Trabajo Final de Cátedra

Computación Orientada a Servicios

Marcelo Javier Rodríguez

Leg: 245-221

mailto:mrodriguez@alumnos.exa.unicen.edu.ar

Tandil, Marzo de 2014

Profesor: Dr. Ing. Cristian Mateos Diaz.

Resumen

El presente escrito expone breve y concisamente aspectos relevantes respecto a la experiencia obtenida durante el desarrollo de un servicio web solicitado por la cátedra, así mismo comentarios teórico-prácticos y una breve discución sobre la implementación del mismo bajo el paradigma REST.

1.Introducción.

El trabajo final de la materia consiste en el desarrollo de un servicio web[1] estilo RPC. Para el desarrollo del servicio se ha facilitado como guía un conjunto preciso de categorías alternativas;dada la libertad de elección sobre la funcionalidad del proyecto, el servicio en cuestión surgió como una combinación de estas.

Taitum (time to move!) es el web service[1] objeto de estudio en este informe. Taitum está destinado a facilitar y reducir el valioso tiempo que demanda el proceso de búsqueda de un inmueble. Actualmente, y por mera simplicidad, Taitum brinda:

- Registrar avisos de inmuebles en alquiler/venta.
- Búsquedas simples sobre metadatos de los immuebles, también búsquedas por proximidad respecto de una dirección (calle,altura,ciudad,etc.).
- Registrar búsquedas con notificación por email ante una eventual coincidencia.
- Simples reportes estadísticos sobre las ofertas registradas

.

La entrega del trabajo se compone del presente documento y el workspace de los proyectos; puede acceder a estos en :https://www.dropbox.com/sh/neap0w5aneb8l60/M_Dy4QNyCK

1 Un servicio web es una tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones: http://es.wikipedia.org/wiki/Servicio_web

2.Taitum.

En el contexto de Computación Orientada a Servicios, lograr el cometido de Taitum requiere de otros servicios web: Emailws, Statisticsws y Geocoderws.Tales servicios han sido desarrollados para dar sentido a Taitum y en consecuencia experimentar el rehúso de software vía composición (sup. que los servicios que integran la aplicación exisistieran).

- **Geocoderws**: permite calcular distancias entre direcciones de localidades arbitrarias, también conversion de coordenadas geográficas a direcciones, y viceversa[2].
- **Statisticsws**: ofrece operaciones para definir espacios muestrales, variables aleatorias simples/conjuntas sobre las cuales aplicar indicadores estadisticos.
- **Emailws**: publica métodos para el envío de emails desde una cuenta gmail, y su búsqueda mediante filtros.

Es de destacar que dichos servicios han sido desarrollados mediante el enfoque **Contract-First**. Todos ofrecen operaciones sencillas,y el código de su implementación está organizado de forma tal que incorporar nueva funcionalidad más compleja se pueda alcanzar con mínimas modificaciones, (precisamente relaciones de composición y extensión) aún repetando sus contratos(particularmente Statisticsws, donde variables aleatorias conjuntas se implementan como filtros que siguen al patrón de diseño composite-visitor).

