Evaluación de herramientas para el desarrollo de servicios grid

Ramos García, Vicente - Bertogna, Leandro M.
Departamento de Ciencias de la Computación, Universidad Nacional del Comahue,
Buenos Aires 1400, Neuquén, Argentina
{vramos, mlbertog}@uncoma.edu.ar

Resumen: La tecnología Grid ha logrado insertarse en el mundo científico y en estos momentos se comienzan a ver implementaciones también en el mundo de negocios, debido al proceso de evolución de esta tecnología en los últimos años. El viejo deseo de compartir recursos e información que llevo a la interconexión de equipos y aplicaciones, cobra un nuevo impulso con la posibilidad de cumplir este deseo también a nivel global, y entre distintas organizaciones en forma cooperativa y coordinada. Grid posibilita la creación de sistemas que aprovechan esta capacidad dando lugar a sistemas distribuidos que antes eran demasiado complejos, tanto desde el punto de vista de sus objetivos como la calidad de servicio con la que pueden alcanzar estas metas. En este trabajo se explora y evalúa distintas herramientas para el desarrollo de los componentes básicos de esta tecnología, los servicios grid.

Palabras Claves: Grid Services, Grid Computing
Workshop de Procesamiento Distribuido y Paralelo (WPDP)

1. Introducción

La palabra Grid fue acuñada en el ambiente académico con el fin de expresar los objetivos de sistemas computacionales distribuidos. De estos sistemas se esperaba que trabajaran bajo demanda al igual que las "mallas" o redes convencionales de agua y electricidad. Grid es una tecnología en constante cambio por estar en sus comienzos y por ello una definición resulta muy difícil, trataremos de englobar su concepto de la siguiente manera:

> *Grid es un sistema que:*
> *1) Coordina recursos que no se encuentran sujetos a un control centralizado. Esto lo hace:*
> *2) Usando protocolos e interfaces estándares, abiertas y de propósito general.*
> *3) Para brindar calidad de servicio (QoS)*

El objetivo de Grid [1] es crear la ilusión de una computadora simple pero grande y

poderosa, que se gestione a si misma. Esta computadora surge de un conjunto heterogéneo de sistemas que se conectan para compartir una gran cantidad de recursos de varios tipos. Podemos resumir diciendo que Grid busca solucionar el problema de: *compartir recursos y resolver problemas coordinadamente en organizaciones virtuales que son dinámicas y abarcan múltiples instituciones.*

Tanto los recursos como las aplicaciones clientes tienen en Grid un gran aliado estratégico. Una aplicación puede distribuir su flujo de trabajo a través de muchos recursos dispersos geográficamente y heterogéneos. Por ejemplo, por medio de un Grid, una aplicación tendrá a su disposición grandes capacidades de almacenamiento o accesos a equipos específicos como por ejemplo bombas extractoras de petróleo; mientras una parte de la aplicación trabaja con la recolección de los costos de distribución del petróleo, otra puede estar obteniendo el total extraído en distintos pozos en tiempo real.

Cada vez resulta más claro que para poder aprovechar al máximo las capacidades de los Grid de cómputo y datos, es necesario que las aplicaciones hagan uso de los servicios que brinda. Para ello muchas veces es necesario transformarlas; pero los costos asociados con estas transformaciones pueden hacer que los beneficios se vean opacados. Sin embargo con el tiempo y los nuevos desarrollos puede ser fundamental para la estrategia del negocio que los servicios prestados se encuentren disponibles para los clientes y los socios de manera estándar y QoS, de modo que no se necesite una tecnología determinada para poder acceder a ellos. Tal vez la manera más eficiente de obtener los beneficios de Grid sin estos costos sea que las nuevas aplicaciones se desarrollen desde un principio como “servicios grid”. Para esto se debe evaluar las herramientas existentes que ayuden y faciliten el desarrollo de dichas aplicaciones.

En el siguiente capítulo se realiza una breve reseña de las tecnologías y elementos principales del presente trabajo. El tercer capítulo presenta el ejemplo de servicio grid con el que se realizaran las experiencias. En el cuarto capítulo se encuentran los comentarios de las evaluaciones con las herramientas de desarrollo y por último se encuentran las conclusiones de las evaluaciones realizadas.

