UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

DISEÑO DE UN MODELO PARA UN SERVICIO MANEJADOR DE OBJETOS PERSISTENTES EN GRIDS COMPUTACIONALES BASADOS EN JAVA.

JULIO CÉSAR VÉLIZ SIRA

Barquisimeto, 2004

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
MENCIÓN INGENIERÍA DEL SOFTWARE

DISEÑO DE UN MODELO PARA UN SERVICIO MANEJADOR DE OBJETOS PERSISTENTES EN GRIDS COMPUTACIONALES BASADOS EN JAVA.

Trabajo presentado para optar al grado de
Magíster Scientiarum

Por: JULIO CÉSAR VÉLIZ SIRA

Barquisimeto, 2004

DISEÑO DE UN MODELO PARA UN SERVICIO MANEJADOR DE OBJETOS PERSISTENTES EN GRIDS COMPUTACIONALES BASADOS EN JAVA.

Por: JULIO CÉSAR VÉLIZ SIRA

Trabajo de grado aprobado

Yudith Cardinale Tutor	Niriaska Perozo Jurado

Euvis Piña
Jurado

Barquisimeto, 31 de Julio de 2004

A la memoria de mi Bebé:

Llegaste a nuestras vidas
como una estrella fugaz, hermosa y llena de luz.
Fuiste como un pequeño ángel que trajo a nuestro hogar
buenas nuevas; las primicias de las bendiciones
que Dios nos dará en tus hermanos y hermanas.

¡Te extrañamos!

Agradecimiento

A Dios, mi padre celestial y amigo, por darme el don de la vida, llenarme de su sabiduría y ayudarme en el desarrollo de este trabajo.

A mi esposa por su amor, comprensión y apoyo en todo momento. Oly Te amo!

A mis padres, hermanos y familiares en especial a mis sobrinas Andrea y Alejandra por llenarme de ternura y ser un motivo de inspiración.

A mis hermanos espirituales por sus oraciones.

A mis compañeros de trabajo por su paciencia en los momentos difíciles.

A mis amigos, por toda su colaboración y ayuda desinteresada e incondicional.

Al Grupo de Sistemas Paralelos y Distribuidos de la USB, por darme la oportunidad de trabajar con ellos en el proyecto SUMA, en especial a mi tutora Yudith.

INDICE

ÍNDICE DE CUADROS

ÍNDICE DE GRÁFICOS

ÍNDICE DE FIGURAS

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
MENCIÓN INGENIERÍA DEL SOFTWARE

DISEÑO DE UN MODELO PARA UN SERVICIO MANEJADOR DE OBJETOS PERSISTENTES EN GRIDS COMPUTACIONALES BASADOS EN JAVA.

Autor: Ing. Julio César Véliz Sira
Tutora: Prof. Yudith Cardinale

RESUMEN

Los Grids Computacionales han sido propuestos, con la finalidad de proveer mecanismos que permitan compartir de manera transparente, diversos recursos heterogéneos, distribuidos geográficamente e interconectados a través de redes de alta velocidad. Con el establecimiento del enfoque orientado a objeto (OO) en la Ingeniería del Software, han proliferado implementaciones de Grids computacionales OO. Estos albergan en su seno, diversos objetos tanto transitorios como persistentes, que brindan el soporte necesario a la ejecución de los servicios ofrecidos por el Grid.

Este trabajo se centra en el estudio del manejo de la persistencia de objetos, en implementaciones de infraestructuras para Grids Computacionales basados en Java. Se hace un especial énfasis en los sistemas de almacenamiento que serán compartidos en el Grid, tales como: Sistemas de Archivos, Sistemas de Gestión de Base de Datos Relacional y Sistemas de gestión de Bases de Datos OO, con el propósito de diseñar un modelo para un servicio que permita administrar los objetos persistentes de manera uniforme y transparente superando así, los problemas de Interoperabilidad entre los recursos y Desadaptación de Impedancias Objeto-Relacional.

Se desarrolló un prototipo funcional del servicio que apoya la verificación del diseño. La información de los recursos disponibles en el Grid, fueron tomados del contexto de algunas Instituciones miembro de la Red Académica de Centros de Investigación y Universidades Nacionales (Reacciun, 2004). Además, se ha planteado el uso de un ambiente de simulación para obtener datos estadísticos.

Palabras Clave: Grid Computacional, Persistencia de Objetos, Java, Ingeniería del Software, Base de Objetos.

INTRODUCCIÓN

Los Grids Computacionales conforman una plataforma distribuida que está teniendo auge en los últimos tiempos. Estos sistemas han sido propuestos con la finalidad de proveer algunos mecanismos, que permitan compartir de manera transparente, diversos recursos heterogéneos, distribuidos geográficamente e interconectados a través de redes de alta velocidad. Estos recursos pueden ser de diverso tipo, pero esta investigación se concentra en los recursos computacionales y de almacenamiento, en especial, recursos de almacenamiento soportados por otros sistemas, tales como: Sistemas de Archivos, Sistemas de Bases de Datos Relacional y Sistemas de Base de Datos OO.

Estos sistemas de almacenamiento han sido diseñados con la finalidad de administrar la información, como recurso, perteneciente a los Dominios Administrativos de las Instituciones participantes en el Grid, siguiendo las políticas de almacenamiento propias de cada Dominio. El término Dominio Administrativo, se refiere a una colección de recursos, entre otros: redes, computadoras y bases de datos, que son gestionados en conjunto, presentando características administrativas comunes, por ejemplo, las políticas de seguridad encontradas en una Intranet de cualquier organización. Este es el gran reto que representa el desarrollo de una infraestructura del Grid: Administrar una cantidad de recursos de almacenamiento heterogéneos como si se tratase de un componente uniforme.

Este trabajo se concentra en el manejo de la persistencia de los objetos en plataformas distribuidas, especialmente enfocándose en los Grids Computacionales basados en Java. Por lo tanto, se estudian las diferentes estrategias empleadas en cada Dominio Administrativo para hacer persistentes los objetos Java en sus sistemas y se busca una manera general o estándar de integrar la administración de estos objetos, poniéndola a disposición de los diferentes componentes del Grid, como un servicio

del mismo, respetando las políticas de almacenamiento en cada Dominio, pero aprovechando, hasta donde sea posible, los principios del paradigma de orientación a objetos. Con esta finalidad, se propone un modelo para un servicio manejador de objetos persistentes en Grids Computacionales basados en Java. Los objetos persistentes representan a los recursos disponibles en el Grid y para los efectos de la validación del modelo, se utiliza la información de estos recursos: Instituciones, Dominios, Almacenes de Objetos, Recursos de Cómputo, Servicios, Usuarios y Roles, entre otros, basados en el contexto de algunas Universidades a nivel nacional, pertenecientes a la Red REACCIUN (Reacciun, 2004).

Este trabajo se ha organizado en cuatro capítulos: El primer capítulo se titula *el problema*, en donde se describe la situación del objeto de estudio, se plantean las metas u *objetivos* trazados en la investigación, se justifica el estudio, se explica la *importancia* del mismo y se establecen los alcances y limitaciones del estudio.

El segundo Capítulo se titula *Marco Teórico*, el cual se presenta como el compendio de una serie de elementos conceptuales que sirven de base a la investigación que se va a realizar, se definen los *antecedentes* de la investigación y además un conjunto de conceptos y proposiciones dirigido a explicar el problema planteado.

El tercer capítulo se titula *Marco Metodológico*, en donde se especifica la *modalidad de la investigación* y la metodología que se empleará para el desarrollo del proyecto. Allí, se revisa el proceso de diseño propuesto, denominado RUP.

El cuarto capítulo se titula *Modelo del Servicio de Objetos Persistentes en Grid Computacionales*, este capítulo representa la Propuesta del Estudio y en el mismo se obtiene el Diseño del Servicio, por medio de RUP.

El quinto capítulo se titula *Implementación del Modelo SMOP*, en el cual se presenta la descripción de un Prototipo Funcional del Servicio y se realiza la demostración del mismo, obteniendo algunos indicadores como resultado.

Finalmente el sexto capítulo se titula *Conclusiones y Recomendaciones*, en donde se expresan las ideas finales del autor en relación a la investigación realizada.

CAPÍTULO I

EL PROBLEMA

Planteamiento del Problema

Los Sistemas Grid surgen con la finalidad de poder compartir, en forma transparente, los recursos disponibles que se encuentran interconectados por medio de una plataforma distribuida de red de alta velocidad. Entre los recursos más importantes compartidos en estos sistemas, se encuentran: los computacionales y los de almacenamiento. Desde el punto de vista de los recursos computacionales, es pertinente mencionar una de las categorías de los sistemas Grid denominada Grid Computacional, la cual es definida como una infraestructura de hardware y software que proporciona acceso dependiente, consistente, generalizado y económico a capacidades computacionales de alto rendimiento. Por otra parte, desde el punto de vista de almacenamiento, existe otra categoría denominada Grid de Datos, la cual provee una infraestructura que permite sintetizar nueva información a partir de repositorios de datos, tales como bibliotecas digitales o almacenes de datos.

Aunque el énfasis de los Grid computacionales está en el poder de cómputo, este tipo de Grid también necesita proveerse de un servicio de datos. La mayor diferencia en este sentido, entre un Grid Computacional y un Grid de Datos, es la infraestructura especializada provista por el Grid de Datos para la administración (almacenamiento y recuperación) de los datos.

Cabe destacar que los recursos que participan en la formación del Grid no se encuentran en la red de forma aislada o aleatoria, sino que forman parte de los activos de alguna Institución u Organización. Por lo tanto, para facilitar el acceso compartido de sus recursos, es necesario que dichas instituciones se sometan voluntariamente al

cumplimiento de algunas reglas mínimas de cooperación. El conjunto de usuarios e instituciones comprometidos en esta actividad, conforman la llamada Organización Virtual.

El éxito de un proyecto de este tipo dependerá entonces, por una parte, de la participación masiva de instituciones que deseen integrarse al Grid, contando de esa manera con una Organización Virtual dinámica y multi-institucional, que comparta sus recursos generalmente heterogéneos y geográficamente distribuidos, de manera coordinada y altamente controlada a través del Grid; y por la otra, que la integración se realice teniendo cierto grado de independencia en el manejo de los recursos, desde el punto de vista local e individual, para cada institución participante. De lo anterior se deriva una serie de aspectos que son indispensables en todo Grid, los cuales mencionamos a continuación: (a) Coordinación de recursos no sujetos a un control centralizado; (b) Uso de protocolos abiertos e interfaces estándares y de propósito general y (c) Obtención de Calidad de Servicio (QoS).

Algunas Instituciones promotoras de un Grid, podrían tratar de persuadir a otras para que participen en el mismo, resaltando los beneficios que obtendrían en esta integración de recursos, pero también haciendo los esfuerzos necesarios para tratar de disminuir, a la mínima expresión, el impacto que puedan ocasionar las reglas de cooperación impuestas por el Grid; es decir, los cambios que deba realizar dicha institución (la candidata) en sus políticas de uso y manejo de los recursos que se encuentran bajo su responsabilidad y resguardo. Estos recursos son desplegados naturalmente en la plataforma o infraestructura interna de la institución, y constituye precisamente su Dominio Administrativo.

Por lo antes descrito, se puede advertir una situación problemática en cuanto a la integración de los recursos que se encuentran presentes en los Dominios Administrativos de las Instituciones participantes (equipos de computación, supercomputadoras, clusters, sistemas operativos, sistemas de almacenamiento, entre otros), debido a que los mismos no son uniformes, y son manejados con políticas de funcionamiento muy particulares, propias de cada Institución. Por lo tanto, al encontrarse, estas Instituciones, inmersas en una posible integración, respetando sus

políticas internas, se produce una pluralidad de recursos que aunque es beneficiosa por su gran variedad, a su vez pudiera convertirse en un obstáculo para que los usuarios la aprovechen en forma inmediata. La solución a este problema ha sido delegada directamente al Grid. Para ello, se han planteado los siguientes requerimientos esenciales para obtener arquitecturas del Grid que sean fiables, consistentes y accesibles:

1. Uniformidad: El usuario del Grid debe ver los diferentes recursos y datos como si se tratase de un único recurso.
2. Fiabilidad: Se refiere a la disponibilidad permanente del Grid, que exige un especial cuidado en la tolerancia a fallas y la redundancia, con servicios de red y almacenamiento robustos.
3. Seguridad: Es fundamental controlar el acceso a los datos y a los recursos compartidos, restringiéndolos sólo a los usuarios autorizados.
4. Ubicuidad: Obtener recursos disponibles para la mayor cantidad de usuarios, en todo lugar.
5. Transparencia: Los datos en sus diferentes formatos y tipos de archivos han de ser integrados en una base de datos virtual, de manera que los usuarios puedan manejarlos con independencia del tipo de fuente de donde provengan.

La implementación de un Sistema Grid particular, implica asumir el reto del desarrollo de un software que permita la satisfacción de todos estos requerimientos. Tal como lo expresa Lindahl et. al.(1998), este software es frecuentemente llamado middleware y su importancia se halla en que precisamente, es el middleware quien transforma una colección de recursos independientes (pertenecientes a los Dominios Administrativos antes mencionados) en una única y coherente máquina virtual.

Un proyecto de esta envergadura puede tener un alto grado de complejidad; no obstante, es posible apoyarse en la arquitectura estratificada (estratos o capas) del Grid para tratar de disminuirla, aprovechando la ventaja que se obtiene al utilizar una estructura por capas como ésta, la cual permite delimitar el trabajo sobre el Grid. En este caso, se seguirá una estrategia que consiste en concentrar los esfuerzos del desarrollo del Grid, tomando los requerimientos de una capa en particular, con la

finalidad de transformarlos por medio de la Ingeniería del Software, en un servicio que satisfaga necesidades específicas ubicadas en ella, pero que a su vez, pueda ser utilizada para apoyar el desarrollo de servicios de otras capas superiores del Grid (la arquitectura del Grid será explicada con mayor profundidad en el marco teórico de este trabajo). Como un adelanto, se tiene que existe una tendencia a ubicar en las capas inferiores los servicios que están vinculados a un nivel cercano de los recursos compartidos y sus sistemas operativos, y que en las capas superiores la tendencia es a ubicar los servicios vinculados a un nivel cercano de las aplicaciones del usuario final.

Tomando en consideración lo antes expuesto, se plantea la necesidad de diseñar un modelo para un servicio que gestione la información sobre los recursos disponibles en el Grid. Para ello se cuenta con los sistemas de almacenamiento participantes en el Grid, que permiten guardar el estado de estos objetos y recuperarlos en el momento en que sea necesario; es decir, manejar la persistencia de estos objetos.

Es importante mencionar, que al hacer referencia al trabajo sobre un Grid Computacional basado en Java, se quiere resaltar que se está en presencia de un Grid, que ha sido elaborado siguiendo el paradigma orientado a objetos y que utiliza Java como lenguaje de programación de alto nivel. La selección de un diseño orientado a objetos se justifica, debido a que tal como lo menciona García y Cardinale (2000), este diseño ha demostrado ser elegante y eficiente para el desarrollo de software distribuido, como lo es el software presente en un Grid. Además, la característica multiplataforma de Java, lo hace ideal para el desarrollo de herramientas que hayan sido diseñadas para ser utilizadas en ambientes heterogéneos.

A través de este servicio se intenta resolver un problema de interoperabilidad entre estos recursos de almacenamiento heterogéneos tales como: (a) Los sistemas de archivos, (b) Los Sistemas de Gestión de Bases de Datos Relacional y (c) Los Sistemas de Gestión de Bases de Datos Orientados a Objetos. El desarrollo de un servicio como éste apunta a satisfacer el requerimiento de Transparencia, con la cual se espera obtener una visión de estos recursos como si se tratase de una Base de

Datos Virtual para el Grid; es decir, una Base de Datos que para el Grid aparece como un único sistema de almacenamiento y recuperación de recursos, pero que en realidad se apoya en los sistemas de almacenamientos locales que se encuentran distribuidos en los diferentes nodos que conforman la red.

Además, por tratarse de un servicio para un Grid Computacional basado en Java, es decir Orientado a Objetos, se hace imprescindible considerar el elemento básico que es manejado por este servicio: El objeto. Más aún, no sólo el objeto en forma aislada e individual, sino también en su relación con otros objetos; esto es, lo que comúnmente es denominado el modelo de objetos. Por lo tanto, el servicio propuesto para este tipo de Grid se encarga del manejo (almacenamiento y recuperación) de la persistencia de los objetos o modelo de objetos Java, sobre los recursos de almacenamiento heterogéneos disponibles en el mismo.

Es importante señalar que algunos de estos recursos de almacenamiento no fueron diseñados originalmente bajo una filosofía orientada a objetos, como es el caso de los Sistemas de Gestión de Bases de Datos Relacional. Por esta razón, estos sistemas no serían los más idóneos para el manejo de la persistencia de los objetos del Grid Computacional. Sin embargo, si volvemos a la parte inicial de este planteamiento, notamos que la intención de una institución promotora de un Grid, no es la de rechazar los recursos que otras instituciones puedan ofrecer, sino buscar la incorporación de una mayor cantidad de ellos. Tampoco la idea sería imponer nuevos paradigmas de desarrollo y administración de los recursos de estas instituciones, como por ejemplo, el paradigma orientado a objetos. En consecuencia, se presenta un nuevo problema que también debe tratar de resolver el servicio que se está proponiendo.

Este tipo de problema, como lo menciona Martínez (2001), es denominado Desadaptación de Impedancias (“Impedence Mismatch”), cuyo término es proveniente de la Ingeniería Eléctrica y adaptado a la Ingeniería del Software, utilizado para nombrar el problema ocasionado al tratar de conectar dos sistemas que presentan bases conceptuales muy diferentes, como es el caso del modelo orientado a objetos (clases, objetos, atributos, métodos, herencia y encapsulamiento, entre otros)

y el modelo relacional (tablas, columnas, campos, filas y registros, entre otros). En el caso de presentarse una situación como esta, los desarrolladores del Grid tienen que realizar una correlación entre los objetos o modelos de objetos y su correspondiente representación en tablas relacionales. Esta acción trae algunos efectos no deseados. Al agregar una mayor cantidad de líneas de código a nivel de programación, se aumenta el nivel de complejidad del software del Grid, especialmente para el mantenimiento del mismo. Además, esto distrae a los desarrolladores del cumplimiento de los requerimientos del Grid más importantes, tales como se mencionan en García y Cardinale (2000): seguridad, resguardo de privacidad, estabilidad del sistema y tolerancia a fallas, entre otros. Por lo tanto, se propuso un servicio que asuma la responsabilidad del manejo de este problema de Desadaptación de Impedancias Objeto-Relacional, con la finalidad de liberar al resto de los desarrolladores del Grid de esta tarea y permitir un manejo transparente de la persistencia de objetos en el Grid.

