedan ser de ayuda ante la resolución de problemas e identificación de errores en el funcionamiento que tendrán como consecuencia una mejor comunicación de fallas al proveedor para que estas sean resueltas.

Etiquetas La utilización de etiquetas en los servicios permite una mejor categorización de los mismos. Los usuarios deben tener la posibilidad de visualizar servicios etiquetados con palabras clave, estableciendo relaciones entre ellos para facilitar las búsquedas. Se abre la posibilidad de realizar el etiquetado en forma conjunta, en una especie de integración social en donde los usuarios aportan sus etiquetas. Será necesario implementar un sistema de moderación para tal característica, de ser implementada, pero permitirá darle un sentido comunitario a la caracterización de servicios.

Búsqueda por palabras El catálogo debe poseer un buscador por palabras que liste a los servicios que coincidan con el criterio de búsqueda, tanto en su nombre como en descripción y otros meta-datos.

La intención del catálogo es permitir a los potenciales clientes encontrar en forma rápida un servicio que le provea la información requerida en la implementación de sus sistemas, así como también brindar información acerca de los requerimientos y mecanismos técnicos necesarios para hacer uso de él, y los procedimientos administrativos requeridos para comenzar con su implementación. La mantención del catálogo favorece a la reusabilidad de servicios y la generación de versiones mejoradas de estos[5], reduciendo así el solapamiento de funcionalidades que de otra manera podría ocurrir incluso dentro de un inventario organizado por un mismo organismo.

3.3. Conclusiones del análisis

Con el aumento en la cantidad de servicios gestionados en la PDI, se ha vuelto necesario definir los procesos de gobernanza que son llevados a cabo sobre estos. La realidad estudiada deja ver algunos puntos de mejora en el monitoreo de la calidad, la gestión de versiones y la información disponible en el catálogo de servicios existente.

Un modelo de calidad definido estandariza los criterios para medir y establece cuáles son los atributos de calidad que interesan sobre los servicios con el fin de llevar a cabo el monitoreo necesario para asegurar que los niveles de servicio son los acordados al momento de hacer uso de la plataforma.

[5]La solicitud de servicios especiales que desemboquen en la repetición de características de otros servicios, puede verse reemplazada por la solicitud de modificaciones a un servicio existente, incitando al versionado y consecuentemente su mejora.

Las estrategias de versionado definidas proveen la guía para evitar la utilización de criterios arbitrarios o diferentes ante situaciones iguales o similares de publicación de nuevas versiones, evitando potenciales inconsistencias en la configuración de los servicios.

Un catálogo de servicios que combine sus meta-datos e información actualizada sobre sus atributos de calidad, permitirá a usuarios y potenciales consumidores de servicios, encontrar la información que buscan, mejorando la evaluación que deben realizar a la hora de publicar un servicio o utilizar uno existente, como consecuencia de la mejora en la etapa de descubrimiento sobre el proceso de gobernanza.

Atender estos aspectos con propuestas que mejoren a cada uno, contribuirá en forma positiva tanto a la gestión como al uso de la PDI.

Capítulo 4

Propuesta de solución

Partiendo de los requerimientos establecidos en el Capítulo 3 se elaboró una propuesta de solución enfocada en los aspectos de *gobernanza* en SOA, *calidad*, *estrategias de versionado* y *descubrimiento* de los servicios administrados bajo la PDI. En la Figura 4.1 se muestra un diagrama con los componentes de la solución y la representación de las interacciones que ocurren entre ellos. En ésta, se representa a los servicios como el área central de la propuesta, junto al proceso de gobernanza, la estrategia de versionado, el sistema de monitoreo de la calidad y el catálogo, conformando el conjunto de los componentes que hacen a la solución dentro de la PDI.

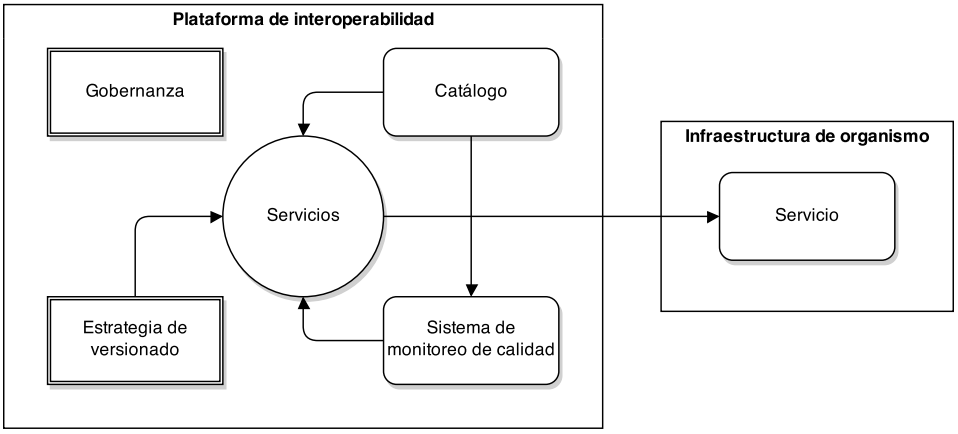

Figura 4.1: Componentes de la propuesta de solución y sus interacciones

El proceso de gobernanza abarca a todos los componentes de la propuesta, por lo que su interacción con éstos está implícita en la Figura. Por otro lado, tanto la estrategia de versionado como el monitoreo de la calidad y el catálogo, tienen interacción directa con los servicios para el cumplimiento de su función: la estrategia de versionado establece las reglas que rigen sobre la actualización y remoción de servicios existentes; el monitoreo mantiene información actualizada sobre los valores de las métricas que interesan de los servicios, y el catálogo funciona como registro con información general y de

funcionamiento de cada servicio registrado en la PDI. Este último, provee además, información sobre los valores obtenidos a través del monitoreo, por lo cual existe una interacción en esa dirección. El funcionamiento ampliado de la PDI, se ve reflejado en la interacción con un servicio alojado en una infraestructura externa, gobernada por un ente del Estado.

4.1. Gobernanza

La solución de gobernanza propuesta se divide en dos aspectos: el *modelo de jurisdicción* y el *ciclo de vida* o proceso aplicados a los servicios alojados en la PDI.

4.1.1. Modelo de jurisdicción

El modelo de jurisdicción consiste en una estructura de SGPO dedicadas a un dominio de servicios independiente, funcionando en forma federada con una SGPO central gestionada por la AGESIC, como se representa en la Figura 4.2.

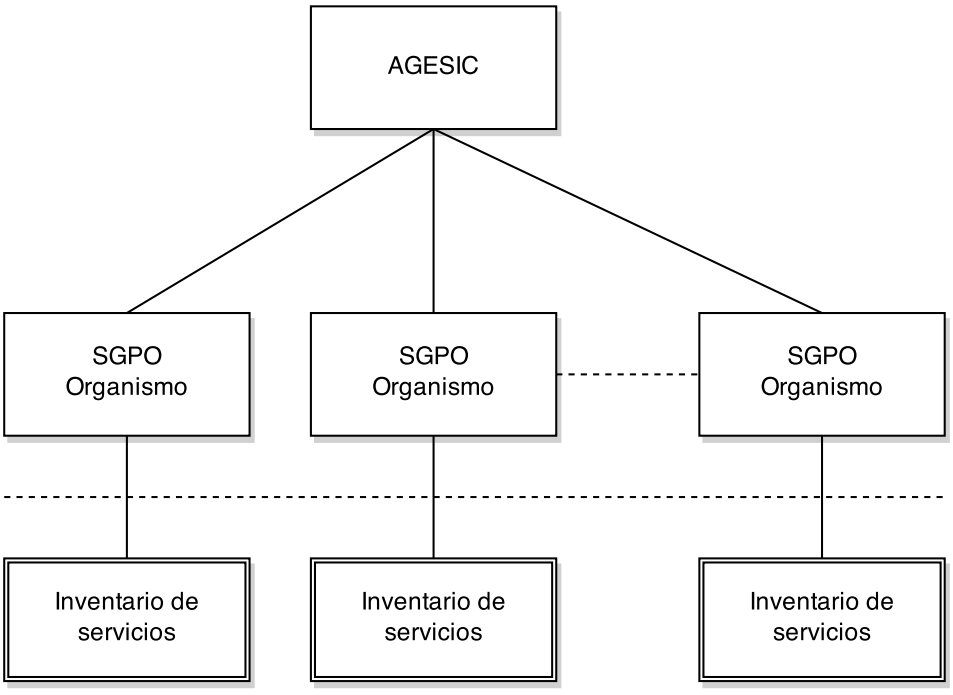

Figura 4.2: Modelo de jurisdicción de SGPO propuesto para la PDI

Para la elaboración de este modelo, se tuvieron en cuenta dos condiciones dadas en la realidad actual:

1. La influencia de AGESIC sobre los procesos de análisis, definición e implementación de servicios, se encuentra limitada a aspectos de opción tecnológica, ya que las etapas iniciales del proceso ocurren en cada organismo por separado.

2. La división de carácter político de los organismos, coincide con las del tipo de información, asegurando que dos de ellos no se encuentren desarrollando servicios tales que ofrezcan información repetida o que se solapen en la función que cumplen.

El último punto da un claro indicio de separación de dominios, en donde cada organismo proveedor representa a un conjunto de servicios relacionados entre sí por una misma área de negocios.

El modelo se ajusta a la realidad actual de los organismos proveedores, en la cual cada uno aplica su proceso administrativo a su dominio de servicios, a la vez que le otorga a la AGESIC la responsabilidad de establecer los lineamientos que rigen a todos los organismos desde la SGPO central.

La propuesta flexibiliza las estructuras de gobernanza internas de cada ente estatal al tomar en cuenta que su autonomía les permite aplicar sus propias subdivisiones administrativas (por ejemplo: SGPO por debajo de la del organismo, ciclos de vida de servicios distintos, entre otros); esta propuesta mantiene una visión de caja negra con respecto a los procesos administrativos en cada organismo, lo que se representa en la Figura como la separación por línea punteada entre la SGPO del organismo y su inventario de servicios correspondiente.

La gestión general aplicada desde la SGPO central se limitará a las actividades sobre las que tiene influencia la AGESIC hoy en día: aspectos técnicos de publicación, de seguridad, procedimientos, y todas las actividades relacionadas a la gestión de servicios desarrollados dentro de la agencia. Todas aquellas decisiones que influyan sobre procesos internos a una SGPO independiente, no podrán ser tomadas por la SGPO central.

La aplicación y comunicación de este modelo a los organismos, les permitirá mantener a éstos a sus estructuras administrativas y de gestión actuales, y le dará visión a la AGESIC sobre la estructura jerárquica de aplicación de la propuesta de gobernanza.

4.1.2. Ciclo de vida de servicios

En la Figura 4.3 se presenta la secuencia propuesta para la publicación, monitoreo y versionado de servicios en la PDI gobernada por la AGESIC. Este proceso es una adaptación del original [19] al cual se le agregan las etapas correspondientes a los procesos administrativos y técnicos.

Las etapas contrastadas (de color más oscuro) son las gobernadas por la AGESIC y comienzan a partir de la solicitud de publicación de un servicio. Cada servicio transita por un ciclo similar al mostrado en la figura, previo al

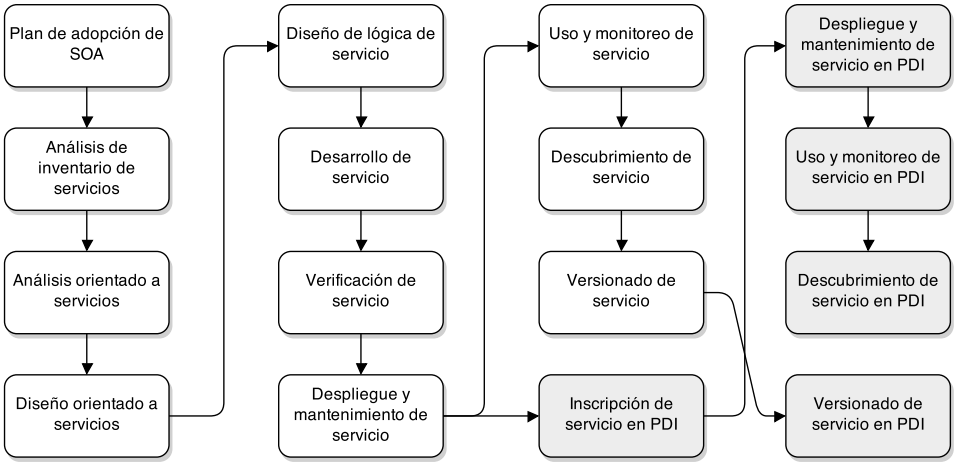

Figura 4.3: Ciclo de vida de proyecto y servicios propuesto

punto de inscripción, y luego cumple con las etapas establecidas en el ciclo propuesto.

La independencia en la gobernanza de servicios de cada organismo (ya establecida en el modelo de jurisdicción propuesto en la Sección 4.1.1), flexibiliza el ciclo de vida anterior a la etapa de inscripción de un servicio, pudiendo un organismo establecer su propio proceso antes de entablar un vínculo con la PDI. Si bien AGESIC establece normas técnicas de compatibilidad para la publicación, no se proponen restricciones a los procesos internos, siguiendo los lineamientos de la autoridad otorgada según el modelo de jurisdicción.

4.1.3. Gobernanza del ciclo de vida en AGESIC

En la Figura 4.4 se muestra un diagrama del proceso propuesto para la gestión del ciclo de vida de un servicio, llevado a cabo por la SGPO central. Durante el mismo, se consideran únicamente las etapas sobre las que la SGPO central tiene jurisdicción, dejando fuera a aquellas que no son gobernadas por la AGESIC. Estas son:

1. Inscripción
2. Despliegue y mantenimiento
3. Uso y monitoreo
4. Descubrimiento
5. Versionado

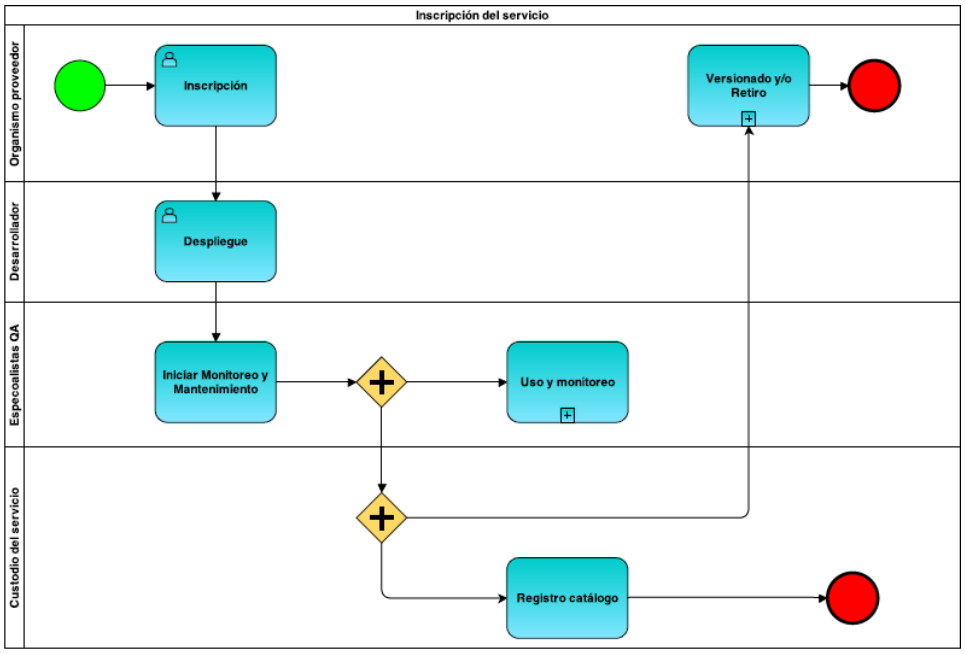

Figura 4.4: Diagrama BPMN del proceso de gestión del ciclo de vida de un servicio en la PDI

Inscripción de servicio

Un organismo proveedor de servicios que desea publicar su servicio deberá ponerse en contacto con AGESIC para dar inicio al proceso. Este comenzará rellenando un formulario vía web en una página alojada bajo el dominio de AGESIC[1] en el cual el personal a cargo de iniciar el trámite deberá detallar datos sobre el servicio a publicar, incluyendo características técnicas, responsables administrativos, ejemplos de uso, entre otros. La información requerida debe tomar como ejemplo el formulario utilizado en la actualidad (incluido como anexo a esta documentación). Este formulario será validado automáticamente verificando la completitud de los datos requeridos, a la vez que enviará una confirmación de recepción a la o las direcciones de correo electrónico especificadas con una copia de los datos ingresados en los campos y un código único de trámite o *ticket* para la realización del seguimiento. La generación de este código es responsabilidad del sistema de gestión de la inscripción de servicios, el cual permitirá al personal administrativo de AGESIC la actualización del estado de los trámites, con el fin de la realización del seguimiento del proceso por parte de los responsables en la parte proveedora.

Los trámites deberán transitar por los estados de la Figura 4.5 que sirven para identificar las tareas restantes. Para cada estado se debe ofrecer un estimativo en días hábiles restantes para la finalización de los procesos

[1] `agesic.gub.uy`

requeridos hasta pasar al siguiente. A continuación se describe la situación en cada estado del proceso:

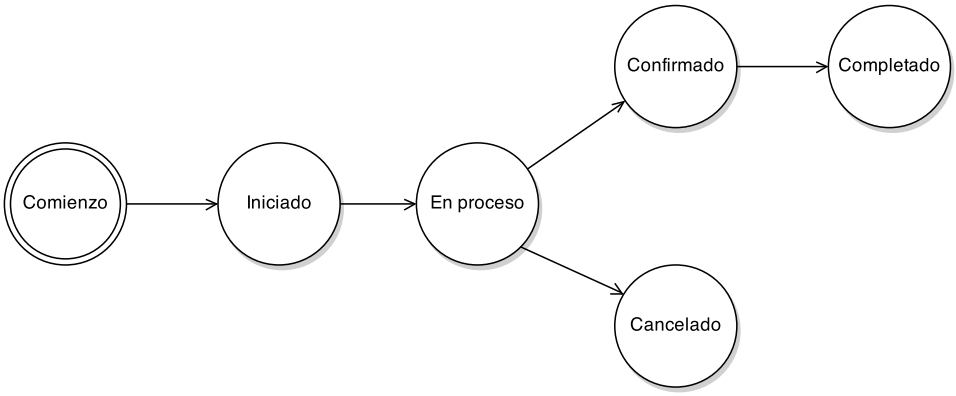

Figura 4.5: Diagrama de estados del proceso de inscripción

Iniciado El formulario ha sido validado y los datos se han almacenado en una base de datos para ser procesados por AGESIC. En esta etapa se debe enviar una confirmación del inicio del trámite a la dirección de correo electrónico provisto por el responsable de la inscripción.

En proceso Los datos han sido recibidos por AGESIC y se han comenzado las tareas necesarias para definir los requerimientos para la publicación. Mientras el proceso se mantiene en este estado, se establecerán una o más comunicaciones con el responsable para negociar las pautas de la publicación.

Confirmado Una vez acordadas las pautas, el servicio pasa a estar confirmado y se comienza con las tareas de definición del acuerdo de nivel de servicios, y la configuración del punto de acceso en la PDI. El acuerdo debe establecer los niveles de servicio desde el proveedor en dirección hacia la PDI de forma tal que sea posible generar nuevos acuerdos entre AGESIC, el proveedor y los potenciales consumidores.

Cancelado En caso de no alcanzar un acuerdo en el estado de procesamiento, o de que el responsable decida retirar la solicitud, el trámite pasa a este estado en forma permanente. No se podrá retomar este trámite, por lo que se deberá volver al estado inicial.

Completado Una vez finalizado el proceso de despliegue del servicio, el trámite pasará a estar completado en forma permanente.

Estos abarcan estados generales, pero se podrán identificar nuevos a medida que el proceso se optimice y así se crea necesario. Este esquema de

funcionamiento es aplicado también para los procesos de versionado de servicios, ya que los mismos deben ocurrir en forma similar a la inscripción de nuevos servicios. Se considera la posibilidad de diseñar el sistema de inscripción vía web de forma que se integre a una aplicación de gestión de trámites existente que así lo permita, adoptando los estados provistos de fábrica por dicha aplicación, en lugar de los propuestos.

Se pretende que durante esta etapa se brinde especial cuidado a la disponibilidad y actualización de la información con respecto al progreso del proceso, preferentemente por intermedio del *sitio Web* y –en forma alternativa– otros medios de comunicación directa.

Despliegue y mantenimiento

Despliegue Esta tarea será asignada a personal de operaciones, teniendo este el entrenamiento y/o experiencia previos requeridos para la configuración del ambiente de producción de servicios en la PDI.

El procedimiento requiere la aplicación del patrón *proxy* (desarrollado en el Capítulo 2); se debe configurar un URL en el dominio de la PDI tal que un cliente pueda enviar una solicitud –utilizando los protocolos establecidos según acuerdos previos entre el proveedor y AGESIC–, que esta sea redirigida a la infraestructura en la que el proveedor aloja al servicio original y que pueda recibir la respuesta entregada por la lógica del servicio y enviada de vuelta al consumidor que originó la solicitud, como se muestra en la Figura 4.6. A cada *proxy* alojado en la PDI se le aplicarán las configuraciones de seguridad y autenticación según lo requieran el servicio y las políticas definidas conjuntamente con el organismo proveedor, en base a los datos provistos en el formulario de inscripción y a los acuerdos de nivel de servicio. Otras consideraciones técnicas quedarán ligadas a la particularidad de cada situación y a las limitaciones de la tecnología utilizada.

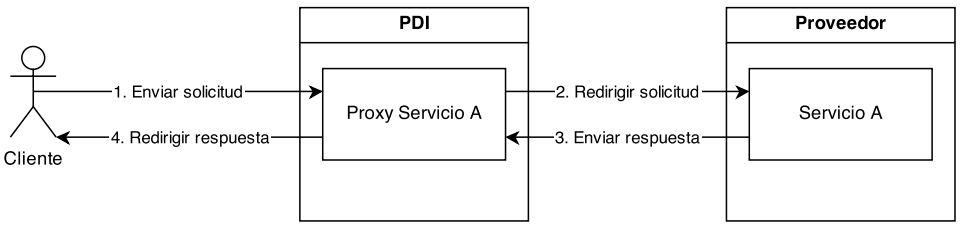

Figura 4.6: Proceso de comunicación Cliente - PDI - Proveedor una vez finalizada la configuración del *proxy*

Al completarse la puesta en producción – verificando el establecimiento de la comunicación en forma exitosa–, un especialista en aseguramiento de la calidad debe trabajar con los operadores en la verificación de cada punto del acuerdo de nivel de servicio, validando el cumplimiento e indicando en caso

contrario aquellos puntos que no sean satisfechos, los cuales serán atendidos por desarrolladores y especialistas técnicos para determinar las acciones correspondientes; esta etapa no debe considerarse completada en tanto los niveles de calidad estipulados no se cumplan. Se deben considerar sólo las métricas cuyas medidas puedan ser tomadas en forma instantánea; se excluyen de esta consideración a aquellas que requieren un periodo de tiempo extendido para su medición.