2. Servicios Grid

Una de las tecnologías de Grid más difundida es el conjunto de herramientas de la alianza Globus [2], también conocido como *Globus Toolkit* en su versión 4.0 (GT4). GT4 es un conjunto de componentes de software que en su mayoría implementan servicios web como mecanismo para construir sistemas distribuidos.

GT4 permite la construcción de dos tipos de aplicaciones:

- *Aplicaciones orientadas a los servicios*: Son aquellas que se construyen a través de la ensamblaje de componentes definidos por una interfase de servicio.
- *Infraestructura orientada a los servicios*: Es la gestión de recursos y mecanismos de abastecimiento usados para alcanzar objetivos de calidad de servicio para componentes y

aplicaciones.

Los servicios web son sistemas de software diseñados para soportar una interacción entre máquinas sobre una red. La forma de lograr esto es publicando su interfase como metadatos a través de los llamados documentos WSDL, e implementando la comunicación por medio del intercambio de mensajes estándares y abiertos. Para esto último se utiliza el protocolo SOAP (simple object access protocol), y para el protocolo de transporte comúnmente se usa HTTP.

En la implementación del servicio web se distinguen dos conceptos importantes:

- Contenedor: Es quien recibe los mensajes, identifica e invoca el código apropiado que procesará el mensaje. Puede llegar a proveer funciones de administración, y está compuesto por los siguiente elementos:
  - Servidor SOAP: Es quien entiende como procesar los mensajes de requerimiento y respuestas.
  - Servidor de Aplicación: Componente de software que provee el "ambiente" a las aplicaciones en nuestro caso los servicios web que deben ser accedidos por los clientes.
  - Servidor HTTP: Es quien conoce como manejar mensajes interpretar los mensajes de dicho protocolo.

  En forma de ecuación sería:

  Contenedor = Servidor SOAP + Servidor de Aplicación + Servidor HTTP.

  Un contenedor para GT4 sería:

  Contenedor GT4 = Axis[3] + Jakarta Tomcat[4] + Servidor HTTP
- Servicio Web: Código que maneja los mensajes y genera las respuestas.

Sin embargo los servicios web "planos" [5] tienen ciertas limitaciones que no los hacen adecuados para la construcción de aplicaciones Grid. Si bien en la arquitectura de servicios web no se menciona que estos no poseen estados, la realidad es que generalmente los servicios web no los poseen. Esto significa que no pueden "recordar" información o guardar estado entre distintas invocaciones.

Para solucionar esto se utiliza una entidad independiente denominada recurso, esta es responsable de guardar la información de estado. Cada recurso tiene una clave única, de manera que cuando se requiera una interacción manteniendo el estado con un servicio web, se lo instruye para que use un recurso en particular. Por otro lado toda la gestión de los recursos se hace a través del *resource home*. El *resource home* es el encargado de conocer todos los recursos existentes y de realizar todas las gestiones de mantenimiento y acceso. Cuando un servicio web solicita un recurso por medio de una clave, el servicio web debe solicitar al *resource home* una referencia al recurso en cuestión. El par servicio web y recurso se lo denomina WS-Resource [6] y en este documento también se lo denomina servicio grid.

3. Diseño de Servicios Grid

Para este trabajo se pensó en un servicio grid con el que se pudieran probar los conceptos más importantes. La idea surgió con una propuesta del sector productivo de la región con el fin de lograr una prueba de concepto y factibilidad de esta tecnología en áreas de complejos requerimientos tecnológicos.

El servicio grid simulará la interacción con pozos petroleros. Si tenemos en cuenta que los pozos petroleros se encuentran dispersos a través de una basta superficie geográfica, y que ellos poseen algún tipo de comunicación, podemos interpretar a cada uno como un nodo productivo conformando una gran red. Cada nodo productivo ofrecerá un servicio por medio del cual se podrá obtener información y solicitar la realización de distintas operaciones: apagado de bomba, encendido de bomba, apertura de llave, cerrado de llave. Como cada pozo petrolero puede tener una o más bombas de extracción, entonces cada servicio ofrecido por un nodo productivo puede tener uno o más recursos asociados.