Objetivos

Objetivo General

Diseñar un modelo para un servicio manejador de objetos persistentes en Grids Computacionales basados en Java.

Objetivos Específicos

1. Diagnosticar la situación actual sobre el manejo de la persistencia de objetos en Grids Computacionales basados en Java.
2. Determinar la arquitectura del modelo propuesto para el Servicio Manejador de Objetos Java Persistentes (SMOP).
3. Determinar las tecnologías estándares para el manejo de la persistencia de objetos java que mejor se integren a la arquitectura seleccionada para el modelo.

4. Determinar la factibilidad de elaboración del modelo propuesto para el SMOP.
5. Especificar el modelo propuesto para el SMOP.
6. Validar el diseño del modelo propuesto para el SMOP.

Justificación e Importancia

Este trabajo se encuentra enmarcado en el proyecto de Grid Computacional basado en Java denominado SUMA ("Scientific Ubiquitous Metacomputing Architecture"), que se ha establecido siguiendo una línea de investigación claramente definida por la Universidad "Simón Bolívar" de Venezuela, a través de su Grupo de Investigación en Sistemas Paralelos y Distribuidos (GI-SPD). SUMA se presenta como un cliente potencial del servicio que se está proponiendo en este trabajo, debido a que actualmente SUMA no utiliza un servicio como éste, por lo tanto, la información persistente (objetos persistentes) de este Grid, tales como: Servicios, Recursos y Usuarios, entre otros, es manejada de una forma muy elemental por medio de archivos planos. Se espera que este servicio proporcione un valor agregado al manejo de estos objetos persistentes, usados por los módulos del núcleo de SUMA para la toma de decisiones de Planificación (El Planificador) y verificación de permisos de usuarios (El Controlador de Usuarios), entre otros.

Actualmente los desarrolladores de Grids Computacionales Orientados a Objetos, que no manejen un mecanismo adecuado de almacenamiento y recuperación de objetos, deben agregar cantidades de líneas de código de programación adicionales para la administración de sus objetos persistentes. De esta manera distraen su atención de los puntos focales para el desarrollo del Grid Computacional, como lo son: seguridad, confiabilidad, estabilidad, rendimiento y tolerancia a fallas, entre otros. Teniendo en cuenta que no todos los componentes de los Grid Computacionales utilizan la misma fuente de datos, para el manejo de sus objetos persistentes, este mecanismo, implica dedicar tiempo y esfuerzo (horas/hombre) extra en su mantenimiento, contribuyendo a aumentar la complejidad del desarrollo, en un ambiente tan heterogéneo y cambiante como el computacional.

Por lo tanto, la importancia del diseño de un modelo para un **Servicio Manejador de Objetos Persistentes en Grids Computacionales basados en Java**, es que busca liberar, al desarrollador del Grid de los detalles internos del manejo de la persistencia de estos objetos haciéndolo, en forma teórica, más productivo su trabajo.

Alcance y Limitaciones

El objetivo principal de los Sistemas Grid es el de proveer una infraestructura coherente entre Instituciones u Organizaciones que deseen participar en una integración de los recursos heterogéneos (computacionales, almacenamiento, entre otros) que se encuentren disponibles en sus Dominios Administrativos que, por lo general, están geográficamente distribuidos y conectados por medio de una red. En relación con el alcance de este proyecto, es oportuno precisar lo siguiente:

1. En el ámbito de esta investigación, la información persistente utilizada para representar recursos disponibles en el Grid, es tomada de algunas Instituciones Universitarias miembros de la Red REACCIUN que hipotéticamente pudieran participar en un Grid a nivel nacional.
2. El tipo de Sistema Grid con el que se trabaja es el Grid Computacional.
3. A pesar de que el cómputo es el principal recurso a ser compartido en los Grid Computacionales, en este trabajo se le da una gran importancia al recurso de almacenamiento en que se apoya el Grid Computacional.
4. Los recursos de almacenamiento que serán revisados, son aquellos que tienen su soporte sobre: Sistemas de Archivos; Sistemas de Gestión de Base de Datos Relacional o Sistemas de Gestión de Base de Datos OO.
5. El tipo de paradigma empleado para la implementación de la infraestructura del Grid es el paradigma orientado a objetos, junto con el lenguaje de programación Java.

Una vez establecidos estos aspectos básicos se puede especificar, claramente, que el modelo que se propone para el servicio manejador de objetos persistentes, se

diseña tomando en consideración como cliente principal, por lo menos uno de los Planificadores presentes en el núcleo de un Grid Computacional basado en Java.

Se plantea una propuesta que satisfaga los requerimientos generales referentes al servicio en estudio, se determinan los elementos que componen el servicio, se establecen las relaciones tanto externas como internas de cada elemento y se modela el servicio de acuerdo a las notaciones proporcionadas por la metodología de desarrollo de software empleada. Una vez finalizado el diseño del modelo para el servicio propuesto, se buscó un mecanismo de validación. En primera instancia, se pensó en la realización de un prototipo funcional que pudiera trabajar en forma independiente de cualquier Grid, pero que demostrara la funcionalidad del servicio, mediante la satisfacción de los requerimientos inicialmente planteados, en especial el almacenamiento y recuperación de objetos o modelo de objetos Java, cuya persistencia sea manejada sobre recursos de almacenamiento heterogéneos. Con respecto al prototipo funcional se buscó proporcionar:

1. Interoperabilidad entre diferentes sistemas de almacenamiento, que se apoyan directamente sobre : Sistema de Archivos, Sistemas de Base de Datos Relacional y OO. En este trabajo se tomó un producto no comercial representante de cada uno de los tipos de sistema de almacenamiento, por ejemplo: (a) “File/Object Store” el cual se instaló directamente sobre Sistemas de Archivos de los Sistemas Operativos Linux y Windows; (b) “MySQL” representando las Bases de Datos Relacional y (c) “ObjectDB”en representación de las Bases de Datos OO.
2. Interfaces para el acceso remoto al Servicio SMOP, que permitan interactuar con otros componentes de software; en especial se buscó simular la interacción con por lo menos, uno de los Planificadores del Grid.
3. Interfaz Gráfica para Administradores locales de cada Servidor en donde se ejecute el Servicio SMOP.

Adicionalmente, se utilizó un Ambiente de Simulación, que permitiera modelar la interacción de un Planificador del Grid con el Servicio SMOP planteado, con la finalidad de obtener algunos datos de tipo estadísticos que, posteriormente, ayuden a los desarrolladores del Grid en la toma de decisiones en relación al uso del Servicio

SMOP para la Planificación general de los Recursos; es decir, la Localización, Asignación o Verificación de la Permisología de los mismos.

CAPÍTULO II

MARCO TEÓRICO

ANTEDECENTES

Los antecedentes del presente estudio son establecidos considerando dos puntos de vistas. Primeramente, es considerado el punto de vista *funcional del servicio* en forma independiente, es decir, cómo se ha venido realizando el manejo de la persistencia de objetos Java sin tomar en cuenta el *contexto* en donde se propone desarrollar el servicio. Luego, considerando el *contexto*, se describe en forma resumida el resultado de la revisión documental de algunos proyectos de implementación de Grid Computacionales o similares conocidos, haciendo énfasis en las soluciones que se plantean en estos proyectos para el manejo de la persistencia de sus objetos.

MECANISMOS DE PERSISTENCIA DE OBJETOS

A continuación, se presentan los mecanismos comúnmente usados para la persistencia de los objetos en el lenguaje Java:

1. Serialización en Sistemas de Archivos: Los sistemas de archivos son considerados una solución de almacenamiento relativamente ligera. Un sistema de archivo tiene la capacidad del almacenamiento de datos usando diversos formatos, incluyendo los definidos por el mismo usuario. La desventaja de trabajar con archivos es que no soportan en forma automática las transacciones, ni las funciones para la verificación de la integridad de los datos. Ahora se define el mecanismo denominado Serialización, el cual trabaja sobre los sistemas de

archivos. La Serialización es el soporte de persistencia básico en el lenguaje Java. Se llama Serialización porque los objetos son escritos o leídos secuencialmente, como una serie de bytes. La técnica de Serialización consiste en la transformación de un grafo de objetos en un flujo de bytes para su almacenamiento o transmisión a través de la red. La Serialización es utilizada para almacenar el estado de un objeto y el grafo de los objetos que referencia a un flujo de salida, manteniendo las relaciones entre los objetos Java de manera que el grafo completo pueda ser nuevamente reconstruido. La Serialización presenta algunas limitaciones, como por ejemplo: No soporta transacciones, ni consultas, ni concurrencia (poder compartir, a la vez, los datos entre los usuarios) y no permite recuperaciones parciales, sino que sólo permite el acceso con el nivel de granularidad original de la Serialización (recuperando el grafo completo), presentado serios problemas a la hora de utilizar la Serialización en múltiples grafos, de manera simultánea.

2. Conectividad de Base de Datos Java ("JDBC") en Sistemas de Gestión de Base de Datos Relacional (SGBDR): Típicamente, se ha empleado un mecanismo combinado para alcanzar la persistencia en Java sobre los SGBDR. Este mecanismo usa la Interfaz de Programación de Aplicaciones ("API") denominada JDBC para el acceso a la base de datos, combinada con el Lenguaje Estructurado de Consultas ("SQL") para la obtención de la información, mediante comandos provenientes de este lenguaje estructurado. El uso de JDBC para la persistencia de objetos presenta algunas dificultades, entre las que se destaca la siguiente: se exige al desarrollador manejar explícitamente los objetos y hacerlos corresponder con las tablas relacionales. De esta manera, el desarrollador es forzado a resolver, por su cuenta, la Desadaptación de Impedancias Objeto-Relacional presente en este caso. Lo que conlleva al manejo de una situación un poco ambigua en varios niveles, debido a que se tienen dos modelos de datos diferentes, dos lenguajes de programación y dos paradigmas de acceso a los datos también diferentes: el modelo de objetos Java y el modelo relacional SQL. El esfuerzo de desarrollo para implementar la correlación entre el modelo de datos relacional y su correspondiente modelo de objetos Java es considerable. Como muestra de ello,

Jordan y Russel (2003) han afirmado que la mayoría de los desarrolladores tienden a no definir un modelo de objetos para sus datos, sino que simplemente escriben el código Java directamente para manipular las tablas de la base de datos. El resultado es que no se benefician totalmente de las ventajas del paradigma orientado a objetos.

3. Sistemas de Gestión de Bases de Datos Orientadas a Objetos (SGBDOO): Estos sistemas parecen ser una posibilidad muy lógica para ser empleada a la hora de seleccionar un método de gestión de la información representada por medio de un modelo de objetos, debido a que con esta opción no se necesita realizar ninguna descomposición artificial de la información en tablas, filas o columnas. Sin embargo, esta estrategia ha presentado desde sus inicios algunos problemas de portabilidad e interoperabilidad, entre las mismas implementaciones de SGBDOO generadas por diferentes compañías. Por esta razón, el grupo ODMG ("Object Data Management Group") dedicó un gran esfuerzo a la estandarización para solventar esta situación. Este grupo finalizó su trabajo en el año 2001, dando como resultado el compendio de una serie de especificaciones estándares para el manejo de los objetos de datos, identificadas como especificaciones ODMG 3.0. Se puede decir que el grupo ODMG proporcionó una API estándar para brindar la funcionalidad de una base de datos en un lenguaje de programación orientado a objetos (por ejemplo, Java). Actualmente el grupo ODMG se ha separado y en su lugar ha tomado esta responsabilidad la "Java Commnunity Proccess" (JCP).

4. "Beans" de Entidad en "Enterprise JavaBean" ("EJB" 2.0) : Los "Beans" de Entidad son parte de las especificaciones EJB de la Edición Empresarial para la plataforma Java ("J2EE"). Esto significa que los "Beans" de Entidad, tienen una manera estándar para la representación de la persistencia de los datos de sus aplicaciones. Esta representación puede ser compartida a través de las conexiones concurrentes de clientes locales o remotos. No obstante, no todos los recursos de almacenamiento a ser compartidos en un Grid se encuentran disponibles en un Servidor de Aplicaciones J2EE; por lo tanto, a pesar que el EJB es una solución (basada en componentes) muy viable al manejo de la persistencia de objetos Java,

por no manejar los ambientes heterogéneos, no será usada directamente para dar soporte a nuestro servicio. Además, Roos (2003) menciona algunas dificultades de trabajar con el modelo de los "Beans" de Entidad (2.0). Entre ellas se pueden mencionar un par: Imposibilidad de almacenamiento de grafos de objetos (relaciones) de los "Beans" de Entidad y Carencia de soporte para la herencia de los objetos.

5. "Java Data Object" ("JDO" 1.0): Es una definición de persistencia de objetos basada en interfaces para el lenguaje Java, la cual describe el almacenamiento, consulta y recuperación de objetos desde un almacén de datos. Es un estándar para la transparencia de la persistencia de objetos Java.

En el Cuadro 1, se hace una comparación entre las principales características, de las diferentes tecnologías utilizadas para alcanzar la persistencia de los objetos:

Características	**Serialización**	**JDBC**	**SGBDOO**	**EJB**	**JDO**
Transacciones	X	✓	✓	✓	✓
Facilidad de consulta	X	✓	✓	✓	✓
API Estándar	✓ java.io	✓ JDBC	X ODMG	✓ EJB	✓ JDO
Lenguaje de Consulta Estándar	X	X SQL	X OQL	✓ EJBQL	✓ JDOQL
Paradigma de almacenamiento de datos soportado	Sistema de archivo	SGBDR	SGBDOO	SGBDR, EAI (Enterprise Application Integration)	SGBDR, SGBDOO, EAI, Sistema de archivos, otros
Transparencia para la conclusión de las instancias persistentes	X	X	✓	X	✓
Transparencia en el modelo del dominio	X	X	✓	X	✓
Verdadera base de datos de objetos	X	X	✓	X	X
Soporte para la estructura de tablas existente	X	✓	X	X	✓

Cuadro 1. Tecnologías utilizadas para alcanzar la persistencia de los Objetos.

Debido a las características observadas anteriormente, se destaca al "JDO" como un mecanismo apropiado para el diseño del modelo del servicio SMOP que se plantea en esta investigación, cuya principal funcionalidad es el manejo de los objetos Java persistentes. Sin embargo, existen ciertas limitaciones que se han detectado, pero que no afectan la decisión de utilizar JDO en el SMOP. El lenguaje de consulta de JDO (JDOQL) requiere el uso intensivo de mayores recursos que otros lenguajes de consulta existentes, como por ejemplo el lenguaje de consulta EJB (EJBQL) y El proceso de mejora o enriquecimiento que se emplea en JDO para proporcionar capacidad persistente a las clases originales, no es muy cómodo para muchos desarrolladores. Estas pueden ser mencionadas entre otras limitaciones.

PLATAFORMAS DE GRIDS COMPUTACIONALES

De aquí en adelante, se presentan algunos antecedentes relevantes para el SMOP, tomando en cuenta los Grid Computacionales OO.

Al especificar que los Grid computacionales OO conforman el ámbito de este estudio, se está reconociendo que de alguna manera, existe una diferenciación del Grid según el enfoque o paradigma que se esté utilizando. Weissman (2000) distingue tres arquitecturas básicas dominantes, según el paradigma que se emplee en la construcción de infraestructuras de Grid Computacionales: (a) Soluciones basadas en servicios globales de alto nivel como el Sistema Globus; (b) Soluciones basadas en la Web como los sistemas operativos web y (c) Soluciones basadas en el paradigma OO, como los sistemas Legion y SUMA.

Se comienza por describir a Globus, por ser uno de los proyectos Grid más importantes, debido a que se ha convertido en un estándar de facto en esta área del conocimiento, como lo señala Sotomayor (2003). El proyecto Globus se describe como un esfuerzo de investigación multi-institucional que procura hacer posible la construcción de un Grid computacional que provea un acceso consistente, confiable y total a los recursos computacionales de alto rendimiento.

Un elemento central del sistema Globus es el "Globus Metacomputing Toolkit" ("GMT"), el cual define los servicios básicos y capacidades requeridas para construir el Grid computacional. Con respecto a su diseño, el "GMT" consta de un conjunto de componentes que implementan estos servicios para la obtención de: seguridad, localización y administración de recursos, comunicaciones, entre otros. Foster y Kesselman (1998) afirman que el Grid Computacional debe soportar una amplia variedad de aplicaciones y modelos de programación. Por esta razón, el "GMT" cuenta con lo que estos investigadores llaman "bolsa de servicios", para que los desarrolladores del Grid puedan seleccionar de esta bolsa las herramientas específicas o aplicaciones más convenientes para satisfacer sus necesidades.

Globus presenta una arquitectura estratificada, en la cual los servicios globales de alto nivel están construidos en el tope de un conjunto esencial de servicios locales. En la base de esta arquitectura estratificada, el "Globus Resource Allocation Manager" ("GRAM") provee el componente local para la administración de los recursos. "GRAM" provee una interfaz estándar, habilitada en redes, para un sistema de administración de recursos locales. Por lo tanto, las herramientas y aplicaciones en el Grid Computacional pueden expresar requerimientos de localización de recursos y administración de procesos en términos de una Interfaz estándar de Programación de Aplicaciones ("API").

En relación al manejo de la persistencia de los datos u objetos en Globus, Stockinger et. al. (1998) han propuesto asumir soluciones basadas en sistemas manejadores de bases de datos orientadas a objetos o almacenes de objetos. A pesar que estas soluciones han sido señaladas para los Grid de Datos, el tipo de servicio que se está estudiando hace que se considere su aplicación, de igual manera, a los Grid Computacionales. Estas soluciones han sido demostradas en el almacenamiento de datos experimentales, como por ejemplo, los datos generados en los experimentos de Física de Alta Energía. En estos casos, han sido empleadas herramientas y lenguajes de Ingeniería del Software OO para desarrollar la infraestructura de software en el análisis final de los datos. Todos los datos son

concebidos como objetos persistentes, en el nivel más elevado de abstracción del modelo de datos del experimento, y pueden ser accedidos por medio de un mecanismo de navegación OO. Asimismo, la visión de los datos del experimento en el más alto nivel no contiene el concepto de archivo, ni de réplica de los datos, sino que todos los objetos son ideados para que existan sin tomar en cuenta cómo son almacenados o cuántas réplicas de los mismos se encuentren. Los conceptos de *archivos* y *replicación* aparecen como mecanismos de implementación de la visión de los objetos del experimento, sólo en las capas más bajas de la abstracción.