Mantenimiento Periódicamente, especialistas técnicos y administradores podrán realizar tareas de diagnóstico de servicios y del funcionamiento de la infraestructura local de la PDI. Estas tareas se consideran como parte del mantenimiento periódico y/o excepcional. En los casos en que la ejecución de estas tareas requiera de la interrupción del acceso normal a los servicios en forma inminente, se deberá tomar contacto con los organismos afectados con una antelación definida como parte de una política general de AGESIC[2]. La información sobre horarios de utilización provista en los formularios de inscripción y consumo de servicios, — y en la aplicación de reportes—, permitirá al equipo una programación adecuada de las tareas, de forma de reducir el impacto al mínimo posible. Se debe tomar en cuenta a las tareas de mantenimiento llevadas a cabo en las infraestructuras esparcidas en los organismos, para lo cual será necesario establecer acuerdos específicos con cada uno de ellos con el fin de coordinar el trabajo conjunto y notificar de interrupciones programadas con la antelación especificada.

Con el fin de facilitar el acceso a la información sobre interrupciones –tanto a proveedores como a consumidores–, se debe contar con un calendario de acceso público en el *sitio Web* de AGESIC, el cual deberá mostrar las fechas programadas de mantenimiento especificando una lista con los servicios afectados y sus proveedores, permitiendo configurar alertas por correo electrónico a direcciones específicas, principalmente las de los responsables en los organismos proveedores y clientes de servicios afectados.

Uso y monitoreo

Un sistema de monitoreo continuo y automatizado debe funcionar sobre los servicios de la plataforma, generando reportes sobre los datos, que servirán luego para determinar que los valores medidos sobre las métricas establecidas en los acuerdos de nivel de servicio, se mantengan en los umbrales acordados. Parte de las tareas que toman lugar durante esta etapa son los planes de acción definidos ante las situaciones en las que los valores medidos escapan a los umbrales establecidos. Estos planes deben establecer por escrito las acciones a tomar con la mayor precisión posible, así como los responsables de los mismos y la forma de verificación de la efectividad

[2]El tiempo de antelación puede ser un acuerdo entre partes o establecido por la agencia.

del plan. Las acciones a tomar deberán ser elaboradas por un equipo especializado y serán dependientes del servicio y métrica de la que se trate; el proceso de definición podrá incluirse como parte de las tareas que rodean a la generación de los documentos de acuerdo de nivel de servicios para cada caso.

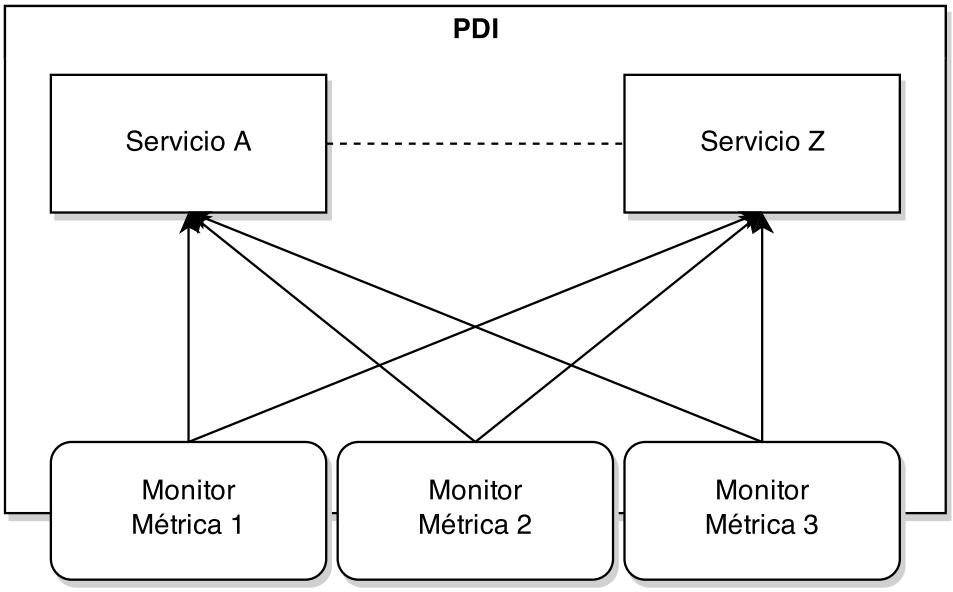

Figura 4.7: Monitoreo de servicios desacoplado

Para evitar la necesidad de acceso a las plataformas administradas por los organismos proveedores, la tecnología utilizada para el monitoreo debe permitir la coexistencia independiente de los módulos de análisis con respecto a los servicios (figura 4.7), esto es, que los primeros puedan realizar su tarea efectivamente y de forma no invasiva para con la lógica del servicio, ya que es una característica deseable que el monitoreo no afecte de forma sensible a la utilización. Dado que la cantidad de servicios puede alcanzar valores que afecten el rendimiento general de la plataforma, se debe establecer un umbral de recursos disponibles para el monitoreo, buscando que su rendimiento máximo no se vea afectado por la sobrecarga producto del análisis. Se pretende un monitoreo efectivo que mantenga la estabilidad, confiabilidad, y comportamiento consistente del servicio monitoreado. [19]

Descubrimiento

Con el fin de facilitar el acceso a información, los servicios deben disponer de meta-datos que describan aspectos acerca de su funcionamiento; ejemplos de este tipo de datos son aquellos que especifican parámetros de entrada y salida. Los meta-datos serán solicitados a los proveedores, tanto

a través del formulario como en forma directa cuando así se requiera (por ejemplo, cuando se agreguen nuevos meta-datos a servicios ya existentes). Personal en el rol de custodio de servicios se encargará de revisar los datos de los servicios para asegurar la completitud y veracidad de los mismos. Esta información será desplegada en un registro central o catálogo de servicios bajo la gestión de la SGPO central y será independiente de los potenciales registros gobernados por cada organismo[3], ya que contendrá sus propios meta-datos predefinidos y almacenados al momento de la publicación/migración; la intención es establecer una tendencia a la centralización del catálogo, absorbiendo todo sub-catálogo existente.

Versionado y retiro

La publicación de nuevas versiones de servicios deberá ajustarse a una serie de criterios definidos en los mecanismos de versionado discutidos en profundidad en la Sección 4.3. Cuando un organismo proveedor planea el despliegue en producción de una nueva versión de un servicio ya publicado en la PDI, debe hacer llegar una notificación a AGESIC, la cual iniciará un proceso administrativo que convocará a especialistas técnicos y desarrolladores para la estimación de tiempos, procesos y recursos necesarios para el despliegue de una nueva versión, tomando como guía al mecanismo introducido en la Sección 4.3. El proceso administrativo de versionado seguirá los mismos estados que el proceso de inscripción de un nuevo servicio (descritos en la Sección 4.1.3), permitiendo hacer el seguimiento del proceso a través de la interfaz ya desarrollada para la gestión de la inscripción de servicios. AGESIC debe establecer los plazos para la realización de la configuración e informar sobre éstos al organismo proveedor para coordinar las acciones conjuntas a tomar, especialmente en los casos en que la nueva versión reemplazará a la ya existente, potencialmente afectando a los consumidores actuales. Con respecto a estos últimos, deberán ser notificados de posibles bajas de servicios en los plazos estipulados en una guía establecida por AGESIC, siguiendo los criterios aplicados a las tareas de mantenimiento desarrolladas en la Sección 4.1.3.

El proceso de retiro de un servicio se rige por las etapas de la Figura 4.8. El retiro debe ocurrir en forma paulatina con el fin de preparar a los consumidores para la utilización de un nuevo servicio que cubra la necesidad satisfecha por el servicio a retirar, o para la búsqueda de una alternativa que permita mantener a los sistemas dependientes en funcionamiento. AGESIC debe solicitar a un organismo que desea retirar un servicio, un calendario en el cual se establezca un período durante el cual el servicio estará disponible hasta ser retirado definitivamente. Durante el tiempo en que el servicio esté disponible con conocimiento de su retiro, se pondrá en un estado especial de

[3]En la actualidad existen registros independientes de servicios como el del Banco de Previsión Social (BPS) y la Dirección General Impositiva (DGI)

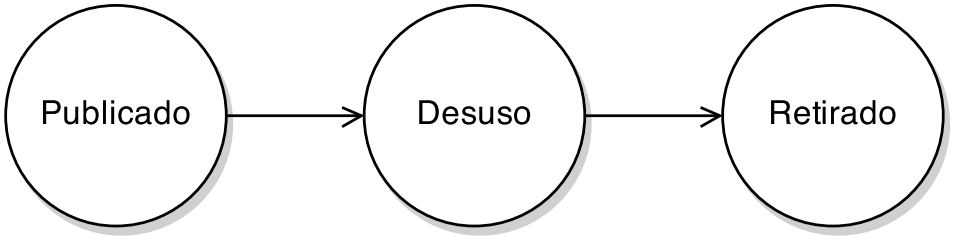

Figura 4.8: Proceso de retiro de un servicio

Desuso, para marcarlo como servicio “a retirar"; esto se verá reflejado también en el catálogo *Web* en caso de que el servicio esté listado en el mismo. Los consumidores del servicio deberán ser notificados inmediatamente, asegurándoles el mayor tiempo posible para realizar las modificaciones necesarias en sus operaciones dependientes. Una vez concluido el periodo establecido por calendario, el servicio debe ser dado de baja en la PDI, eliminando cualquier dependencia interna (ejemplo de ello es la actividad de monitoreo, ya que no será necesaria una vez retirado). En los datos de registro, el servicio debe actualizar su estado a *Retirado*, deshabilitando la posibilidad de iniciar el proceso de suscripción.

En esta etapa se debe enfatizar la sincronización (automática o manual) entre el alojamiento de puntos de acceso a servicios en la PDI y el registro público de servicios, ya que la información sobre éstos permitirá buscar alternativas en los casos en que un servicio del que dependen pase a tener una nueva versión que requiera de la introducción de cambios, o pase a ser retirado.

4.2. Modelo de calidad

Basándose en el meta-modelo de calidad abstracto, el análisis de requerimientos, y sucesivas reuniones de intercambio con el cliente, se propone el conjunto de dimensiones y factores definido en la Figura 4.9.

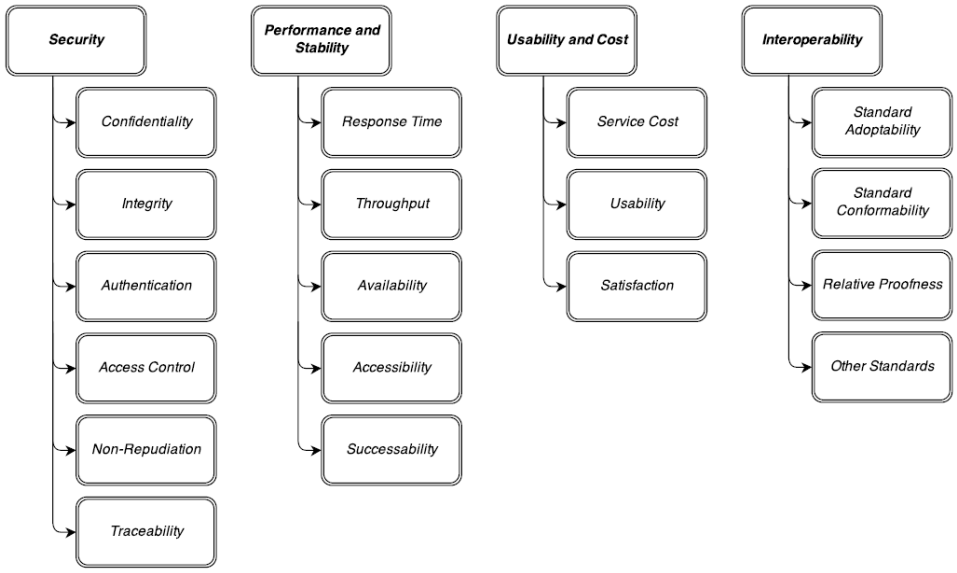

Figura 4.9: Propuesta de modelo de calidad: dimensiones y factores

Las dimensiones presentadas en la solución fueron agrupadas en base a los criterios expresados por el cliente, consistiendo en:

- Factores vinculados con la seguridad de servicios
- Factores vinculados con la inalterabilidad de servicios
- Factores que involucran y/o afectan a un cliente que utiliza servicios
- Factores cuyos resultados varían con el tiempo (ejemplos: disponibilidad, rendimiento)

Los factores de cada dimensión están basados en los modelos de calidad analizados en la Sección 3.2.1, a partir de los cuales —a través del intercambio de ideas con el cliente y en base a los atributos que son de interés para la operativa de la PDI— fueron seleccionados y agrupados, tomando su definición del modelo al cual pertenece cada uno. El conjunto resultante está compuesto por los factores de los siguientes modelos:

OASIS: Tiempo de respuesta *(Response Time)*, Disponibilidad *(Availability)*, Grado de éxito *(Successability)*, Confidencialidad *(Confidentiality)*, Integridad *(Integrity)*, Autenticación *(Authentication)*, Control de acceso *(Access Control)*, No repudio *(Non-Repudiation)*, Trazabilidad *(Traceability)*, Adaptabilidad a estándares *(Standard Adoptability)*, Conformidad con estándares *(Standard Conformability)*, Prueba

relativa *(Relative Proofness)*, Otros estándares *(Other Standards)*[4].

S-Cube: Rendimiento *(Throughput)*, Satisfacción *(Satisfaction)*, Usabilidad *(Usability)*, Costos de servicio *(Service Cost)*.

IBM: Accesibilidad *(Accessibility)*.

4.2.1. Descripción de las dimensiones y factores de calidad

La solución consta de cuatro perspectivas de análisis o dimensiones:

Seguridad *(Security)*
Enfocada en el monitoreo de los aspectos de seguridad, de forma de poder analizar diferentes controles que permitan proteger al servicio.

Rendimiento y estabilidad *(Performance and Stability)*
Enfocada en el análisis del rendimiento de los servicios y su comportamiento a lo largo del tiempo.

Usabilidad y costos *(Usability and Cost)*
Análisis de los aspectos que aportan a la facilidad, aprendizaje y conformidad por parte de los clientes que consumen los servicios, junto con los costos que potencialmente representa su utilización.

Interoperabilidad *(Interoperability)*
Evalúa la adaptación de los servicios a estándares, conformación estándar y que los servicios se comunique de forma exitosa con distintas plataformas (*Relative Proofness*) [37].

Dentro de cada dimensión se establecen los factores que la componen en el nivel inmediatamente inferior:

Seguridad

Confidencialidad *(Confidentiality)*
Propiedad en la cual los datos no son puestos a disposición de usuarios no autorizados, entidades o procesos [38]. En general se utilizan técnicas de encriptación para lograr este cometido.

Integridad *(Integrity)*
Protección de servicios/mensajes contra modificación, eliminación o creación [37]. El mensaje debe ser exactamente el mismo que se envió desde el cliente.

[4]Si bien no es un factor directo del modelo de calidad de OASIS, en la propuesta de solución se agrega este factor para dar un mayor análisis en interoperabilidad de servicios basados en distintas relaciones con otros estándares cómo son los de WS-I, según menciona OASIS [37].

Autenticación *(Authentication)*
Es la identificación del consumidor/proveedor de los servicios y la verificación de las credenciales que pueden ser aseguradas por la identificación y confiadas por la transmisión [37].

Control de acceso *(Access Control)*
Control sobre el acceso al servicio/mensaje según el permiso de cada actor [37].

No repudio *(Non-Repudiation)*
Es la propiedad que permite prevenir que emisores y receptores rechacen haber enviado o recibido mensajes [37].

Trazabilidad *(Traceability)*
Capacidad de registrar actividades y grabar los eventos relacionados con la seguridad [37].

Rendimiento y estabilidad

Tiempo de respuesta *(Response Time)*
Es la duración de tiempo desde el momento de enviar una solicitud al servicio hasta momento de recibir una respuesta [37].

Rendimiento *(Throughput)*
Número de solicitudes completadas que tiene un servicio en un periodo de tiempo [26].

Disponibilidad *(Availability)*
Proporción de tiempo en el cual el servidor del *Web Service* se encuentra en funcionamiento [37].

Accesibilidad *(Accessibility)*
Representar el grado en cual es capaz de procesar un pedido en un *Web Service* [27].

Grado de éxito *(Successability)*
Análisis de la capacidad de éxito del servicio para resolver solicitudes en forma correcta [37].

Usabilidad y costos

Costos de servicio *(Service Cost)*
Análisis de los costos que aplican al uso del servicio.

Usabilidad *(Usability)*
Análisis de los elementos que permiten facilitar el uso del servicio.

Satisfacción *(Satisfaction)*
Evaluación del nivel de conformidad o desconformidad con el servicio por parte del uso de los usuarios [26].

Interoperabilidad

Adaptabilidad a estándares *(Standard Adoptability)*
Evalúa cuantas funciones implementadas en el servicio cumplen con la adopción de estándares. Siendo funciones del servicio la necesidad de requerir encriptación de datos, autenticación, formato de documentación, etc [37].

Conformidad con estándares *(Standard Conformability)*
Aquellas funciones que sean adaptables a estándares se evalúa cuantas se ajustan por completo y correctamente a los estándares.

Prueba relativa (*Relative Proofness*)
Indica si los *Web Services* son exitosos en intercambiar y usar información entre dos plataformas de sistemas especiales [37].

Relaciones con otros estándares (*Other Standards*)
Indica si el *Web Service* se relaciona con otros estándares como por ejemplo *WS-I Basic Profile*, *WS-I Basic Security Profile*, *WS-I Reliable Secure Profile*, *WS-I Simple SOAP Binding Profile* [37].

La *Relación con otros estándares* no es un factor de interoperabilidad directo del modelo OASIS, sino que es agregado al modelo para otorgar un mejor análisis de interoperabilidad de servicios, evaluando relaciones con otros estándares (por ejemplo, WS-I Basic Profile) [37].

4.2.2. Definición de las métricas de calidad

Dentro de cada dimensión y para cada factor, se establecen las métricas que lo definen. Algunas de ellas resultan de mayor interés desde el punto de vista del proveedor de servicios que desde el del consumidor, pero se introducen con la intención de asistir en procesos de mantenimiento de la PDI y la comunicación con la misma.

A continuación se listan las métricas de calidad para la dimensión *Rendimiento y estabilidad*:

- **Factor: *Response Time***
  - **Métrica**: *Average Response Time (Client)*
    - **Semántica**: Tiempo promedio en el cual resuelve una solicitud al servicio invocado. Es el tiempo que espera el cliente
    - **Granularidad**: Por cada operación del servicio

    - **Unidad**: Segundos
  - **Métrica**: *Max Response Time (Client)*
    - **Semántica**: Máximo tiempo que llevó resolver una solicitud al servicio invocado
    - **Granularidad**: Por cada operación del servicio
    - **Unidad**: Segundos
  - **Métrica**: *Min Response Time (Client)*
    - **Semántica**: Mínimo tiempo que llevó resolver una solicitud al servicio invocado
    - **Granularidad**: Por cada operación del servicio
    - **Unidad**: Segundos
  - **Métrica**: *Average Response Time (Provider)*
    - **Semántica**: Tiempo promedio en el cual resuelve una solicitud al servicio invocado sin tener en cuenta los tiempo de latencia que puede sufrir el servicio (ejemplo transformaciones)
    - **Granularidad**: Por cada operación del servicio
    - **Unidad**: Segundos
  - **Métrica**: *Max Response Time (Provider)*
    - **Semántica**: Máximo tiempo que llevó resolver una solicitud al servicio invocado sin tener en cuenta los tiempo de latencia que puede sufrir el servicio
    - **Granularidad**: Por cada operación del servicio
    - **Unidad**: Segundos
  - **Métrica**: *Min Response Time (Provider)*
    - **Semántica**: Mínimo tiempo que llevó resolver una solicitud al servicio invocado sin tener en cuenta los tiempo de latencia que puede sufrir el servicio
    - **Granularidad**: Por cada operación del servicio
    - **Unidad**: Segundos

- **Factor:** ***Throughput***
  - **Métrica**: *Throughput (Client)*
    - **Semántica**: Cantidad de solicitudes al servicio, por segundo que puede resolver
    - **Granularidad**: Por cada operación del servicio
    - **Unidad**: Cantidad transacciones/Seg
  - **Métrica**: *Throughput (Provider)*

        - **Semántica**: Cantidad de solicitudes al servicio , por segundo que resuelve el endpoint del proveedor
        - **Granularidad**: Por operación
        - **Unidad**: Cantidad transacciones/Seg

- **Factor:** ***Availability***
    - **Métrica**: *Availability*
        - **Semántica**: Proporción de tiempo en que el servidor de *Web Service* está en funcionamiento
        - **Granularidad**: Servicio
        - **Unidad**: Numérico

- **Factor:** ***Accessibility***
    - **Métrica**: *Accessibility*
        - **Semántica**: Proporción de tiempo en que se invoca al *Web Service* y este se encuentre accesible para ser invocado
        - **Granularidad**: Servicio
        - **Unidad**: Numérico

- **Factor:** ***Successability***
    - **Métrica**: *Successability*
        - **Semántica**: Proporción de tiempo en que el servicio es invocado y recibe una respuesta sin error por parte del proveedor del mismo
        - **Granularidad**: Por operación
        - **Unidad**: Numérico

Las métricas que forman parte de la dimensión *Interoperabilidad y Usabilidad y costos* son las siguientes:

- **Factor:** ***Service Cost***
    - **Métrica**: *Price*
        - **Semántica**: Indica si el servicio tiene costo
        - **Granularidad**: Servicio
        - **Unidad**: Booleano
    - **Métrica**: *Cost for Transaction*
        - **Semántica**: Indica a partir de que cantidad de solicitudes al servicio, se le debe cobrar al cliente
        - **Granularidad**: Por invocación de cada operación del servicio
        - **Unidad**: Numérico

- **Métrica**: *Penalty*
  - **Semántica**: Precio de sanciones al proveedor por incumplimiento de acuerdos de calidad del servicio (SLA)
  - **Granularidad**: Por operación del servicio
  - **Unidad**: Numérico

- **Factor:** ***Usability***
  - **Métrica**: *Documentation Services*
    - **Semántica**: Indica si el servicio dispone de una buena documentación que permita entender e invocar el servicio
    - **Granularidad**: Servicio
    - **Unidad**: Booleano

- **Factor:** ***Satifaction***
  - **Métrica**: *Rating*
    - **Semántica**: Indica el puntaje promedio de satisfacción de los usuarios que evaluaron el servicio
    - **Granularidad**: Servicio
    - **Unidad**: Numérico

- **Factor:** ***Standard Adoptability***
  - **Métrica**: *Standard Adoptability*
    - **Semántica**: Indica si el *Web Service* cumple con algún estándar de adaptabilidad
    - **Granularidad**: Servicio
    - **Unidad**: Boolean

- **Factor:** ***Relative Proofness***
  - **Métrica**: *Relative Proofness*
    - **Semántica**: cantidad de pruebas exitosas en que el servicio funciona correctamente en distintas plataformas
    - **Granularidad**: Servicio
    - **Unidad**: Numérico

- **Factor:** ***Standard Conformability***
  - **Métrica**: *Standard Conformability*
    - **Semántica**: Indica si para los servicios con adaptabilidad de estándares cumple completamente con los estándares
    - **Granularidad**: Servicio

        - **Unidad**: Boolean

- **Factor:** ***Other Standards***

    - **Métrica**: *WS-I Basic Profile*
        - **Semántica**: Indica si el servicio cumple con los estándares de perfil definidos por *Web Services Interoperability Organization*
        - **Granularidad**: Servicio
        - **Unidad**: Boolean

Para la dimensión *Seguridad* las métricas son:

- **Factor:** ***Confidentiality***

    - **Métrica**: *Confidentiality*
        - **Semántica**: El servicio aplica o no la propiedad de seguridad de confidencialidad del mismo
        - **Granularidad**: Servicio
        - **Unidad**: Boolean

- **Factor:** ***Integrity***

    - **Métrica**: *Integrity*
        - **Semántica**: El servicio aplica o no la propiedad de seguridad de integridad del mismo
        - **Granularidad**: Servicio
        - **Unidad**: Boolean

- **Factor:** ***Authentication***

    - **Métrica**: *Authentication*
        - **Semántica**: El servicio aplica o no la propiedad de seguridad de autenticación del mismo
        - **Granularidad**: Servicio
        - **Unidad**: Boolean

- **Factor:** ***Access Control***

    - **Métrica**: *Access Control*
        - **Semántica**: El servicio aplica o no la propiedad de seguridad de control de acceso del mismo
        - **Granularidad**: Servicio
        - **Unidad**: Boolean

- **Factor:** ***Non-Repudiation***
  - **Métrica**: *Non-Repudiation*
    - **Semántica**: El servicio aplica o no la propiedad de seguridad de no repudio del mismo
    - **Granularidad**: Servicio
    - **Unidad**: Boolean
- **Factor:** ***Traceability***
  - **Métrica**: *Traceability*
    - **Semántica**: El servicio aplica o no la propiedad de seguridad de trazabilidad del mismo
    - **Granularidad**: Servicio
    - **Unidad**: Boolean

La implementación de los métodos que se utilizan en las métricas depende de la aplicación en concreto [39], por lo que según la que se trate, se deberán que establecer las operaciones que permitan calcular las métricas del modelo.