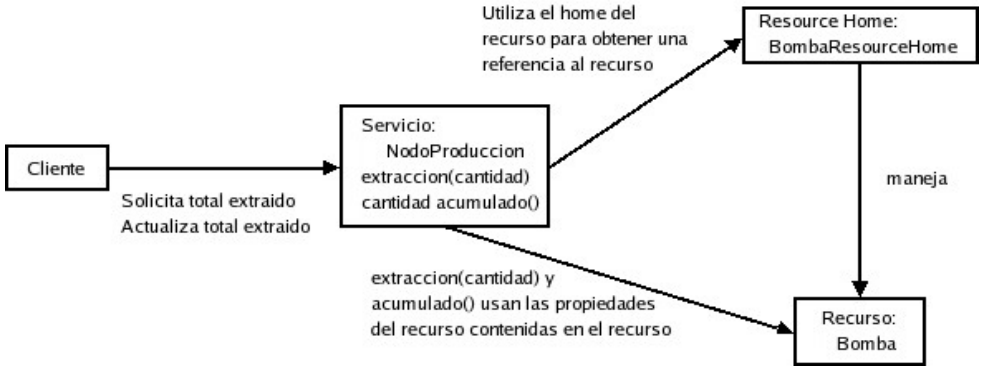

Figura 1

Cada recurso tendrá una propiedad que será el total acumulado de las extracciones realizadas por esa bomba. Por medio de esta propiedad se podrá ver una de las características más importantes que diferencia a los servicios grid de los servicios web: los primeros tienen estado. Un esquema de este diseño se puede observar en la figura 1.

Cuando un recurso es creado, el valor de la propiedad del recurso se inicia con valor cero. La operación de *agregar una cantidad extraída (extracción)* espera como parámetro un entero. Este parámetro es sumado al valor del total acumulado. Para obtener el valor del total acumulado se hace uso de la operación *obtener total extraído (acumulado)* que no espera ningún parámetro pero que devuelve un entero representando el total extraído hasta ese momento.

Cada pozo puede estar compuesto por una o más bombas. Esto se implementará de modo que cada *nodo productivo* será un *contenedor* donde se ejecutará el *servicio grid*. Dentro de este *contenedor* existirá un recurso por cada bomba del pozo petrolero. Se implementará un *servicio grid* con un único recurso utilizando un patrón *singleton [7]*. A pesar de que exista un único recurso se manejará un *resource home* y el *resource* (recurso) como clases separadas para que se pueda identificar el papel de cada uno de estos elementos y como interactúan entre sí.

Un diagrama de secuencia simplificado de la interacción entre el cliente, sevicio, resource

home y el recurso se puede observar en la figura 2.

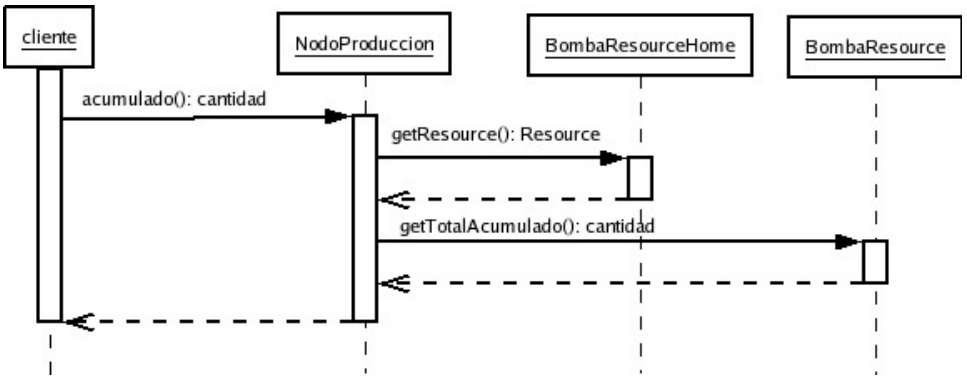

Figura 2

4. Implementaciones

En esta sección describiremos los distintos procesos a los que se ven sometidos los servicios grid durante su desarrollo como: creación, construcción y despliegue. En este caso los elementos provistos por el desarrollador serán: la interfase del servicio (documento WSDL), la implementación del servicio y sus recursos (clases java), el descriptor de despliegue (documento WSDD), y el descriptor de nombres y directorios de interfaces (JNDI).