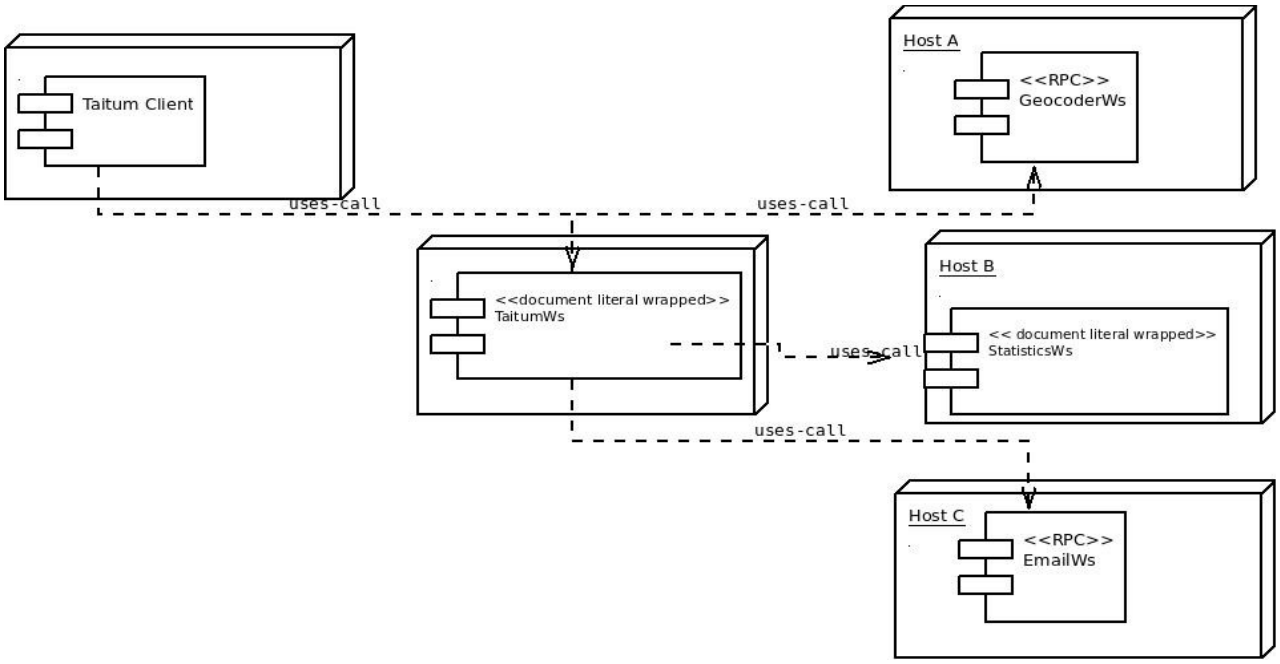


Fig 1. Diagrama de despliegue.

Desde la perspectiva de Taitumws, Geocoderws promueve consultas del estilo :“quiero vivir a no más de 5 cuadras de General Pinto,399,Tandil”; Emailws, “avisar a arminvanbuuren@armada.nl que encontramos un departamento por el precio solicitado”; y Statisticws, “porcentaje de personas que prefieren la zona céntrica, pero $5000”,etc.

2 Conocido como Reverse Geocoding.http://en.wikipedia.org/wiki/Reverse_geocoding

•

2.1.El Servicio.

A continuación se explican detalles sobre las operaciones disponibles en el servicio, sus tipos de datos y estilo del contrato.

2.1.1.Métodos publicados en Taitum.

En este punto es importante destacar que según la operación, y con fines didácticos, en algunas operaciones se ha variado el patrón de comunicacion sin alterar la semántica.

- **Ads getAds()**: lista todas ofertas que existen en el servidor.

- **void publish(ad,adType)**: registra un aviso de oferta de inmueble o una solicitud de notificación poremail ante una búsqueda satisfactoria. Los datos de publicación se encuentran en el tipo de dato ad; el tipo de publicación (alerta por búsqueda, oferta,etc.) está determinado por adType.
  El valor de adType es clave en una hashtable de **factory method**[3]s; así, las publicaciones se asociana métodos que se ejecutarán ante una eventual operación sobre dichas publicaciones.
  publish(...) es una operación rápida, no involucra cálculos con operaciones que puedan emitir excepciones, por lo tanto se ha decidido variar el patrón de comunicación e implementarla como **one-way**.
  Al momento de registrar un aviso, se consultan el resto de las solicitudes; al registrar una solicitud, se consultan las ofertas.El patrón de diseño **observer**[4] plasma este comportamiento.

- **Report getReport(ad,adType)**: esta operación tiene una semántica similar a la operación **Ads search(ad,adType)** y su finalidad es calcular una distribución muestral de las ofertas que satisfacen búsquedas según adType con el valor ad. Si no hay un reporte que coincida con la búsqueda, la operación retorna el reporte global sobre todas las ofertas registradas

- **Ads search(ad,adType)**: lista todas las ofertas en el servidor que coinciden con el valor del dato ad, según el atributo o metadato indicado por adType.

2.1.2.Tipos de datos.

Para fomentar el rehúso del schema y minimizar el tamaño del archivo WSDL, las definiciones de los tipos de datos se encuentran en un archivo separado. Los tipos de datos que se utilizan en el servicio son los siguientes.