4.3. Versionado

La solución del proceso de versionado se divide en la elección de una estrategia, una descripción sobre los tipos de cambio existentes proponiendo para cada uno de ellos cómo actuar ante éstos y la especificación de un formato para la identificación de versiones.

4.3.1. Estrategia de Versionado

A partir del estudio realizado en la Sección 2.8.4 y la comparación entre las estrategias de versionado presentado en la Sección 3.2.2, se concluye que ninguna coincide en forma completa con los requerimientos a cumplir. Debido a esto, se plantea una combinación de los enfoques de las estrategias *estricta* y *flexible*, adquiriendo lo mejor de ambas.

De la *flexible* se integra el comportamiento ante cambios compatibles, generando nuevas versiones menores de contrato, produciendo un bajo impacto sobre la gestión y la forma de trabajo actual. Se utiliza la estrategia *estricta* para aquellos cambios que se consideren inseguros y/o en donde existe un salto importante entre una versión y otra, quebrando con la compatibilidad hacia atrás. Ante un escenario de un cambio incompatible, ambos enfoques actúan igual: se gesta una nueva versión mayor del contrato. Esta combinación de las estrategias soporta la compatibilidad hacia atrás ante cambios compatibles aunque no lo hace sobre compatibilidad hacia adelante debido a la complejidad de implementación de la estrategia *relajada*.

4.3.2. Tipos de cambios

En base a las situaciones descritas por el cliente sobre los casos típicos de versionado de servicios en la PDI, se introducen tres tipos de cambio: *compatibles*, *incompatibles manejables* e *incompatibles*.

Los cambios compatibles se pueden dar en distintos escenarios, donde el proveedor realiza el cambio sobre el *WSDL* original o crea una nueva versión del servicio a partir de la nueva definición del *WSDL* con el nuevo cambio. En el primer caso AGESIC crea un nuevo *proxy* para esta nueva versión exponiéndolo hacia nuevos consumidores, por el otro escenario se procede de la misma manera que al crear un nuevo servicio: se crea un nuevo *proxy* que apunte al nuevo *endpoint* del proveedor. Para ambos eventos los consumidores existentes seguirán utilizando el *proxy* original sin que este cambio los afecte, entretanto los nuevos consumidores harán uso del nuevo pudiendo beneficiarse del cambio introducido.

Los cambios incompatibles manejables son aquellos cambios incompatibles que mediante una configuración en la plataforma se "vuelven" manejables; se dice que un cambio es «compatibilizable» si existe tal configuración. El fin de esta característica es manejar cambios de forma transparente para los consumidores. Las configuraciones pueden implicar la redirección de tráfico y transformación de mensajes a través de plantillas en lenguaje *XSLT*, por lo que es necesario que la tecnología de base de la plataforma de soporte a este tipo operación. Al completar la configuración, además de compatibilizar el cambio —evitando que se quiebre el uso de la versión con los consumidores actuales—, se crea un nuevo *proxy* para futuros consumidores.

Por último, los cambios incompatibles exigirán la creación de un nuevo *proxy* en la plataforma, quebrando la compatibilidad con los consumidores actuales, forzando el pasaje a la versión siguiente. La realización de la transición forma parte de las políticas de gobernanza, por lo que existirá un período durante el cual conviven dos o más versiones incompatibles de un mismo servicio. Existen cambios de este tipo que se vuelven incompatibles por el agregado de parámetros, pero que constituyen un caso en el que se les podría aplicar una transformación para que correspondan a cambios incompatibles manejables, a través de la utilización de valores por defecto para los parámetros agregados.

La Tabla 4.1 contiene una descripción de posibles cambios sobre contratos, ordenados según la categorización propuesta.

En la Figura 4.10 se muestra el estado inicial de la plataforma antes de producirse un cambio de versión. Le siguen las descripciones ante los tipos de cambio introducidos.

Categoría	**Cambio**	**Por ejemplo**
Message Schema	*Cambio Compatible*	Agregado de parámetro opcional
		Cambiar opcionalidad Parámetro (Requerido a opcional)
	Incompatible Manejable	Renombrado de parámetro
		Cambio en tipo de parámetro (Existe transformación posible)
		Cambio en el targetNamespace
		Borrado de parámetro
	Incompatible no Manejable	Agregado de un nuevo parámetro obligatorio
		Cambiar Opcionalidad Parámetro (Opcional a requerido)
		Cambio en tipo de parámetro (No existe transformación posible)
WSDL Definition	Cambio compatible	Agregado de operación
	Incompatible Manejable	Renombrado de operación
		Cambio de EndPoint
	Incompatible no Manejable	Borrado de una operación
		Cambio de Semántica

Cuadro 4.1: Tipos de cambios

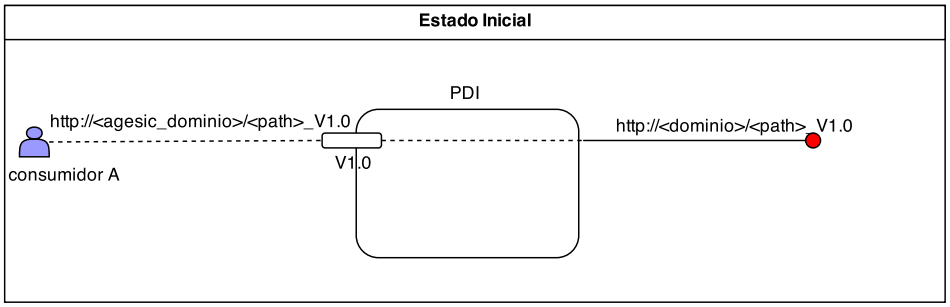

Figura 4.10: Estado inicial

Cambio Compatible

Agregar operación/parámetro opcional Este cambio involucra agregar una nueva operación o un parámetro opcional al *WSDL* original. Como se discutió anteriormente existen dos casos, en el primer el proveedor modifica la definición del *WSDL*, la Figura 4.11 muestra que los consumidores existentes no se ven afectados ya que siguen invocando el servicio a través del *proxy* original, mientras que los nuevos consumidores utilizarán el nuevo *proxy* con la nueva versión del contrato del servicio.

El segundo caso se da sí el proveedor toma la decisión de crear una nueva versión del servicio, por lo que es necesario agregar un nuevo *proxy* en la PDI, de forma que existe un tiempo de convivencia entre la versión actual y la anterior del servicio a través de los *proxies*, como se puede notar en la Figura 4.12. Los consumidores existentes podrán seguir invocando el *proxy* original mientras que este no pase a desuso.

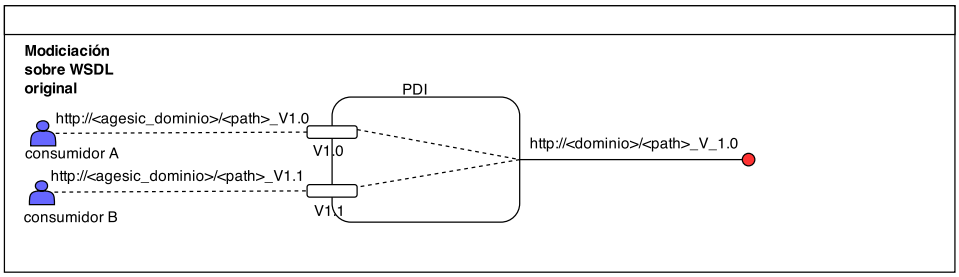

Figura 4.11: Se agrega una operación/parámetro opcional modificando el WSDL

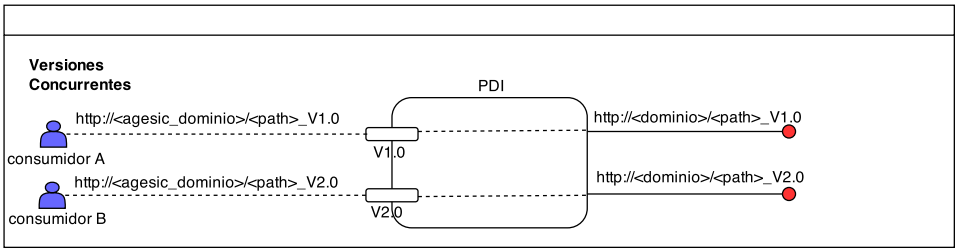

Figura 4.12: Se agrega una operación/parámetro opcional creando un nuevo servicio

Cambio Incompatible Manejable

Cambio de endpoint El proveedor modifica el *endpoint* del servicio provisto a la plataforma. En consecuencia, AGESIC debe modificar la plataforma para que los pedidos sobre el *proxy* se redirijan al nuevo *endpoint* del proveedor, tal como se puede ver en la Figura 4.13.

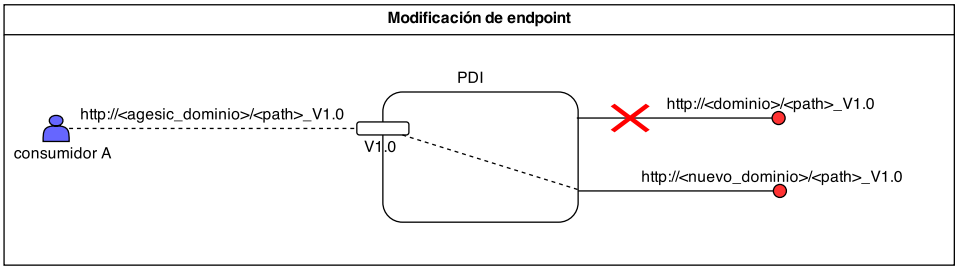

Figura 4.13: Cambio de *endpoint*

Renombrado de operación/parámetro El proveedor renombra una operación o un parámetro del servicio que se expone en la plataforma. En la Figura 4.14 se puede apreciar el diagrama sobre un renombre de operación o parámetro en donde la definición en el *proxy* original contiene el elemento con el nombre anterior, mientras que en el servicio modificado por el proveedor se encuentra el elemento renombrado. Este cambio quebraría con la compatibilidad de los consumidores existentes, por lo que se utiliza el módulo de transformaciones donde el pedido del consumidor y la respuesta de la plataforma son modificados. En la Figura 4.15 se muestra la transformación del pedido, mientras que en la Figura 4.16 se muestra la respuesta.

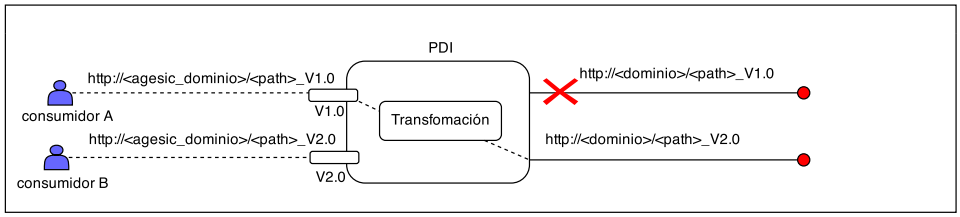

Figura 4.14: Cambio de nombre de una operación o parámetro

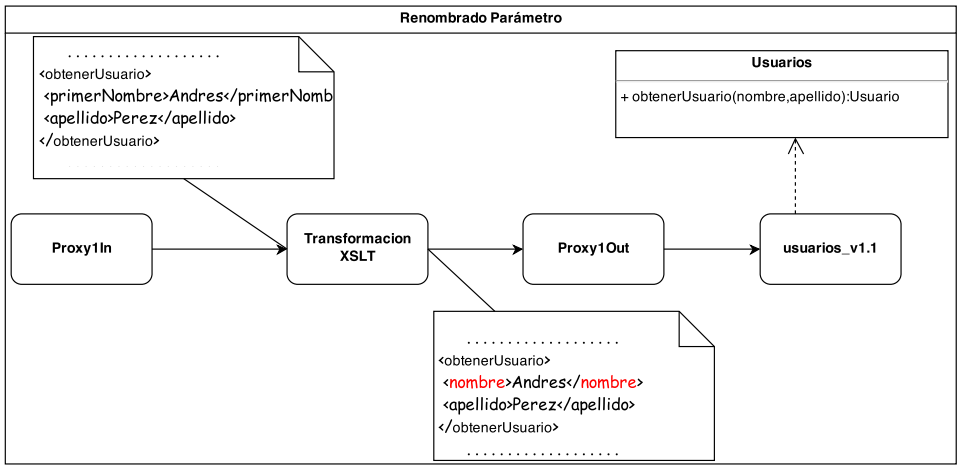

Figura 4.15: Transformación *XSLT* sobre un cambio de nombre de parámetro en el pedido por parte del consumidor

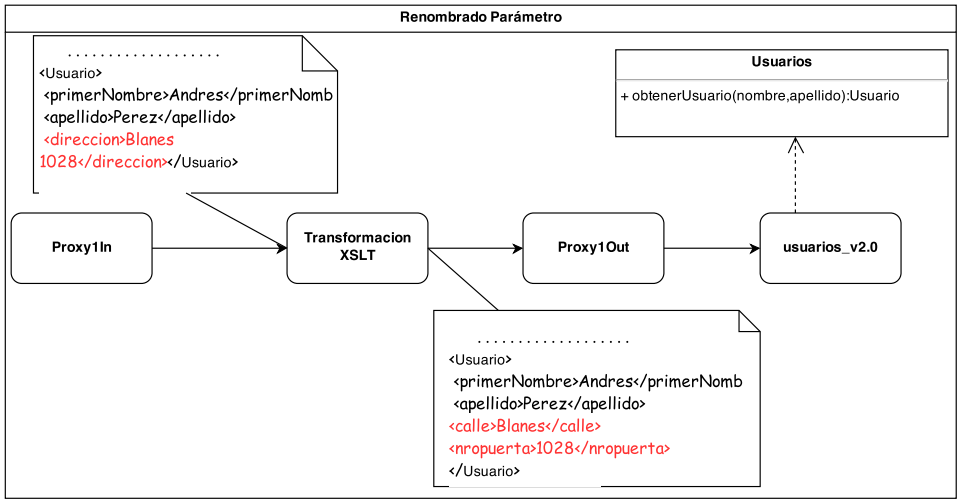

Figura 4.16: Transformación *XSLT* sobre un cambio de nombre de parámetro en respuesta del organismo

Utilización de parámetro por defecto Utilizando el caso del agregado de un parámetro obligatorio y definiendo un parámetro por defecto, es posible compatibilizar el cambio a través de una transformación. En la Figura 4.17 se muestro cómo el parámetro *country* fue agregado luego de la utilización del servicio por parte de algunos consumidores. En este caso se tiene definido el valor AR por defecto, por lo que este cambio pasa a estar comprendido entre los cambios incompatibles manejables.

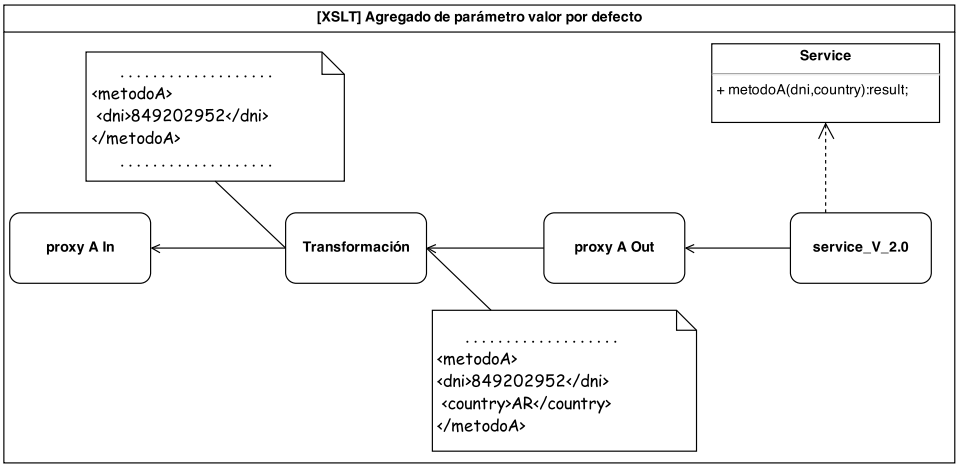

Figura 4.17: Transformación *XSLT*, utilizando valores por defecto sobre el nuevo parámetro.

Cambio Incompatible

Los cambios incompatibles presentan un periodo de migración de servicio durante el cual la plataforma tendrá dos versiones distintas expuestas como se muestra en la Figura 4.18 (basado en la política de gobernanza en SOA propuesta). Luego de que haya caducado el tiempo de transición, el *proxy* anterior ya no estará disponible y los consumidores deberán actualizarse, escenario representado en la Figura 4.19.

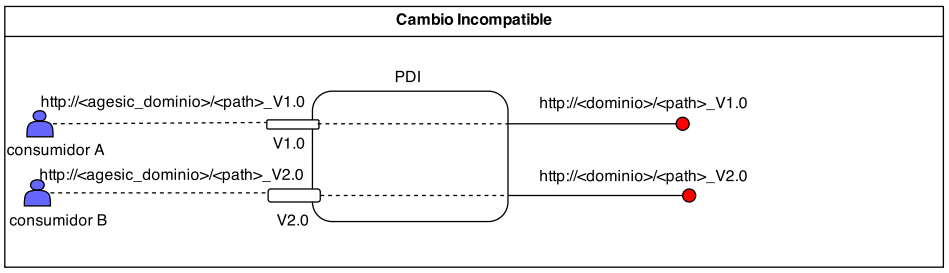

Figura 4.18: Cambio incompatible, versiones concurrentes

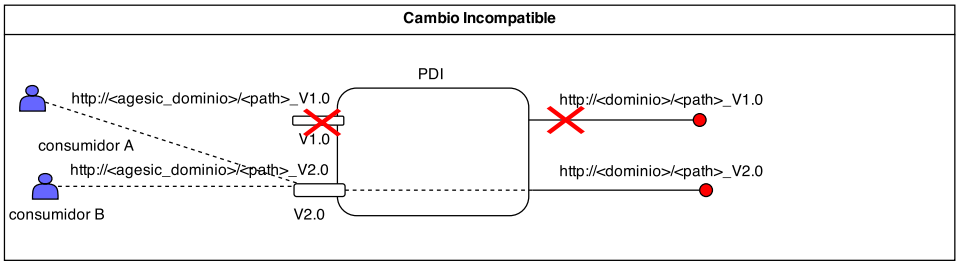

Figura 4.19: Cambio incompatible, caduca el tiempo para la versión anterior

4.3.3. Identificación de Versiones

Se utilizará un formato de tres números separados por puntos, en donde el primer número marca los cambios de mayor versión (X), el segundo corresponde a cambios de menor versión (Y) y el tercero refiere a cambios por correcciones (Z) como muestra la Figura 4.2.

X	Y	Z

Cuadro 4.2: Identificación de versiones

Esta identificación es una medida o magnitud del cambio. Los cambios de versión mayor deberán tener como resultado la modificación del *target-namespace*, quebrando con la compatibilidad de sus consumidores, mientras que los cambios de versión menor se reflejarán en el tag *documentation*, especificando la versión y opcionalmente alguna descripción. Por ejemplo, la definición del *WSDL* de la Figura 4.20, muestra como el servicio se encuentra en la versión *2.1.0*

```
<definitions name="service" targetNamespace="http://<dominio>/<path>_V_2.0"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://<dominio>/<path>_V_2.0"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<documentation>Version 2.1.0</documentation>
...
</definitions>
```

Figura 4.20: Identificación de versión sobre la definición de un *WSDL*

4.4. Catálogo de servicios

En base a la información generada a partir de la etapa de descubrimiento en el ciclo de gobernanza, se concibe un *catálogo de servicios de la PDI*: un registro en línea con una serie de funcionalidades que se adecúan a las características necesarias establecidas en el análisis de la Sección 3.2.3. El acceso a este registro será público a través de la Internet y se hará disponible utilizando un *servidor Web* gestionado por AGESIC bajo su dominio. A continuación se describen las distintas características que se contará con el catálogo de servicios:

Meta-datos De los meta-datos recolectados como parte de la etapa de descubrimiento de un servicio, el catálogo debe requerir en forma obligatoria un subconjunto de ellos:

Nombre Nombre con el que se identifica.

Versión Número de la última versión pública.

Estado Estado actual del servicio (Ver Figura 4.8).

Descripción Un desarrollo sobre la funcionalidad que cumple.

Proveedor Ente estatal que provee el servicio.

Ambiente de publicación Si se encuentra en producción o verificación.

Mecanismos de seguridad Requerimientos especiales de autenticación, encriptación, entre otros.

Parámetros de entrada/salida Ejemplo de invocación y retorno de datos exitoso.

Archivo WSDL Vínculo a la versión correspondiente del archivo WSDL.

Comentarios Comentarios adicionales.