En este trabajo se resalta el modelo estratificado en cinco capas, propuesto por Globus para la arquitectura del Grid (ver Bases Teóricas), como un aporte importante a ser tomado en cuenta en el desarrollo del modelo para el servicio en estudio. Por otro lado, la concepción del Conjunto de Herramientas, mejor conocido como "Toolkit" de Globus, también da una buena referencia a ser considerada en la implementación del SMOP, porque ésta representa una solución que ya ha sido puesta en marcha con éxito en el área de la investigación del Grid. Finalmente, se observa como en Globus también se le da importancia a la solución del manejo de la persistencia de objetos en los Sistemas Grids intensivos en datos (Grid de Datos). De esta solución, se deriva el tener presente a los sistemas de bases de datos OO, como uno de los sistemas de almacenamiento más acordes con la conservación y recuperación de los estados de los objetos persistentes, a ser empleados en los sistemas Grid.

Pasando a otro enfoque, ahora se presentan a los proyectos vinculados con las soluciones basadas en Web. Se destaca el Sistema Operativo Web (SOW) debido a que el mismo forma parte del esfuerzo de investigación que se está realizando en la Universidad de los Andes (ULA) de Venezuela. A pesar de que este proyecto no es considerado directamente como un Grid Computacional OO, se observan algunas similitudes en sus características (sistemas distribuidos, finalidad de compartir recursos en forma transparente, entre otras) que hacen que pueda ser considerado dentro de los antecedentes contextuales del servicio en

estudio. Muestra de ello se evidencia en la definición proporcionada por Kropf (citado por Perozo, 2003) , quien dice que un SOW: "Es un ambiente de metacomputación que soporta y maneja adecuadamente el procesamiento paralelo y distribuido sobre redes de área amplia ". Se han subrayado los términos comunes entre el Grid y SOW, según esta definición.

En una investigación reciente (Perozo, 2003) se presenta al SOW como una corriente de desarrollo emergente, que tiene como objetivo principal proveer una plataforma que permite a los usuarios beneficiarse del potencial computacional ofrecido en la web, a través de compartir los recursos y solucionar problemas de heterogeneidad y adaptabilidad dinámica presentes en la misma.

Cabe destacar que en el estudio al cual se hace referencia, se plantea el diseño de dos subsistemas: El subsistema manejador de comunidades y El Subsistema Manejador de Repositorios Locales (SMRL). La propuesta de este último subsistema (SMRL) evidencia una especial preocupación por brindar nuevas soluciones a los mecanismos de almacenamiento tradicionales en estos ambientes tan heterogéneos.

Uno de los aspectos que llama poderosamente la atención en el diseño del SOW, es el concepto de repositorio local y repositorio remoto, debido a la posibilidad de adaptarlo al modelo del servicio en estudio; el SMOP. En el SOW propuesto, un repositorio local es visto como una colección integral de datos, variable en el tiempo y no volátil, con las siguientes características básicas: (a) Debe ser íntegro: recolectarse en un espacio de almacenamiento; (b) Debe ser organizado, en torno al objetivo de la aplicación y variables involucradas: es el soporte para la toma de decisiones y (c) Se debe incrementar en el tiempo: almacenar información de varios períodos.

Adicionalmente, se mencionan las diversas funcionalidades del repositorio de la propuesta del SOW: (a) Catalogar y describir la información disponible; (b) Especificar el propósito de la misma; (c) Indicar las relaciones entre los distintos datos; (d) Establecer quién es el propietario; (e) Relacionar las estructuras técnicas de datos con la información de la aplicación; (f) Establecer las relaciones

con los datos operacionales y las reglas de transformación; y (g) Limitar la validez de la información.

Es importante resaltar que en la propuesta general del SOW no se mantiene el enfoque de un catálogo global de los recursos disponibles, ya que no se considera conveniente hacerlo, debido a la enorme cantidad de recursos presentes en la Internet, y los constantes cambios en la información de los mismos realizados en períodos de tiempo muy breves, lo que traería como consecuencia una actualización ineficiente de la información de los recursos. En su lugar, la propuesta complementa el manejo de repositorios locales para cada nodo involucrado en el sistema, con el manejo de repositorios remotos para almacenar la información referente a los nodos vecinos de un nodo dado.

Antes de presentar las soluciones Grid basadas en el paradigma OO, se abre un paréntesis para describir al proyecto Gaia, que aunque no es considerado un Sistema Grid, posee un servicio denominado "Data Object Service" ("DOS") que maneja su información persistente siguiendo un enfoque OO. Gaia se plantea como una infraestructura que exporta y coordina los recursos contenidos en un espacio físico, de modo que define un ambiente computacional genérico. Gaia convierte los espacios físicos y los dispositivos de computación ubicua que ellos contienen en un sistema de computación programable. Gaia es análogo a un sistema de computación tradicional. Esto significa que tal como un computador está compuesto de dispositivos de entrada y salida, recursos y periféricos, de esa misma forma se encuentra en Gaia un espacio físico poblado con muchos dispositivos. Gaia es la extensión de un trabajo previo realizado por los investigadores sobre el sistema operativo 2k (sistema operativo middleware que utiliza CORBA como mecanismo de comunicación y se ejecuta como el tope de plataformas existentes tales como Windows NT y Solaris).

El DOS como servicio de distribución de datos de Gaia, es un servicio de datos middleware que hace uso de un sistema operativo nativo para administrar los datos sobre el disco. Sin embargo, el servicio ofrece más que un simple acceso a los archivos de datos. En general, los datos no son transportados como largos

flujos de bytes (aunque DOS puede soportar esta modalidad) sino que son manejados como objetos de datos ("Data Object"). Las interfaces tradicionales de los sistemas de archivos (por ejemplo "Open", "Read", "Write", "Close") son reemplazadas por abstracciones orientadas objetos, tales como: Contenedores ("Containers") e Iteradores ("Iterators").

Los sistemas de archivos distribuidos tradicionales son diseñados generalmente para ambientes homogéneos y para simplemente transferir datos a un nodo local. Sin embargo, la naturaleza heterogénea dominante de los ambientes de computación hacen parecer inapropiada las configuraciones estáticas de los tradicionales sistemas de archivos distribuidos. Por eso, sería mejor opción tener un servicio de acceso a los datos que sea configurable dinámicamente como "DOS" y que funcione para diferentes tipos de dispositivos.

Tal como lo describen Hess et. al (2000) un componente clave para los sistemas distribuidos es el acceso remoto a los datos. Ellos afirman que los sistemas de archivos distribuidos tradicionales son más estáticos y no son capaces de adaptarse a los recursos de los dispositivos disponibles de hoy en día. Los archivos de datos son tratados como un flujo continuo de bytes y las interfaces para acceder a ellos son diseñadas para datos no estructurados. Estos investigadores opinan que resulta muy difícil proveer un contenido variable y adaptable usando estas interfaces. En consecuencia, han propuesto un servicio adaptable de objetos de datos para ambientes computacionales usando objetos distribuidos. Este servicio es considerado por ellos, como un servicio general de distribución de información para ambientes heterogéneos, que incorpora adaptación automática de contenido, conciencia de localidad y conocimiento del entorno.

La información es manipulada por medio de interfaces orientadas a objetos basadas en Contenedores e Iteradores. Los Contenedores proveen operaciones de manipulación de datos, mecanismos de análisis gramatical y transformación del contenido a estructuras de datos y los Iteradores proveen un acceso más conveniente. Los Contenedores pueden ser instanciados en la localidad más

apropiada, y el acceso a estos componentes puede ser transferido entre nodos, permitiendo que los Contenedores sean ubicados en varios nodos para comunicarse.

En resumen, los aspectos más importante a ser considerados en el proyecto Gaia, son el servicio de almacenamiento y recuperación de los datos "DOS" y el uso de las interfaces OO denominadas Contenedores e Iteradores.

Se concluye este recorrido de los proyectos de implementación de la infraestructura de los Grid Computacionales, con dos de los proyectos que siguen el paradigma de OO: Legion y SUMA.

Primeramente, se destaca al proyecto Legion, como el ejemplo más representativo de las soluciones de infraestructura del Grid, basada en el paradigma de OO. El proyecto Legion consiste en el desarrollo de un software de metacomputación, dirigido por un grupo de investigación de la Universidad de Virginia, en los Estados Unidos, y ha sido implementado sobre Mentat (sistema de procesamiento paralelo orientado a objetos desarrollado también por la misma universidad). Según Grimshaw et. al. (1994), los objetivos centrales para el diseño de Legion son los siguientes: (a) Ambiente computacional uniforme y de fácil uso, (b) Alto rendimiento a través del paralelismo, (c) Espacio de nombres único y persistente, (d) Seguridad para los usuarios y propietarios de los recursos, (e) Gestión y explotación de la heterogeneidad de los recursos y (f) Mínimo impacto sobre la computación local de los propietarios de los recursos.

La construcción de Legion, como ya se ha mencionado, está fundamentada sobre los principios del paradigma de orientación a objetos, enfocándose en la explotación de las propiedades de encapsulamiento y herencia que proporciona el paradigma, con la finalidad de obtener beneficios tales como la reutilización del software, contención de fallas y reducción de la complejidad. El uso de este paradigma aplica en desarrollos de sistemas tan largos y complejos como lo son los Grids Computacionales.

Tal como lo presenta Chapin et. al. (1998), Legion tiene previsto conectar miles y quizás millones de recursos de computación que van desde simples PCs

hasta supercomputadores paralelos. Por lo tanto, en un ambiente de metacomputación OO como éste, se podrán manejar hipotéticamente, entre millones y hasta billones de objetos. De allí, se percibe la necesidad de contar con un servicio eficiente, para el manejo de la persistencia de estos objetos, que proporcione un buen rendimiento en el almacenamiento y recuperación de los mismos. Karpovich (1996) presenta a los objetos, como la unidad básica del sistema Legion. A su vez, describe a los objetos de Legion como instancias de clases bien definidas, que tienen un nombre universal dentro del sistema y pueden persistir por largos períodos de tiempo. También, muestra a cada objeto Legion con una interfaz determinada por su clase, que define las funciones miembro o métodos con las cuales el objeto responderá.

Con respecto al enfoque empleado por Legion para la implementación de un servicio de almacenamiento y recuperación de sus objetos, señalamos que Legion reconoce el acceso transparente de la localización de los archivos de datos, como el servicio más simple que un Grid Computacional puede ofrecer (Lindahl et. al. 1998). Este servicio es implementado normalmente en los llamados sistemas de archivos distribuidos, como por ejemplo, el NFS ("Network File System"). No obstante, a pesar del gran beneficio brindado por estos sistemas, se presenta el problema de requerir la participación del administrador o súper usuario ("root") del sistema para el uso del servicio. En consecuencia, Legion plantea un nuevo enfoque denominado Espacio Compartido de Objetos Persistentes, el cual provee un acceso seguro y compartido a los archivos u objetos de datos, sin la intervención del administrador del sistema.

Brevemente se describe este modelo, el cual consiste en poder compartir o manejar entidades u objetos (programas, bases de datos, instrumentos, etc), en lugar de usar sólo archivos y sin hacer la distinción tradicional entre los que son los archivos y los objetos. Los archivos representan a los datos persistentes y se utilizan para hacer que los mismos sobrevivan en dispositivos de almacenamiento secundario. Así que a pesar que el acceso a los archivos es más lento que el acceso a la memoria principal, estos tienen la ventaja de que pueden persistir en el

tiempo en el caso de cualquier falla. En un espacio de objetos compartidos, un objeto puede soportar las operaciones estándar de un archivo, tales como “leer” y “escribir”.

En resumen, sobre la temática de Legion se resalta la identificación de Legion como un ejemplo representativo de las soluciones de infraestructura Grid OO y adicionalmente se considera importante destacar el aporte de Legion de su nuevo enfoque de Espacio Compartido de Objetos Persistentes, lo cual puede ayudar a explorar nuevas opciones para el desarrollo de la propuesta de este trabajo.

Finalmente, se concluyen los antecedentes contextuales, con otro de los proyectos que sigue el enfoque OO para la construcción de la infraestructura del Grid. Se presenta a SUMA (“Scientific Ubiquitous Metacomputing Architecture”), como un proyecto de Investigación y Desarrollo (I+D) de la Universidad “Simón Bolivar” (USB) de Venezuela. Este proyecto es dirigido por el Grupo de Investigación en Sistemas Paralelos y Distribuidos (GI-SPD) de esa casa de estudios y consiste en un sistema de metacomputación (Grid Computacional) OO para la ejecución de bytecode de Java de manera remota y transparente. La implementación de este Grid Computacional extiende el modelo de la máquina virtual de Java para obtener un sistema de componentes distribuidos con comunicación basada en CORBA (“Common Object Request Broker Architecture”).

SUMA posee las siguientes características:

1. Capacidad de ejecución de aplicaciones compiladas a *bytecode* de Java, sobre un conjunto de recursos distribuidos.
2. Una interfaz de ejecución de programas sencilla y transparente. La visión de máquina virtual presentada al usuario es la de una máquina poderosa en su escritorio.
3. Servicios implementados en modo nativo, por ejemplo librerías de álgebra lineal como *PLapack.*

4. Adicionalmente, SUMA provee herramientas para el perfil de desempeño *(performance profiling)* de aplicaciones, así como de monitoreo y administración de la máquina virtual.

El objetivo principal de SUMA es ofrecer acceso a recursos distribuidos de alto desempeño para la ejecución de *bytecode* de Java, incluyendo código paralelo con comunicación a través de MPI de Java *(mpiJava).* Además de cumplir con otros objetivos (SUMA, 2002):

1. Aprovechar eficientemente las capacidades de los recursos con que se cuenta en una red local o global, poniendo al alcance de todos los usuarios la velocidad de plataformas de cómputo intensivo.
2. Compartir recursos computacionales bajo convenios de cooperación entre universidades y centros de investigación.
3. Aprovechar el tiempo ocioso de la capacidad computacional instalada.

Para el manejo de la persistencia de sus objetos Java, SUMA no utiliza un servicio adicional que se encargue de esta funcionalidad, sino que se apoya principalmente en el uso de la Serialización de manera directa sobre los Sistemas de Archivos.

Observando con detenimiento las características y los objetivos de SUMA, se puede determinar la importancia de haber incluido a este proyecto como parte de los antecedentes contextuales de esta investigación. Básicamente, por ser un proyecto para la construcción de un Grid Computacional basado en Java, cuya Institución promotora es una Universidad Nacional (USB); este proyecto caracteriza perfectamente el entorno en el cual se estima que funcione el SMOP. Por lo tanto, se piensa en SUMA como un posible cliente potencial del mismo. Por esta razón, nos detendremos un poco más en este Grid Computacional para describir su funcionamiento interno.

SUMA presenta tres niveles de componentes: (a) Los Clientes, (b) El Núcleo del Sistema y (c) los Agentes de Ejecución. Los clientes (Interno y Externo) son aplicaciones de diversos tipos y funcionalidades que se comunican con el Núcleo del Sistema, entregando la información necesaria para solicitar la ejecución de

una Aplicación Java. El Núcleo del sistema SUMA es el encargado de recibir las peticiones y encontrar un servidor de aplicaciones adecuado para la ejecución de cada petición, monitoreando su estado y garantizando la ejecución. El núcleo esta conformado por El Planificador, El Control de Usuario y El Representante (Ver Glosario). Los Agentes de Ejecución cumplen la función de iniciar la ejecución de los programas.

La figura 1 muestra la arquitectura general de SUMA, allí se identifican una serie de pasos para la ejecución remota en el Grid, explicados a continuación:

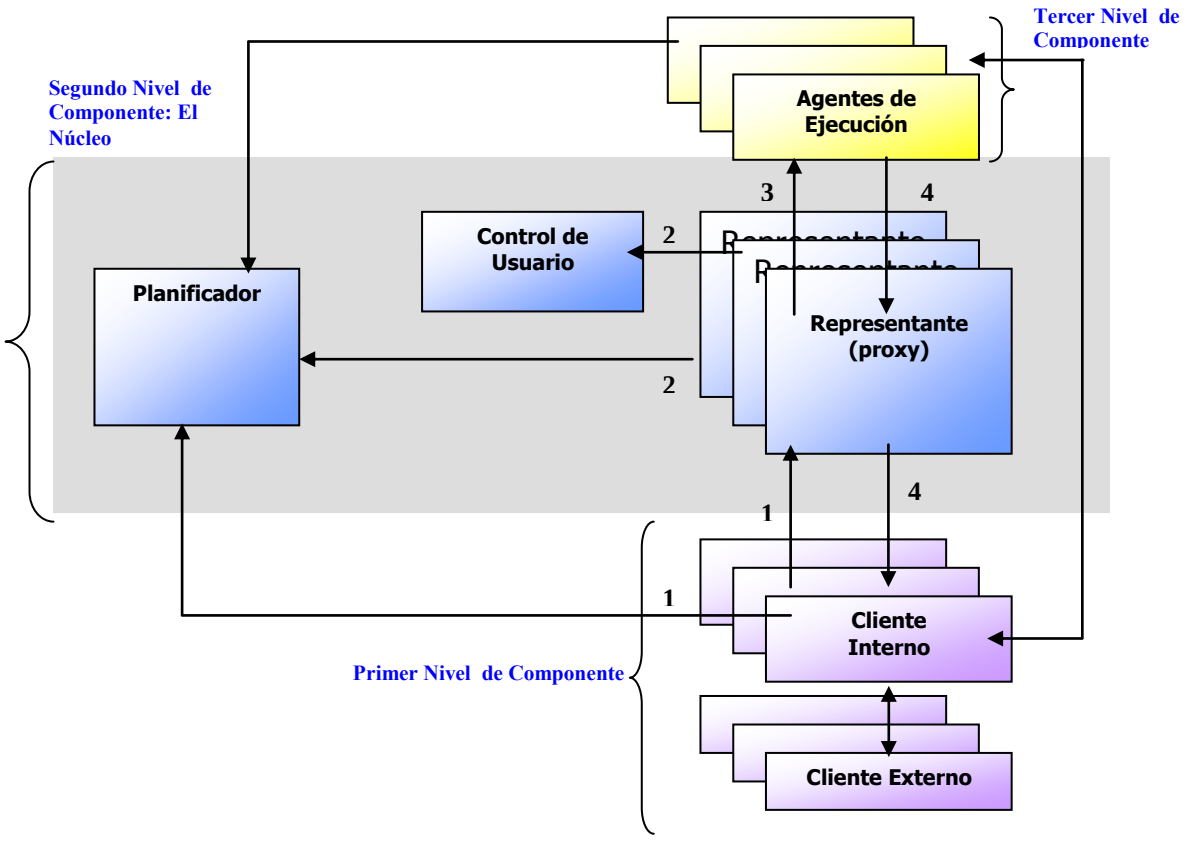


Figura 1. Arquitectura General de SUMA.