- **Ad**:modela una publicación: dirección/tel de contacto del interesado; dirección del inmueble, precio ofrecido/solicitado, etc.
- **AdType**:tipo de datos que catagoriza una publicación. La clase de publicación sirve para despacho de métodos. Ej: si se registra una publicación con solicitud de notificación vía email según proximidad a una dirección, se disparará una invocacion a Geocoderws; si es una búsqueda por precio, se utilizará un filtro que tiene en cuenta el atributo precio de cada publicacióndato;etc.
- **Ads**: es un arreglo de Ad. Es decir, una lista de avisos.
- **Report**:fundamentalmente es una distribucion de probabilidad, donde cada evento no está representado por un valor X=xi entero, sino más bien por el equivalente en cadena de caracteres de la publicación, Ad correspondiente.

3 Metódo fábrica:http://en.wikipedia.org/wiki/Factory_method_pattern
4 Observer-Observable en Java. http://en.wikipedia.org/wiki/Observer_pattern

2.2.Diseño del contrato.

Se ha decantado por el enfoque **Contract-First** antes que Code-First principalmente porque el proyecto comienza desde cero,y entre otras porque, durante la cursada se entendió que conlleva a desarrollar servicios menos propensos al cambio beneficiando, en consecuencia al desarrollo de aplicaciones cliente además de facilitar mayor presición en la definicion del contrato y evitar vicios sobre los tipos de datos dependientes de un de un lenguaje de programación en particular.

2.3.Estilo RPC.

En el diagrama de despliege de la fig.1. aparecen cuatro servicios web, de los cuales dos siguen el estilo RPC Literal, y dos, Document Literal Wrapped. Esta división se debe sóo a fines didácticos, y a partir de esta experiencia se pudo resumir que:

Tanto en Taitum y Statisticws hay operaciones con los mismos argumentos, y dado que en el contexto de servicios el overriding de funciones no es posible, al escoger el estilo Document , fue necesario utilizar Document literal Wrapped para cual envolver el mensaje SOAP dentro de otro elemento que contenga el nombre del método objetivo y así facilitar el dispatch del método asociado en el servicio web. Esta elección permitie obtener la misma estructura de mensaje SOAP que el estilo RPC literal.

Respecto al documento WSDL. En RPC literal los messages de las operaciones puede tener un part element para cada argumento y su “type”, esto clarifica la correspondencia con el código; es decir, es más amigable programaticamente. En contraste, en Document literal Wrapped los message elements no pueden ser multipartes, esto implica crear un element más, dificultando la legibilidad del contrato con su impacto sobre el código autogenerado.

3.Detalles de implementación y diagrama de clases.

3.1.Detalles de implementación.

Cabe aclarar que tanto los servicios y la aplicación que los consume, y sus respectivas implementaciones son sencillas y acotadas en número por razones de simplicidad y tiempo; ya que se considera que el objetivo es demostrar el conocimiento adquirido para realizar servicios web, no implementar la lógica de los mismos.

Las herramientas para el desarrollo de los servicios fueron el **IDE Eclipse WTP**, utilizando el framework para web service **Apache AXIS 2**[5]**,** y **Apache Tomcat 7**.

Nota: Tal como se muestral en Fig 1., Taitumws se compone de otros servicios y *es imprescindible que previo a su ejecución estos ya estén listos y ejecutandose.*

3.1.1.Servidor Taitum.

Actualmente la implementación de Taitum **single-thread** y por lo tanto capaz de manejar secuencialmente las solicitudes del cliente. La lógica de Taitum, invoca sincrónicamente los servicios que lo integran y en consecuencia ante cada invocación remota todo el procesamiento se detiene. Esto resulta en un obstáculo para la performance del servicio y a su vez una oportunidad pendiente de mejoras.

El código de los clientes de los servicios Geocoderws,Emailws y Statisticsws, que componen la lógica de Taitum fueron generados mediante la herramiente **wsimport**[6] del framework JAX-WS[7].