Atributos de calidad El modelo de calidad debe ser destacado en el detalle de los servicios, presentando valores tomados de las métricas correspondientes, organizados según las dimensiones y factores propuestos en la Sección 4.2. Interesa un subconjunto de factores, dejando el resto a criterio de la gestión:

1. Seguridad
   a) Confidencialidad
   b) Autenticación

   c) Control de acceso

2. Rendimiento y estabilidad

   a) Tiempo de respuesta
   b) Rendimiento
   c) Disponibilidad
   d) Accesibilidad
   e) Grado de éxito

3. Usabilidad y costos

   a) Costo de servicio
   b) Satisfacción (medida a través de la calificación de los usuarios)

Histórico de versiones Si bien la intención es que los usuarios sólo puedan suscribirse a una única versión de un servicio, por la estrategia de versionado propuesta, es posible encontrarse ante un escenario de múltiples versiones de un servicio coexistiendo en la plataforma. Por lo tanto se contará con la posibilidad de acceder al histórico de versiones en la vista de detalle de un servicio, permitiendo que usuarios de versiones en *Desuso* accedan a los meta-datos y detalles completo de las mismas, así como también otros usuarios puedan observar los cambios en la evolución desde versiones ya retiradas, asegurándose de que su estado actual (retirada o en desuso) sea correctamente indicado.

Integración y sincronización con la PDI Se podrá agregar servicios al catálogo, aunque esto depende directamente con la creación del punto de acceso en la PDI, por lo que se brinda la posibilidad de realizarlo en forma manual como automática. En el último caso existirá un mecanismo de sincronización disponible en la tecnología utilizada en la PDI como ser un API pública o cola de mensajes que soporte la suscripción, y la capacidad de la aplicación de registro de aprovechar dicho mecanismo.

Solicitud e inicio de trámites desde la interfaz A través de la aplicación se podrá iniciar la gestión administrativa para obtener información adicional sobre un servicio, e incluso enviar los formularios correspondientes al inicio del proceso de inscripción para el consumo de un servicio que han encontrado en el catálogo.

Capítulo 5

Implementación

En este capítulo se describe el proceso de construcción de un prototipo de software con el objetivo de validar los aspectos considerados críticos de la propuesta introducida en el Capítulo 4. En primer lugar, se presenta el diseño del prototipo objetivo, seguido del desarrollo de un análisis sobre las principales herramientas disponibles en el mercado actual para cumplir con las características del diseño, continuando con la toma de decisiones a partir de dicho análisis, una breve reseña sobre limitaciones encontradas en la implementación, y un caso de estudio que describe al diseño en funcionamiento.

5.1. Diseño de implementación de solución

En la Figura 5.1 se presenta un diagrama con el diseño del prototipo y la interacción entre sus componentes.

El *ESB* es el *middleware* que soporta a los servicios de la PDI. La elección de una tecnología de este tipo tiene su base en la sencillez de uso para el despliegue y administración de servicios en funcionamiento, así como también, en su capacidad de monitorear directamente los factores de calidad relacionados con los aspectos técnicos de los servicios (tiempo de respuesta, rendimiento, etc.).

Por otro lado, existe un cliente del ESB que provee una interfaz para la administración de los servicios alojados en él. Dicha interfaz es el punto de acceso de operadores de la AGESIC para la configuración de nuevos servicios y el mantenimiento de los ya existentes.

Además, en un nodo independiente del ESB y su aplicación cliente, existe una aplicación *Web* que ofrece un catálogo con los servicios disponibles en la PDI, mostrando sus meta-datos y datos de calidad, a través de la sincronización periódica con el ESB. Dicha aplicación es accesible por usuarios dentro y fuera del contexto de las actividades de la AGESIC.

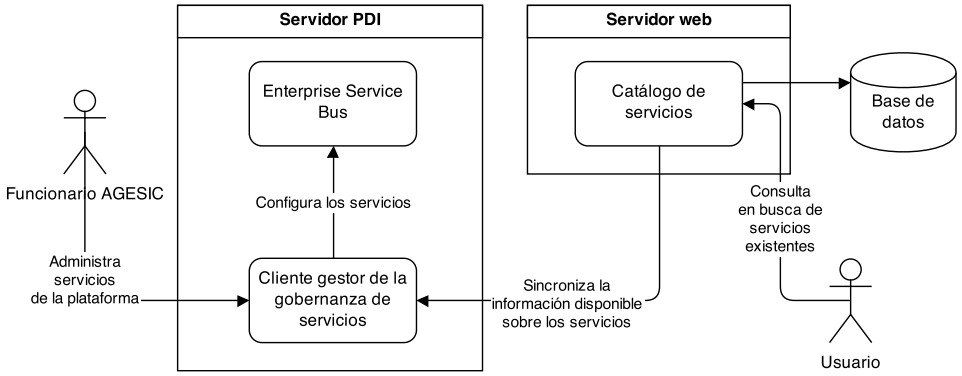

Figura 5.1: Diseño de implementación del prototipo de solución

5.2. Alternativas de implementación

Para la elección de una aplicación que permita administrar servicios aplicando los procesos de gobernanza en SOA, se debió realizar un análisis sobre las principales alternativas disponibles en la industria actual.

La selección de aplicaciones a analizar se basó en el estudio presentado en el artículo *Magic Quadrant for Application Services Governance* [40], en donde se evalúan varias aplicaciones enfocadas en la gobernanza de servicios, dando como resultado el cuadro comparativo de la Figura 5.2.

Figura 5.2: *Magic Quadrant for Application Services Governance* [40]

Para el análisis, se tomaron dos herramientas visionarias y dos líderes del mercado: *WSO2* y *MuleSoft* por un lado, y *SOA Software* e *IBM WebSphere Service Registry and Repository* por el otro. Se incorporó además a *Petals Master*, la cual no está incluida en el artículo mencionado, y trata de una herramienta de código libre orientada al registro de servicios, similar al objetivo de las demás.

Los criterios aplicados en la valoración de cada aplicación tienen en cuenta las funcionalidades ofrecidas por estas. Las mismas son agrupadas en:

1. Registro y repositorio
    - *a*) Posibilidad de acceder al registro de servicios de forma programática (API para el acceso al registro).
    - *b*) Mecanismos de dependencia entre servicios.
    - *c*) Posibilidad de añadir información adicional (meta-datos) a los servicios registrados.
    - *d*) Mecanismos para estructurar y organizar el conjunto de servicios.
    - *e*) Mecanismos ágiles para la búsqueda dentro del repositorio de servicios.
2. Gestión del ciclo de vida de los servicios
    - *a*) Gestión el ciclo básico de servicios: análisis/diseño, desarrollo, pruebas y explotación.
    - *b*) Administración de múltiples versiones de un mismo servicio.
    - *c*) Integración de un flujo de trabajo que permita la promoción de los servicios (que permita la aprobación por parte de un arquitecto SOA del paso de un servicio de diseño a desarrollo, de desarrollo a pruebas, etc.)
    - *d*) Integración de herramientas para la ejecución de pruebas sobre servicios.
3. Administración de políticas, métricas, monitorización y configuración
    - *a*) Implementación de estándares de políticas relacionadas con *Web Services* (por ejemplo, el cumplimiento del *Basic Profile 1.1*).
    - *b*) Posibilidad de añadir o configurar nuevas políticas.
    - *c*) Presencia de indicadores o métricas para monitorizar los servicios en ejecución (registros, auditoría, tiempos de respuesta, etc.).
    - *d*) Presencia de una interfaz gráfica que muestre dichas métricas de forma amigable.
    - *e*) Notificaciones sobre ocurrencia de eventos mediante el envío de mensajes (email, RSS, SMS, etc.).

4. Otros criterios

   a) Disponibilidad de edición Community/Enterprise

   b) Tipo de licencia

   c) Coste de edición Community/Enterprise

   d) Calidad de la documentación

   e) Soporte multi-idioma

   f) Disponibilidad del código fuente

   g) Comunidad activa

   h) Coste del soporte

   i) Disponibilidad de empresas que brinden soporte para la herramienta

Las Tablas 5.1, 5.2 y 5.3, son el resultado de aplicar los criterios de evaluación anteriores sobre cada una de las herramientas seleccionadas. Para las tres tablas rige la nomenclatura siguiente: el ícono ✓ indica que la herramienta cumple con la funcionalidad requerida; el ícono ✓ indica que cumple la funcionalidad en forma parcial o similar; el ícono ? indica que no se tiene información suficiente para determinar si cumple con la funcionalidad o no; finalmente, el ícono ⊗ indica que la herramienta no cumple con la funcionalidad.

Para finalizar el análisis de alternativas, se incluye la Tabla 5.4 en donde se realizan comparaciones adicionales entre las herramientas.

Funcionalidad	**SOA Soft**	**IBM WSRR**	**PM**	**Mule Soft**	**WSO2 G-Reg**
API para el acceso	✓	✓	✓	✓	✓
Gestión de meta-datos	?	✓	✓	✓	✓
Gestión de dependencias entre servicios	?	✓	⊗	⊗	✓
Estructuración y organización del repositorio	?	✓	✓	⊗	✓
Búsqueda ágil de servicios	✓	✓	✓	⊗	✓

Cuadro 5.1: Funcionalidades de registro y repositorio

Funcionalidad	SOA Soft	IBM WSRR	PM	Mule Soft	WSO2 G-Reg
Gestión del ciclo de vida	✓	✓	✓	✓	✓
Gestión de versiones	✓	✓	⊗	✓	✓
Promoción del servicio	✓	✓	⊗	✓	✓

Cuadro 5.2: Funciones de gestión de ciclo de vida de servicios

Funcionalidad	SOA Soft	IBM WSRR	PM	Mule Soft	WSO2 G-Reg
Implementación de políticas estándares	✓	✓	⊗	✓	✓
Configuración de nuevas políticas	?	✓	⊗	✓	✓
Monitoreo de ejecución	?	✓	⊗	✓	✓
Interfaz gráfica de monitorización	✓	✓	⊗	⊗	✓
Notificación de eventos en ejecución	?	✓	⊗	⊗	✓

Cuadro 5.3: Administración de políticas, métricas, monitorización y configuración

Característica	**SOA Soft**	**IBM WSRR**	**PM**	**Mule Soft**	**WSO2 G-Reg**
Disponibilidad de edición Community/Enterprise	No Libre	Versión Demo.	Libre	Libre Sin Código	Libre
Tipo de licencia	Sin Datos	V8.0.0.2	V1.1	V3.4	V4.6.0
Coste edición Community/Enterprise	Sin Datos	62900 dols. (100 PVU)	Sin Costo	Sin Costo	Sin Costo
Documentación	Muy buena	Muy buena	Buena	Buena	Muy buena
Soporte multi-idioma	Sin Datos	Inglés Español	Inglés Francés	Inglés	Inglés
Disponibilidad del código fuente	No	No	Sí	No	Sí
Comunidad activa	Sí	Sí	Sí	Sí	Sí
Coste del soporte	Sin Datos	Incluida primer año	Sin Costo	Sin Datos	Sin Datos

Cuadro 5.4: Cuadro comparativo adicional entre herramientas

5.3. Decisiones de implementación

Tras el análisis de las herramientas seleccionadas, se concluyó que la herramienta de software libre más completa para llevar a cabo un proceso de gobernanza de servicios en la actualidad es *WSO2*; en particular, una combinación de *WSO2 Enterprise Service Bus (ESB)* y *WSO2 Governance Registry*.

En primer lugar, la arquitectura de software de la plataforma SOA implementada por WSO2 se inspira en la especificación *OSGi (Open Services Gateway Initiative)*. Se trata de una plataforma completamente modular, extensible y, por tanto, personalizable, que proporciona un gran número de componentes combinables para cubrir varias necesidades de una SOA actual. Esta característica representa una gran ventaja, ya que todos sus componentes siguen una misma arquitectura con el fin de lograr las combinaciones mencionadas. Otras plataformas requieren de la integración de productos de diferentes fabricantes para cubrir sus necesidades, los cuales son potencial-

mente concebidos con arquitecturas sustancialmente distintas, limitando la interoperabilidad entre las herramientas utilizadas, por lo que presentan un aspecto más heterogéneo, su crecimiento es más irregular y se hace necesario un mayor conocimiento sobre cada producto distinto.

WSO2 ESB es un *Enterprise Service Bus*, disponible bajo licencia *Apache v2.0*. Está basado en el ESB *Apache Synapse* de la *Apache Software Foundation*, añadiéndole una mejor gestión y soporte de configuración para la gobernanza de una SOA. *WSO2 Governance Registry* se integra de forma sencilla con *WSO2 ESB* para generar una solución completa para la validación de la solución propuesta: una plataforma para alojar puntos de acceso a servicios, y una herramienta para la gestión de dichos servicios, aplicando un proceso de gobernanza en SOA.

5.3.1. Implementación del catálogo

Si bien *WSO2 Governance Registry* es un registro de los servicios disponibles en *WSO2 ESB*, se trata de una aplicación fundamentalmente orientada a operadores y administradores de la SOA, más que a potenciales consumidores de servicios. Esto se ve en limitaciones como la búsqueda, el acceso al listado de servicios sin necesidad de autenticación, entre otras cuestiones relacionadas con la usabilidad y no cumplimiento de las características expresadas como "deseables" en una aplicación que funcione como catálogo.

Esta situación trajo consigo la necesidad de implementar una aplicación a medida con este fin —a la vez que se continúa utilizando una herramienta genérica para la gestión de la PDI y sus procesos de gobernanza—, haciendo obligatoria la investigación para lograr la comunicación entre las distintas herramientas. Esta arrojó que tanto *WSO2 Governance Registry* como *WSO2 ESB* disponen de puntos de acceso a *Web Services SOAP* que proveen información acerca de los servicios gestionados, obtenida directamente desde sus bases de datos.

Con el fin de centralizar el conjunto de métodos que devuelven los datos que requiere el catálogo, se implementó un módulo que publica un nuevo *Web Service SOAP* que contiene dichos métodos y es el encargado de derivar las consultas a los servicios correspondientes de *WSO2 ESB* y *WSO2 Governance Registry*, logrando la comunicación del catálogo con ambas aplicaciones a través de un único punto de acceso como se muestra en la Figura 5.3.

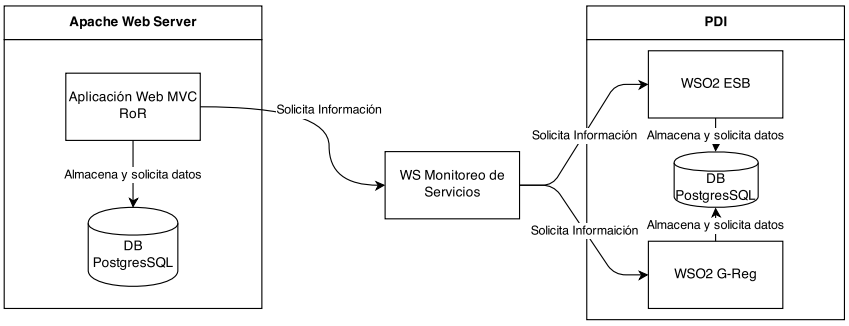

Figura 5.3: Diagrama de componentes del prototipo implementado

5.3.2. Implementación de métricas de calidad

WSO2 ESB incluye módulos de monitoreo sobre los *Proxies* de servicios, a partir de los cuales se obtienen los valores de las métricas del modelo de calidad definido. Sin embargo, no todas las métricas que componen al modelo están disponibles "de fábrica". Con el fin de obtener los valores requeridos, el módulo intermedio contiene lógica que agrega información sobre valores de métricas de calidad no incluidos en *WSO2 ESB*, basadas en la información ya provista por el ESB. Las Tablas 5.5, 5.6 y 5.7, contienen las métricas del modelo de calidad definido en el Capítulo 4, implementadas por las herramientas de *WSO2* en forma directa, junto con aquellas que fueron agregadas a través del módulo intermedio.

Métrica	**Método**
Average Response Time (Client)	Implementado por WSO2 ESB
Max Response Time (Client)	Implementado por WSO2 ESB
Min Response Time (Client)	Implementado por WSO2 ESB
Average Response Time (Provider)	Implementado por WSO2 ESB
Max Response Time (Provider)	Implementado por WSO2 ESB
Min Response Time (Provider)	Implementado por WSO2 ESB
Throughput (Client)	*Custom* Cantidad de solicitudes que recibe el ESB en un segundo
Throughput (Provider)	*Custom* Cantidad de solicitudes que procesa el punto de acceso del proveedor del servicio en un segundo
Availability	*Custom* (1- (Tiempo que estuvo no disponible/Tiempo total de monitoreo))*100
Accessibility	Implementado por WSO2 ESB
Successability	*Custom* (Cant. solicitudes enviadas al punto de acceso del proveedor del servicio /Total de solicitudes)*100

Cuadro 5.5: Métodos de implementación para métricas de performance y estabilidad

Métrica	**Método**
Confidentiality	Dato obtenido por WSO2 ESB
Integrity	Dato obtenido por WSO2 ESB
Authentication	Dato obtenido por WSO2 ESB
Access Control	Dato obtenido por WSO2 ESB
Non-Repudiation)	Dato obtenido por WSO2 ESB
Traceability	Dato obtenido por WSO2 ESB

Cuadro 5.6: Métodos de implementación para métricas de seguridad

Métrica	**Método**
Price	Dato obtenido por WSO2 G-Reg
Rating	Dato obtenido por WSO2 G-Reg
WS-I Basic Profile	Dato obtenido por WSO2 G-Reg

Cuadro 5.7: Métodos de implementación para métricas de usabilidad y costos e interoperabilidad

5.3.3. Plataforma de interoperabilidad

La plataforma de interoperabilidad del prototipo (o *Back-End*), está compuesta por las herramientas de *WSO2*: *WSO2 Governance Registry* y *WSO2 Enterprise Service Bus*.

Agregar un nuevo servicio a la plataforma requiere del seguimiento de una serie de pasos como los que se describen a continuación:

1. En la aplicación *WSO2 Governance Registry*, agregar el *WSDL* del servicio que se desea crear.

2. Acceder al servicio con su nombre y *namespace* extraídos del *WSDL* y versión *1.0.0-SNAPSHOT* por defecto. Ingresar el conjunto de metadatos necesario para inicializar la información del servicio.

3. Promover el estado del servicio hasta llegar al estado *Production*.

4. Acceder a la administración de *WSO2 ESB* para publicar un *Proxy* del servicio creado en el *WSO2 Governance Registry*.

5. En *WSO2 ESB* crear la entidad *Endpoint* en el ESB con la opción *Address Endpoint*.

6. Publicar el *Proxy* utilizando el template *Custom Proxy*.

7. Seleccionar el *Proxy* creado en *List* y habilitar el monitoreo de las métricas de calidad del servicio.

8. Verificar que el *Proxy* se haya publicado correctamente invocando alguna operación del servicio.

El procedimiento para agregar una versión de un servicio ya existente comparte los pasos con el de agregado de uno nuevo. Según la estrategia de versionado propuesta, será necesario evaluar si los cambios agregados en nuevas versiones corresponden a las categorías *compatible*, *incompatible manejable* o *incompatible*, para tomar las medidas necesarias en cuanto a la configuración de la versión. En los casos en que un cambio es incompatible manejable, será necesario aplicar una transformación *XSLT* para cumplir con la estrategia de compatibilización en la cual los consumidores de la versión inmediatamente anterior no se ven afectados por el cambio. El procedimiento para agregar una transformación *XSLT* es como sigue:

1. Agregar el/los código/s *XSLT* de la/s transformación/es del servicio en *Local Entries*

2. Establecer una instancia llamada *Sequences* y marcar *Enable Statistics* para monitorear métricas de calidad.

3. En *List*, seleccionar el *Proxy* al que se le desea aplicar la transformación y editarlo para agregar la transformación creada.

Estos procedimientos fueron definidos de esta forma para poder adaptar las funcionalidades de las herramientas con los requerimientos establecidos en la propuesta. Esto no significa que se trate de la única secuencia de pasos posible para trabajar con la herramienta, ya que existen varias alternativas para crear un *Proxy*, *Endpoint* o transformación *XSLT* en *WSO2 ESB*. Los procedimientos descritos fueron seleccionados por ser los que permiten monitorear las métricas y estadísticas.

Adicionalmente, *WSO2 Governance Registry* requiere de agregar los meta-datos *Organismo proveedor* y *Costos del servicio* para cumplir con los requerimientos de meta-datos para el descubrimiento.

El ciclo de vida es definido extendiendo el *XML* del servicio que define *WSO2 Governance Registry*, agregando los estados:

- CREADO (*CREATED*)
- PUBLICADO (*PUBLISHED*)
- RETIRADO (*RETIRED*)
- DESUSO (*DEPRECATED*)
- BLOQUEADO (*BLOCKED*)

Los pasos en cada procedimiento son una breve descripción de la lógica de uso a seguir. Con este informe se adjunta un anexo en donde se incluye un manual de usuario desarrollando cada paso con mayor detalle.

5.3.4. Catálogo de servicios (Front-End)

El *Front-End* de la arquitectura, es una aplicación que funciona como catálogo de los servicios alojados en la PDI. Se trata de una aplicación de acceso público –a nivel de la Internet global y/o REDuy– que funciona en forma sincronizada con las aplicaciones *WSO2 Governance Registry* y *WSO2 ESB*.

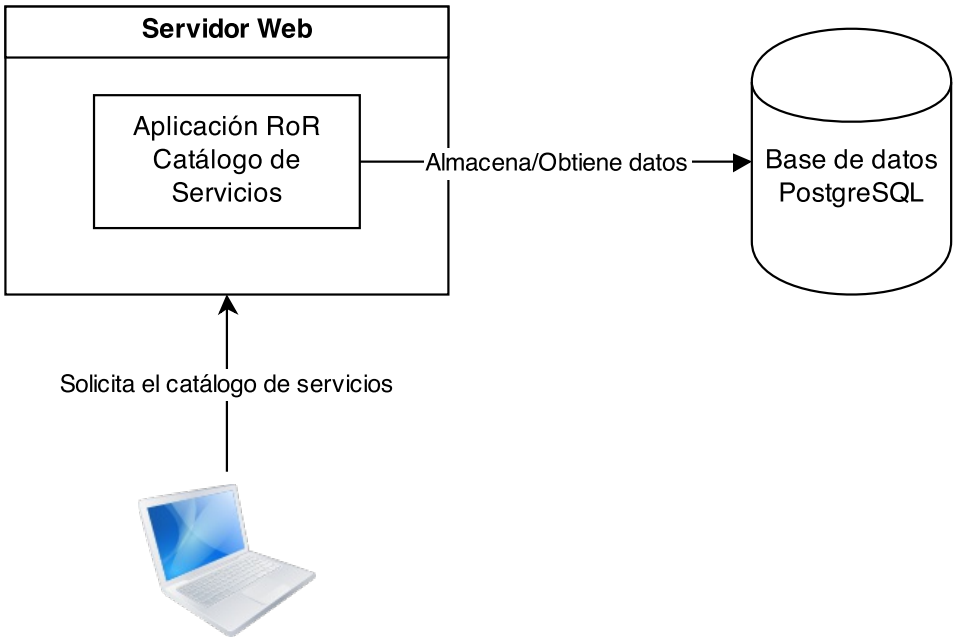

Figura 5.4: Arquitectura del catálogo de servicios

El diseño del sistema de la Figura 5.4 consiste en una aplicación *Web Ruby on Rails* que utiliza una base de datos relacional *PostgreSQL*.