Para el desarrollo de la evaluación se utilizó la plataforma Eclipse [8], esta es una herramienta de código abierto. Eclipse es una plataforma universal para el desarrollo de aplicaciones ya que da la libertad de elegir distintos lenguajes, plataformas y entornos de programación. También provee un *framework* basado en *plugins* que permite la creación, integración y utilización de herramientas de software. Provee componentes y una base para la construcción y ejecución de herramientas de desarrollo de software integradas. Además esta plataforma esta desarrollada en el lenguaje Java que facilita su ejecución sobre múltiples plataformas.

El primer método de desarrollo que se realizó es el de implementación manual, es decir sin asistentes automáticos. En este caso se busca realizar el proceso de la forma más elemental, llevando a cabo cada uno de los pasos involucrados en el desarrollo de un servicio grid. El segundo método utiliza un plugin para el manejo del contenedor. Esto permitirá explorar las ventajas de trabajar en un entrono de desarrollo integrado (IDE) para la creación de un servicio grid y los beneficios de poder integrar al IDE el despliegue de estos en el contenedor. El tercer caso se usa la plataforma Eclipse, sumando al IDE capacidad para el desarrollo de servicios grid Proyecto Globus Service Build Tool (GSBT) [9] en lugar de integrar el manejo del contenedor.

A continuación se expresan los cinco pasos básicos que componen el ciclo de desarrollo de un servicio grid. Estos pasos siguen la metodología utilizada en el documento de GT4 [10] y el patrón se puede observar casi de la misma manera en los tres casos de prueba, excepto en el último, donde la generación del documento WSDL es realizada automáticamente.

- *Definición de la interfase WSDL*

La primera tarea que se debe realizar al implementar un servicio grid es definir que interfaz tendrá. Esto se hace creando un documento WSDL. A través de este documento se informa cuales son los servicios ofrecidos. Por cada servicio se indica cuales son las operaciones disponibles, y que mensajes se utilizan para la invocación de dichas operaciones y para el retorno de los resultados. Los mensajes de las operaciones están compuestos por ningún, uno o varios parámetros de tipos simples o compuestos. Todo esto también se encuentra definido en el documento WSDL. El estándar de WSDL llama a la interfase del servicio como *PortType* (tipo de puerto). Es decir, un *PortType* está compuesto por las operaciones del servicio en cuestión, las cuales a su vez lo están por mensajes, y finalmente estos están compuestos por parámetros simples o complejos.

- *Implementación del servicio en Java.*

En la implementación que se realiza aquí se mantiene el código del servicio separado del código del recurso para destacar que los recursos y el servicio son conceptos independientes, pero que trabajan en forma cooperativa para implementar el servicio con estado. El siguiente paso es la implementación del servicio en Java. En el primer y segundo caso se implementan tres clases en sus correspondientes archivos:

`BombaResource.java:` En esta clase se encuentra el código que implementa el recurso: la bomba extractora. Aquí se implementan las propiedades necesarias para mantener el estado.

`BombaResourceHome.java:` Esta clase se encarga de la gestión de los recursos. Todo aquel servicio que quiere interactuar con el recurso lo debe hacer a través de su *home*. El servicio web (sin estado) utiliza el *home* para obtener el recurso (que tiene estado) y así poderlo manipular convirtiéndose en un servicio grid con estado: WS-Resource.

`NodoProduccion.java:` Esta es la clase que implementa el servicio grid.

`NodoQName.java:` Esta interfase tiene como único propósito el de simplificar el manejo de los espacios de nombre dentro de las clases.

- *Configuración del despliegue*

En el tercer paso se deben armar los archivos necesarios para instruir al contenedor como se deben publicar los servicios y como se utilizaran los recursos. Para esto se utilizan dos archivos:

`deploy-jndi-config.xml`: Responsable de especificar cual es el *resource home* que el servicio debe utilizar para tener acceso al o los recursos, y los parámetros relacionados con como el *resource home* maneja los recursos.

`deploy-server.wsdd`: Es el encargado de indicarle al contenedor como debe publicar el servicio. Por ejemplo, entre otras cosas, especifica con que URI debe ser publicado el servicio. A este archivo se lo suele denominar *deployment descriptor.*

- *Creación del archivo grid (GAR)*

En este paso se crea un único archivo que contiene a todos los elementos que se necesitan

para que un contenedor ponga a disposición del resto del mundo el servicio grid. Estos elementos van desde las clases compiladas y stubs hasta información de como el contenedor debe desplegar el archivo en cuestión.