5 Apache **AXIS 2**:http://axis.apache.org/axis2/java/core/

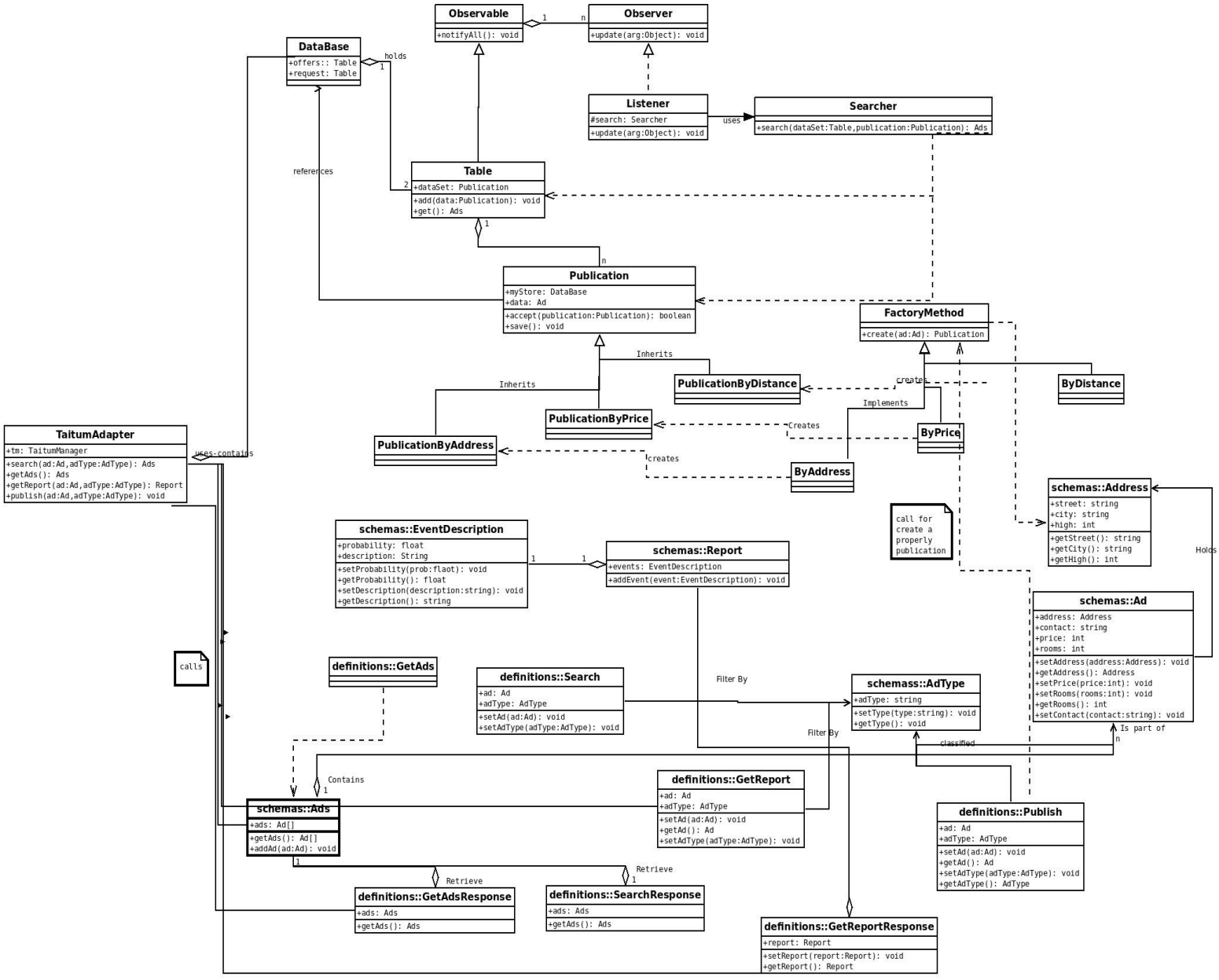
6 **wsimport** permite generar clientes con invocacion no bloqueante o asincrónica al servicio correspondiente. En estos casos, la signatura de los métodos tiene un sufijo Async,y para completar la implemetación se requiere de un handler donde recibir el callback desde el web service.

7 Fuente: http://en.wikipedia.org/wiki/Java_API_for_XML_Web_Services . Sitio del proyecto: https://jax-ws.java.net/

3.2.Diagrama de clases.

3.2.1.Diagrama de clases del servicio web[8].