Paso 1: El *Cliente Interno* solicita un *Representante* al *Planificador*; luego el *Cliente Interno* entrega la solicitud de ejecución a ese *Representante*.

Paso 2: El *Representante* autentica y verifica los permisos del usuario, solicita al *Planificador* un *Agente de Ejecución.*

Paso 3: El *Representante* entrega solicitud al *Agente de Ejecución*.

Paso 4: El *Representante* devuelve la referencia del *Agente de Ejecución* al *Cliente Interno*, quien establece la conexión. La ejecución comienza cargando remotamente las clases, y sirviendo la E/S desde el *Cliente Externo.*

BASES TEÓRICAS

PARADIGMA DE OBJETOS

Definición

El mundo en que vivimos está rodeado por objetos. Estos objetos son creados por el hombre o ya existen en forma natural. Pueden ser clasificados, descritos, organizados, combinados, manipulados y creados. Por eso no es sorprendente que se haya propuesto una visión orientada a objetos para la creación de software de computadora, la cual es una abstracción que modela el mundo de forma tal que nos ayuda a entenderlo de una mejor manera. Cervera y Marcos (2000) recuerdan que desde el nacimiento de las ideas básicas de orientación a objetos, en la década de los sesenta, esta tecnología se ha presentado como la solución a muchos de los problemas de software, consolidándose en la década de los noventa, como el paradigma de desarrollo para sistemas de complejidad media y alta, tales como: los sistemas para el Diseño y Manufactura Asistida por Computador (CAD/CAM), aplicaciones multimedia y distribuidas entre otras.

Pressman (2002) destaca el concepto de reutilización como uno de los grandes logros de las tecnologías de objetos en relación a la ingeniería del software, debido a que la reutilización lleva a un desarrollo de software más rápido y a programas de mejor calidad. El software orientado a objetos es más fácil de

mantener debido a que su estructura es inherentemente poco acoplada. Esto lleva a menores efectos colaterales cuando se deben hacer cambios. De acuerdo a Peralta (1995) la orientación a objetos se basa en tres pilares fundamentales: (a) Encapsulación y Ocultación de la información; (b) Abstracción y Clasificación y (c) Herencia y Polimorfismo.

Persistencia de Objetos

El concepto de persistencia según Booch (1994) es asegurar la supervivencia de los datos de un programa a la propia ejecución del mismo. Dicho de otra manera, la persistencia es la propiedad por la que la información del sistema puede persistir durante el tiempo que sea requerida. Se necesitará un almacén de datos permanente. Este tipo de datos son considerados como datos persistentes

Si se aplica este concepto a un lenguaje orientado a objetos, se podrá entender que posea esta propiedad, siempre y cuando sea posible que el estado de los objetos creados en un programa, por medio de este lenguaje, pueda persistir una vez finalizado el mismo. Por consiguiente, este tipo de persistencia, siguiendo una metodología orientada a objetos, será referida como Persistencia de Objetos.

Enfoques sobre la persistencia de objetos Silberschatz et .al (2002):

Persistencia por Clases

Los autores califican a este enfoque como el más sencillo pero el menos conveniente. Consiste en declarar que una clase es persistente. En consecuencia, todos los objetos de la clase serán persistentes de manera predeterminada y todos los objetos de las clases no persistentes serán transitorios. Igualmente, consideran que este enfoque no es flexible, debido a que suele resultar útil disponer en una misma clase tanto de objetos transitorios como de objetos persistentes. Como información adicional, para evitar inconsistencias, estos investigadores, opinan que es necesario hablar

con propiedad cuando se haga referencia a una clase llamada persistente, ya que, en muchos sistemas de bases de datos orientados objetos la declaración que una clase es persistente se interpreta como si se afirmara que los objetos de la clase pueden hacerse persistentes en vez de que todos los objetos de la clases son persistentes.

Persistencia por Creación

En este enfoque se introduce una sintaxis nueva para crear los objetos persistentes mediante la extensión de la sintaxis para la creación de objetos transitorios. Por tanto, los objetos son persistentes o transitorios en función de la manera de crearlos. Este enfoque se sigue en varios sistemas de bases de datos orientados a objetos.

Persistencia por marcas

Una variante de un enfoque anterior es marcar los objetos como persistentes después de haberlos creado. Todos los objetos se crean como transitorios , pero, si un objetos tiene que persistir más allá de la ejecución del programa, hay que marcarlo de manera explícita antes de que éste concluya. A diferencia del enfoque anterior, la decisión sobre la persistencia o transitoriedad se retrasa hasta después de la creación del objeto.

Persistencia por alcance

Uno o varios objetos se declaran objetos persistentes (objetos raíz) de manera explícita. Todos los demás objetos serán persistentes si (y solo si) son alcanzables desde el objeto raíz mediante una secuencia de una o más referencias.

Por tanto, todos los objetos a los que se haga referencia desde los objetos persistentes raíz (es decir, aquellos cuyos identificadores se almacenen en el objetos persistentes raíz) serán persistentes. Pero también lo serán todos los objetos a los que se haga referencia desde ellos, y los objetos a los que estos últimos hagan referencia también serán persistentes, etc. Por tanto, los objetos persistentes son exactamente los alcanzables desde una raíz persistente.

Este esquema tiene la ventaja de que resulta sencillo hacer que sean persistentes estructuras de datos completas con sólo declarar persistente la raíz de las mismas. Sin embargo, el sistema de bases de datos sufre la carga de tener que seguir las cadenas de referencias para detectar los objetos que son persistentes y eso puede resultar costoso.

Posibilidades para la Gestión de Objetos Persistentes

Con la llegada de la orientación a objetos la información debe seguir siendo almacenada y recuperada y para ello se han planteado diferentes alternativas (Martínez,2001) :

Basada en Archivos Planos

Cada objeto tiene la capacidad de serializarse y también de iniciarse a partir de una representación serializada D'souza y Wills (1999). La desventaja es que No provee facilidades multiusuarios, concurrencia, metadata, evolución de esquemas, transacción y recuperación que provee un manejador de bases de datos.

Basadas en el modelo relacional

Las bases de datos relacionales tienen y han tenido una presencia y relevancia en el ambiente computacional, por eso es lógico que ante el

surgimiento del paradigma de orientación a objetos, éstas intentasen dar soluciones para la gestión de los objetos.

Bases de Datos Objeto-Relacionales

La esencia de estos sistemas está centrada en la extensión del modelo relacional, de hecho antes eran denominados sistemas de gestión de bases de datos relacionales extendidos. Realmente, existen muchas formas de extender un modelo relacional, sin embargo, la elegida por los promotores de las bases de datos objeto-relacionales está muy centrada en las relaciones, de forma que este tipo de base de datos todavía consta de un conjunto de tablas, y las características de orientación a objetos que soportan son proporcionadas como una extensión al núcleo del modelo relacional (tablas multi-filas, referencias entre filas y herencia entre tablas, entre otras)

Desventajas: (a) Ampliación artificial del modelo de datos relacional; (b) Falta de uniformidad en los mecanismos para la extensión del modelo relacional; (c) Falta de claridad conceptual y (d) Desadaptación de impedancias entre los lenguajes.

Mediadores o Envoltorios (Wrappers) Objeto-Relacionales:

Los mediadores objeto-relacionales, envoltorios o wrappers son productos (tales como: Persistence, ObjectDriver, JavaBlend, etc) que se basan en colocar capas de software (orientado a objetos) sobre una base de datos relacional. Estos sistemas a nivel de programación proporcionan la sensación de estar trabajando con un SGBDOO (Sistema de Gestión de Bases de Datos Orientados a Objetos), sin embargo, internamente las clases son traducidas en tablas relacionales, y los objetos son traducidos en

tuplas. El envoltorio generalmente emplea SQL (Structured Query Language) para interactuar con la base de datos relacional.

Desventajas: (a) Descomposición del esquema conceptual orientado a objetos en tablas; (b) Restricciones en el modelo de datos de la aplicación.

Basadas en el modelo de objetos

Lenguajes Persistentes

Una perspectiva muy prometedora para los programadores ha sido desde siempre la idea de que el propio lenguaje de programación con el que están trabajando les proporcione la posibilidad de que sus datos persistan en el tiempo. Esta posibilidad es proporcionada por los lenguajes persistentes. Dichos lenguajes pueden ser por lo tanto, una posibilidad interesante para conseguir la persistencia de los objetos, dejando de lado otras prestaciones ambiciosas que nos proporcionan los SGBDOO.

Atkinson (citado por Martínez, 2001) propone la gestión de datos persistentes integrada a nivel de lenguaje, de forma que un lenguaje persistente se puede definir como un lenguaje que proporciona la habilidad de preservar datos entre las sucesivas ejecuciones de un programa e incluso permite que tales datos sean empleados por programas muy diferentes. Los datos en un lenguaje de programación persistente son independientes del programa y capaces de existir más allá de la ejecución y tiempo de vida del código que los creó. Trabajos en ese sentido han producido algunos resultados, como por ejemplo un sistema de programación persistente, a nivel experimental, para el lenguaje de programación Java, denominado Pjama.

Ventajas: (a) Acercamiento de los lenguajes de programación a las bases de datos;(b) Eliminación de la Desadaptación de Impedancias.

Desventajas: Dificultad para conseguir interoperabilidad entre diferentes lenguajes de programación.

Gestores de Objetos

Los gestores de objetos, también llamados almacenes de objetos persistentes proporcionan menos funcionalidad que los SGBDOO. Están diseñados para gestionar el almacenamiento persistente de un número de objetos relativamente pequeño y con bajos requerimientos de concurrencia. Son por tanto, poco apropiados para muchos usuarios. Normalmente ofrecen una distribución de datos limitada, un bloqueo de grano muy grueso, no proporcionan evolución de esquemas y carecen de un lenguaje de consulta o de un lenguaje de programación. Pueden asociarse con una extensión de un sistema de fichero.

La mayoría de estos gestores vienen en forma de API que será utilizada desde el lenguaje de programación correspondiente (C++ o Java principalmente), y actualmente existe una gran cantidad de ellos, sobre todo para Java (entre ellos, PSE Pro, Jeevan, StreamStore, etc.)

Bases de Datos Orientadas a Objetos

A la hora de desarrollar software en el que el modelo de objetos recoge bien su semántica, parece lógico emplear un método de gestión de la información en el que no sea necesario hacer una descomposición artificial de la misma. Para ello la posibilidad más lógica sería la de un SGBD (Sistema de Gestión de Base de Datos) que permitiese gestionar directamente los objetos.

Una base de datos orientada a objetos es una colección de objetos en los que su estado, comportamiento, y relaciones son definidas de acuerdo con un modelo de datos orientado a objetos, y un SGBDOO es un SGBD

que permite la definición y manipulación de una base de datos orientada a objetos. En el manifiesto del sistema de base de datos orientada al objeto (The Object-Oriented Database System Manifesto), se especificaron por primera vez las características que debería presentar un SGBDOO: Obligatorias (Cuadro 2), Optativas, y Abiertas. Sin embargo, durante los primeros años de existencia de estos sistemas no había en lo absoluto portabilidad ni interoperabilidad entre ellos. Con el fin de solucionar estos problemas surge el primer intento de estandarización llevado a cabo por ODMG ("Object Data Management Group")

Los SGBD orientados a objetos o lenguajes de programación orientados a objetos según Cattel, combinan características de orientación a objetos y lenguajes de programación orientadas a objetos con capacidades de bases de datos. Estos sistemas están basados en la arquitectura de un lenguaje de programación de base de datos. Las aplicaciones son escritas en una extensión de un lenguaje de programación existente, y el lenguaje y su implementación (compilador, preprocesador, entorno de ejecución) han sido extendidos para incorporar funcionalidad de bases de datos. El objetivo de estos sistemas es llegar a integrarse con múltiples lenguajes, aunque esto puede suponer un problema debido a lo cercano de la asociación que se requiere entre el sistema de tipos de la base de datos y el sistema de tipos del lenguaje de programación.

Actualmente existen una gran cantidad de productos etiquetados como SGBDOO (ObjectStore, POET, Jasmine, GemStone, etc.) y aunque la mayoría de ellos se proclaman conformes con el estándar, sigue existiendo una falta de interoperabilidad entre sus modelos de objetos.

Cuadro 2. Características Obligatorias en un SGBDOO

Sistemas Java Persistentes

El consorcio ODMG ha establecido las normas para que se defina la persistencia en java. Silberschatz et .al (2002) explica algunos detalles de la persistencia en Java. Java usa la persistencia por alcance. Los objetos no se crean explícitamente en la base de datos. En su lugar se dan los nombres a los objetos en la base de datos que sirven como raíces para la persistencia. Estos objetos son persistentes y también algunos objetos que sean alcanzables a partir de ellos.

La persistencia por alcance implica que los apuntadores persistentes deben ser del mismo tipo que los apuntadores transitorios. Otro resultado de la persistencia por alcance es que los objetos en la base de datos pueden volverse basura si el objeto se vuelve inalcanzable desde todas las raíces persistentes en la base de datos y ninguna transacción activa guarda un apuntador hacia el objeto. Tales objetos deben ser eliminados ejecutando periódicamente un procedimiento de recogida de basura en la base de datos. La recogida de basura se efectúa generalmente de forma concurrente con otras actividades de la base de datos.

Si un objeto de una clase se alcanza desde una raíz persistente, la clase debe hacerse persistente. Una clase normalmente se hace persistente (es decir, los objetos de esa clase pueden volverse persistentes) ejecutando un postprocesador en el código de esa clase generado por la compilación del programa java. También es posible hacer manualmente una clase insertando instrucciones apropiadas en el código java, pero este enfoque es relativamente complicado y no se recomienda. El postprocesador también inserta código para marcar automáticamente los objetos como modificados si son actualizados.

Cuando una transacción desea acceder a los objetos de una base de datos, se debe comenzar por uno de los objetos raíz desde la base de datos que la transacción busca por su nombre. El sistema va por el objeto raíz desde la base de datos en memoria. Las implementaciones pueden elegir ir a buscar todos los objetos referenciados inmediatamente desde el objeto raíz, o buscar los objetos referenciados en forma perezosa o tardía.

SISTEMAS GRID ("Grid Computing")

Definición

El término de sistemas Grid ("Grid Computing") fue establecido a mediados de los años 90 y actualmente se ha convertido en un área activa de investigación. Existen diferentes definiciones para los Sistemas Grid, pero la gran mayoría de los autores coinciden en que son colecciones de recursos heterogéneos, distribuidos geográficamente e interconectados como una sola unidad, para satisfacer las necesidades de los usuarios. Otra definición de Sistemas Grid que se trae a colación está basada, a su vez, en el llamado Sistema de Computación en Red ("NCS"), el cual consiste en un sistema virtual conformado por máquinas y redes en los que se acuerdan realizar trabajos en conjunto, compartiendo sus recursos en forma mancomunada. De allí, que un Sistema Grid también pueda ser visto como un Sistema de Computación en Red generalizado, diseñado para

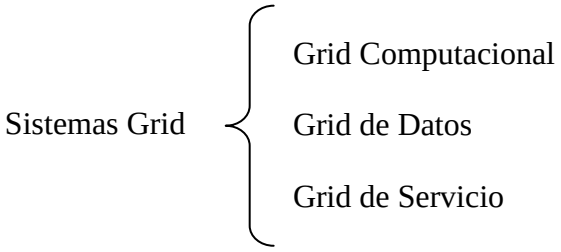
funcionar a niveles de gran escala y manejar computación y datos en forma transparente, Krauter et. al (2002).

Adicionalmente, se acota que el término "Grid" proviene de una analogía con la red que distribuye energía eléctrica ("Electric Grid"); esta comparación sugiere que el uso de todos los recursos disponibles a gran escala (escala mundial) debe llegar a ser tan natural como se obtiene la energía de la red eléctrica. Algunos Sistemas Grid también han sido conocidos con el nombre de metasistemas o metacomputadores; por esta razón, en el ámbito de este trabajo serán utilizados indiferentemente cualquiera de estos términos, ya que apuntan a un mismo significado; no obstante, existe una preferencia en usar el término Sistemas Grid o simplemente Grids, por considerarlo más actualizado.

Para complementar estas definiciones, seguidamente se presenta el punto de vista de los miembros de dos proyectos de desarrollo del software de Sistemas Grids ampliamente conocidos como lo son Globus y Legion. Por una parte, los miembros del proyecto Globus, Foster and Kesselman (1997), definen el término metacomputador para denotar a un supercomputador virtual conectado en red, construido dinámicamente a partir de recursos distribuidos geográficamente, enlazados por redes de alta velocidad. Además, definen el Grid desde un punto de vista arquitectural diciendo que "Un Grid Computacional es una infraestructura de hardware y software que provee un acceso seguro, consistente, penetrante y económico a las altas capacidades computacionales", Foster and Kesselman (1998). Por otra parte, los miembros del grupo Legion, Lindahl et. al. (1998) definen físicamente a los metasistemas como una colección de recursos (personas, computadores, instrumentos, bases de datos) separados geográficamente, conectada por una red de alta velocidad. Además, hacen una distinción entre lo que es un metasistema, en comparación a una simple colección de computadores. Ellos resaltan la importancia de la intervención del middleware o software intermedio, el cual hace posible identificar claramente esta diferencia; esta capa de software transforma una colección de recursos independientes en una única y coherente máquina virtual.

Clasificación

Según (Krauter, 2002) los Grids pueden ser clasificados, de acuerdo a su uso, en: (a) Grids Computacionales: Esta categoría denota a los sistemas que, en forma global, poseen una capacidad computacional más alta que la capacidad de cualquier máquina, constituyente del sistema, tomada en forma individual; (b) Grid de Datos: Se utiliza esta categoría en sistemas que proveen una infraestructura para sintetizar información a partir de repositorios de datos tales como bibliotecas digitales o almacenes de datos y (c) Grid de Servicios: Esta categoría es para sistemas que proveen servicios que no pueden ser provistos por una máquina en forma individual. Se ilustra esta clasificación en el Cuadro 3.



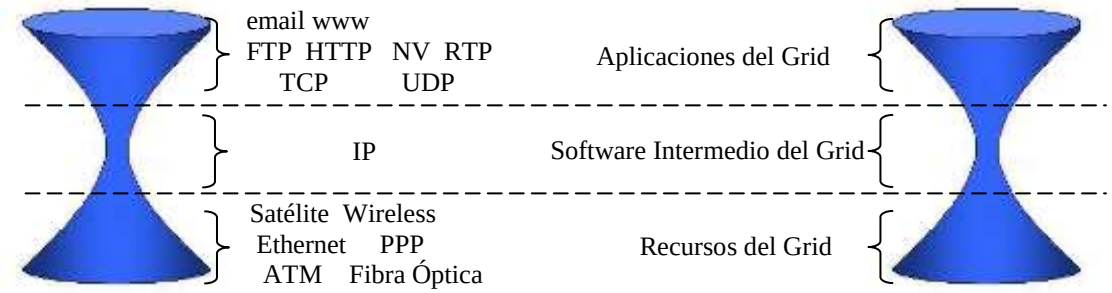
Cuadro 3. Clasificación de los Sistemas Grid.