- *Despliegue del servicio en un contenedor*

Un contenedor puede albergar uno o más servicios. El termino *desplegar* se refiere a la tarea de instalar e inicializar la ejecución de un servicio en un contenedor en particular. Este proceso es realizado con una herramienta del GT4 que utilizando la herramienta Ant descomprime el archivo con extensión *gar* y copia los archivos dentro de la instalación del contenedor.

4.1 Implementación en Forma Manual

En esta sección se describe la implementación del servicio grid *Nodo Productivo* de manera manual. Denominamos *manera manual* debido a que en las siguientes secciones se verán formas distintas de implementar el mismo servicio grid por medio de herramientas que ayudan y automatizan dicho proceso.

En esta implementación para el servicio NodoProduccion se definió un PortType denominado **NodoProduccionPortType**. Este *PortType* está compuesto por las dos operaciones del servicio: extracción y acumulado.

También se necesita crear las clases *stubs* con las cuales se comunica nuestro servicio para recibir requerimientos y devolver resultados. Estas clases son generadas desde el documento WSDL utilizando una herramienta del GT4. El archivo **Namespace2package.properties** indica a la herramienta de GT4 donde ubicar dichos *stubs*.

Para realizar el GAR se utiliza el script **globus-build-service** y los archivos que a continuación se detallan:

- **`globus-build-service.sh`**: Encargado de comenzar la ejecución de todos los pasos que construyen el archivo GAR. Este script hace uso de las herramientas provistas por GT4 y del Ant que es una herramienta de construcción para Java.
- **`build.xml`**: Este archivo también es parte del GSBT y es utilizado por el script anterior. La mayor importancia de este archivo es que evita tener que escribir un build.xml cada vez que utilicemos el Ant con un proyecto.
- **`build.mappings`**: En este caso el archivo es opcional y su único propósito es reducir la cantidad de parámetros que deben ser pasados al script globus-build-service.sh

  Finalmente para la formación del archivo *gar* se ejecuta el script **`globus-build-service.sh`** indicando como parámetro el nombre del servicio que se encuentra en el archivo **`build.mappings`**. Como salida de este proceso resulta el archivo con extensión *gar* .

Para el despliegue del archivo GAR se utiliza una herramienta provista por GT4. Se debe ejecutar el comando **`globus-deploy-gar`** usando como parámetro el archivo GAR del servicio. Al finalizar el contenedor tendrá todos los elementos e información necesarios para publicar el

servicio.

4.2 Implementación con Eclipse y Tomcat

En esta sección se explica como se realizó la implementación del servicio *NodoProduccion* en el IDE Eclipse utilizando un plugin que permite manejar e interactuar con el servidor Tomcat desde la plataforma Eclipse. Para este método se utilizaron las versiones 3.0.2 y 3.1 de Eclipse.

Este método se puede dividir en dos pasos. El primer paso es la configuración del Eclipse y el plugin de Tomcat, el segundo el armado del proyecto, compilación, despliegue, y depuración.

El primer paso se hace una única vez y se reutiliza la configuración para todos los desarrollos posteriores. Las tareas que se realizan durante este paso son:

- Configurar el servidor Tomcat en el sistema.
- Instalar y configurar el plugin *Sysdeo Tomcat Launcher* en la plataforma Eclipse.
- Instalar las herramientas del proyecto GSBT.
- Crear en el Eclipse una librería de usuario llamada GT4 que contenta a todos los archivos JAR con las librerías de Globus.

  La estructura de directorios y los archivos necesarios son casi idénticos a los utilizados durante el desarrollo manual.

  El segundo paso se puede dividir a su vez en tres etapas:

- Creación y configuración de un nuevo proyecto son: dentro de Eclipse se crea un nuevo proyecto y se lo vincula al plugin de Tomcat, esto hace que el servidor Tomcat ejecute los archivos con extensión *class* del proyecto cuando esto se hayan compilado. Agregar el código fuente del proyecto al entorno del servidor Tomcat, permite que el depurador tome la última versión de fuentes cuando este sea invocado, y finalmente se incluye en el proyecto las clases del GT4.
- En las tareas de desarrollo necesarias para el proyecto se realiza: La codificación de las clases que implementarán el servicio y sus recursos. La configuración del destino del proceso de compilación. Esto hace que los archivos con extensión *class* generados por Eclipse durante la compilación del código fuente sean ubicados en un directorio especial. Y la inclusión de otros archivos del proyecto son WSDD y JNDI.
- Para realizar la compilación y el despliegue del servicio, se realizan las siguiente tareas: Se crea una configuración de lanzamiento, esto hace que se invoque la ejecución de la herramienta Ant y se realice las tareas según lo instruya el archivo **`buildservice.xml`**. Y para realizar la compilación, la construcción del archivo GAR y su despliegue solo basta con presionar el botón creado a tal fin durante la configuración de lanzamiento. El proceso que se lanza no solo compilará los archivos fuentes, sino que creará el archivo GAR y realizará las tareas necesarias para que el servidor Tomcat publique los servicios grid registrados.

Una vez realizada la construcción y el despliegue se puede agregar al camino de librerías del proyecto los *stubs* que recién se acaban de crear.

- Los siguientes pasos tienen como objetivo ejecutar el servidor Tomcat, compilar el cliente y realizar una prueba. Para ejecutar este servidor y todos los servicios registrados solo se debe hacer clic en el icono de lanzamiento. Para compilar y utilizar el cliente, se ubica su código fuente en un directorio creado a tal fin, y si el servidor Tomcat está ejecutando, al compilar y correr el cliente se verá desde la plataforma Eclipse el resultados de esta ejecución.
- Para realizar depuración solo se debe agregar los puntos de quiebre en el código del servicio grid y ejecutar el cliente, esto hará que cuando se llegue a esos punto durante la ejecución del servicio grid (que a su ves es invocada por el cliente) el Eclipse realice un cambio a la vista de depuración y nos permita depurar el servicio grid.
- Todas las correcciones y cambios que se realicen en el servicio grid se verán reflejados inmediatamente sobre servicio en ejecución. Esto se debe a que en la plataforma Eclipse cada vez que se realiza una modificación al código fuente automáticamente se compila, envía un evento al plugin *Sysdeo Tomcat* para realizar las actualizaciones pertinentes.

4.3 Implementación con Eclipse y GT4IDE plugin

Esta sección desarrolla la utilización del plugin de Eclipse creado por el proyecto GSBT. El propósito de este plugin es que Eclipse realice una asistencia en el desarrollo de proyectos Grid (GT4), hasta aquí se utilizo la plataforma solo como herramienta integradora junto al contenedor.

Al igual que en la sección anterior se debe configurar el plugin, pero a diferencia del anterior en este caso la configuración consta de un solo paso, indicar al plugin donde se encuentra el Globus Toolkit.

Durante la creación de un proyecto se ingresa el nombre del proyecto y se da la opción de crear el primer servicio. Al optar por esto, se presenta una segunda pantalla donde se ingresan la información básica del servicio grid:

- Paquete base: donde se ubican los archivos de implementación.
- Base del espacio de nombres: Es la dirección URI que servirá como base de todos los espacios de nombres que se utilizaran.
- Nombre del *Port Type*: Aquí se ingresa el nombre del servicio, al final se le concatenará la cadena "*PortType*".
- Camino del servicio: Es la cadena que se concatenará al URL.
- Patrón de diseño: Con el fin de mantener las experiencias semejantes para la creación del servicio se selecciona "*Singleton with ServiceResourceHome*" como patrón de diseño.
- Proveer un esqueleto a la implementación: El plugin ofrece la posibilidad de que se genere automáticamente un conjunto de archivos con código básico, luego solo habrá que codificar los métodos específicos del servicio.

Una vez creado el proyecto y los servicios se puede pasar a la siguiente etapa que es la implementación de métodos. Esto se hace creando métodos privados a los que se le antepone el prefijo “do_”, al solicitar la actualización del documento WSDL, una acción provista por el plugin, automáticamente se crea un método que está vinculado con la operación del documento WSDL. Esta operación de actualizar el documento WSDL también realiza el manejo de los parámetros si estos son de los tipos básicos de Java.

Finalmente para generar los *stubs* y el archivo GAR solo hace falta hacer un clic en los botones correspondientes, en la figura tres podemos ver una pantalla de ejemplo y estos botones se encuentran ubicados en los últimos lugares de la barra de herramientas.