La vista principal se trata de una cuadrícula cuyos espacios contienen datos básicos sobre cada servicio publicado (nombre, calificación y descripción breve). En la barra superior se ofrece la posibilidad de realizar búsquedas por palabras o etiquetas (palabras clave) asignadas a cada servicio a la vez que estos se encuentran asociados a un proveedor, por lo que se permite también visualizar todos los servicios administrados por un mismo organismo. Tanto desde la página principal como desde los resultados de búsqueda, el sitio permite obtener una vista rápida de los datos básicos de un servicio para que, a partir de allí, sea posible desplazarse a una página con más detalles (a través de un enlace), o regresar a la búsqueda realizada sin perder el contexto de los resultados obtenidos, gracias a la utilización de ventanas superpuestas (*modals*), las cuales pueden ser ocultadas para regresar a la búsqueda y continuar la visualización de otros servicios (ver Figura 5.5).

Una vez en la página de detalles de un servicio (ver Figura 5.6), los datos permanentes (nombre, introducción y descripción) se presentan en una columna junto con la calificación, las etiquetas asociadas, otras versiones publicadas, y fechas de publicación y actualización. Sobre la columna siguiente, se muestra una tabla dinámica que permite visualizar los valores de las medidas de calidad tomadas para el servicio, organizadas por categorías. Su organización responde al modelo de calidad aplicado lo cual es configurable a través de archivos *YAML* disponibles en la estructura de directorios de la

aplicación. Por último, el sitio ofrece un enlace para contactar con AGESIC para iniciar la suscripción a un servicio o solicitar más información[1].

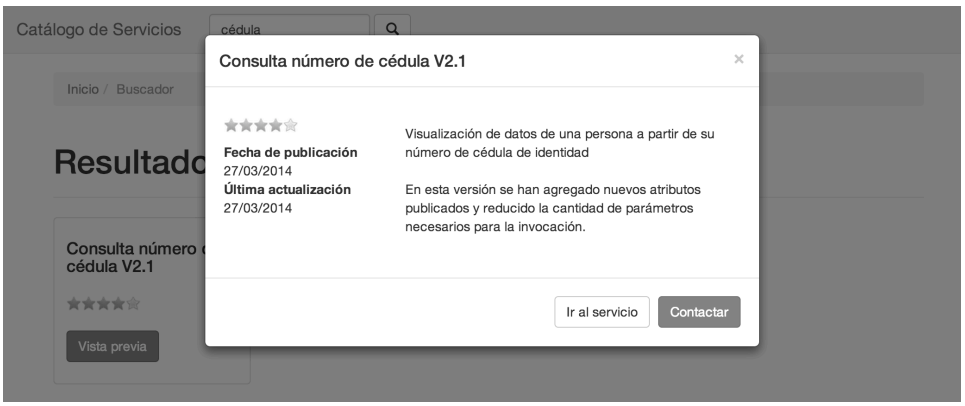

Figura 5.5: Ventanas flotantes sobre los resultados permiten visualizar los detalles mínimos de un servicio y continuar con la búsqueda en forma inmediata

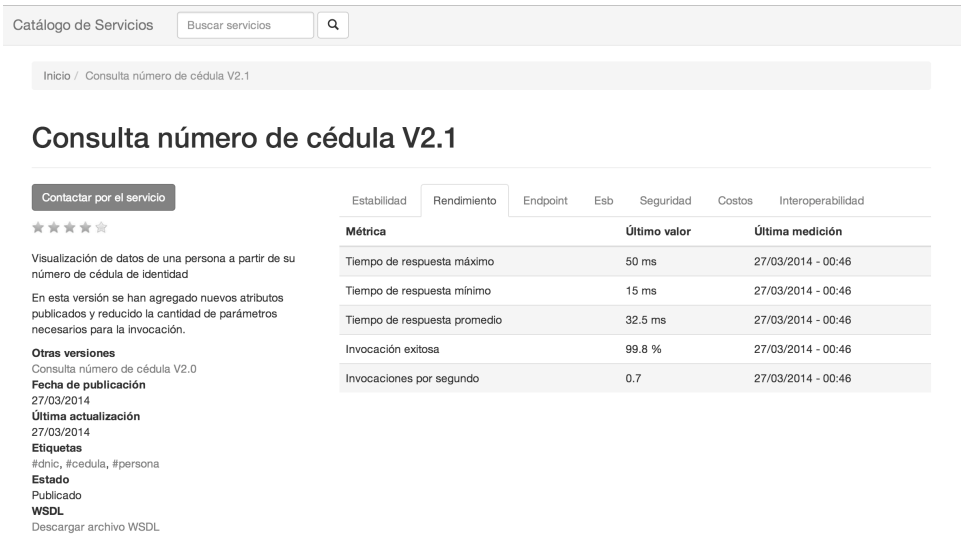

Figura 5.6: Página de detalles de un servicio

Los datos de los servicios publicados son obtenidos automáticamente desde las aplicaciones de *WSO2* que componen el prototipo de la PDI. La

[1] De existir el sistema de gestión de trámites de inscripción descrito en la Sección 4.1.3, es posible realizar una integración con el sistema de catálogo de modo que quien lo navega tenga la opción de ser redirigido al formulario correspondiente de inscripción o solicitud de cambios en una versión existente.

integración funciona a través de puntos de acceso a *Web Services SOAP* publicados por un módulo implementado a medida que actúa como intermediario entre la aplicación y las aplicaciones de *WSO2*, como se ve en la Figura 5.3. Periódicamente, la aplicación ejecuta un llamado a un servicio de listado, obteniendo así los servicios publicados para determinar los nuevos en la plataforma desde la última ejecución.

Una vez agregados sus datos básicos, formarán parte del conjunto de servicios catalogados, los cuales son incluidos en un proceso de actualización de meta-datos y medidas de calidad que ocurre también en forma periódica. La configuración de la aplicación permite establecer servicios y meta-datos "sensibles" que serán almacenados en la base de datos pero se mantendrán ocultos de la interfaz pública. La actualización se realiza mediante la ejecución de un comando de consola por lo que su periodicidad es configurable a través de llamadas del sistema operativo[2]. La sincronización incluye a los datos de calidad actualizados, obtenidos desde el módulo intermedio comunicado con *WSO2 ESB* y *WSO2 Governance Registry*.

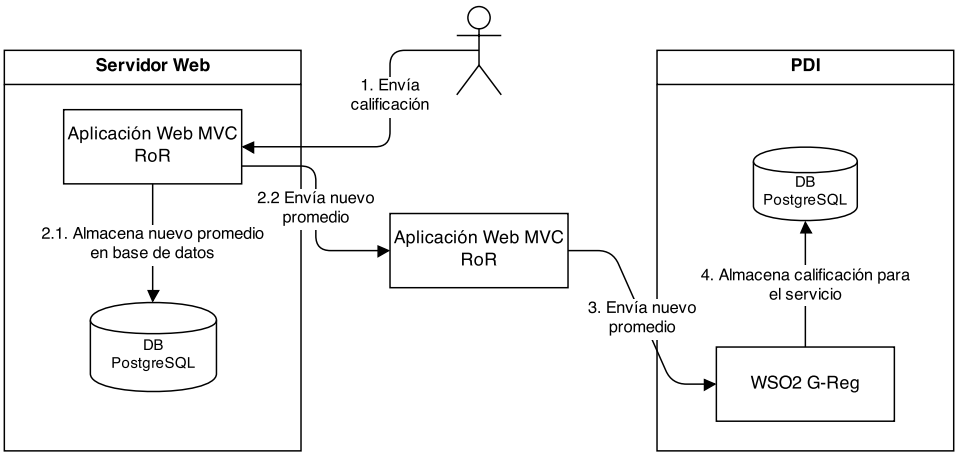

Figura 5.7: Página de detalles de un servicio

Si bien el flujo principal de datos ocurre desde la plataforma hacia la aplicación, la gestión de calificaciones de servicios es almacenada en la base de datos local, ya que los usuarios no tendrán acceso al conjunto de aplicaciones de la plataforma en forma directa para dar su opinión. Cada vez que un servicio es sincronizado, el promedio calculado será actualizado en *Governance Registry* para ser visualizado por el administrador, representado en la Figura 5.7.

Por su diseño modularizado, es posible adaptar cambios en los requerimientos de integración en forma ágil. En un caso extremo, ante la eventual necesidad de migrar la tecnología que da soporte a la PDI, la aplicación

[2]En un servidor Linux, será posible configurar la ejecución como un *cronjob*

puede ser adaptada a una nueva forma de comunicación, implementando los módulos correspondientes a la sincronización, sin la necesidad de recurrir a herramientas externas o la migración inminente de la aplicación junto con el resto de los componentes. La tecnología adoptada debe proveer alguna interfaz de interoperabilidad para el intercambio de datos (de forma similar a como lo resuelve *WSO2*), sea a través de *Web Services SOAP*, *REST* u otro estándar implementable en lenguaje de programación *Ruby*.

Esta aplicación, aunque a modo de prototipo, se diferencia del catálogo actual en varios aspectos sustanciales:

- El sistema de calificación es un concepto nuevo con respecto al catálogo integrado al portal actual de AGESIC, buscando acercarse a las necesidades de un catálogo expresadas en la Sección 4.1.3.
- Las búsquedas han sido mejoradas, ya que al tratarse de un sitio específicamente dedicado al catálogo, éstas se concentrarán en los servicios y no en todo el contenido del portal de la AGESIC como sucede en la actualidad.
- Las etiquetas permiten categorizar de nuevas formas a los servicios, agregando palabras clave que los identifiquen y hagan más fácil encontrar otros servicios relacionados.
- La presentación en columnas permite visualizar mayor cantidad de información sin necesidad de desplazar la página gracias al aprovechamiento del espacio horizontal.
- La integración con la plataforma permite automatizar el proceso de alta de servicios en el catálogo y la cantidad y calidad de la información presentada, en especial, las medidas del modelo de calidad, sincronizado de forma periódica.
- La vista de detalle permite visualizar las distintas versiones existentes de un mismo servicio.

La aplicación puede ser desplegada en distintos tipos de servidores que soportan al framework *Ruby on Rails* como *Apache*[3] (con *Phusion Passenger*[4]), *Unicorn*[5], entre otros. En cuanto a la tecnología utilizada del lado del cliente, el sitio requiere de *JavaScript* para el comportamiento dinámico de ventanas superpuestas y otras acciones que implican la comunicación con el servidor, por medio de solicitudes de tipo *AJAX*. Los navegadores que ingresen a este sitio deben ser capaces de interpretar *HTML5* y *CSS3*, utilizados para la diagramación y presentación de las páginas.

[3] `http://www.apache.org/`
[4] `https://www.phusionpassenger.com/`
[5] `http://unicorn.bogomips.org/`

El resultado es un sistema que implementa varias de las características deseadas en un catálogo, mostrando las mejoras con respecto al existente, junto con su capacidad de integración con el resto de las herramientas utilizadas en el prototipo.

5.4. Dificultades y limitaciones encontradas

Sobre el desarrollo del prototipo se presentaron algunos problemas y dificultades que se describen a continuación:

Curva de aprendizaje pronunciada: Se presentaron varias dificultades, principalmente derivadas de la inexperiencia con *WSO2*, las cuales, combinadas con la escasa documentación, hicieron del uso de la aplicación una dificil tarea en los comienzos.

Versionado de servicios poco intuitivo: El manejo de versiones presenta desafíos a la hora de comprender lo que se está llevando a cabo, debido a que la interfaz resulta poco intuitiva.

Publicación de *Proxy* de servicio confusa: La publicación de un *Proxy* requiere de alternar entre configuraciones en *WSO2 Governance Registry* y *WSO2 ESB* ya que ambos no proveen del acceso a todas las configuraciones necesarias.

Integración con bases de datos *MySQL* con dificultades: La experiencia mostró que *WSO2* presenta dificultades al integrarse con un motor de bases de datos *MySQL*. La solución encontrada fue utilizar un motor *PostgreSQL*.

Implementación de API con errores Se encontraron problemas con algunos *Web Services* de la API que no tenían definido su atributo *soapAction* en el *WSDL* correspondiente, dando un error al momento de invocar alguna operación. Este problema fue solucionado agregando el atributo en forma manual en aquellos *WSDL* que así lo requerían.

Configuración de API poco intuitiva: No existe documentación de fácil acceso acerca de cómo habilitar la API para extraer los datos requeridos desde *WSO2 ESB*. El descubrimiento de esta funcionalidad requirió de aplicar métodos de prueba y error hasta lograrlo.

Documentación escasa: En general, la documentación oficial pública es escasa, y los creadores referencian a sitios públicos[6] como fuente principal de información para la comunidad.

[6]Stack Overflow (`http://www.stackoverflow.com/`)

5.5. Caso de estudio

El caso de estudio está basado en la publicación de un servicio. El *Ministerio de Trabajo y Seguridad Social (MTSS)* desea publicar en producción un *Web Service* para luego hacerlo disponible en la PDI. [41].

El servicio consta de seis métodos: el primer y segundo método, informan si un empleado se encuentra o no registrado en la planilla de trabajo presentada ante el ministerio, realizando la búsqueda por el número de MTSS (además del número de cédula) o por RUT, respectivamente. El tercer y cuarto método, retornan la lista de empleados que se encuentran registrados en la planilla de trabajo presentada ante el ministerio, realizando la búsqueda por el número de MTSS o por RUT, respectivamente. El quinto método, retorna la información del cabezal de la planilla de trabajo presentada ante el ministerio. El cabezal de la planilla contiene toda la información de interés de una empresa [42]. En las Tablas 5.8, 5.9, 5.10, 5.11 y 5.12 se describen los parámetros de cada una de las operaciones disponibles en el servicio, cuyos nombres son *getEmpleadoPorNumMTSS*, *getEmpleadoPorRUT*, *getEmpleadosPorNumMTSS*, *getEmpleadosPorRUT* y *getCabezalPlanilla* respectivamente.

Parámetro	**Tipo de dato**	**Descripción**	**¿Obligatorio?**
mtss	Entero Largo	Número de empresa registrado en el MTSS	Sí
ordinal	Entero Largo	Número de planilla de trabajo (ej.: 1: casa central, 2: sucursal)	Sí
cedula	Texto	Número de documento de identidad. El formato esperado es XXXXXXX-X	Sí

Cuadro 5.8: Descripción de operación *getEmpleadoPorNumMTSS*

Parámetro	**Tipo de dato**	**Descripción**	**¿Obligatorio?**
rut	Entero Largo	Número de RUT	Sí
cedula	Texto	Número de documento de identidad. El formato esperado es XXXXXXX-X	Sí

Cuadro 5.9: Descripción de operación *getEmpleadoPorRUT*

Parámetro	Tipo de dato	Descripción	¿Obligatorio?
mtss	Entero Largo	Número de empresa registrado en el MTSS	Sí
ordinal	Entero Largo	Número de planilla de trabajo (ej 1: casa central, 2:sucursal)	Sí

Cuadro 5.10: Descripción de operación *getEmpleadosPorNumMTSS*

Parámetro	Tipo de dato	Descripción	¿Obligatorio?
rut	Entero Largo	Número de RUT	Sí

Cuadro 5.11: Descripción de operación *getEmpleadosPorRUT*

Parámetro	Tipo de dato	Descripción	¿Obligatorio?
mtss	Entero Largo	Número de empresa registrado en el MTSS	Sí
ordinal	Entero Largo	Número de planilla de trabajo (ej 1: casa central, 2:sucursal)	Sí

Cuadro 5.12: Descripción de operación *getCabezalPlanilla*

5.5.1. Publicación de servicio

El servicio es publicado siguiendo el procedimiento descrito para el prototipo, simulando su comportamiento a través de la funcionalidad *Mock Service* de la herramienta *SoapUI*, la cual permite crear un punto de acceso a un *Web Service* en un puerto local. En la Figura 5.8 se muestra el estado de la configuración de la PDI con *WSO2 Governance Registry* y *WSO2 ESB* en el prototipo.

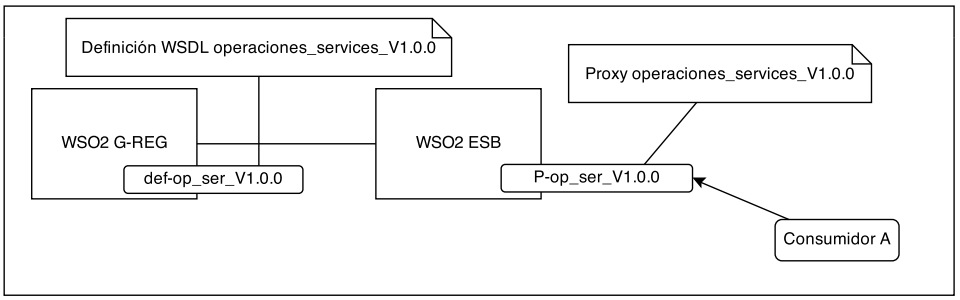

Figura 5.8: Estado de la PDI prototipada al publicar un servicio

5.5.2. Cambio en la versión del servicio

Cambio compatible

Suponiendo que transcurre un tiempo suficiente durante el cual el *Proxy* del servicio publicado se encuentra en producción, se presenta un escenario en el cual existe una cantidad de consumidores del servicio, utilizándolo para sistemas propios. Bajo esta situación, el ministerio considera agregar una operación al servicio, tal como se muestra en el Cuadro 5.13.

getEmpleadosPorEdad retorna la lista de empleados de una empresa según un rango de edad ingresado. Esta operación estará disponible en una nueva versión del WSDL ofrecida desde los servidores del ministerio, y por tratarse de un cambio compatible, —siguiendo la estrategia de versionado propuesta— las solicitudes serán redirigidas de tal forma que los consumidores futuros utilizarán la nueva versión del WSDL, mientras que los consumidores de la versión anterior podrán seguir utilizando la versión que ya tienen del mismo WSDL, en forma transparente, tanto para los usuarios como para la infraestructura que soporta al servicio en el ministerio.

Parámetro	**Tipo de dato**	**Descripción**	**¿Obligatorio?**
rut	Entero Largo	Número de RUT	Sí
rangoEdad	Entero Largo	Número del rango de edad según el Instituto Nacional de Estadísticas (INE) (ej 1: 15-19 años, 2: 20-24 años, 3: 25-29 años, .. , 11: 65-69 años)	Sí

Cuadro 5.13: Nueva operación agregada al servicio: *getEmpleadosPorEdad*

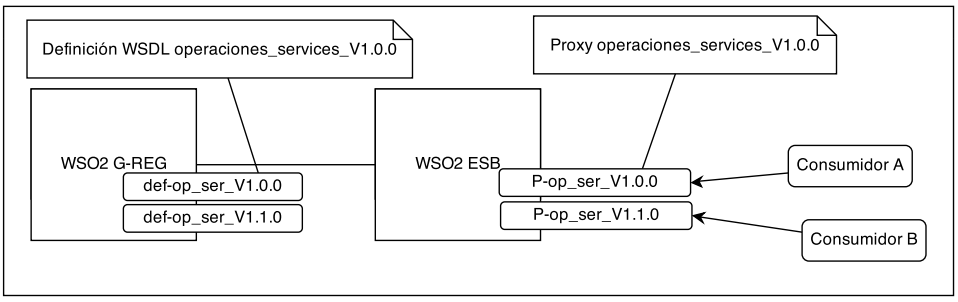

Figura 5.9: Estado de la PDI prototipada al producirse un cambio compatible en el servicio

La nueva versión del WSDL debe ser publicada en *WSO2 Governance Registry* con versión _V1.1.0, dado que se trata de un tipo de cambio compatible. Luego se publica el *Proxy* en *WSO2 ESB* con versión _V1.1.0. Como

forma de notificar a los clientes actuales utilizando la versión _V1.0.0, se cambia el estado del servicio, pasando de *Publicado* a *En desuso*. De esta forma, el cliente sabrá que en algún momento la versión que utiliza del servicio puede dejar de estar disponible, ya que existe una más actualizada. El estado de la PDI con las dos versiones del servicio coexistiendo se muestra en la Figura 5.9.

Cambio incompatible manejable

Frente a un nuevo escenario en el cual existen varios consumidores de las versiones _V1.0.0 y/o _V1.1.0 del servicio, el ministerio decide combinar las operaciones *getEmpleadosPorRut* y *getEmpleadosPorEdad* en una sola, ya que tienen una lógica similar.

En la nueva versión, cuando el parámetro *rangoEdad* está vacío, la operación se comporta como *getEmpleadosPorRut* en la versión anterior, mientras que cuando está presente, se aplica el filtro de rango de edad al resultado de dicha operación. Se crea entonces una operación *getEmpleadosPorRut_RangoEdad* como se describe en la Tabla 5.14, la cual recibe un parámetro *rut* correspondiente al número de RUT y otro *rangoEdad* correspondiente al rango de edad.

Parámetro	**Tipo de dato**	**Descripción**	**¿Obligatorio?**
rut	Entero Largo	Número de RUT	Sí
rangoEdad	Entero Largo	Número del rango de edad según el Instituto Nacional de Estadísticas (INE) (ej null: Todas las edades, 1: 15-19 años, 2: 20-24 años, 3: 25-29 años, .. , 11: 65-69 años)	No

Cuadro 5.14: Nueva operación getEmpleadosPorRut_RangoEdad

La nueva versión del WSDL debe ser publicada en *WSO2 Governance Registry* con versión _V1.2.0, dado que se trata de un tipo de cambio incompatible manejable. Este tipo de cambio requiere además de la aplicación de transformaciones *XSLT* a las solicitudes correspondientes a las versiones con las cuales es incompatible manejable.