En esta sección se muestra la estructura de clases del servicio.Como resultado de la generación automática del código se tiene una clases para cada operación publicada, esqueletos de los datos complejos que aparecen en el WSDL (argumentos y retornos, definidos en el schema). No se incluyen el resto de las clases que completan la implementación del servicio ya que no son autogeneradas desde el contrato y corresponden a la lógica del servicio.



8 Por cada nueva publicación se invoca una instancia de tipo **FactoryMethod** almacenada en un HashTable según el valor adType (argumento en cualquier metodo de TaitumAdapter): esto registra un aviso y le asocia con alguna subclase de Publication;posteriormente, el patrón **observer-observable** implementa las búsquedas requeridas: si es una oferta, revisará entre las solicitudes; si es una solicitud de búsqueda-notificación por email, entre las ofertas; la idea es mantener una **complejidad temporal O(n)** en vez de buscar siempre entre todos los registros y llegar a **O(n^2)**:http://es.wikipedia.org/wiki/Cota_superior_asintótica

Fig. 2 Diagrama de clases del web service Taitumws.

3.1.2.Cliente Taitum.

La forma estándar de consumir servicios se basa en el patrón **Proxy**[9], el cual es una clase que puentea los servicios publicados para permitir la comunicación remota aparentemente local; el cliente desconoce la implementación real, y el proxy encuentra los métodos remotos,y empaquta y desempaqueta los datos que se comunican.

En todos los casos el proxy y los stubs del lado cliente fueron generadas automáticamente con la herramienta **wsimport** del framework JAXWS.

3.2.2.Diagrama de clases del cliente[10].