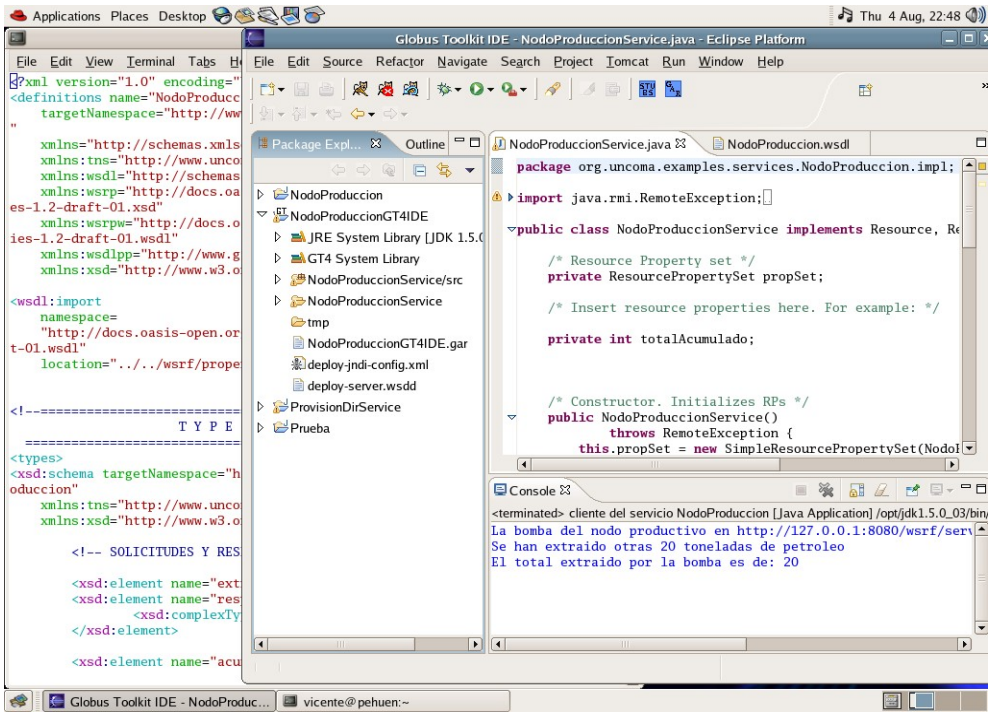

Figura 3

5. Conclusiones

En el primer método las herramientas GSBT nos permite librarnos de la tediosa tarea de crear y configurar los archivos que se necesitan para desarrollar y desplegar un servicio grid. Sin embargo este método resultó ser muy limitado. No hay beneficios palpables en la implementación de servicios grid de esta manera. Aun ante la necesidad de desarrollar servicios grid de pequeña escala este método resulta complejo y factible a inducir a errores, así como también carece de facilidades para la compilación y/o el despliegue.

En el caso de la implementación con Eclipse y plugin *Sysdeo Tomcat* inicialmente se debe realizar toda una serie de tareas que nada tienen que ver con el desarrollo de un servicio grid y es la configuración del IDE Eclipse. Pero es importante destacar que esto no afecta las conclusiones relacionadas con esta forma de desarrollar y desplegar servicios grid ya que toda esa tarea se realiza una única vez. Una de las ventajas de este plugin es tener en línea al servidor Tomcat y que las

modificaciones realizadas al servicio puedan estar inmediatamente publicadas en el contenedor. Otro beneficio es que las tareas de compilación y despliegue del servicio son simplificados al realizarse en un solo paso. Por último lo más destacable de la utilización del plugin *Sysdeo Tomcat* es que permite hacer depuración y evaluación rápida y sencilla del impacto que hayan tenido las modificaciones de los servicios grid publicados por el servidor.

Por otro lado, presta poca asistencia en el desarrollo servicios grid. Otro punto en contra es que solo para windows se lograr una correcta configuración, en plataformas Unix existe conflicto con los permisos del sistema de archivos ya que generalmente el rol del desarrollo no coincide con el de administración de Globus o el servidor Tomcat y este plugin necesita una combinación de los tres para funcionar correctamente. Finalmente cabe destacar que actualmente existen problemas para ejecutar el GT4 y el Tomcat, estos problemas serán resueltos en próximas versiones.