De aquí en adelante se hará un mayor énfasis en el uso del término Grid Computacional o simplemente Grid, en lugar de hacer referencia a los Sistemas Grid en general; ya que, el Grid Computacional es el tipo de sistema que concierne a este trabajo.

Arquitectura

En la arquitectura del Grid se identifican los componentes fundamentales del sistema, se especifica el propósito y la función de estos componentes, y además se indica cómo estos componentes interactúan unos con otros. La arquitectura estándar del Grid presenta un enfoque estratificado. No obstante, existen diversas propuestas en cuanto al número y al nombre usado para la identificación de cada estrato o capa del Grid.

Se comienza por presentar una sencilla forma de visualizar la arquitectura estratificada para el software del Grid Computacional, representada en el modelo denominado "Reloj de Arena", mostrada en la figura 2a. Este modelo fue tomado de la filosofía de diseño de la Internet, en donde se coloca al protocolo IP ("Internet Protocol") como el centro del modelo; gráficamente, se ubica en el cuello del reloj, cumpliendo así con el siguiente principio: *"IP está sobre todo y todo está sobre IP"* Bellovin(2003); esto significa que: (a) Todas las aplicaciones dependen del IP; (b) El IP se ejecuta o "corre" sobre todas las redes; (c) El IP es el corazón o núcleo de todas las comunicaciones.



a. Internet ***b. Software del Grid***

RTP:Real Time Protocol.
PPP: Point-to-Point Protocol.
NV:Network Video.

ATM: Asynchronous Transfer Model.
UDP: User Datagram Protocol.
TCP: Transmission Control Protocol.

Figura 2. Modelo "Reloj de Arena" (HourGlass)

En la figura 2b, se muestra la relación del Grid con este modelo, presentando numerosas herramientas de alto nivel (Aplicaciones del Grid), las cuales

dependen de un conjunto de herramientas intermedias (Software Intermedio del Grid). Las herramientas intermedias proveen una colección de interfaces estandarizadas para acceder a una amplia variedad de servicios subyacentes ofrecidos por los recursos del Grid y sus sistemas operativos (Recursos del Grid).

Por otro lado, se presenta otra forma de visualizar la arquitectura del Grid, que se ha convertido en un estándar de facto, Sotomayor(2003). Foster et. al. (2001), han definido esta arquitectura, en un modelo de cinco capas, las cuales se muestran a continuación, en la figura 3:

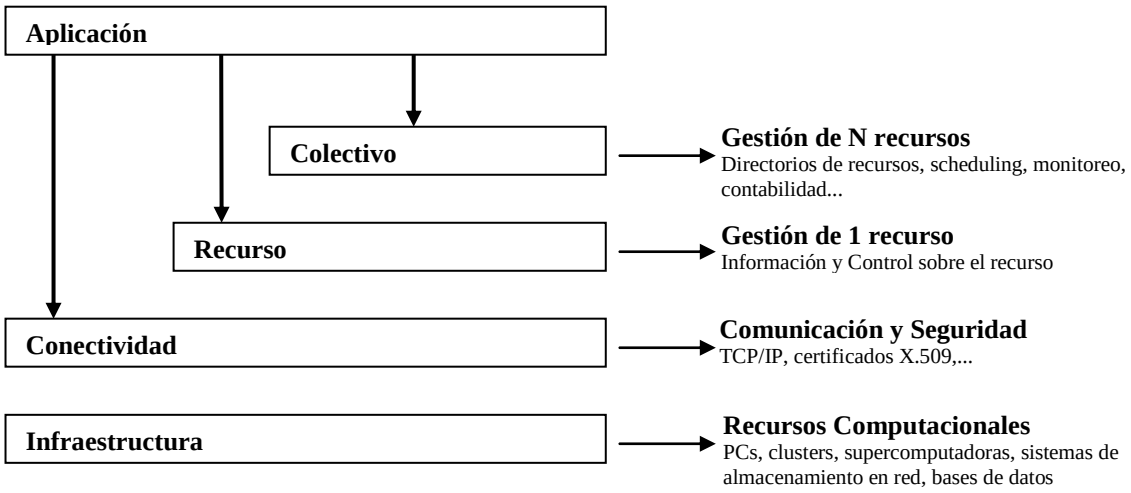


Figura 3. Arquitectura del Protocolo Grid

1. Infraestructura: Esta capa esta compuesta por los recursos computacionales que se desea compartir: PCs, clusters, supercomputadoras, y sistemas de almacenamiento, entre otros. También, se incluye la infraestructura de red y sus mecanismos de gestión y control.

2. Conectividad: Esta capa incluye los protocolos de comunicación y de seguridad que permite que los recursos computacionales se comuniquen; por ejemplo: protocolos TCP/IP (se están estudiando nuevos protocolos como Ipv6, que proporcionen mejor rendimiento en redes de alta velocidad), SSL (Secure Sockets Layer), certificados X.509.

3. Recurso: Esta capa incluye protocolos para el control y gestión de recursos individuales. Debe ser posible obtener *información* sobre un recurso (características técnicas, carga actual, etc), y también *control* sobre el recurso (acceso, arranque, gestión, parada, monitorización, contabilidad, auditoria del recurso, etc)

4. Colectivo: Esta capa engloba todos los servicios que permiten gestionar a su vez, a varios recursos: (a) Servicio de Directorio: localización de los recursos de interés; (b) Planificadores distribuidos: asignación de tarea a cada recurso; (c) Monitorización y diagnóstico de la ejecución de las distintas tareas de una aplicación; (d) Contabilidad: cálculo del costo de la utilización de varios recursos heterogéneos; (e) Acceso a datos distribuidos: gestión de réplicas.

5. Aplicación: Acceden a la infraestructura del Grid a través de las distintas capas. Según la exigencia de la aplicación, puede ser necesario pasar por todas las capas, o conectarse directamente a la infraestructura.

Para concluir con las diferentes opciones de arquitecturas presentadas en este trabajo, se muestra la propuesta de Lorch (2002), la cual consiste en un modelo de seis (6) capas, que será descrito a continuación:

1. Sistema Operativo y Acceso a los Recursos: Es la capa más baja del Grid y consiste en los recursos físicos junto con sus sistemas operativos respectivos. Puede proveer servicios simples para la transferencia de archivos (ejemplo, FTP), acceso remoto (ejemplo, SSH), recuperación de información a través de bases de datos o servidores web.

2. Administración de Recursos a bajo nivel: En esta capa se abarcan los paquetes de software que proveen mecanismos de paso de mensajes (ejemplo, librerías MPI) y sistemas de planificación que no forman parte del sistema operativo. Esta capa es muy importante, en especial, si el recurso físico está conformado, en sí mismo, por otros recursos individuales; este es el caso cuando un conglomerado (“cluster”) de equipos, es conectado en red representado como un único recurso.

3. Servicios Grid y Comunicación: Esta capa constituye el corazón o núcleo de un sistema Grid. Comprende servicios que proveen información sobre los recursos disponibles y brinda soporte para la reservación y sumisión del trabajo de estos recursos. Además, provee protocolos que habilitan una transferencia de datos eficiente en el Grid y métodos de comunicación que proveen seguridad, servicios de paso de mensajes entre los nodos del Grid, a un nivel más alto. Sin embargo, los servicios más importantes que son proporcionados en esta capa, son los servicios de autenticación y autorización. Globus y Legion proveen extensos servicios en esta capa.

4. Administración de Recursos a un alto nivel: Esta capa se coloca sobre la capa de Servicios del Grid y Comunicación, y provee servicios de asignación de recursos, basada en la información suplida por el servicio de información del Grid. Un servicio como este, usará los mecanismos de la capa inferior más inmediata, para localizar y reservar los recursos apropiados para cumplir con su cometido.

5. Interfaz de usuario del Grid y Flujo de trabajo: Los componentes de software para esta capa proveen servicios que permiten a los desarrolladores o usuarios definir, modificar y ejecutar estructuras de programas a un alto nivel de abstracción. Los servicios usan las capas inferiores para adquirir en forma transparente los recursos necesarios para iniciar la ejecución de programas.

6. Aplicación: Es la capa más elevada del Grid. Este es el lugar donde se ubican las interfaces de usuario, que proporcionan un servicio para dar solución a problemas específicos, ocultando al usuario la complejidad de la arquitectura subyacente.

La figura 4, muestra una correspondencia entre la arquitectura de 5 capas y la de 6 capas antes descritas, por un lado, y por el otro, muestra la ubicación de algunos de los numerosos proyectos de implementación de Grids Computacionales, existentes en la actualidad. Tomando en consideración los niveles de desarrollo alcanzados por las herramientas de software generadas en

cada proyecto, se han ubicando las mismas, abarcando los diferentes estratos o capas, de acuerdo a la funcionalidad o servicios que brindan al Grid.

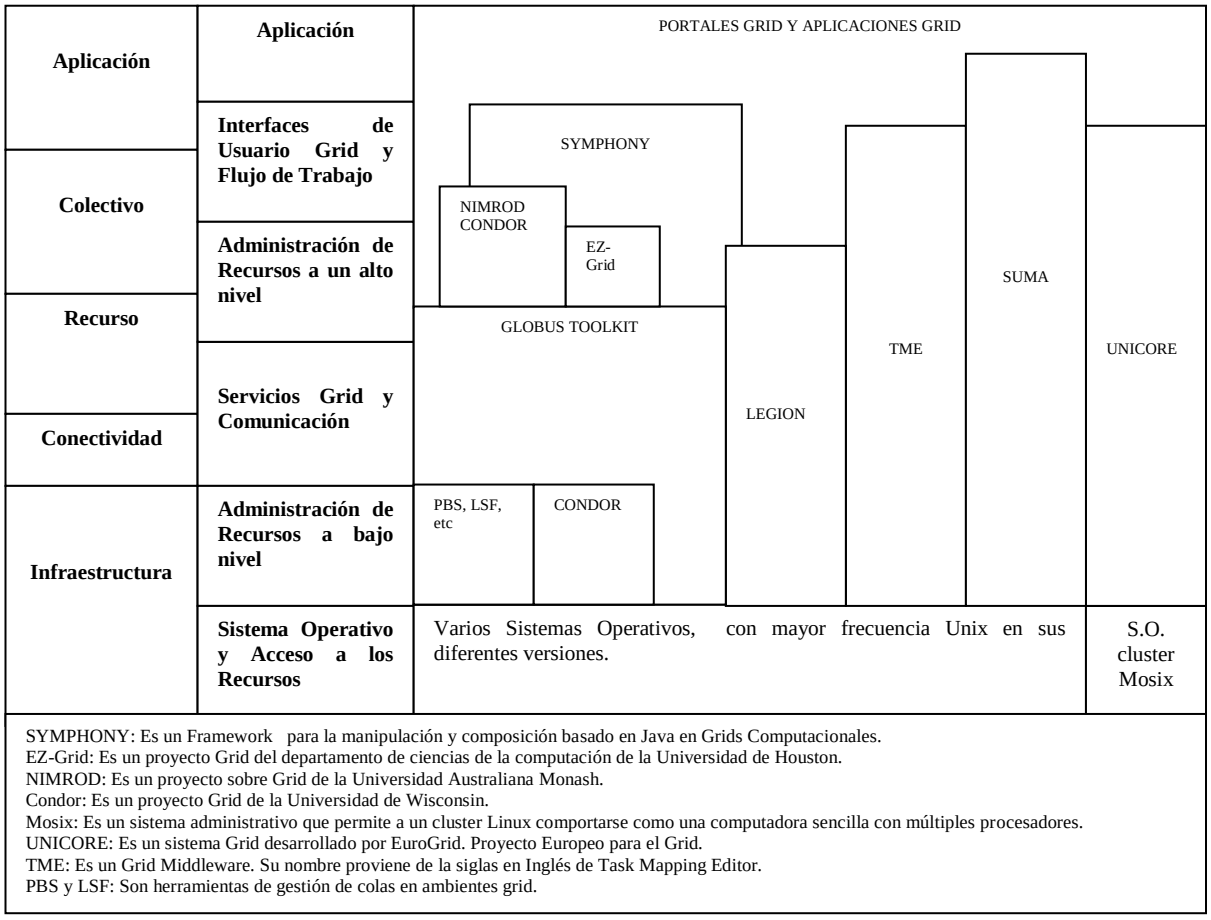


Figura 4. Implementaciones de Software Grid y su correlación con los modelos de arquitectura de cinco y seis capas.

Servicios del Grid

Uno de los aspectos más importantes que deben ser tomados en cuenta, en relación a los Servicios en general, son los protocolos. Un protocolo es un conjunto de reglas establecidas en un sistema de telecomunicación para intercambiar información. Una definición de protocolo describe cómo los elementos, de un sistema distribuido como el Grid, interactúan unos con otros para lograr un comportamiento

especificado, y la estructura de la información intercambiada durante esta interacción. De manera general, un servicio puede ser definido como una entidad disponible en una red para proveer una capacidad determinada. Según Foster et. al (2001), un servicio del Grid debería ser definido únicamente por el protocolo que el servicio maneja o "habla" y el comportamiento que éste implementa; esta idea pudiera ser expresada en forma de ecuación de la siguiente manera: *Servicio = Protocolo + Comportamiento.* Una definición de un servicio puede permitir una variedad de implementaciones.

Una Interfaz de Programación de Aplicaciones o "API" define una interfaz estándar para invocar un conjunto de funcionalidad específica; en el caso de un "API" orientada a objetos, se harían llamadas a un conjunto de subrutinas o invocación de los métodos de los objetos. Es importante entender la relación entre "API"s y protocolos; una definición de protocolos no nos dice nada sobre las "API"s que pueden ser llamadas desde un programa para generar mensajes a protocolos. Un único protocolo puede tener muchas "API"s; una única "API" puede tener múltiples implementaciones que eligen protocolos diferentes. Un "API" estándar hace posible la *portabilidad*; protocolos estándar hacen posible la *interoperabilidad.*

CAPÍTULO III

MARCO METODOLÓGICO

En este capítulo se describe la metodología empleada para llevar a cabo la recopilación de la información durante la investigación, así como también el método de organización y análisis de la información.

Naturaleza del Estudio

La modalidad de investigación establecida en este trabajo es la de Estudio de Proyectos, con la cual se presentó una solución factible a la problemática planteada apoyándose en un diseño de tipo documental, con lo cual se profundizó el conocimiento del área de estudio, se conoció el estado del arte en ese campo de la investigación y se nutrió con la experiencia de otros investigadores en trabajos similares.

Fase Diagnóstica

Esta investigación se encuentra enmarcada en el contexto de los Servicios de Sistemas Grid. A nivel global existen diversos proyectos, como los mencionados en el Marco Teórico, para el desarrollo de estas implementaciones. No obstante, el Universo de esta investigación se circunscribió a los desarrollos de implementaciones del Software del Grid (Middleware), que permitieran compartir los recursos disponibles de la Red Académica de Centros de Investigación y Universidades Nacionales (REACCIUN, 2004). Esto es debido a que el autor de esta investigación consideró más viable tratar de verificar los beneficios de la misma en este ámbito (el

ámbito nacional); es decir, tomando la información basada en los recursos de las instituciones miembros de la Red REACCIUN, que potencialmente puedan ser compartidos vía Grid. Para el desarrollo de esta investigación fue seleccionado el Grid Computacional denominado SUMA, como punto referencial o Muestra representativa de los Grids Computacionales basados en Java a nivel Nacional.

Métodos seleccionados para recopilar información

Para realizar el levantamiento de información en la investigación se utilizaron los siguientes métodos de recopilación:

1. **Entrevista a Expertos:** En esta investigación se contó con el apoyo del Grupo de Investigación de Sistemas Paralelos y Distribuidos de la Universidad Simón Bolívar, en especial la Profesora Yudith Cardinale (tutora de este trabajo) y el Profesor Carlos Figueira miembro de este grupo. Ambos profesores forman parte importante del equipo de desarrollado del Grid Computacional SUMA; es decir, son expertos en el área de los Grid Computacionales basados en java. Se realizaron una serie de entrevistas estructuradas con ellos, para obtener toda la información necesaria para precisar, en detalle, los aspectos más importantes a considerar para el diseño del Servicio propuesto en esta investigación.

   Adicionalmente, se asistió a alguno de los seminarios organizados por este grupo, en donde se abordaron temas que ayudaron al enriquecimiento de la percepción global del problema. De igual manera también se asistió como observador a algunas defensas de tesis de grado, a nivel de pregrado, orientados en esa misma dirección. Finalmente se participó en algunas de las reuniones ordinarias del grupo y se intercambio información vía Foros, Grupos de Discusión y Charlas (Chats) en Internet.

2. **Revisión Bibliográfica:** Consistió principalmente en la revisión de publicaciones de investigaciones relacionas con el área, Revisión de Tesis de Pregrado, Maestría

y Doctorales, y también la revisión de artículos publicados en los sitios web oficiales de proyectos relacionados con Grid Computacionales.

Fase de Factibilidad

Tecnológica

La factibilidad de este proyecto fue considerada desde el punto de vista tecnológico; es decir, los recursos tanto de hardware y software necesarios para confirmar la factibilidad tecnológica en el desarrollo de esta investigación.

Recursos

La existencia y disponibilidad de recursos de hardware y software hicieron posible la realización de este trabajo y la confirmación de su factibilidad tecnológica para considerar su implantación en trabajos futuros.

Plataforma Instalada

- Existe en el país una plataforma tecnológica ya instalada a nivel Nacional: La Red Académica de Centros de Investigación y Universidades Nacionales (REACCIUN, 2004), la cual permite la interconexión de las distintas redes nacionales de los sectores académicos, científicos y tecnológicos. La figura 5 muestra la topología de la red REACCIUN.