Para las solicitudes dirigidas a la versión _V1.0.0, se debe configurar una transformación en *WSO2 ESB* tal que las invocaciones a *getEmpleadosPorRut* sean redirigidas a *getEmpleadosPorRut_RangoEdad* con el parámetro *rangoEdad* vacío, utilizando una plantilla *XSLT* como la que se muestra en la Figura 5.10. Para el caso de solicitudes dirigidas a la versión _V1.1.0 no es necesario realizar transformaciones, sino redirigir las solicitudes, ya que la operación mantiene el formato de parámetros de invocación. La versión

_V1.1.0 pasa al estado *En desuso* y el estado final de la PDI con las tres versiones del servicio se muestra en la Figura 5.11

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:web="http://webservices.venetus.mtss.gub.uy/" version="1.0">
      <xsl:template match="node()|@*">
         <xsl:copy>
            <xsl:apply-templates select="node()|@*"></xsl:apply-templates>
         </xsl:copy>
      </xsl:template>
      <xsl:template match="web:getEmpleadosPorRut">
         <web:getEmpleadosPorRut_RangoEdad xmlns="http://ws.apache.org/ns/synapse">
            <xsl:apply-templates select="@*|node()"></xsl:apply-templates>
         </web:getEmpleadosPorRut_RangoEdad>
      </xsl:template>
   </xsl:stylesheet>
```

Figura 5.10: Transformaciones *XSLT* a solicitudes dirigidas a la versión _V1.0.0

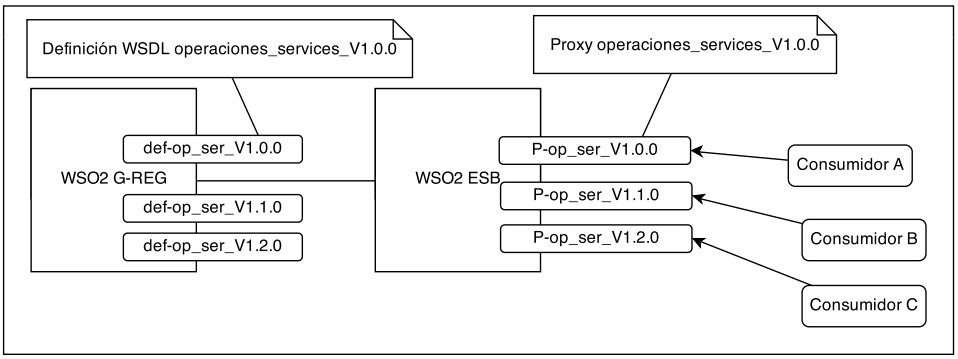

Figura 5.11: Estado de la PDI prototipada al producirse un cambio incompatible manejable en el servicio

Cambio incompatible no manejable

Por último, ya teniendo en producción una versión del servicio con la operación *getEmpleadosPorRut_ RangoEdad* soportando la funcionalidad de *getEmpleadosPorRut* y *getEmpleadosPorEdad*, el ministerio decide eliminar ambas operaciones en su próxima versión del WSDL. Un cambio de este tipo se considera incompatible manejable, por lo que el WSDL se publicará con versión 2.0.0, y los consumidores que actualmente utilicen las versiones 1.0.0, 1.1.0 y 1.2.0 deberán actualizar sus sistemas para migrar hacia la nueva versión en caso de que quieran seguir utilizando el servicio. Las versiones anteriores pasarán su estado a *Retirado* a la vez que la nueva versión será marcada como *Publicado* en cuanto esté disponible.

5.5.3. Conclusiones y pruebas realizadas

Luego de la publicación del servicio, ante la necesidad de agregar versiones con cambios sustanciales al WSDL, la estrategia de versionado de la propuesta pudo aplicarse sobre el prototipo de prueba elaborado. En todos

los casos se utilizaron servicios de prueba, cuyos pasos de publicación se desarrollan en el Apéndice D.

Se adjunta a este informe un documento anexo en el cual se presenta una serie de pruebas realizadas sobre un conjunto de servicios del catálogo actual de la AGESIC, simulados con la funcionalidad *Mock Service* de *SoapUI*.

Capítulo 6

Conclusiones y trabajo a futuro

A medida que la cantidad de servicios en una SOA crece, se hace necesario estandarizar y controlar a los procesos involucrados, estableciendo guías de procedimientos que ayuden a alcanzar los objetivos de la SOA en forma óptima. El conjunto de procesos administrativos que gobiernan a los servicios de una organización es lo que llamamos Gobernanza en SOA.

La preferencia de las organizaciones por las SOA para alojar a sus servicios, ha hecho de la gobernanza de éstas arquitecturas un área de interés en crecimiento en los últimos años. Como se ha visto, los procesos abarcan todo el ciclo de vida de un servicio, desde su concepción hasta su retiro, incluyendo los relacionados al control de calidad (QA), el versionado y el registro (catálogo). El caso particular de AGESIC presenta desafíos en cuanto a la jurisdicción sobre los servicios, ya que no es el autor ni administrador directo de estos, por lo que la gobernanza se debe limitar a las áreas en donde tiene posibilidad de ejercer cambios. En este proyecto se presenta una propuesta de gobernanza adaptada a estas circunstancias, enfocándose en las etapas que están bajo la órbita de la agencia. Se toma como punto de partida a la estructura general propuesta en el marco conceptual, a conciencia de la existencia de otras similares por diferentes autores, variando en las responsabilidades y estructuras de los procesos.

El análisis parte de los requerimientos planteados por el cliente: una propuesta de gobernanza en SOA, el establecimiento de un modelo de calidad, la definición de una estrategia de versionado y la mejora del catálogo de servicios existente. Luego de sucesivas reuniones que permitieron el refinamiento, el relevamiento permitió visualizar con mayor detalle las implicancias de estos requerimientos, expandiéndose hacia cuestiones como la publicación y consumo de los servicios, la ejecución de tareas de mantenimiento periódicas, entre otras.

Con el análisis de la realidad completado, se estudian los requerimientos para consolidar las partes que compondrán la propuesta. En lo referente a modelos de calidad, se estudian tres modelos alternativos y sus atributos como punto de partida para elaborar una propuesta. En estrategias de versionado se exponen estrategias conocidas y se comparan con las necesidades

de AGESIC para aplicarlas a sus servicios. Para el catálogo se analizan las características deseables en base al relevamiento. Estos requerimientos terminan de cerrar la definición de los objetivos del proyecto, dando lugar al comienzo de la investigación para la elaboración de la solución.

La propuesta consta de una componente conceptual en la que se introducen la distribución y los procesos de gobernanza que aplican a la jurisdicción de AGESIC, así como también el modelo de calidad que se debe aplicar sobre los servicios y la estrategia de versionado recomendada para los mismos. La otra trata de un prototipo de pieza de software que muestra las características deseables para un catálogo en funcionamiento, la cual es presentada como parte del prototipo de software en el Capítulo 5.

Para la elaboración de la propuesta de gobernanza, se recurrió a la búsqueda de experiencias en otras plataformas, la bibliografía existente y la experiencia actual de AGESIC, que resulta fundamental para la introducción de ideas que se acerquen a lo aplicable. Dada la naturaleza de cómo surge la PDI, no es fácil encontrar ejemplos de plataformas similares. Desde un principio se comprendió que la propuesta de un proyecto de gobernanza en SOA que abarque todas las etapas del ciclo de vida de un servicio, no estaba al alcance de lo aplicable por la agencia, por lo que se buscó acotar el alcance del proyecto a la jurisdicción actual, limitada a la PDI. En ella se busca atender a los aspectos administrativos y de funcionamiento más notorios, como la definición de los roles que cumplen los actores con respecto a la gestión de servicios, la modernización del proceso de publicación, la disponibilización de información para los organismos usuarios, y en forma general, la definición de un ciclo de vida completo de un servicio desde el punto de vista de la agencia, sentando las bases que conforman la mejora de procesos en cada etapa por separado.

El estudio de los modelos de calidad propuestos por organizaciones influyentes en el área, brindó los recursos para la elaboración de un modelo personalizado adecuado a las necesidades del cliente, seleccionando los atributos que son del interés de la agencia, sobre todo para lo que refiere a la generación de SLA y la publicación de meta-datos de servicios en el catálogo.

La estrategia de versionado introduce las definiciones y procedimientos a llevarse a cabo ante cambios en los servicios, aumentando la robustez y por tanto la confianza de parte de los consumidores, a la vez que se reduce la complejidad de la configuración de múltiples versiones de un mismo servicio para múltiples clientes.

Como forma de validar el cumplimiento de los objetivos, se presenta una implementación en la que se estudian herramientas integradas para la gobernanza y sus posibilidades de comunicación con una aplicación de registro de servicios desarrollada como parte del prototipo. Entre las herramientas de gobernanza, se hace un análisis comparativo con paquetes líderes en el mercado (*IBM* y *SOA Software*), y por otro lado, otras dos con gran potencial pero menos promocionadas (MuleSoft y WSO2). En las pruebas, WSO2

resultó ser el más conveniente, no sólo por ser de código abierto y con licencia gratuita, sino también por sus características técnicas que lo hacían una alternativa viable. En esta herramienta se integraron los conceptos del ciclo de vida propuestos, el manejo de versiones de la estrategia y sobre todo, su capacidad de apertura de la información a través de Web Services para comunicarse con otras aplicaciones, lo que permitió desarrollar la aplicación de catálogo con las características deseadas, consumiendo información directamente desde la herramienta de gobernanza. Si bien esto permitió validar algunos de los objetivos, no es posible validar toda la propuesta ya que parte de ella se trata de procesos teóricos cuya efectividad sólo puede ser estudiada mediante su implantación en un ambiente similar al de producción; para justificar su validez, se debe recurrir a los textos de base utilizados. El prototipo resultado puede ser utilizado como muestra de la efectividad del catálogo y de cómo el ciclo de vida propuesto se adapta a las necesidades de AGESIC, cumpliendo así con los objetivos planteados al inicio del proyecto.

6.1. Trabajo a futuro

Si bien los objetivos del proyecto se consideran alcanzados, es posible identificar aspectos a desarrollar a partir de las conclusiones alcanzadas:

Sincronización en tiempo real de datos de servicios En el prototipo elaborado para la validación, la obtención de los datos de los servicios desde la aplicación de catálogo se realiza en forma periódica por lo que los datos presentados no reflejan el estado en forma inmediata. Una sincronización en forma de suscripción de la aplicación de catálogo a la herramienta de gestión de servicios permitiría a la primera detectar cambios en los datos de la segunda y mostrarlos en tiempo real.

Roles de usuario en la aplicación de registro Permitir la gestión del catálogo requiere de la introducción del concepto de usuarios administrativos con acceso a un área de back-end desde la cual configurar aspectos variables de la aplicación.

Suscripción a eventos del catálogo Ofrecer la posibilidad de que usuarios registrados en la aplicación de catálogo se suscriban a eventos de interés sobre sus servicios.

Dashboards con reportes estadísticos sobre los servicios Éstos permitirían mostrar la evolución y cumplimiento de acuerdos de nivel de servicio a lo largo del tiempo en forma gráfica y con datos reales en base a automatización.

Análisis de otros procesos de gobernanza en SOA Los distintos autores y sus propuestas de gobernanza en SOA brindarían una visión

más amplia sobre las alternativas aplicables a los procesos. Analizar las distintas aproximaciones es una fuente de refinamiento para la propuesta elaborada.

Apéndice A

Modelos de calidad

A.1. Modelo OASIS

Los factores de calidad se dividen en tres grandes grupos:

Business Value Quality Es un conjunto de factores de calidad que pueden definirse como determinantes para el valor del negocio del servicio cuando un usuario selecciona un *Web Service* específico.

Service Measurement Quality Es el nivel de la calidad en que el usuario percibe con la utilización del *Web Service*. La calidad general significa la rapidez en que ofrece el servicio y/o que tan estable es.

System Information Quality

Dentro del grupo *Business Value Quality* se encuentra los factores:

- Costo del Servicio (*Service Cost*)
  Es el factor de calidad que se refiere al costo a ser pagado por el uso del servicio, incluye precio del servicio, convenio, compensación y facturación, penalizaciones.

- Evaluación del servicio (*Service Suitability*)
  Es el factor de calidad que se refiere a qué tan apropiado es el objetivo del negocio o estrategia y servicios. Determina si un servicio en particular es apropiado para las operaciones comerciales de un consumidor. Incluye medidas como aplicación al negocio, o conveniencia al usuario.

- Uso del servicio (*Service After Effect*)
  Es una medida cualitativa o cuantitativa de valor de negocio que puede obtener con el uso del servicio. Expresa los efectos de un servicio web particular del negocio. Efectividad, satisfacción del usuario son algunas medidas

- Marca del servicio (*Service Brand Value*)
  Es la medida según el reconocimiento o grado de reputación dada por los usuarios.

Los factores que integran el grupo *Service Measurement Quality* son:

- Rendimiento (*Performance*)
  Factores de calidad tales como tasa de transferencia, tiempo de respuesta, accesibilidad, y disponibilidad son objetos de medición del servicio.
- Estabilidad (*Stability*)
  Indica que el servicio puede proveer un servicio en forma estable. (Disponibilidad, accesibilidad, exitosamente)

El último grupo *System Information Quality* se encuentra formado por los factores:

- Calidad de proceso de negocio (*Business Process Quality*)
  se encuentra definida por
  - Mensajería confiable (*Reliable Messaging*)
    Envío y recepción realizado en orden en el tiempo, sin solapamiento y asegurar la trasmisión. Se refiere a garantizar el intercambio de mensajes entre los servicios web y las de sus clientes.
  - Transacción (*Transaction*)
    Su objetivo es determinar la habilidad para mantener los datos consistentes en todos los *Web Services* involucrados, en caso de que el servicio consista de una combinación de varios *Web Services*. Se refiere a la capacidad de procesar las tareas relacionadas en un flujo de proceso como una única unidad lógica.
  - Colaboración (*Collaboration*)
    Determina la habilidad de *Web Services* relacionados en forma conjunta sobre un mismo proceso de negocio. La posibilidad de incluir la ejecución de servicios distribuidos dentro de un proceso de negocio.
- Adaptación a Estándares (*Suitability for Standards*)
  - Conformidad (*Saccordance*)
    Se evalúa si un servicio web fue construido de conformidad con las normas. Conformidad con estándares como Intercambio de mensajes, definición de servicios, búsqueda y registro.
  - Interoperabilidad (*Interoperability*)
    Mide el grado de adhesión, de un servicio web, con las mejores prácticas definidas por las normas. Indica cuáles mensajes son apropiadamente intercambiados y usados en una especificación. Interoperabilidad de servicio básica y de servicio seguro.
- Calidad en la seguridad (*Security Quality*)

- Confidencialidad (*Confidentiality*)
  Capacidad de prevenir qué usuarios no autorizados vean o accedan al servicio/mensajes. Usa control de acceso y encriptación.
- Integridad (*Integrity*)
  Capacidad de proteger servicios/mensajes contra modificación, eliminación o creación. Integridad basada en mensaje o basada en transporte.
- Autenticación (*Authentication*)
  Verificar que un objeto es confiable para la transmisión.
- Control de acceso (*Access Control*)
  Verifica el control sobre el acceso a servicio/mensajes para cada permiso del actor.
- No-repudio (*Non-Repudiation*)
  Asegurar que quien envía o recibe es quien dice ser. Capacidad que permite que los receptores y remitentes no pueden negar que han enviado o recibido mensajes.
- Disponibilidad (*Availability*)
  Permitir acceso a servicio/mensajes solo a personal autorizado y proteger el servicio contra intentos de acceso no autorizados.
- Trazabilidad (*Traceability*)
  Capacidad de registro de actividades.
- Privacidad (*Privacy*)
  Es la protección de la información entre el usuario del *Web Service* y el proveedor.
- Autorización distribuida (*Distributed Authorization*)
  Es una característica de seguridad, que permite el uso de varios *Web Services* en diferentes plataformas mediante un único acceso de autenticación.

▪ Calidad en la administración (*Manageability Quality*)

- Administración (*Manageability*)
  Es un factor de medida del grado de manejo a través de un sistema de administración por medio de atributos. Se refiere a la disponibilidad de información de *Web Service*, desde una perspectiva de gestión.
- Observable o introspección
  Indica que tan claro se puede observar el interior del servicio. Es La función de proporcionar la información sobre la gestión de los *Web Services* y su recurso relacionado. Puede ser evaluado por recursos, interfaces, y métricas de información medibles.

- Capacidad de control (*Controlability*)
  Facilidad para control del sistema y su contenido desde fuera del mismo. La característica que puede cambiar la información interna de los *Web Service* y sus recursos relacionados del exterior.

A.2. Modelo IBM

Para IBM los principales requisitos para el soporte de QoS en los *Web Services* son:

- Disponibilidad (*Controlability*)
  La disponibilidad es el aspecto de la calidad de si el *Web Service* está presente o listo para su uso inmediato. Disponibilidad representa la probabilidad de que un servicio está disponible. Los valores más altos representan que el servicio es siempre listo para su uso, mientras que valores menores indican incertidumbre de si el servicio estará disponible en un momento determinado. También asociado con la disponibilidad es el tiempo hasta la reparación (TTR). TTR representa el tiempo que se necesita para reparar un servicio que ha fallado. Lo ideal sería que los valores más pequeños de TTR son deseables.

- Accesibilidad
  La accesibilidad es el aspecto de la calidad de un servicio que representa el grado en que es capaz de servir a una solicitud de *Web Service*. Se puede expresar como una medida de probabilidad que indica la tasa de éxito o probabilidad de un éxito de instancias de servicio en un punto en el tiempo. Puede haber situaciones en las que un *Web Service* está disponible, pero no se puede acceder. Alta accesibilidad a los servicios Web se puede lograr mediante la creación de sistemas altamente escalables. Escalabilidad se refiere a la capacidad de servir constantemente las peticiones pesar de las variaciones en el volumen de solicitudes.

- Integridad
  La integridad es el aspecto de la calidad de cómo el servicio mantiene la exactitud de la interacción con respecto a la fuente. Ejecución correcta de las transacciones de *Web Service* proporcionará la corrección de interacción. Una transacción se refiere a una secuencia de actividades que se van a tratar como una sola unidad de trabajo. Todas las actividades tienen que ser completados para hacer la transacción con éxito. Cuando una transacción no se completa, todos los cambios realizados se deshacen.

- Rendimiento
  El rendimiento es el aspecto de la calidad del servicio Web, que se mide

en términos de rendimiento y latencia. Mayor rendimiento y valores de latencia más bajos representan el buen desempeño de un *Web Service*. Throughput representa el número de solicitudes de servicios web que se sirve en un periodo de tiempo determinado. Latencia es el tiempo de ida y vuelta entre el envío de la solicitud y la recepción de la respuesta.

- Fiabilidad
  La fiabilidad es el aspecto de la calidad de un *Web Service* que representa el grado de ser capaz de mantener el servicio y la calidad del servicio. El número de fallos por meses o años representa una medida de la fiabilidad de un *Web Service*. En otro sentido, la fiabilidad se refiere a la entrega asegurada y ordenada de los mensajes que son enviados y recibidos por los solicitantes de servicios y proveedores de servicios.

- Regulador
  Regulador es el aspecto de la calidad del *Web Service* de conformidad con las reglas, la ley, el cumplimiento de las normas y el acuerdo de nivel de servicio establecidos. Los servicios Web utilizan una gran cantidad de estándares como *SOAP, UDDI y WSDL*. La adhesión estricta a corregir versiones de las normas (por ejemplo, *SOAP* versión 1.2) por los proveedores de servicios es necesaria para la invocación adecuada de los servicios web de solicitantes del servicio.

- Seguridad
  La seguridad es el aspecto de la calidad del *Web Service* de proporcionar confidencialidad y no repudio mediante la autenticación de las partes involucradas, el cifrado de mensajes, y proporcionar control de acceso. La seguridad ha cobrado más importancia porque la invocación del servicio se produce a través de Internet. El proveedor de servicios puede tener diferentes enfoques y niveles de proporcionar seguridad en función del solicitante del servicio.

A.3. Modelo S-Cube

A continuación se detallan un conjunto de principales atributos de distintas categorías del modelo de calidad S-Cube.

Performance contiene atributos de calidad que caracterizan el nivel de desempeño de un servicio:

- Tiempo de respuesta (*Response Time*)
  La cantidad de tiempo necesario para completar una solicitud de servicio desde el punto de vista del usuario.

- Rendimiento (*Throughput*)
  Número de solicita completados que tienen un servicio en un período de tiempo.

- Tiempo de transacciones (*Transaction Time*)
  Tiempo que transcurre mientras el servicio está completando una transacción.

Dependability se refiere a la capacidad de un sistema informático para ofrecer un servicio que justificadamente se puede confiar:

- Escalabilidad (*Scalability*)
  Capacidad de incremento de la capacidad de cálculo del sistema informático del proveedor de servicios y la capacidad del sistema para procesar más operaciones o transacciones en un período determinado.

- Confiabilidad (*Reliability*)
  Capacidad de un servicio para realizar sus funciones requeridas bajo las condiciones establecidas por un período de tiempo especificado.

- Disponibilidad (*Availability*)
  Disponibilidad del servicio prestado a los clientes.

- Disponibilidad continua (*Continuous Availability*)
  Se evalúa la probabilidad con la que un cliente puede acceder a un servicio de un número infinito de veces durante un período de tiempo determinado.

- Semántica de fallos (*Failure Semantics*)
  Describen las capacidades generales de un servicio para controlar los errores.

- Flexibilidad (*Robustness/Flexibility*)
  Se refiere a la capacidad del servicio a comportarse de una manera aceptable en situaciones anómalas o inesperadas.

- Precisión (*Accuracy*)
  Define la tasa de error producida por el servicio calculado sobre la base del resultado esperado.

Security se ocupa de los aspectos de seguridad de servicios. Los atributos de calidad reconocidas en esta categoría son:

- Seguridad (*Safety*)
  La ausencia de consecuencias catastróficas en los usuarios y el entorno.

- Autorización (*Authorization*)
  El proceso de determinar, mediante la evaluación de la información de control de acceso aplicable, si se permite un sujeto a tener los aspectos especificados de acceso a un determinado recurso del servicio.

- Confidencialidad (*Confidentiality*)
  La ausencia de la divulgación no autorizada de información.

- Integridad (*Integrity*)
  Ausencia de alteraciones indebidas del estado del sistema, incluyendo la alternancia o eliminación de la información accidental o maliciosa.

- Responsabilidad (*Accountability*)
  El estado de ser responsable; riesgo de ser llamados a rendir cuentas; la obligación de asumir las consecuencias en caso de no funcionar como se espera.

- Trazabilidad y auditabilidad (*Traceability and Auditability*)
  Capacidad del servicio para monitorear y generar de una manera fiable y segura de los eventos que producen un registro de auditoría de la que una secuencia de eventos puede ser reconstruida y examinada.

- Encriptación de información (*Data encryption*)
  Se refiere a los algoritmos adoptados para proteger los datos de accesos maliciosos.

Usability recoge los atributos de calidad que se pueden medir subjetivamente según la opinión de los usuarios.

- Facilidad de aprendizaje (*Learnability*)
  Capacidad del servicio para que el usuario pueda aprender a aplicar / usarlo.

- Eficacia (*Effectiveness*)
  La precisión y la exhaustividad con la que los usuarios a lograr objetivos específicos.

- Eficiencia de uso (*Efficiency of use*)
  Recursos invertidos en relación con la exactitud y la exhaustividad con la que los usuarios a lograr sus metas.

- Accesibilidad contenido (*Content accessibility*)
  Asegurar que el contenido del servicio se puede navegar y leído por todos, independientemente de su ubicación, la experiencia, o el tipo de tecnología informática utilizada.

- Satisfacción (*Satisfaction*)
  La libertad de la desconformidad y las actitudes positivas hacia el uso del servicio.

Cost incluye atributos de calidad relacionados con el coste del servicio

- Modelo del costo (*Cost model*)
  Define un conjunto de funciones que transforman los recursos (servicios) en los costos. Los costos fijos

- Costos fijos (*Fixed costs*)
  Los costos que constituyen una cantidad fija de los costos generales para la prestación de un servicio.

- Costos variables (*Variable costs*)
  Los costos que cambian durante la prestación de un servicio, además de los costos fijos

Apéndice B

Sub procesos BPMN

El Diagrama B.1 representa los sub procesos BPMN que forman parte del proceso de ciclo de vida de gobernanza.

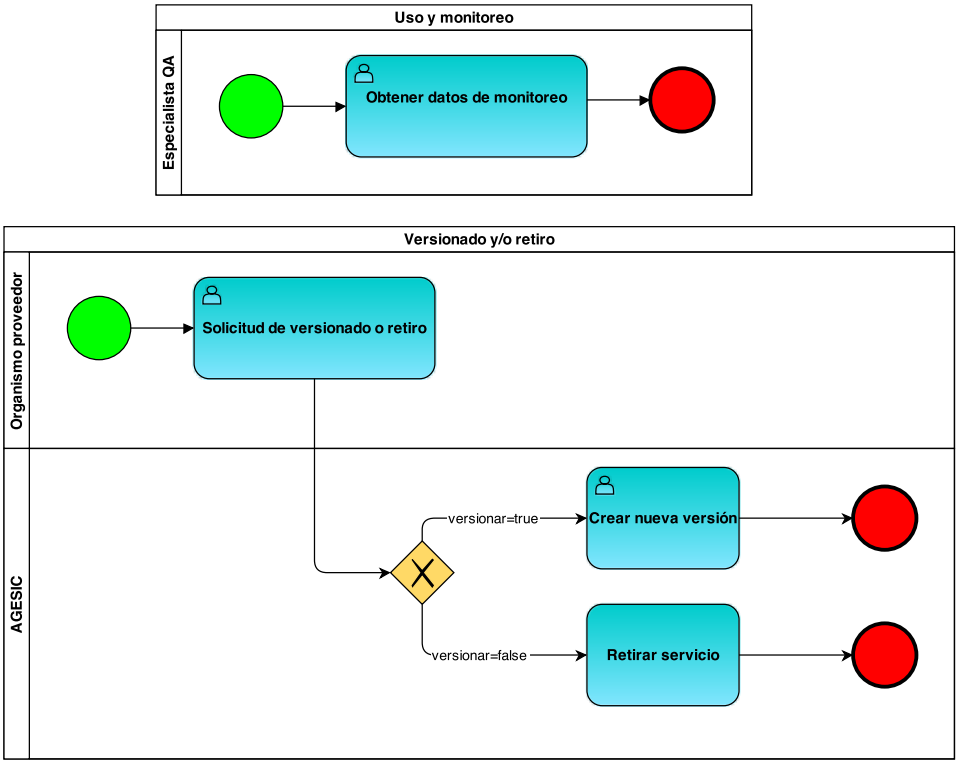

Figura B.1: Subproceso BPMN de ciclo de vida gobernanza

Apéndice C

Formularios de solicitud de servicios en AGESIC

agesic
agencia de gobierno electrónico y sociedad de la información

PRESIDENCIA
REPÚBLICA ORIENTAL DEL URUGUAY

Solicitud de Publicación de Servicio en PGE

Fecha Solicitud:		
Nombre Organismo solicitante:		
Persona Solicitante:	**Nombre:**	
	Mail:	
	Teléfono:	
Información General:		
Datos del Servicio		
Nombre del Servicio:		
Descripción Funcional del Servicio:	*[* *Describia aquí el propósito del servicio. Considerar que esta decripción será la que se incluya en el Catálogo de Servicios.* *]*	
Dirección Física (url):		
WSDL	Adjuntarlo a la solicitud	
Tecnología utilizada:	*[.NET Framework (indicar versión)/Java (indicar versión)/Genexus (indicar versión), etc]*	
Testing/Producción:	<T/P>	
Transferencia de archivos:	*[Indicar si el servicio involucra la transferencia de archivos. En casos que sí, indicar el tamaño promedio y máximo de los archivos a transferir]* *<SI/NO>* ***Tamaño promedio:*** ***Tamaño máximo:***	
Datos de Entrada:	*Para cada campo de entrada indicar:* *•Nombre* *•Descripción* *•Tipo* *•Valor por defecto* *•Rango o conjunto de valores válidos*	
Ejemplos de Entrada:	*Especificar un conjunto de valores de entrada representativos*	
Datos de Salida:	*Para cada campo de entrada indicar:* *•Nombre* *•Descripción* *•Tipo* *•Rango o conjunto de valores válidos*	

Figura C.1: Formulario de solicitud de consumo de servicios PGE v2.5

Solicitud de Publicación de Servicio en PGE

Fecha Solicitud:		