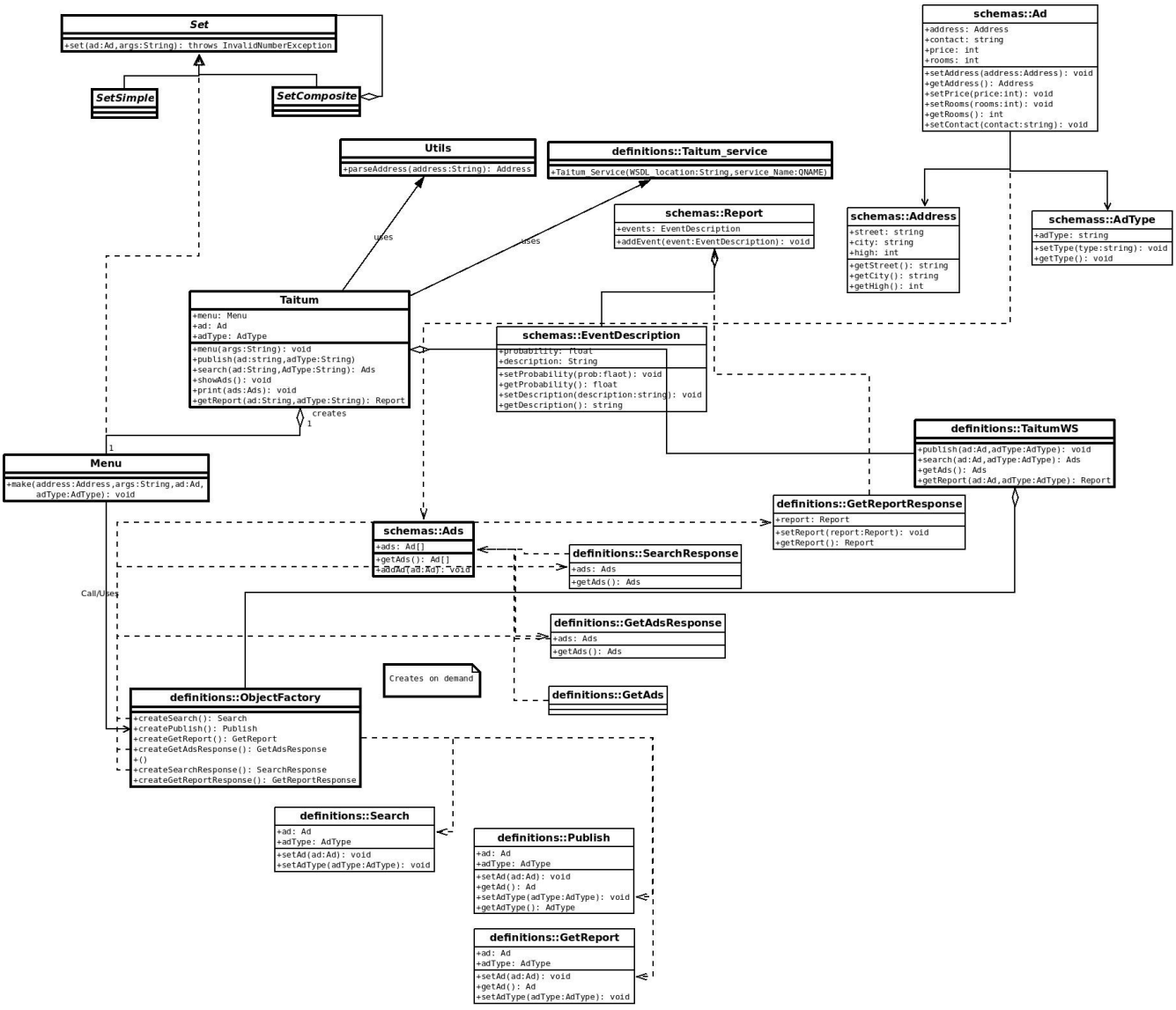


Fig. 3 Diagrama de clases del cliente de Taitumws.

9 Patrón de diseño Proxy:http://en.wikipedia.org/wiki/Proxy_pattern

10 Para consumir el servicio, se han generado esqueletos de los tipos de datos definidos en el schema del Web Service. También, aparecen las clases que permiten la creación de objectos para realizar las llamadas y comunicacion con el servicio. La clase menu toma los datos del usuario, parsea y prepara argumentos para luego invocar las operaciones del servicio.

4.Implementación utilizando el paradigma REST.

El paradigma REST es bastante diferente de SOAP. REST es una arquitectura en la cual los recursos de interés se encuentran conectados utilizando URLs; utiliza HTTP como transporte y dado que le provee de todos los datos necesarios, también como API para ultimar la funcionalidad en los extremos. Así, REST es más complejo programáticamente, pero reduce la carga de transporte. En contraste, SOAP es un protocolo de mensajes; es más sencillo de programar, pero introduce más carga en los mensajes que se transportan pues se envia un mensaje SOAP dentro de un mensaje HTTP.

4.1.Consideraciones sobre REST.

Para movernos a una implementación REST debemos considerar:

1. Identificar los recursos que participan en cada método de nuestros servicios.
2. Determinar una URL representativa para cada recurso.
3. Definir una correspondencia entre los métodos HTTP (repetando su semántica) y los métodos del servicio involucrado.
4. Precisar cúales códigos de estado del HTTP se corresponderán eventualmente con las salidas de algunas operaciones del servicio.
5. Escoger el tipo de datos adecuado que *represente* el estado actual de cada recurso involucrado.

4.2.La transición a REST.

En concordancia y orden con el apartado anterior tendríamos:

1. Los recursos que maneja el servicio son las publicaciones de ofertas o alertas de búsqueda.Estos recursos son los tipos de datos **Ad** definidos en el schema.
2. Determinar una URL que incluya la dirección de despliegue del servicio, y un nombre figurativo para las para cada publicación registrada. Ej: http://localhost:8080/Taitum**/ofertas**
3. Taitum actualmente sólo expone operaciones de búsqueda/listado y registro de nuevas publicaciones/alertas de búqueda; no hay borrados ni actualizaciones. Entonces, las operaciones de HTTP cuya semántica es coherente con las publicadas en el servicio serán únicamente GET y POST
   a.GET: búsquedas, listados y generación de reportes (operaciones idempotentes).
   b.POST: registro de nuevas ofertas y/o alertas de búsqueda (operaciones con efectos secundarios).

   En este momento, es conveniente evidenciar que el servicio debe tener acceso completo al mensaje HTTP que arriba, así podrá determinar qué método del servicio despachar en función del verbo HTTP. Por ejemplo: el caso de GET, el mapeo entre un listado completo y una búsqueda viene determinado por la query string del mensaje.
4. Cuando se intenta publicar un aviso de oferta o registrar una alerta por búsqueda respecto de algún campo en particular (calle,precio,etc.), esto asocia el método correspondiente en el server (patrón Factory Method). Ante una categoria aún no implementada, podemos lanzar una excepción HTTP con el código 405:"Method not supported"
5. La carga del mensaje HTTP puede ser representada utilizando el sistema MIME, en nuestro caso particular la elección será "**text/xml**" . Aquí, es necesario considerar que las representaciones de los recursos que se intercambian entre cliente y servidor deben ser codificados y decodificados en XML.