En el caso de utilizar la Plataforma Eclipse con el plugin GT4IDE la configuración es muy simple, solo un paso. Y la creación de la estructura de archivos y directorios está integrada en la creación del proyecto, es decir, se realiza automáticamente. Se destaca la facilidad para crear un servicio nuevo; pero la ayuda que pueda brindar al momento de la creación de servicios complejos puede ser poca o nula. Crear automáticamente los métodos remotos a partir de las definiciones de métodos privados resulta en un gran avance liberando al desarrollador de la necesidad de mantener sincronizado el documento WSDL del servicio grid, sin embargo es muy limitado el manejo de parámetros. Otro detalle es que si bien automatiza el manejo del documento WSDL para las operaciones, no lo hace para las propiedades de los recursos y tampoco colabora en la generación del código de las propiedades de los recursos, estas tareas deben codificarse en forma manual.

	Configuración	Desarrollo	Despliegue	Depuración
Manual	Nula	Compleja	Compleja	Compleja
Plugin Sysdeo-Tomcat	Compleja, pero una sola vez	Complejo	Sencillo	Sencillo
GT4 IDE	Sencilla	Sencillo con limitaciones	Complejo	Complejo

Tabla 1

Poder generar los stubs y el archivo GAR sin problemas de permisos ni tener que modificar la seguridad del equipo, hace que este plugin sea más práctico que la utilización del *Sysdeo Tomcat*. La utilización de patrones durante la creación del servicio y la poca disponibilidad de opciones hace que sean muy limitadas la cantidad de arquitecturas de implementación que se pueden utilizar. Aunque una vez generado el código inicial se puede modificar el mismo en forma manual. Una gran desventaja de este plugin respecto del *Sysdeo Tomcat* es que carece de facilidades para desplegar y publicar el servicio. Finalmente se destaca que no tiene ninguna forma de ayudar en la depuración del servicio.

Como conclusiones finales sería muy interesante si se pudiera integrar el plugin GT4IDE con el *Sysdeo Tomcat*, esto daría muchos beneficios, el primero en la generación del servicio y el

segundo en el despliegue y depuración. Si bien el GT4IDE parece ser de poca ayuda, es clara la necesidad de crear un IDE capaz de desarrollar servicios grid y solo es una cuestión de tiempo para que sea una alternativa valida al momento de desarrollar aplicaciones grid. Es de suma importancia que las nuevas versiones del plugin GT4IDE ayuden en el manejo de las propiedades de los recursos por lo menos brindando este servicio para los tipos básicos de java.

En la tabla 1 se puede observar un resumen de las características del desarrollo de los servicios y la complejidad para los métodos evaluados

6. Trabajo Futuro

Una vez concluida las tareas de evaluación de herramientas y desarrollo de servicios grid, el siguiente paso es acceder a estos servicios a través de un portal Grid. Actualmente se encuentra en evaluación los portales orientados a grid: GridSphere Portal[11], Gridport[12] y de uso general Jetspeed[13], todos estos basados en portlets.

Referencias

[1] I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001.

[2] A Globus Toolkit Primer. I. Foster, 2005.

[3] Web Services Axis.http://ws.apache.org/axis/java/architecture-guide.html

[4] Jakarta Tomcat.http://jakarta.apache.org/tomcat/tomcat-5.5-doc/index.html

[5] W3C http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

[6] WS-Resource Specification.http://www-128.ibm.com/developerworks/webservices/library/specification/ws-resource/ws-modelingresources.html

[7] Gamma, E., Helm, R., Johnson, R. & Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Reading.1995.

[8] Gamma, E. and K. Beck, “Contributing to Eclipse: Principles, Patterns, and Plug-ins,” Addison Wesley Longman, 2003.

[9] Proyecto Globus Service Build Tool (http://gsbt.sourceforge.net/)

[10] GT4 Tutorial (http://gdp.globus.org/gt4-tutorial/multiplehtml/index.html)

[11] GridLab. The gridsphere portal. http://www.gridsphere.org/gridsphere/gridsphere.

[12] Thomas, M. P., Mock, S., Dahan, M., Mueller, K., Sutton, D., The GridPort Toolkit: a System for Building Grid Portals. Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing, August, 2001.

[13] Apache Jetspeed open-source portal serve: http://jakarta.apache.org/jetspeed.