Nombre Organismo solicitante:		
Persona Solicitante:	**Nombre:**	
	Mail:	
	Teléfono:	
Información General:		
Datos del Servicio		
Nombre del Servicio:		
Descripción Funcional del Servicio:	*[* *Describia aquí el propósito del servicio. Considerar que esta decripción será la que se incluya en el Catálogo de Servicios.* *]*	
Dirección Física (url):		
WSDL	Adjuntarlo a la solicitud	
Tecnología utilizada:	*[.NET Framework (indicar versión)/Java (indicar versión)/Genexus (indicar versión), etc]*	
Testing/Producción:	<T/P>	
Transferencia de archivos:	*[Indicar si el servicio involucra la transferencia de archivos. En casos que sí, indicar el tamaño promedio y máximo de los archivos a transferir]* <*SI/NO*> ***Tamaño promedio:*** ***Tamaño máximo:***	
Datos de Entrada:	*Para cada campo de entrada indicar:* •*Nombre* •*Descripción* •*Tipo* •*Valor por defecto* •*Rango o conjunto de valores válidos*	
Ejemplos de Entrada:	*Especificar un conjunto de valores de entrada representativos*	
Datos de Salida:	*Para cada campo de entrada indicar:* •*Nombre* •*Descripción* •*Tipo* •*Rango o conjunto de valores válidos*	

Figura C.2: Formulario de solicitud de publicación de servicios PGE v2.7

Apéndice D

Publicar servicio en el WSO2: caso de estudio

Para la publicación del servicio primero se accede a la plataforma de *WSO2 G-Reg* como se muestran en las Figuras D.1, D.2 y D.3

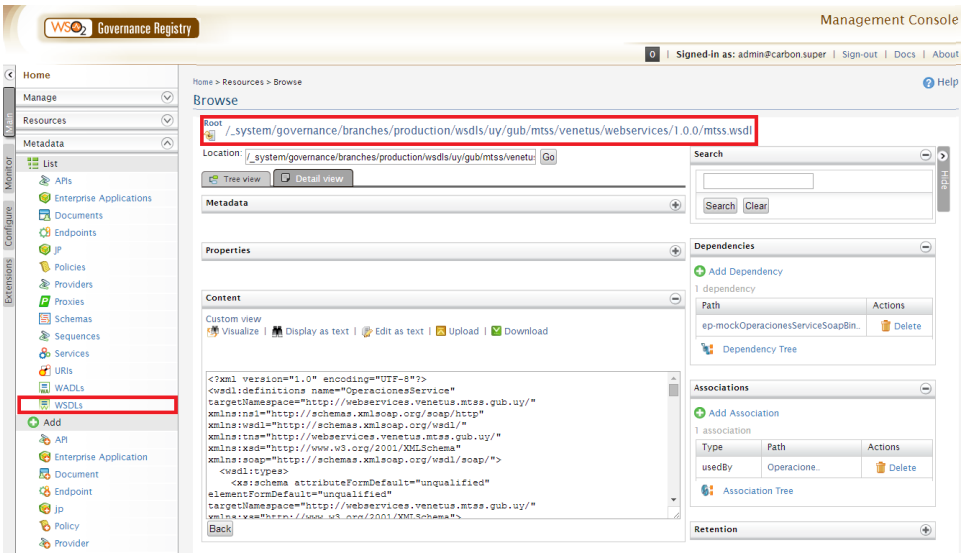

Figura D.1: *WSO2 G-Reg* creación del *WSDL*

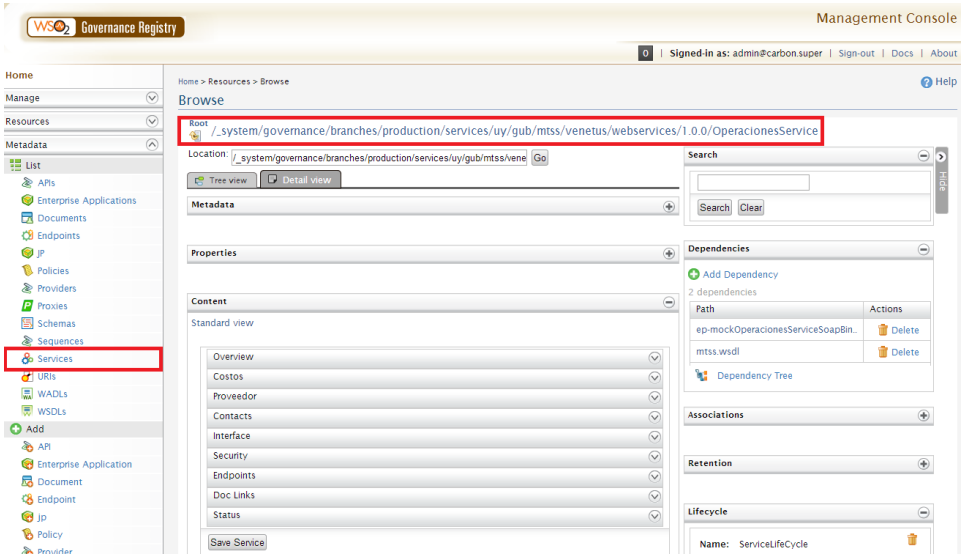

Figura D.2: *WSO2 G-Reg* descripción del servicio OperacionesService V1.0.0

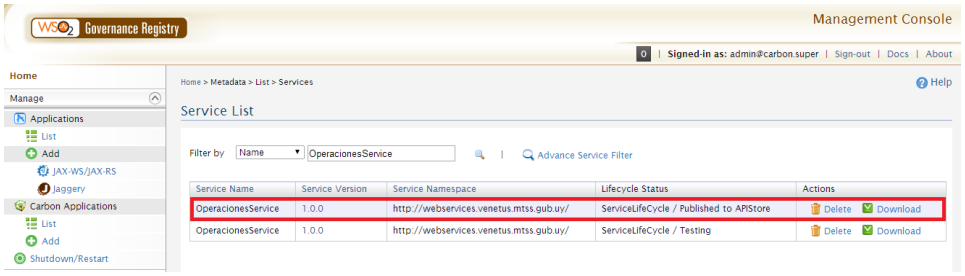

Figura D.3: *WSO2 G-Reg* servicio OperacionesService V1.0.0 en producción

En el *WSO2 ESB* se debe habilitar las estadísticas del servicio y verificar que haya quedado disponible invocando alguna operación como se puede observar en las Figuras D.4 y D.5.

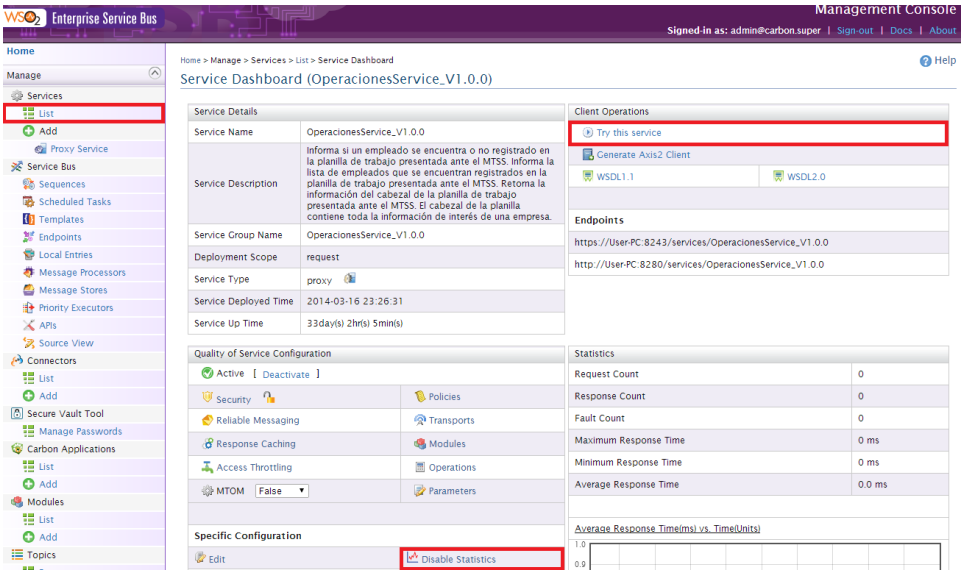

Figura D.4: *Proxy* del servicio OperacionesService_V1.0.0

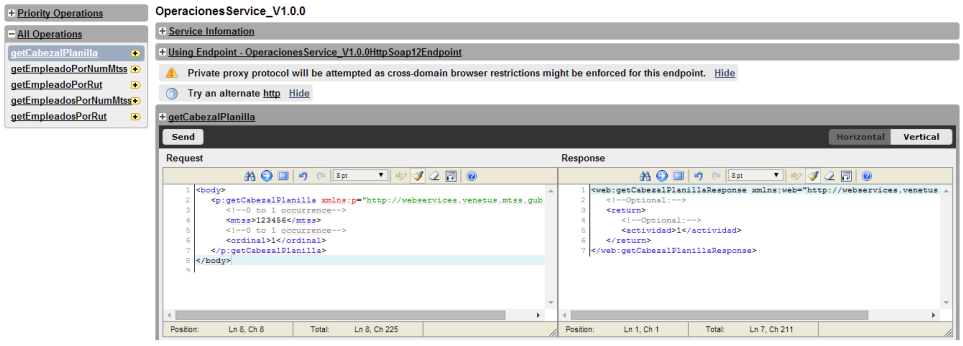

Figura D.5: *Test* con *Try this service* del proxy OperacionesService_V1.0.0

Bibliografía

[1] Agencia para el Desarrollo del Gobierno de Gestión Electrónica y la Sociedad de la Información y del Conocimiento. Misión y visión.

[2] Andrés Segurola Mónica Canto, Daniel Pereda. Service oriented architecture. *Facultad de Ingeniería - Universidad de la República*, 2006.

[3] Área de CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL. Servicios web. http://ccia.ei.uvigo.es/docencia/SCS/0910/transparencias/Tema4.pdf, October 2008.

[4] Bishop Karne. A survey of middleware. *Computer and Information Science Dept*, 2003.

[5] Matías Leandro Varela. Conceptos fundamentales de un middleware y razones de su importancia en el mundo de hoy. http://lisidi.cs.uns.edu.ar/chaco/Trabajos/Evaluaci2007.

[6] O'Reilly. Enterprise service bus,by dave chappell,publisher, June 2004.

[7] Diego Rojas. ¿qué es un esb – enterprise service bus? http://icomparable.blogspot.com/2009/04/que-es-un-esb-enterprise-service-bus.html, November 2009.

[8] Laura González. *Plataforma ESB Adaptativa para Sistemas Basados en Servicios*. PhD thesis, Instituto de Computación – Facultad de Ingeniería - Universidad de la República, 2011.

[9] Bartolomé Sintes Marco. Xslt: Transformaciones xsl. http://www.mclibre.org, April 2012.

[10] W3C. Extensible markup language (xml). http://www.w3.org/XML/, November 2008.

[11] W3. Xml path language (xpath). http://www.w3.org/TR/xpath/, November 1999.

[12] W3. Xml schema. http://www.w3.org/TR/xmlschema-1, October 2004.

[13] W3. Namespaces in xml 1.0, December 2009.

[14] W3C. Web services architecture. http://www.w3.org/TR/ws-arch/, February 2004.

[15] W3C. Web services description language (wsdl) 1.1. http://www.w3.org/TR/wsdl, March 2001.

[16] Alex O'Ree Kurt T Stam. Apache juddi guide. http://juddi.apache.org/, 2014.

[17] Oracle. Interoperability guidelines. http://docs.oracle.com/, 2011.

[18] Keith Ballinger et al. Basic profile version 1.1. http://www.ws-i.org/Profiles/BasicProfile-1.1.html, 2006.

[19] Thomas Erl, Stephen G. Bennett, Benjamin Carlyle, Clive Gee, Robert Laird, Anne Thomas Manes, Robert Moores, Robert Schneider, Leo Shuster, Andre Tost, Chris Venable, and Filippos Santas. *SOA Governance: Governing Shared Services On-Premise and in the Cloud*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition, 2011.

[20] Jim; Graham Steve Holley, Kerrie; Palistrant. Effective soa governance. 2006.

[21] Todd Biske. *SOA Governance*. Packt Publishing, 10 2008.

[22] Michael R. Zheng, Zibin / Lyu. *QoS Management of Web Services*. Zhejiang University Press, 2013.

[23] Instituto de Computación Facultad de Ingeniería. Calidad de servicios web.

[24] Organization for the Advancement of Structured Information Standards. About oasis, 2014.

[25] Laura González. Quality factors for services. *Instituto de Computación – Facultad de Ingeniería*, 2010.

[26] Andreas Gehlert. About. http://www.s-cube-network.eu/, April 2012.

[27] Arun Nagarajan Anbazhagan Mani. Understanding quality of service for web services. https://www.ibm.com/developerworks/library/ws-quality/, January 2002.

[28] Priscilla Walmsley Hugo Haas Umit Yalcinalp Canyang Kevin Liu David Orchard Andre Tost James Pasley Thomas Erl, Anish Karmarkar. *Web Service Contract Design and Versioning for SOA*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition, 2008.

[29] ANJALI ANAGOL-SUBBARAO CHRIS PELTZ. Design strategies for web services versioning. http://chrispeltz.sys-con.com/node/44356, April 2004.

[30] Randy A. Steinberg, David Cannon, Vernon Lloyd, Lou Hunnebeck, and Stuart Rance. *ITIL Lifecycle Suite, 2011 Edition (5 Volume Set)*. The Stationery Office, 2nd edition, 7 2011.

[31] Agencia para el Desarrollo del Gobierno de Gestión Electrónica y la Sociedad de la Información y del Conocimiento. Plataforma de egob.

[32] gencia de gobierno electrónico y sociedad de la información Agencia de gobierno electrónico y sociedad de la información. Guía de uso de la plataforma de ge del estado uruguayo.

[33] Agencia de gobierno electrónico y sociedad de la información. Formulario de solicitud de consumo de servicios en la pge v. 2.5.

[34] Agencia de gobierno electrónico y sociedad de la información. Formulario de solicitud de publicación de servicios, v2.7.

[35] G. de proyectos Agencia de gobierno electrónico y sociedad de la información. Requerimientos para la publicación de servicios sobre pge.

[36] Agencia de gobierno electrónico y sociedad de la información. Catálogo de artículos, servicios y obras que utiliza el estado.

[37] Yeongho Kim Yongkon Lee. Web services quality factors version 1.0. http://docs.oasis-open.org/wsqm/WS-Quality-Factors/v1.0/cos01/WS-Quality-Factors-v1.0-cos01, 2011.

[38] Ronald Monzillo Phillip Hallam-Baker Carlo Milono Anthony Nadalin, Chris Kaler. Web services security: Soap message security. http://docs.oasis-open.org/wss-m/wss/v1.1.1/os/wss-SOAPMessageSecurity-v1.1.1-os.html, May 2012.

[39] InCo. Dimensiones de calidad. http://www.fing.edu.uy/inco/cursos/caldatos/.

[40] Gordon Van Huizen Paolo Malinverno, Daryl C. Plummer. Magic quadrant for application services governance. http://www.gartner.com/technology/reprints.do?id=1-1IBQ7OQ&ct=130809&st=sg, August 2013.

[41] AGESIC. Servicios mtss, November 2012.

[42] Sergio Pío Alvarez Adriana Redín. Servicio básico de información, Dicember 2012.

Glossary

AGESIC Agencia de gobierno electrónico y sociedad de la información. 3

API Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. 14

Backward Compatibility Ante modificaciones en el contrato de servicio la nueva versión continúa soportando las aplicaciones diseñadas para trabajar con su versión anterior. 37

catálogo de servicios Es una base de datos o documento estructurado con información acerca de todos los servicios activos en la organización. 41

Cloud Service Owner Propietario del servicio en la nube. 48

Dimensión Aspecto de la calidad correspondiente a la perspectiva que brinda el mayor nivel de abstracción de la misma. 32

endpoint Terminal donde se encuentra alojado la definición del servicio. 23

Enterprise Architect Quien define la arquitectura de la SOA. 49

Enterprise Design Standards Custodian Quien verifica el seguimiento de los estándares definidos por las políticas que gobiernan la SOA. 49

ESB Bus de servicios de empresa consiste en un combinado de arquitectura de software que proporciona servicios fundamentales para arquitecturas complejas a través de un sistema de mensajes (el bus) basado en las normas y que responde a eventos. 14

Factor Aspecto particular de la dimension de la calidad. 33

Flexible Esta estrategia de versionado toman medidas diferentes de acuerdo a los distintos cambios. 39

Forward Compatibility Es una propiedad que presentan aquellos contratos de servicio que son diseñados de manera que soportan a futuros consumidores sin la necesidad de algún cambio. 37

GE Gobierno Electrónico. 9

Gobernanza Estudia todos los mecanismos, procesos y reglas a través de los cuales se administra una organización. 3

Granularidad Especificidad a la que se define un nivel de detalle. 33

HTTP Es el método más común de intercambio de información en la world wide web (WWW), el método mediante el cual se transfieren las páginas web a un ordenador. 25

inventario Representa una colección de servicios independientemente estandarizados, administrados y gobernados. 26

ITIL Biblioteca de Infraestructura de Tecnologías de Información es un conjunto de conceptos y prácticas para la gestión de servicios de tecnologías de la información, el desarrollo de tecnologías de la información y las operaciones relacionadas con la misma en general. 41

LDAP Protocolo Ligero de Acceso a Directorios que hacen referencia a un protocolo a nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar diversa información en un entorno de red. 14

Middleware Es un software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. 14

Método Forma de implementar una métrica de calidad. 33

Métrica Corresponde a un instrumento que define una forma de medir un factor de calidad. 33

namespace Es un medio para organizar clases dentro de un entorno, agrupándolas de un modo más lógico y jerárquico. 19

PDI Plataforma de interoperabilidad. 41

Policy Custodian Rol encargado de mantener las políticas de seguridad de acceso a la PDI. 48

proxy Interceptar las solicitudes de un servicio que un cliente hace a un servidor de destino, actuando de intermediario entre el cliente y servidor. 50

QoS Calidad de servicios estudia el rendimiento. 32

Quality Abstract Model Modelo abstracto de calidad. 32

Relajada En esta estrategia los cambios incompatibles generan una nueva versión mayor de servicio y los cambios compatibles una nueva versión menor. 39

Schema Custodian Se encarga de definir esquemas normalizados de metadatos para los servicios, aunque estos existen en carácter de recomendación para los organismos. 48

Semántica Define como se mide la métrica. 33

Service Administrator Encargando del funcionamiento de los mismos sobre la plataforma. 48

Service Custodian Encargados de los servicios del lado de los organismos. 48

SGPO Es un área encargada de uno o más dominios de servicios dentro de la organización, según el modelo de jurisdicción adoptado por el sistema de gobernanza. 28

SLA Un acuerdo de nivel de servicio es un contrato escrito entre un proveedor de servicio y su cliente con objeto de fijar el nivel acordado para la calidad de dicho servicio. 9

SOA La arquitectura orientada a servicios es un paradigma de arquitectura para diseñar y desarrollar sistemas distribuidos. 9

SOA Governance Specialist Encargado de dirigir las tareas de gobernanza en SOA en toda de la institución. 49

SOAP Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. 21

TI Tecnicos en informatica. 9

UDDI Es la especificación de una plataforma estándar interoperable que permite publicar, encontrar y utilizar *Web Service*. 20

Unidad de medida Corresponde al proceso por el cual se implementa la métrica. 33

URI Identifica los recursos de una red de forma unívoca. 18

Web Services Es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. 32

WS-I Propone la interoperabilidad entre Web Services. 24

WS-I profiles Propone la interoperabilidad entre Web Services a través de criterios que debe cumplir.. 24

WSDL Es un formato XML que se utiliza para describir Web Services. 20

XML Es un lenguaje de marcas utilizado para almacenar datos en forma legible. 16

XPath Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML. 15

XSD Es el nombre técnico de los lenguajes de esquema de XML. 18

XSLT Es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros e incluso a formatos que no son XML. 15, 16

Universidad de la República

Facultad de Ingeniería

Instituto de Computación

Informe de Proyecto de Grado

ANEXO: Gobernanza de servicios en la plataforma de interoperabilidad de gobierno electrónico

Autores:
Arian Bessonart
Juan Pablo Lucas
Miguel Renom

Tutores:
MSc. Ing. Laura González
MSc. Ing. Guzmán Llambías

Agosto, 2014

Índice general

Capítulo 1
Anexos

1.1. WSO2 Web Services

Para consumir los *Web Services* que ofrecen las herramientas de *WSO2* se debe configurar el archivo *carbon.xml* ubicado en *CARBON_HOME/repository/conf*, siendo *CARBON_HOME* la raíz del directorio donde se encuentra instalada la plataforma. Editar el archivo en la sección *tag HideAdminServiceWSDLs* al valor *false*, y reiniciar la aplicación. Al iniciar el servicios *WSO2* con la opción *DosgiConsole*, la función *listAdminServices* lista el conjunto de *Web Services* disponibles para ser consultados:

1. ProvisioningAdminService
   https://<host:port>/services/ProvisioningAdminService?wsdl
2. CarbonAppUploader
   https://<host:port>/services/CarbonAppUploader?wsdl
3. OperationAdmin
   https://<host:port>/services/OperationAdmin?wsdl
4. MediationSecurityAdminService
   https://<host:port>/services/MediationSecurityAdminService?wsdl
5. SequenceAdminService
   https://<host:port>/services/SequenceAdminService?wsdl
6. StatisticsAdmin
   https://<host:port>/services/StatisticsAdmin?wsdl
7. TopicManagerAdminService
   https://<host:port>/services/TopicManagerAdminService?wsdl
8. MediationStatisticsAdmin
   https://<host:port>/services/MediationStatisticsAdmin?wsdl
9. ApplicationAdmin
   https://<host:port>/services/ApplicationAdmin?wsdl

10. ServiceGroupAdmin
    https://<host:port>/services/ServiceGroupAdmin?wsdl
11. ClassMediatorAdmin
    https://<host:port>/services/ClassMediatorAdmin?wsdl
12. CustomMeteringService
    https://<host:port>/services/CustomMeteringService?wsdl
13. BAMMediatorConfigAdmin
    https://<host:port>/services/BAMMediatorConfigAdmin?wsdl
14. STSAdminService
    https://<host:port>/services/STSAdminService?wsdl
15. FileDownloadService
    https://<host:port>/services/FileDownloadService?wsdl
16. CachingAdminService
    https://<host:port>/services/CachingAdminService?wsdl
17. MessageStoreAdminService
    https://<host:port>/services/MessageStoreAdminService?wsdl
18. CustomUIAdminService
    https://<host:port>/services/CustomUIAdminService?wsdl
19. RMAdminService
    https://<host:port>/services/RMAdminService?wsdl
20. ReportingResourcesSupplier
    https://<host:port>/services/ReportingResourcesSupplier?wsdl
21. ThrottleAdminService
    https://<host:port>/services/ThrottleAdminService?wsdl
22. JMSTransportAdmin
    https://<host:port>/services/JMSTransportAdmin?wsdl
23. ExternalTryitService
    https://<host:port>/services/ExternalTryitService?wsdl
24. DiscoveryAdmin
    https://<host:port>/services/DiscoveryAdmin?wsdl
25. ResourceAdminService
    https://<host:port>/services/ResourceAdminService?wsdl
26. FileUploadService
    https://<host:port>/services/FileUploadService?wsdl

27. DataSourceAdmin
    https://<host:port>/services/DataSourceAdmin?wsdl
28. MediationLibraryUploader
    https://<host:port>/services/MediationLibraryUploader?wsdl
29. ConfigServiceAdmin
    https://<host:port>/services/ConfigServiceAdmin?wsdl
30. EventBrokerService
    https://<host:port>/services/EventBrokerService?wsdl
31. TracerAdmin
    https://<host:port>/services/TracerAdmin?wsdl
32. ProxyServiceAdmin
    https://<host:port>/services/ProxyServiceAdmin?wsdl
33. RepositoryAdminService
    https://<host:port>/services/RepositoryAdminService?wsdl
34. PriorityMediationAdmin
    https://<host:port>/services/PriorityMediationAdmin?wsdl
35. DeploymentSynchronizerAdmin
    https://<host:port>/services/DeploymentSynchronizerAdmin?wsdl
36. ServerAdmin
    https://<host:port>/services/ServerAdmin?wsdl
37. FlowsAdminService
    https://<host:port>/services/FlowsAdminService?wsdl
38. UserAdmin
    https://<host:port>/services/UserAdmin?wsdl
39. LogViewer
    https://<host:port>/services/LogViewer?wsdl
40. SynapseArtifactUploaderAdmin
    https://<host:port>/services/SynapseArtifactUploaderAdmin?wsdl
41. SynapseApplicationAdmin
    https://<host:port>/services/SynapseApplicationAdmin?wsdl
42. MediationLibraryAdminService
    https://<host:port>/services/MediationLibraryAdminService?wsdl
43. LoggedUserInfoAdmin
    https://<host:port>/services/LoggedUserInfoAdmin?wsdl

44. MessageProcessorAdminService
    https://<host:port>/services/MessageProcessorAdminService?wsdl
45. NDataSourceAdmin
    https://<host:port>/services/NDataSourceAdmin?wsdl
46. RegistryAdminService
    https://<host:port>/services/RegistryAdminService?wsdl
47. RMAdminGlobal
    https://<host:port>/services/RMAdminGlobal?wsdl
48. LoginStatisticsAdmin
    https://<host:port>/services/LoginStatisticsAdmin?wsdl
49. SearchAdminService
    https://<host:port>/services/SearchAdminService?wsdl
50. CGAgentAdminService
    https://<host:port>/services/CGAgentAdminService?wsdl
51. TaskAdmin
    https://<host:port>/services/TaskAdmin?wsdl
52. ModuleAdminService
    https://<host:port>/services/ModuleAdminService?wsdl
53. MediationTracerService
    https://<host:port>/services/MediationTracerService?wsdl
54. RestApiAdmin
    https://<host:port>/services/RestApiAdmin?wsdl
55. LoggingAdmin
    https://<host:port>/services/LoggingAdmin ?wsdl
56. SampleDeployer
    https://<host:port>/services/SampleDeployer?wsdl
57. KeyStoreAdminService
    https://<host:port>/services/KeyStoreAdminService?wsdl
58. SecurityAdminService
    https://<host:port>/services/SecurityAdminService?wsdl
59. JrxmlFileUploader
    https://<host:port>/services/JrxmlFileUploader?wsdl
60. EndpointAdmin
    https://<host:port>/services/EndpointAdmin?wsdl

61. ServerRolesManager
    https://<host:port>/services/ServerRolesManager?wsdl
62. ThemeMgtService
    https://<host:port>/services/ThemeMgtService?wsdl
63. MediationStatPublisherAdmin
    https://<host:port>/services/MediationStatPublisherAdmin?wsdl
64. WSDLValidatorService
    WSDLValidatorService
    https://<host:port>/services/WSDLValidatorService?wsdl
65. CommandMediatorAdmin
    https://<host:port>/services/CommandMediatorAdmin?wsdl
66. MultipleCredentialsUserAdmin
    https://<host:port>/services/MultipleCredentialsUserAdmin?wsdl
67. TransportAdmin
    https://<host:port>/services/TransportAdmin?wsdl
68. Java2WSDLService
    https://<host:port>/services/Java2WSDLService?wsdl
69. UserStoreConfigAdminService
    https://<host:port>/services/UserStoreConfigAdminService?wsdl
70. TemplateAdminService
    TemplateAdminService
    https://<host:port>/services/TemplateAdminService?wsdl
71. TenantMgtAdminService
    https://<host:port>/services/TenantMgtAdminService?wsdl
72. DBReportingService
    https://<host:port>/services/DBReportingService?wsdl
73. RelationAdminService
    https://<host:port>/services/RelationAdminService?wsdl
74. ServiceAdmin
    https://<host:port>/services/ServiceAdmin?wsdl
75. PropertiesAdminService
    https://<host:port>/services/PropertiesAdminService?wsdl
76. WSDL2CodeService
    https://<host:port>/services/WSDL2CodeService?wsdl

77. TransportStatisticsAdmin
    https://<host:port>/services/TransportStatisticsAdmin?wsdl

78. LocalEntryAdmin
    https://<host:port>/services/LocalEntryAdmin?wsdl

79. EndpointTemplateAdminService
    https://<host:port>/services/EndpointTemplateAdminService?wsdl

Donde el *host* y *port* corresponden al *ip* y puerto en el cual se encuentra levantado el servidor *WSO2*. Los servicios enumerados corresponden a la aplicación de *WSO2 ESB*, análogamente se puede enumerar los *Web Services* que ofrece *WSO2 G-Reg* iniciando el servicio con la opción *DosgiConsole*.

1.2. Manual de usuario WSO2

Basado en el manual que ofrece en la web oficial de *WSO2 G-Reg* y *WSO2 ESB* se establece una guía de pasos a seguir que involucra todas las etapas que atraviesa un servicio en la propuesta de solución del proyecto, utilizando las herramientas mencionadas.

1. Un funcionario de AGESIC ingresa a la plataforma de *WSO2 G-Reg* y agrega el *WSDL* del servicio que se desea crear. Se puede importar el *WSDL* desde un archivo *zip/wsdl* o desde una *URL* donde se encuentre publicado el *Web Service*. Como buena practica se sugiere que el nombre del *WSDL* en el *WSO2 G-Reg* se de la forma: *NombreWSDL_ VX.X.X*, donde X.X.X indica el número de versión. El *WSDL* debe tener definido el *soapAction* de las operaciones ya que encontramos un problema con los servicios que tienen vació el campo en el *WSDL*.

2. Finalizado el punto anterior, se crea de forma automática el servicio con el nombre de servicio y *Namespace* extraído desde el *WSDL* y versión *1.0.0-SNAPSHOT* por defecto. También de forma automática se crea el endpoint del servicio generando una dependencia al servicio y al *WSDL* creado. La entidad servicio que brinda *WSO2 G-Reg* consta de un conjunto de meta-datos que tienen que ser ingresados como *costos, organismo proveedor, URL de documentación, estado del servicio, etc.* Establecida las propiedades del servicio, se procede a guardar y comenzar el proceso de ciclo de vida *ServiceLifeCycle* definido en WSO2 G-Reg:

   a) *Development* es el estado por defecto que tiene el servicio cuando es creado luego del punto 1. Para completar el estado, se ingresan los datos del servicio en los formularios y luego se promueve con la opción *Promote* indicando con que número de versión pasa al siguiente estado. Si el servicio se esta dando de alta por primera vez, continúan con la versión 1.0.0, sino con el número de versión que corresponda.

   b) *Testing* es el segundo estado en que se encuentra el ciclo del servicio. En esta etapa se pueden hacer cambios de propiedades si corresponde. Luego se promueve el servicio con la opción *Promote* asignando el número de versión con el mismo criterio anterior.

   c) *Production* con la opción *Publish* se promueve al último estado del ciclo de vida del servicio.

3. Para publicar el *proxy* del servicio creado en el *WSO2 G-Reg*, se debe acceder a la plataforma del *WSO2 ESB*.

4. Desde el *WSO2 ESB* se debe crear la entidad *Endpoint* en la plataforma con la opción *Address Endpoint*; indicando el nombre del endpoint y la URL donde se encuentra publicado el *Web Service*. Como buena practica se sugiere asignar el mismo nombre de endpoint que se definió en el *WSO2 G-Reg*. Cuando haya quedado creado, habilitar la opción *Enable Statistics* para monitorear métricas de calidad vinculadas al endpoint del proveedor del servicio.

5. Finalmente se procede a publicar el *proxy* utilizando el template *Custom Proxy*.

   a) *Proxy Service Name* es el nombre del servicio *proxy* con que se identifica a la instancia de servicio en el *WSO2 ESB*. La estructura debe ser *NombreServicio_ VX.X.X* indicando en X.X.X el numero de versión que corresponda.

   b) *Publishing WSDL* se debe establecer *Pick from Registry* para poder seleccionar el *WSDL* creado en el WSO2 G-Reg.

   c) Reference Key seleccionar desde la opción *Governance Registry* el *WSDL* creado en el *WSO2 G-Reg*.

   d) *Add Parameter* agregar un parámetro con nombre *PathGREG* y en valor la locación del repositorio donde se encuentra el servicio en el*WSO2 G-Reg*.

   e) En el step siguiente se establece las transformaciones y endpoint del proxy. En *Define Endpoint* seleccionar la opción *Existing Endpoint* y buscar el endpoint creado en el punto 4.

   f) Finalmente en el último step simplemente seleccionar *Finish*.

6. En *List* debería estar el *proxy* creado. Al seleccionarlo, en *Specific Configuration* habilitar *Enable Statistics* para monitorear las métricas de calidad del servicio.

7. Por ultimo, verificar que el *proxy* haya quedado bien publicado con *Try this service* invocando alguna operación del servicio.

Se define el procedimiento para agregar una transformación en el *WSO2 ESB*:

1. En *Local Entries* agregar *Add In-lined XML Entry* el código *XSLT* correspondiente que realiza las transformaciones necesarias para que el servicio pueda continuar siendo utilizado sin hacer afectado por la nueva versión. Se debe definir una transformación que permita modificar el mensaje soap de forma que el proveedor del servicio pueda resolver la solicitud. Y dependiendo del cambio con respecto se debe definir otra transformación para procesar el mensaje soap de respuesta que se

le devuelve al cliente.
Como buena practica se siguiere establecer como nombre de la transformación *XSLT* el mismo nombre del servicio y versión para poder llevar un registro más ordenado.

2. Luego se establece una instancia llamada *Sequences* el cual permite establecer un flujo de acciones a realizar cuando llega un mensaje soap de un proxy. En *Add Child* se selecciona la opción *Transformation->XSLT*. Sobre *Key* seleccionar *Governance Registry* y sobre *Local Entry* hay un combo donde se visualizara el *XSLT* creado en el punto 1. Para el nombre de la *Sequences* se propone mantener el mismo criterio de creación de *Local Entries*. Finalmente creado el flujo, habilitar el *Enable Statistics* para monitorear métricas de calidad.

3. En *List* seleccionar el *proxy* que se desea aplicar la transformación y *Edit*.

4. En el step 2, sobre *Define In Sequence* seleccionar en *Use Existing Sequence* la secuencia creada en el punto 2. Esta transformación es para modificar los menasjes soap a entregar al proveedor.

5. En el step 3, si corresponde, sobre *Define OUT Sequence* seleccionar en *Use Existing Sequence* la secuencia creada en el punto 2. Esta transformación es para modificar los menasjes soap a entregar al cliente que invoco el servicio.

6. Finalmente seleccionar *Finish*.

1.3. Extensión de meta-datons en WSO2 G-Reg

Los meta-datos de los servicios creados en *WSO2 G-Reg* pueden extenderse utilizando *XML* en el témplate *service* que se encuentra en *Artifact Source*. Las Figuras 1.1 y 1.2 muestran los fragmentos de código *XML* que fueron agregados al témplate original de *service* para agregar el formulario de datos sobre costos y proveedores.