5.Conclusiones

La experiencia obtenida durante el desarrollo de este trabajo ah permitido comprobar la flexibilidad con la que se puede hacer converger servicios distintos en una aplicación nueva,disminuyendo el esfuerzo y tiempo de realización de la misma. Los web-services nos permiten interoperabilidad a través de plataformas y tecnologías diversas facilitando el desarrollo de sistemas totalmente heterogéneos, aprovechando las ventajas de distintos lenguajes de programación según el contexto.

En lo personal, puedo afirmar que la incursión en el área de la Computación Orientada a Servicios, me ah provisto de de conocimientos en tecnologías relativamente nuevas, y progresivamente crecientes en la informatica; estoy seguro de haber sacado partido de la materia, y poder así aprovechar estos conocimientos en mi vida profesional.

6.Apéndice.

Dentro del directorio taitumClient se encuentra el ejecutable taitum.jar .
Algunos ejemplos para ejecutar el cliente TaitumWs son:

“Ejecutar el test con la funcionalidad del web service".
$>java -jar taitum.jar -ts emailAddr

“Registrar una notificación por email cuando se publique algo: Ó en Mar del Plata, ó en Mar del Plata por la calle Juan B Justo"
$>java -jar taitum.jar -pa "Juan B Justo,1700,Mar del Plata" emailAddr

// Para disparar una notificación por la consulta anterior.
“invap@ieee.in ha publicado una oferta en Mar del Plata, en la calle Juan B Justo al 500 por $7000 con **4** ambientes. Deberíamos recibir un email."
$>java -jar taitum.jar -po "Juan B Justo,500,Mar del Plata" 7000 invap@ieee.in 4

“Registrar una notificación por email cuando se publique alguna oferta a máximo 3 cuadras de General Pinto al 2000, en Tandil"
$>java -jar taitum.jar -pd "Pinto,2000,Tandil" 3 emailAddr

// Para disparar una notificación por la consulta anterior.
"acmicpc@mit.us ha publicado una oferta en Tandil, en la calle San Martin al 1**7**60 por $9800 y **6** ambientes. debemos recibir un email."
$>java -jar taitum.jar -po "San Martin,1700,Tandil" 9800 acmicpc@mit.us 6

“Listar las ofertas que estén a maximo 6 cuadras de San Martin 1**3**00 en Tandil"
$>java -jar taitum.jar -sd "San Martin,1300,Tandil" 6

"Reporte de viviendas en San Martín, Tandil o Simplemente en Tandil. Si no hay viviendas registradas en Tandil, se entrega un reporte completo”.
$>java -jar taitum.jar -ra "San Martin,133,Tandil"

Notas (en orden de importancia):

1. *Todos los servicios se desarrollaron utilizando el framwork Apache AXIS2, JAX-WS para los clientes,en el IDE Eclipse Kepler. Al momento de importar los proyectos al IDE Eclipse, asegurese de incluir la dirección del directorio de Apache AXIS2:* **Window->Preferences->Web Services->Axis2 Preferences**.
2. *Taitumws se compone de otros servicios, es imprescindible que previo a su ejecución,el resto de los servicios ya estén listos y ejecutandos e. Se utilizaron 2 instancias de Tomcat las cuales tienen asignados los servicios (no resultó efectivo el orden de carga de los servicios impuesto en el archivo* **Servers.xml**):
    1. *Tomcat(1): Geocoderws,Emailws,Statisticsws.*
    2. *Tomcat(2):Taitumws. (8880 port).*
3. *El WSDL de Taitum indica el puerto 8880; y el cliente está construido desde esa dirección (wsimport -keep http://localhost:8880/taitumws/services/taitumws?wsdl)*
4. *EmailAddr es direccion de email en la cual recibir notificaciones, si lo desea puede utilizar: marcelojavierrodriguez@gmail.com.*