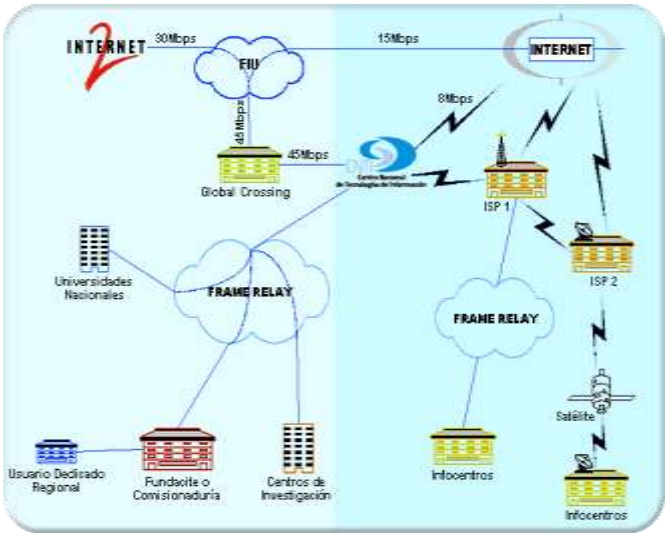


Figura 5. Topología de la Red REACCIUN

➢ Existen proyectos establecidos que buscan la mejora continua de esa plataforma (REACCIUN). Entre ellos se destaca el proyecto de REACCIUN2 (Red Académica de Centros de Investigación y Universidades Nacionales de Alta Velocidad), que no es más que el proyecto de Internet2 del Centro Nacional de Tecnologías de Información (CNTI) con el cual se interconectarán varios laboratorios de universidades nacionales y centros de investigación, con las redes internacionales experimentales de Internet de alta velocidad (Internet2). Es importante señalar, que según la información obtenida del sitio web oficial de Reacciun2 (REACCIUN, 2004) desde Abril del 2003, este proyecto fue suscrito al proyecto Ampath de la Universidad Internacional de Florida (FIU) y ya se ha instalado un enlace de 45 Mbps (DS3) con la empresa Global Crossing, de los cuales ya se están utilizando 15 Mbps para el Internet comercial y 30 Mbps están en reserva para ser utilizados en el proyecto de Internet2. La figura 6 muestra la topología de la red REACCIUN2.

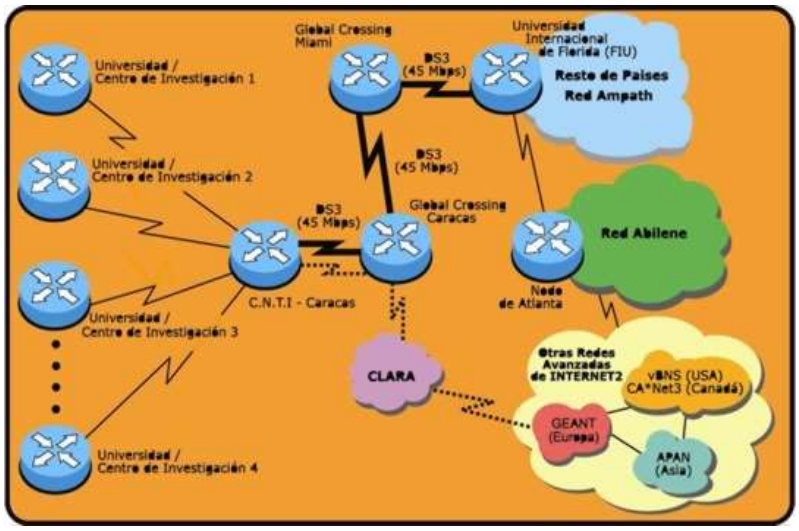


Figura 6. Topología de la Red REACCIUN2

Software Disponible

Cabe destacar que un gran apoyo para la verificación de la factibilidad de este proyecto, fue haber tenido la disponibilidad de acceder a un conjunto de herramientas de software, previamente implementadas, que proporcionaron el soporte necesario para avanzar con rapidez en el proyecto, al obtener la funcionalidad de las mismas, sin que el autor de esta investigación haya tenido que desarrollarlas personalmente. Adicionalmente, se recibió una ventaja de tipo económica al seleccionar este software bajo la modalidad de licencia libre, con lo cual se evitaron los grandes gastos relacionados con la adquisición de sistemas de software comerciales.

Entre otros el software usado fue:

- Visual Paradigm (Visual Paradigm, 2004): Esta herramienta fue empleada, especialmente, para representar el diseño del modelo propuesto mediante el lenguaje unificado UML. La versión utilizada fue la Edición Estándar 3.1, la cual proporciona una licencia académica. La única desventaja encontrada con

el uso de esta versión fue la impresión de una línea identificando a este producto en todos los diagramas diseñados con la herramienta.

- TJDO (TJDO, 2004): Es una implementación de código abierto de las especificaciones JDO, diseñado para obtener la persistencia de objetos, en bases de datos relacionales.
- MySQL (MySQL, 2004): Este el Sistema Manejador de Bases de Datos Relacionales utilizada en el proyecto para almacenar los objetos persistentes en tablas relacionales. La versión usada es la 1.4.
- ObjectDB (ObjectDB, 2004): Este es un Sistema Manejador de Bases de Datos Orientadas a Objetos, que implementa las especificaciones JDO y es empleada por el proyecto para obtener un espacio de almacenamiento que permita manipular directamente los objetos persistentes. Se usa la versión libre del producto.
- JDORI (JDORI, 2004): Es la implementación referencial de las especificaciones JDO (JSR12, 2004) de Sun, establecidas por la comunidad java (JCP, 2004). Es el estándar de java empleado en este proyecto para obtener una alternativa al manejo de la persistencia de los objetos java.
- Java J2SE (JAVA, 2004): Es la edición estándar para plataformas java, el cual es el lenguaje utilizado en el proyecto para realizar el prototipo funcional que permite la demostración de la funcionalidad del servicio propuesto.

Proceso de Desarrollo del Software

Para alcanzar los objetivos planteados en esta investigación se siguió un proceso de desarrollo de software denominado Proceso Unificado o RUP (Rational Unified Process), del cual se describen a continuación los aspectos más relevantes que fueron considerados para la realización del diseño del modelo propuesto en esta investigación.

Un proceso de desarrollo de software define *Quién* debe hacer *Qué*, *Cuándo* y *Cómo* debe hacerlo. No existe un proceso de desarrollo de software universal, el

mismo se determina por las características de cada proyecto: el equipo de desarrollo y los recursos, entre otras. En la Ingeniería de Software el objetivo es construir un producto de software o mejorar alguno existente, esto se ilustra en la siguiente figura 7:

Figura 7. Proceso de Ingeniería de Software.

RUP captura varias de las mejores prácticas en el desarrollo moderno del software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Puede ser visto como una guía de cómo usar efectivamente UML. Provee a cada miembro de un equipo, un fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Crea y mantiene modelos en lugar de enfocarse en la producción de una gran cantidad de papeles de documentación.

Con el Proceso Unificado se busca realizar un conjunto de actividades con la finalidad de transformar los requisitos de usuarios en un sistema de software. RUP se caracteriza por ser: Iterativo e Incremental, Dirigido por los Casos de Uso y Centrado en la Arquitectura.

Iterativo e Incremental: Pequeños proyectos que incorporan incrementalmente una nueva funcionalidad y cuyo desarrollo es una Iteración. De esta manera se obtiene un sistema robusto, se reduce el riesgo de tener un mal producto, también se reduce el riesgo de no obtener el producto en el tiempo previsto y permite atacar problemas con requerimientos incompletos. Dados los sistemas de software sofisticados de la actualidad, no es posible hacer de manera secuencial la definición completa del problema, diseñar la solución completa, construir el software y por último, probarlo. Ya que el descubrimiento de defectos en fases posteriores de diseño dan como resultado un aumento en los costos y la posible cancelación total del proyecto. La figura 8, ilustra el desarrollo iterativo.

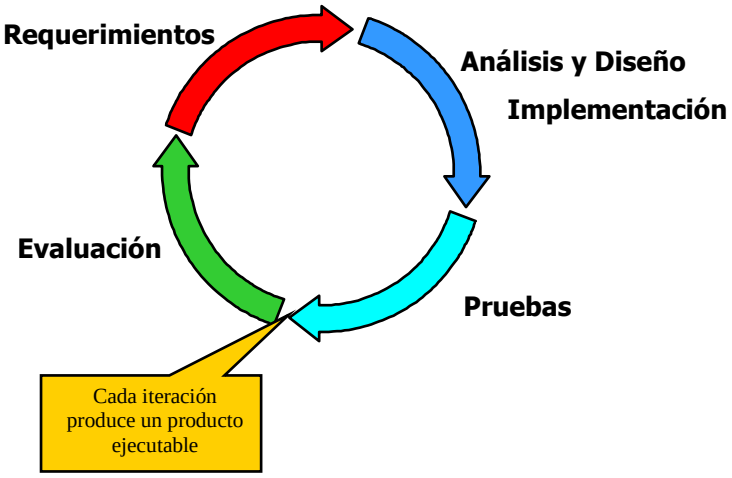


Figura 8. Desarrollo Iterativo.

Algunas de las características del Desarrollo Iterativo:

- ✓ Permite un entendimiento incremental del problema a través de refinamientos sucesivos.
- ✓ Habilita una fácil retroalimentación del usuario.
- ✓ Las metas específicas permiten que el equipo de desarrollo mantenga su atención en producir resultados.
- ✓ El progreso es medido conforme avancen las implementaciones.

Dirigido por Casos de Uso: Los Casos de Uso pueden ser vistos como los servicios que los actores requieren de un sistema y le proporcionan un resultado; es decir, proporcionan los requerimientos funcionales del sistema. Los requerimientos son fácilmente capturados y comunicados a través de los Casos de Uso. Por lo tanto, toda la funcionalidad del sistema puede ser descrita mediante un modelo de Casos de Uso. Los Casos de Usos pueden funcionar como instrumentos importantes de planeación, ya que los mismos dirigen el trabajo desde el análisis hasta las pruebas. Esto se ilustra en la figura 9:

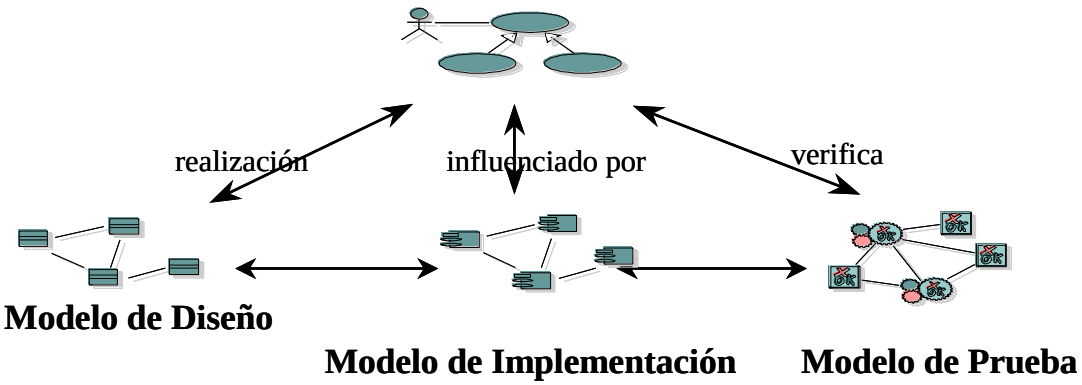


Figura 9. Dirección del trabajo mediante Casos de Uso.

Centrado en la Arquitectura: La arquitectura define la Forma o Estructura del sistema. Se describe mediante Vistas que incorporan entre 5 y el 10 % de los Casos de Uso. Las vistas de los modelos de Casos de Uso contienen a los Actores y los casos de Uso relevantes; la vista del modelo de Análisis: Clases de Análisis (Interfaz, Control y Entidad) de los Casos de Uso anteriores; la vista del modelo de Diseño: Subsistema y Clases de Diseño derivadas de las Clases de Análisis; la vista del modelo de Despliegue: Arquitectura Física por medio de nodos; la vista del modelo de Implementación: Componentes de las Clases relevantes.

El Proceso Unificado consta de cuatro fases: Inicio, Elaboración, Construcción y Transición, las cuales pueden ser apreciadas en la figura 10. Cada fase puede tener una o varias iteraciones y las cuatro fases producen una generación. Una iteración es un ciclo de desarrollo completo dando como resultado una entrega de producto ejecutable (interna o externa).

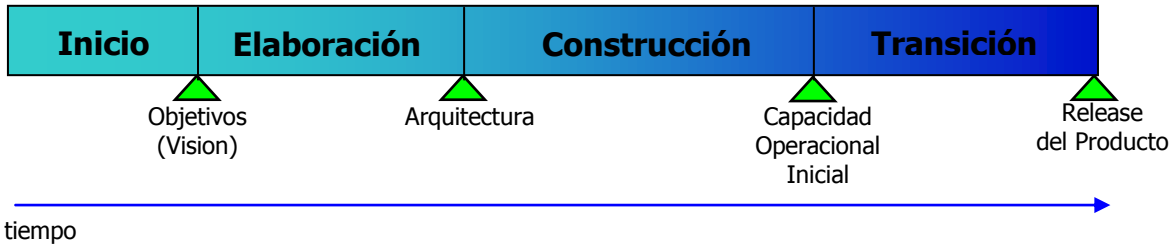


Figura 10. Fases del Proceso Unificado RUP.

Fases: Cada una de las fases busca alcanzar algunos objetivos específicos los cuales se describen a continuación:

Fase de Inicio: También puede ser llamada fase de inspección o concepción. Su finalidad básica es Comprender el Problema y Determinar su Ámbito; es decir, se establece la idea o la visión del proyecto. En esta fase se realiza una planificación del Proyecto y su Alcance. Las principales actividades a realizar en esta fase son: Determinar las funciones fundamentales del sistema; Realizar un Esquema tentativo de la Arquitectura, Realizar un Plan del Desarrollo del Proyecto y el Costo. Algunos de los artefactos (piezas de información) obtenidos en esta fase son: Un documento con la visión del proyecto, y El modelo de casos de uso.

Fase de Elaboración: Entre sus objetivos básicos se encuentran: Capturar los Requerimientos y Producir un Prototipo. Se establece la línea base de la arquitectura. Las principales actividades son: Analizar el Dominio del Problema; establecer una Base Arquitectónica. Desarrollar un plan comprensivo mostrando cómo el proyecto será completado. Durante las fases de inicio y elaboración se capturan el 80% de los requerimientos. Entre los artefactos obtenidos en esta fase se destacan: Descripción de la arquitectura del software y un prototipo ejecutable de la arquitectura.

Fase de Construcción: Diseño e Implementación. Primer Producto Operacional. Se desarrolla el producto completo. Las actividades básicas en esta fase: Incrementar la línea base de la arquitectura hasta obtener el sistema completo. Describir los requerimientos restantes, Refinar el diseño, Completar la implementación y las pruebas del software. Construir el producto, la arquitectura y los planes hasta que esté listo para ser enviado a la comunidad de usuarios. La implementación es el flujo de

trabajo fundamental en la fase de construcción. El artefacto principal es el producto de software integrado a la plataforma adecuada.

Fase de Transición: La finalidad de esta fase es Producir y Suministrar el Producto Final; es decir, se realiza la transición del producto a los usuarios, en lo cual se incluye: la manufactura, el envío, el entrenamiento, el soporte y el mantenimiento del producto, hasta que el cliente esté satisfecho. Las actividades básicas son: prueba de la versión beta; fabricación y suministro del producto; formación del cliente y ayuda técnica, mantenimiento y corrección de errores; determinar si ya se han satisfecho todos los objetivos o si se debe iniciar otro ciclo de desarrollo.

En RUP se pueden observar algunos flujos de trabajos fundamentales: Modelado del Negocio, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Despliegue y la Configuración y Gestión de Cambios, Gestión del Proyecto y Entorno. El proceso puede describirse en dos dimensiones, o a lo largo de dos ejes, tal como se muestra en la figura 11: el eje horizontal, representa el tiempo y muestra el aspecto dinámico del proceso, expresado en términos de ciclos, fases, iteraciones y metas; el eje vertical, el aspecto estático del proceso: cómo está descrito en términos de actividades, artefactos, trabajadores y flujos de trabajo.

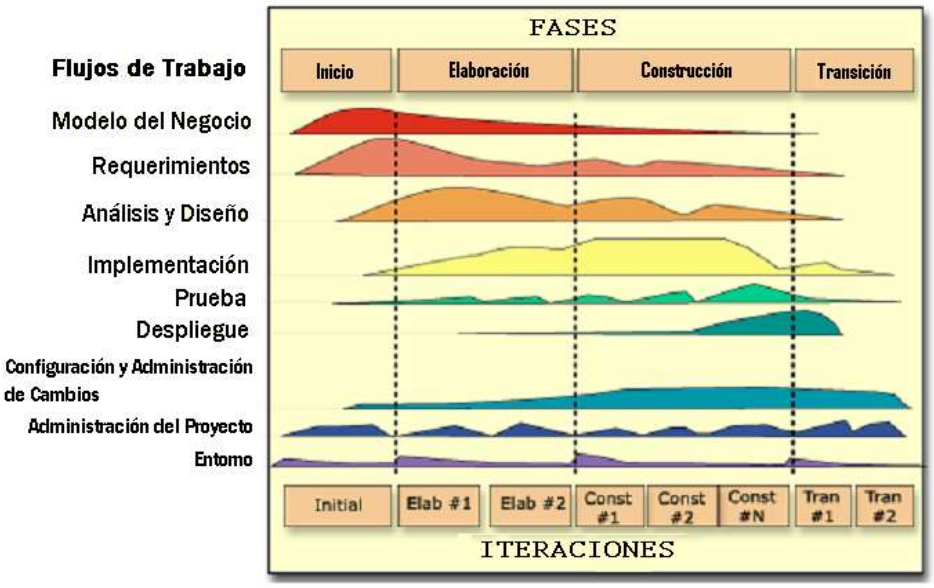


Figura 11. Descripción del Proceso Unificado en dos dimensiones.

Flujos de Trabajo: Un flujo de trabajo es una secuencia de actividades que producen un resultado de valor observable. En términos de UML un flujo de trabajo puede ser expresado como un diagrama de secuencia, un diagrama de colaboración o un diagrama de actividades.

Requerimientos: Este flujo de trabajo tiene la finalidad de: establecer lo que el sistema debe hacer (Especificar Requerimientos); definir los límites del sistema; definir las interfaces de usuario y la realización de una estimación del costo y tiempo de desarrollo.

Análisis y Diseño: En este flujo de trabajo se busca trasladar los requerimientos en especificaciones de implementación. El análisis transforma los casos de uso en Clases y el diseño refina el análisis para poderlo implementar.

Implementación: Los objetivos principales en este flujo son: implementar las clases de diseño como componentes (ejemplo: archivos fuentes); asignar los Componentes a los nodos; probar los componentes individualmente e integrarlos en un sistema ejecutable.

Pruebas: Verificar la integración de los componentes; verificar que todos los requerimientos hayan sido implementados y Asegurar que todos los defectos detectados hayan sido resueltos antes de la distribución.

Desplieque: Asegurar que el producto esté preparado para el Cliente y proceder a su entrega y recepción por parte del cliente.