```
<table name="Proveedor">
      <field type="options" required="true">
              <name label="Organismo">Organismo</name>
                      <values>
                        <value></value>
                        <value>DNIC</value>
                        <value>DGI</value>
                        <value>DGREC</value>
                        <value>BPS</value>
                        <value>MSP</value>
                        <value>DNA</value>
                        <value>CGN</value>
                        <value>MTSS</value>
                        <value>DNB</value>
                        <value>ACCE</value>
                        <value>Otros</value>
                      </values>
      </field>
</table>
```

Figura 1.1: *XML* proveedores

```
<table name="Costos">
      <field type="options" readonly="false">
              <name label="Tiene Costos">Costo</name>
                      <values>
                              <value>si</value>
                              <value>no</value>
                      </values>
      </field>
      <field type="text" required="true" readonly="false">
              <name>Descripción del costo</name>
       </field>
</table>
```

Figura 1.2: *XML* costos

1.4. Transformaciones

Durante el desarrollo del proyecto, se realizó pruebas con servicios que requirieron aplicar transformaciones. De forma de aporte se describen los *XSLT* para diferentes cambios específicos. En la Figura 1.3 se presenta el código *XSLT* para aplicar en caso de un cambio incompatible manejable: renombrar operación. La Figura 1.4 aplica para cambios de eliminación de parámetros opcionales de una operación. La Figura 1.5 permite transformar parámetros de tipo *Integer* a *Float* en una operación.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0" xmlns:unk="http://unknown.namespace/">
<xsl:template match="node()|@*">
        <xsl:copy>
            <xsl:apply-templates select="node()|@*"/>
        </xsl:copy>
    </xsl:template>
<xsl:template match="unk:sum">
        <unk:multiply>
            <xsl:apply-templates select="@*|node()" />
        </unk:multiply>
    </xsl:template>
</xsl:stylesheet>
```

Figura 1.3: *XSLT* Renombrar método

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
<xsl:template match="node()|@*">
        <xsl:copy>
            <xsl:apply-templates select="node()|@*"/>
        </xsl:copy>
    </xsl:template>
<xsl:template match="metAModificadoReturn">
    </xsl:template>
</xsl:stylesheet>
```

Figura 1.4: *XSLT* Remover parámetro

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="@* | node()">
    <xsl:copy>
      <xsl:apply-templates select="@* | node()"/>
    </xsl:copy>
  </xsl:template>
  <xsl:template match="paramB">
    <b>
      <xsl:value-of select="format-number(., '#.00')" />
    </b>
  </xsl:template>
</xsl:stylesheet>
```

Figura 1.5: *XSLT* Cambiar tipo de parámetro

1.5. Pruebas Realizadas

Se utilizó varios servicios publicados en el catálogo de AGESIC para simular las pruebas sobre el prototipo implementado. Los servicios fueron simulados por la herramienta *soapUI* a través de la funcionalidad *MockService*. Los objetos *Mock* permite simular los servicios externos al código que queremos probar. Al no disponer conectividad ni acceso a datos de los servicios, esta alternativa permite simular el comportamiento de la base de datos utilizando objetos simulados que tenga el comportamiento esperado para el sistema real antes determinado conjunto de datos de entrada.

Los sevicios seleccionados fueron:

- ConsultaProveedores
- comprobanteWSService
- operacionesService
- RUTPersonaGetNombre

Cada servicio fue publicado en el *WSO2 G-Reg* en su versión 1.0.0 sin cambiar la definición del wsdl, con la salvedad de el endpoint que para ser simulado con *MockService* se modifico el *host* a *localhost*. A cada servicio se le aplicaron diversos cambios que se detallan en la Tabla 1.1.

Servicio	Versión	Tipo de Cambio	Descripción
ConsultaProveedores	1.1.0	Incompatible manejable	Renombrado de operación
comprobanteWSService	1.1.0	Incompatible manejable	Cambio de tipo de atributo
operacionesService	1.1.0	Compatible	Nueva operación
operacionesService	1.2.0	Incompatible manejable	Cambio de nombre de operación
operacionesService	2.0.0	Incompatible no manejable	Eliminación de operación
RUTPersonaGetNombre	1.0.1	Compatible	Agregar parámetro en operación

Cuadro 1.1: Servicios de testing de AGESIC

Por otra parte se realizaron pruebas de rendimiento a través de la funcionalidad *LoadTest* de *soapUI* permitiendo ejecutar varias solicitudes a un servicio en un periodo de tiempo. Con respecto a la disponibilidad se fue dando de baja/alta el servicio en un periodo de tiempo. Para medir el grado de éxito (*Successability*) de un servicio a lo largo del tiempo se realizó invocaciones al *proxy* de un servicio publicado en *WSO2 ESB* con el servicio del proveedor no disponible, notando disminución del porcentaje de éxito de las solicitudes.

En las Figuras 1.6, 1.7 y **??** muestran las mediciones del servicio tras una sola invocación a una operación.

Figura 1.6: Prueba 1 cantidad de solicitudes

Figura 1.7: Prueba 1 disponibilidad del servicio

Figura 1.8: Prueba 1 rendimiento del servicio

Luego se ejecutó con *sopUI* 500 solicitudes como muestra la Figura 1.9, dando como resultado lo que indican en las Figuras 1.12, 1.10 y 1.11. Se puede apreciar claramente que se aumentaron a 501 solicitudes.

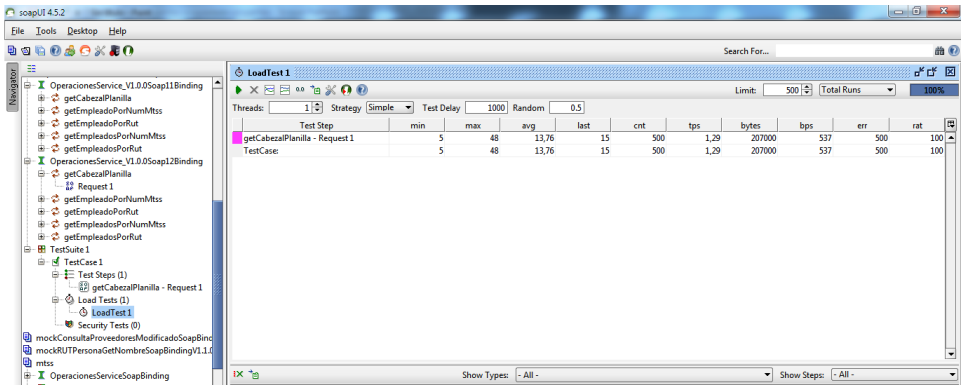

Figura 1.9: Prueba 2 ejecución de solicitudes al servicio

Figura 1.10: Prueba 2 disponibilidad del servicio

Figura 1.11: Prueba 2 rendimiento del servicio

Figura 1.12: Prueba 2 cantidad de solicitudes

En la siguiente prueba se detiene el servicio y se realizan solicitudes (7 pedidos) al mismo. El resultado se muestra en las Figuras 1.13, 1.14 y 1.15.

Figura 1.13: Prueba 1

Figura 1.14: Prueba 1

Figura 1.15: Prueba 1

La cantidad de fallos se incrementa a siete por parte del ESB y mantiene cero fallos por parte del proveedor del servicio, ya que no se encuentra disponible. Se aprecia como disminuye la disponibilidad y la cantidad de pedidos exitosamente.