CAPÍTULO IV

PROPUESTA DEL ESTUDIO

MODELO DEL SERVICIO DE OBJETOS PERSISTENTES EN GRID COMPUTACIONALES

Introducción

La Ingeniería del Software, como cualquier otra Ingeniería, requiere *modelar* tanto el problema como sus posibles soluciones. Los modelos ayudan a determinar qué información es requerida y qué datos se necesita recoger para obtener esa información. Adicionalmente la *abstracción* y el *rigor* en los modelos, permite que los mismos sean usados como medios para *especificar, analizar y evaluar* las propiedades del producto final que se esté desarrollando.

En este capítulo se presenta la aplicación de un enfoque sistemático y disciplinado de principios y métodos de Ingeniería del Software para el desarrollo de un modelo que surge como una posible solución al problema planteado en esta investigación. La aplicación del Proceso de Desarrollo de Software Unificado (RUP), en sus fases de Inicio y Elaboración orientadas a la obtención de un Diseño que modele el Servicio Manejador de Objetos Persistentes (SMOP) propuesto, conforma principalmente la estructura en que se divide este capítulo.

Se comienza con la descripción de los aspectos más relevantes que deben ser considerados en el Desarrollo del SMOP y se termina con los resultados obtenidos de una demostración de la funcionalidad de este Servicio, mediante el uso de un prototipo funcional.

Definición del Servicio Manejador de Objetos Persistentes (SMOP)

Descripción

Un Servicio Manejador de Objetos Persistentes es una solución de software que permite realizar las operaciones básicas para la administración de la persistencia de un objeto, las cuales son: Inserción, Consulta, Eliminación y Modificación. En el contexto de los Grids Computacionales basado en Java, un Servicio Manejador de Objetos Persistentes tiene la misión de gestionar la persistencia de los Objetos que administra el Grid. Sin embargo, para efectos de esta investigación este Servicio es aplicado sólo a aquellos objetos que representan la información persistente de los Recursos disponibles en el Grid, tales como: *Institución, Dominio, Recurso, Almacén, Servicio, Rol y Usuario.*

A continuación se describen los escenarios en los que puede operar el SMOP, con la finalidad de obtener las especificaciones funcionales del sistema y establecer algunas pautas que permitan una mayor comprensión de las características presentes en esta propuesta.

En este trabajo el SMOP puede ser apreciado desde dos puntos de vista, tomando en consideración el tipo de acceso que se tenga al mismo:

1. **Acceso Local:** Desde este punto de vista el Servicio se caracteriza por presentar un único *actor* principal el cual es llamado Administrador Local. Ciertamente, este administrador tiene acceso a todas las operaciones (*métodos*) que brinda el Servicio, y para ello utiliza una Interfaz Gráfica de Usuario ("GUI") que facilita la ejecución de las mismas. En este caso, el Servicio se desempeña con una especie de arquitectura "monolítica", ya que los componentes del software que lo conforman y que son necesarios para su ejecución: Interfaces, Lógica y Base de Objetos (Ver Glosario), estarían ubicados en el mismo espacio físico; es decir, en el mismo Servidor.

Cualquier Institución propietaria de una cantidad determinada de Recursos, es responsable de la custodia de los mismos, utilizando para ello las políticas de

administración de recursos propias de cada Institución. Si la Institución decide participar en un Grid Computacional debe especificar de alguna manera, cuáles son los recursos con los que va a participar. Para ello debe: (a) Agrupar sus recursos como mejor le convenga a la Institución; (b) Seleccionar un recurso de computación y colocarlo como Servidor en cada grupo seleccionado; (c) Asignar un responsable de la Administración Local a cada Servidor.

El Administrador Local utiliza las bondades que ofrece esta modalidad para realizar un registro detallado de los recursos de la Institución que están bajo su responsabilidad, en el equipo asignado para tal fin (Servidor). Cada recurso es representado a través del SMOP como un objeto persistente y es el deber del Administrador Local mantener actualizada la información propia de cada recurso. Debe existir una sola instancia del SMOP por cada Servidor. Cada Servidor maneja una sola Base de Objetos.

Para el soporte de cada Base de Objetos, se debe escoger alguno de estos tipos de Sistema de Almacenamiento: Base de Datos Relacional, Base de Datos OO o un Sistema de Archivo. El MySQL (http://www.mysql.com) versión de Software Libre (Open Source GNU GPL) es el producto utilizado en este proyecto para representar a las Bases de Datos Relacionales. El ObjectDB (http://www.objectdb.com) en su edición libre, es el producto utilizado en esta investigación para representar a las Bases de Datos OO y el File/Object Store (FOStore) es un mecanismo de almacenamiento que se coloca directamente sobre el Sistema de Archivos. FOStore viene con la implementación referencial del JDO (http://jcp.org/aboutJava/communityprocess/final/jsr012/index2.html).

Finalmente, el Administrador Local es el encargado de registrar en el Grid, la Base de Objetos que contiene la información de los recursos que pertenecen al Servidor de la Institución, para permitir el acceso remoto a la misma por parte de los componentes del Grid; para este proyecto, solo se toma el Planificador del Grid.

2. Acceso Remoto: Desde este punto de vista el Servicio se caracteriza por presentar como *actor* principal al Planificador del Grid, el cual es el componente del Grid encargado de la localización y asignación de los recursos disponibles para la

ejecución de procesos en el Grid. Este trabajo no es tarea fácil, debido a que los recursos que maneja el Grid cambian constantemente; es decir, éstos pueden ser incorporados y liberados del Grid en cualquier momento y sin previo aviso. Por lo tanto los mecanismos que emplean los Planificadores deben atender diligentemente a estas variaciones, para permitirles reorganizarse de manera eficiente y a su vez permitir la toma de buenas decisiones en las asignaciones de los recursos, basadas en diversos factores de decisión, tales como: rendimiento de la CPU, rendimiento de la red, cantidad de memoria y adecuación de la arquitectura al proceso que se desea ejecutar, entre otros.

Debido a la naturaleza distribuida de un Grid computacional, por lo general existen más de un Planificador de Recursos, por tal motivo se debe mantener una relación lo suficientemente coordinada para realizar sus labores sin tropiezos. La figura 12 muestra un ejemplo de la coordinación existente entre diferentes Planificadores (Sch) del Grid Computacional SUMA, utilizando una estructura jerárquica entre ellos, en donde se destacan unos Planificadores denominados Meta-Planificadores o Meta-Schedulers (DRBroker).

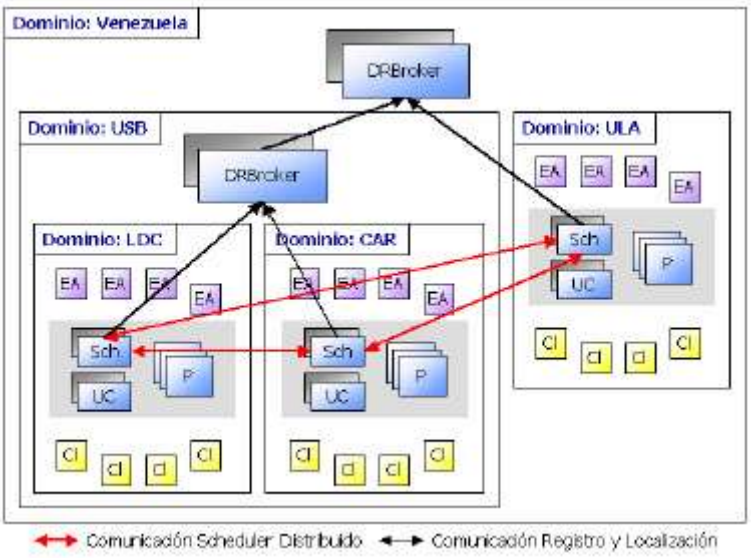


Figura 12. Vista de un Planificador distribuido (Planificadores y Meta-Planificadores) para SUMA.

Es importante establecer con mucha claridad, que la atención de los mecanismos de localización y asignación de Recursos por parte de los Planificadores del Grid y la coordinación entre ellos, se encuentra fuera del alcance de esta investigación, debido a que la misma forma parte de las responsabilidades de otra de las capas del Grid. No obstante, con la finalidad de realizar algunas pruebas utilizando un Prototipo Funcional del Servicio que se derive de esta investigación, ha sido necesario utilizar una estrategia de "simulación" o Demostración, en donde se pueda apreciar el acceso remoto que realiza un Planificador del Grid al Servidor del SMOP y las operaciones que se pueden realizar con el mismo. Para esto se tomó en cuenta la participación de un solo Planificador del Grid para n cantidad de recursos. Aunque en teoría este Servicio puede funcionar considerando el número total de Planificadores que contenga el Grid, al no tomar en cuenta para este proyecto la coordinación entre estos Planificadores por las razones ya expuestas con anterioridad, no tendría mucha relevancia considerar más de un Planificador en forma aislada y por lo tanto, se considera suficiente realizar el estudio con un solo Planificador.

Características

La propuesta se caracteriza por los siguientes aspectos:

- **Es Escalable:** Pueden ser incorporados tantos Servidores SMOP como se requiera, simplemente con registrarlo al Planificador del Grid. Desde la perspectiva del Planificador, el total de Servidores SMOP registrados conforman una gran Base de Objetos, con la que cuenta para atender sus necesidades del manejo de objetos persistentes. La única restricción es un límite máximo de atención a los Servidores pre-establecido para cada Planificador.
- **Usa Tecnologías Abiertas:** De esta manera se reduce el costo del proyecto, sin problemas de licencias, facilita su implementación en forma independiente de la plataforma y se mantiene un mayor control sobre el código que maneja este Servicio.

- **Se basa en Estándares:** El uso de estándares proporciona *portabilidad*, permitiendo que una aplicación se pueda ejecutar sobre sistemas distintos con mínimas modificaciones. Los estándares también proporcionan *interoperabilidad* permitiendo que una aplicación pueda acceder a varios sistemas diferentes.
- **Maneja Sistemas de Almacenamiento heterogéneos:** Esta característica se deriva justamente de la anterior, ya que debido al uso de estándares es posible que el Servicio pueda trabajar apoyándose en diversos Sistemas de Almacenamiento heterogéneos, como por ejemplo: Las Bases de Datos Relacionales, Las Bases de Datos Orientadas a Objetos o directamente sobre un Sistema de Archivos.

Estructura del Modelo

El resto de este capítulo se fundamenta en el seguimiento de las fases del proceso de desarrollo de Software de la metodología RUP. En la figura 13 se muestran unos cuadrados de líneas punteadas sobre la ilustración del Proceso general de RUP, esto quiere decir, que esta investigación se concentró en generar el modelo esperado del Servicio, fundamentalmente desarrollando las fases de inicio y elaboración por un lado, y los flujos de trabajo denominados Requerimientos, Análisis y Diseño por el otro. De esta manera se han obtenido una serie de diagramas o "artefactos" que proporcionan una especie de planos del Servicio, que en definitiva conforman "El Modelo del Servicio" que permite la implementación final del mismo. Para la realización de los diagramas se ha utilizado una herramienta de modelaje UML llamada Visual Paradigm 3.1 Edición Estandar (www.visual-paradigm.com).

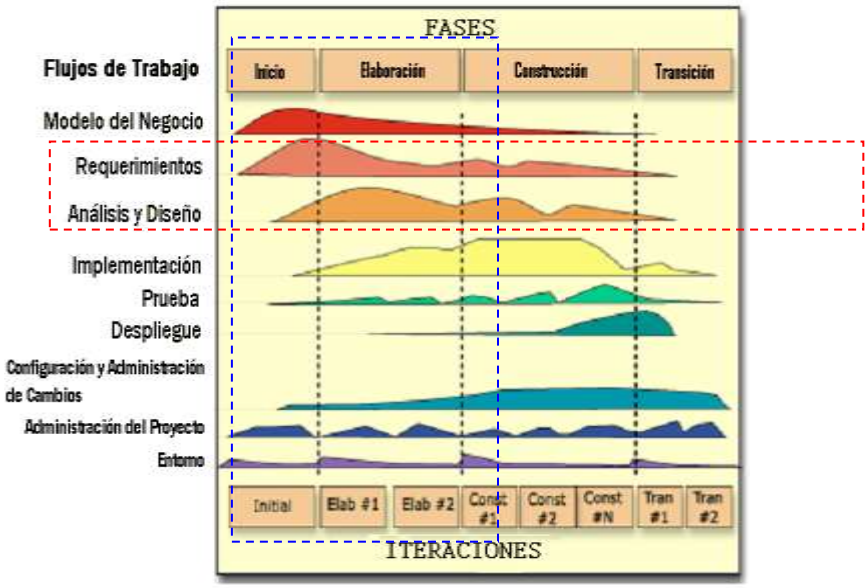


Figura 13. Fases del Proceso Unificado usadas en el Proyecto.

Fase de Inicio

Componentes del Proceso

1. Visión
2. Arquitectura
3. Análisis de Requerimientos

Fase de Elaboración

Componentes del Proceso

4. Análisis del Sistema
5. Diseño del Sistema

Fase de Inicio

Seguidamente se presenta la visión del proyecto como uno de los artefactos esenciales utilizados en esta fase:

1. Visión

La administración de la información de los recursos compartidos en un Grid Computacional basado en Java, implica el manejo de los Objetos Persistentes que representan a dichos recursos. El término manejo de objetos en este proyecto también puede ser utilizado igualmente como administración o gestión de objetos y significa la realización de operaciones que permiten mantener actualizados a estos objetos persistentes. Estas operaciones a las cuales se hace referencia son: Inserción, Modificación, Eliminación y Consulta de Objetos. Estos Objetos Persistentes se alojan en los diversos sistemas de almacenamiento que son colocados a disposición por las Instituciones participantes en el Grid. Estos sistemas de almacenamiento son generalmente heterogéneos y se encuentran dispersos geográficamente. Al tratar de operar sobre Objetos Persistentes sería lógico pensar en que los sistemas de almacenamiento involucrados en estas operaciones deberían ser Orientados a Objetos. Sin embargo, en un Grid computacional pueden ser encontrados sistemas de almacenamiento de otro tipo como por ejemplo, las Bases de Datos Relacionales.

El problema del manejo de los Objetos Persistentes en Grid computacionales basados en Java, es un problema que afecta directamente a los desarrolladores del Grid y trae consigo inconvenientes adicionales al tratar de administrar los Objetos persistentes sobre sistemas de almacenamiento tan heterogéneos. Uno de estos inconvenientes es la llamada desadaptación de impedancias Objeto-Relacional encontrada al tratar de manipular entidades que se encuentran bajo un enfoque Orientado a Objetos en sistemas que no lo soportan completamente.

Como consecuencia, los desarrolladores del Grid deben agregar cantidades de líneas de código de programación extra para solventar estos problemas, en lugar de

dedicar su esfuerzo a la atención de los puntos focales del desarrollo del Grid, en niveles o estratos más elevados, como lo son: La Planificación, Rendimiento, Seguridad, Confiabilidad, Estabilidad y Tolerancia a Fallas, entre otros. Todo esto conlleva a un incremento de la complejidad del sistemas que dificultan aún más su mantenimiento.

Una solución exitosa, sería proporcionar un Servicio Manejador de Objetos Persistentes (SMOP) que libere a los desarrolladores del Grid de los detalles internos de la administración de estos objetos, sobre los sistemas de almacenamiento heterogéneos, permitiendo un acceso uniforme y transparente, en forma tanto local como remota, a los Servidores de los almacenes de objetos, que se encuentran geográficamente distribuidos y que conforman la Base de Objetos del Grid, conteniendo la información de sus recursos disponibles.

El acceso a este servicio se pudiera proporcionar utilizando una Interfaz Gráfica de Usuario ("GUI") para los Administradores locales (de los Servidores de cada almacén de objetos) y también disponiendo de métodos de interfaces remotas para el acceso de los Componentes del Grid (en sus diferentes estratos) que lo requieran, en este caso los Planificadores.

2. Arquitectura

Desde el punto de vista de la arquitectura del servicio ha sido necesario considerar algunos elementos importantes para seleccionar una posible alternativa. Entre ellos se destacan, por un lado, el contexto en donde funcionará el SMOP, es decir, los Grids Computacionales. Los Grids pertenecen a la categoría de sistemas distribuidos y por esta razón los servicios asociados al trabajar con Grid deben presentar mecanismos que permitan la ejecución de procesos en distintos espacios físicos o equipos de computación. Por otro lado, el hecho de que el tipo de Grid Computacional sea basado en Java, implica el uso de un paradigma Orientado a Objetos a través del Lenguaje de Programación Java. Por esta razón se ha seleccionado el uso de un mecanismo de Java para la Invocación de Métodos Remotos ("RMI"), cuya

arquitectura es presentada en la figura 14 y determina a su vez, la arquitectura Cliente / Servidor del SMOP.

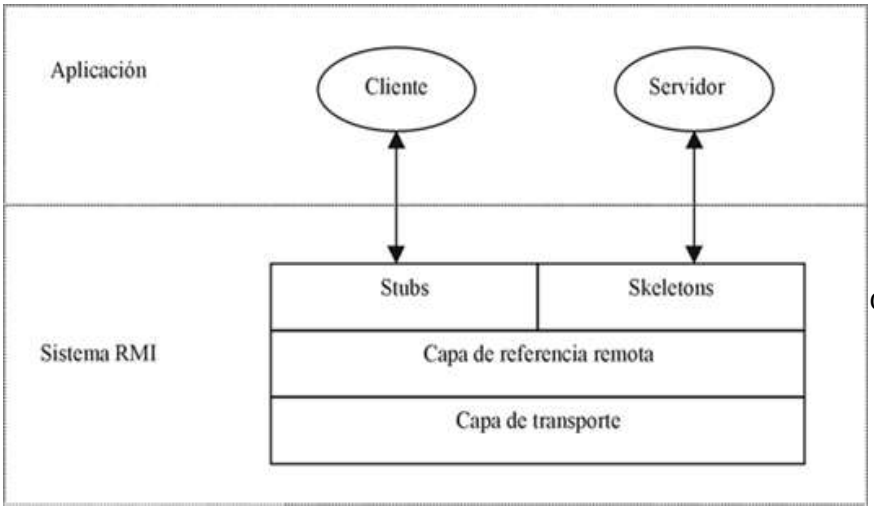


Figura 14. Arquitectura RMI.

En la figura 14 se pueden apreciar los componentes Stubs y Skeletons (Ver Glosario) propios de la arquitectura RMI, los cuales son los representantes del Servidor en el Cliente y representante del Cliente en el Servidor respectivamente. Además, los Stubs y Skeletons permiten realizar los procesos de Marshalling/UnMarshalling (Ver Glosario) de la información, para garantizar que la misma sea utilizable en ambientes heterogéneos.

El sistema RMI seleccionado conforma la capa de conectividad del SMOP propuesto como se observa en la figura 15. Allí se puede apreciar adicionalmente como esta capa de conectividad pudiera ser sustituida perfectamente por cualquier otro sistema como: CORBA, SOAP, entre otros.

Esta figura 15, ilustra una visión general del conjunto de capas que intervienen en el SMOP conformándose así, un diseño estructural del mismo. Se comienza con la instalación del servicio en un Servidor que se pretenda integrar al Grid y que posea algún sistema operativo que soporte la ejecución de la Máquina Virtual de Java (JVM). Una vez instalado el servicio podrá ser configurado para aprovechar el sistema de almacenamiento que sea elegido para su funcionamiento, entre ellos se

mencionan los siguientes: (a) El sistema de archivos dispuesto por el sistema operativo; (b) Un sistema de almacenamiento basado en una Base de Datos Relacional o (c) Un sistema de almacenamiento basado en una base de datos Orientada a Objetos. Estos elementos conforman otra de las capas del servicio denominada base de objetos.

Finalmente, se observa una capa de servicios de persistencia de objetos java, la cual está orientada, en este proyecto, para la satisfacción de los requerimientos remotos del Planificador del Grid, y para la satisfacción de los requerimientos locales del Administrador del Servidor. Todo esto por medio de la utilización de implementaciones estándares de las especificaciones JDO de java, el cual fue el mecanismo utilizado en este proyecto para el manejo de la persistencia.

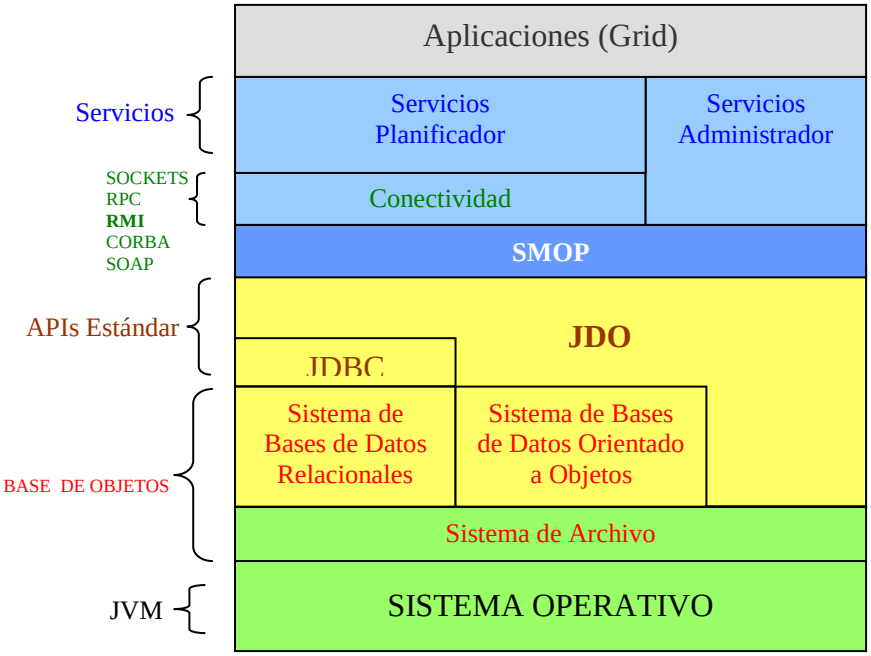


Figura 15. Capas de Software que interactúan en el SMOP.

3. Análisis de Requerimientos

En esta fase se han definido los límites del sistema, se han especificado los requisitos o requerimientos del sistema, es decir se ha establecido lo que el sistema debe hacer (*el qué*). Estos requerimientos se capturaron a través de algunos Casos de

Uso. El modelo de casos de uso contiene un conjunto coherente de los roles que desempeñan los usuarios (actores) de los casos de uso, cuando interactúan con el sistema. Este modelo de casos de uso, se presenta como un diagrama en donde se muestran sus actores, casos de uso y relaciones. La siguiente figura 16, presenta el modelo de casos de uso del sistema:

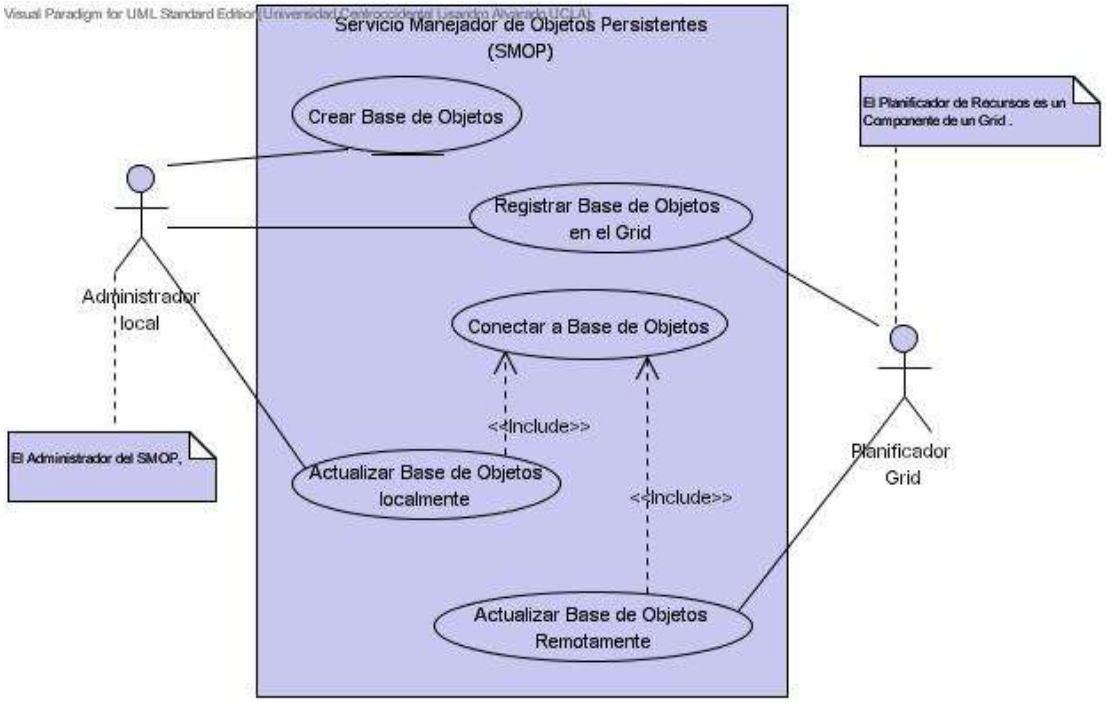


Figura 16. Modelo Contextual de Caso de Usos del Sistema.

Diccionario de Actores:

Los actores del Sistema SMOP fueron identificados:

- **Administrador Local:** Como su nombre lo indica, es la persona encargada de administrar los objetos persistentes que representan los recursos disponibles con los que la institución participará en el Grid. Estos objetos persistentes se encuentran alojados en los diversos sistemas de almacenamiento que se hayan colocado a la disposición, y para cada uno de ellos es necesario desplegar un Servidor SMOP que permita realizar la gestión local de dicho sistema de almacenamiento según corresponda. El Administrador local cuenta con una interfaz gráfica mediante la cual puede mantener un registro actualizado de los recursos de la Institución (*Institución*), utilizando las operaciones básicas del Servicio: Insertar Objeto, Modificar Objeto, Eliminar Objeto y Consultar Objeto, también utilizando el apoyo de un árbol jerárquico, como se muestra en la 17, para organizar los recursos según el orden particular de cada institución y especificar: los Dominios Administrativos (*Dominio*) de la institución; los *Recursos* asignados a ese Dominio según sea el tipo de *Recurso*: *Almacén*, *Computo* o *Servicio*; los *Usuarios* que pueden utilizar esos *Recursos* y los *Roles* que desempeñan para cada *Recurso*.

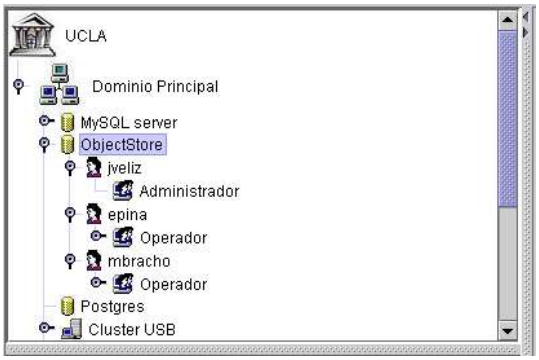


Figura 17. Árbol jerárquico de Recursos disponibles en una Institución del Grid.

- **Planificador del Grid:** El planificador representa al Componente del Grid encargado de la administración de los recursos del Grid. El planificador tiene la

responsabilidad de localizar un recurso disponible en el Grid, y luego asignarlo a la ejecución de una aplicación solicitada por un Cliente del Grid. Por esta razón el Planificador se convierte en el Cliente por excelencia del SMOP, debido a que con este servicio se podrá obtener una alternativa para controlar, en forma remota, los recursos disponibles en el Grid (Objetos Persistentes), en forma transparente de los sistemas de almacenamiento heterogéneos. El Planificador puede ejecutar todas las operaciones básicas del servicio: Insertar Objeto, Modificar Objeto, Eliminar Objeto y Consultar Objetos, con la particularidad que estas operaciones se realizan utilizando la capa de Conectividad para alcanzar la comunicación con el Servidor SMOP, el cual contiene el repositorio de los Objetos Persistentes.

Diccionario de Casos de Uso:

En base al modelo contextual de los casos de uso del sistema, se hace una breve explicación de cada uno de ellos. La descripción detallada en forma textual de todos los casos de uso que aparecen en esta sección pueden ser ubicados en el ANEXO B.

- **Crear Base de Objetos:** El Administrador local utiliza este caso de uso para crear el repositorio o almacén de objetos sobre el sistema de almacenamiento que se haya colocado a disposición para tal fin. En el mismo se van a administrar los objetos persistentes (Recursos del Grid) de acuerdo al Esquema Lógico de la Base de Objetos del Grid o los metadatos, especificados en un archivo (XML) descriptor de las relaciones entre estos recursos, al cual se hace referencia en este trabajo como *metadatos XML o descriptor XML* (Ver Glosario).
- **Registrar Base de Objetos en el Grid:** Este caso de uso lo inicia el Administrador local y es el que permite al Planificador del Grid conocer el conjunto de nuevos recursos que se incorporan al Grid y que pueden ser considerados por él para asignarlos a la ejecución de procesos que requieran de su utilización.

- **Conectar a la Base de Objetos:** Este caso de uso se incluye <<include>> dentro de las actividades que son desarrolladas en los casos de uso Actualizar Base de Objetos en forma tanto local como remota. Es muy importante la realización de este caso de uso, ya que el mismo abre una conexión con la Base de Objetos, de acuerdo a los estándares establecidos, y a partir de esa conexión es que pueden ser llevadas a cabo el resto de las operaciones de actualización.

- **Actualizar Base de Objetos Localmente:** El caso de uso Actualizar Base de Objetos Localmente es un caso de uso que, como su nombre lo indica constituye, las actividades u operaciones necesarias para mantener actualizada la Base de Objetos, que en este trabajo también puede ser referenciada con el nombre de Repositorio de Objetos. Este caso de uso es iniciado por el Administrador del Servicio, en forma local. Está definido, a su vez, por un conjunto de casos de usos con los cuales interactúan específicamente los actores del sistema para realizar la actualización de la Base de Objetos. El conjunto de casos de usos se describen a continuación y se identifican con los siguientes nombres: Insertar Objeto Localmente, Consultar Objeto Localmente, Eliminar Objeto Localmente y Modificar Objeto Localmente.

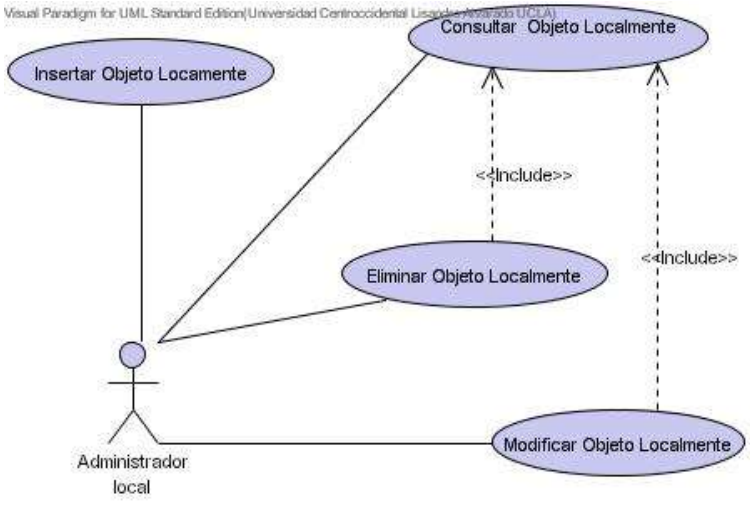


Figura 18. Detalles del Caso de Uso (SubCaso de Uso) de *Actualización de Base de Objetos Localmente.*

- **Insertar Objeto Localmente:** Este caso de uso puede ser iniciado localmente por el Administrador del Servicio. Consiste en el almacenamiento de los Objetos Persistentes en un Servidor SMOP en particular. Para ello es necesario: (a) Escoger el tipo de recurso que representa ese Objeto Persistente (OP): *Institución, Dominio, Almacén, Computo, Servicio, Usuario o Rol*. (b) Escoger la posición o lugar de almacenamiento dentro del repositorio de objetos, de acuerdo a las relaciones de *asociación, dependencia, generalización o agregación* presentes entre los objetos involucrados en la transacción de inserción y los ya contenidos dentro del repositorio, respetando el orden según la jerarquía y restricciones que representa el recurso en el Dominio Administrativo de la Institución. (ver figura 18) (c) Escoger el Servidor SMOP en el que se va a realizar la operación.
- **Consultar Objeto Localmente:** Este caso de uso tiene como finalidad la recuperación de los objetos persistentes que han sido almacenados previamente (ver caso de uso *Insertar Objeto*), en un Servidor SMOP determinado. Al igual que el caso de uso anterior, éste es utilizado en forma local por el Administrador del Servicio. Para ello es necesario: (a) Seleccionar el tipo de recurso que se desee recuperar, (b) Indicar el criterio de selección o búsqueda, por medio del cual serán evaluados los objetos persistentes ubicados en el repositorio, para determinar si son seleccionados para la recuperación, basado en el cumplimiento o no de las condiciones expuestas en dicho criterio de selección. Cabe destacar que en este caso de uso se obtendrá como resultado una Colección de los objetos persistentes que cumplen con el criterio de selección utilizado. Para especificar el criterio de selección es necesario conocer los atributos del recurso que se desea recuperar.
- **Eliminar Objeto Localmente:** En ocasiones será necesario Eliminar alguno de los Objetos persistentes del repositorio de objetos. Para ello es utilizado este caso de uso por parte del Administrador del Servicio en forma local, tomando en consideración los siguientes aspectos: (a) Determinar el tipo de recurso que va a ser eliminado; (b) Determinar el objeto o conjunto de objetos persistentes que serán seleccionados para ser eliminados, apoyándose en el caso de uso anterior

<<include>> (ver *Consultar Objeto*). (c) Indicar el Servidor SMOP en donde se va a realizar esta operación de eliminación de objetos.

- **Modificar Objeto Localmente:** Al igual que todos los casos de uso locales éste también es iniciado por el Administrador Local, el cual debe conocer los atributos de los Recursos que vaya a modificar para poder indicar al sistema el nuevo valor que se le vaya a colocar al mismo. Este caso de uso se apoya en la Consulta de Objetos <<include>> para realizar modificaciones a conjuntos de objetos persistentes que cumplan con el criterio de selección de la consulta.

- **Actualizar Base de Objetos Remotamente:** El caso de uso Actualizar Base de Objetos Remotamente es un caso de uso que presenta la misma funcionalidad que la de Actualizar Base de Objetos Localmente. La principal diferencia entre ellos es el tipo de acceso a la Base de Objetos y el actor principal que inicia el caso de uso. Este caso de uso también está definido por un conjunto de subcasos de usos con los cuales interactúan específicamente los actores del sistema para realizar la actualización de la Base de Objetos en forma remota. Estos casos de usos se describen a continuación y se identifican como: Insertar Objeto Remotamente, Consultar Objeto Remotamente, Eliminar Objeto Remotamente y Modificar Objeto Remotamente.

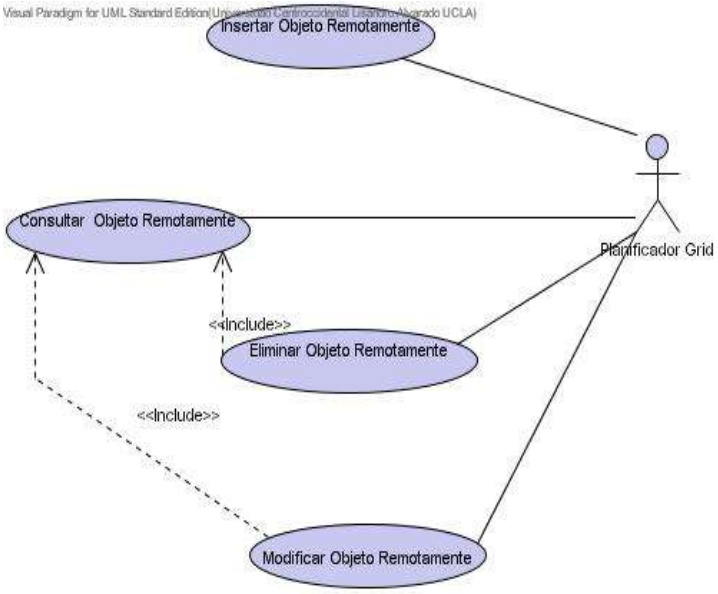


Figura 19. Detalles del Caso de Uso (SubCaso de Uso) de *Actualización de Base de Objetos Remotamente*

- **Insertar Objeto Remotamente:** Este caso de uso lo inicia el Planificador del Grid, por medio de este caso de uso se puede realizar el almacenamiento de los Objetos Persistentes en el Servidor SMOP que haya sido seleccionado por el Planificador. Para que un Servidor SMOP pueda ser seleccionado para realizar una Inserción de objetos, es necesario que el mismo haya sido registrado previamente (ver caso de uso registrar Base de Objetos en el Grid). Para poder realizar la Inserción debe abrirse una conexión con la Base de Objetos (ver caso de uso, Conectar a la Base de Objetos).
- **Consultar Objeto Remotamente:** Este caso de uso tiene como finalidad la recuperación de los objetos persistentes que han sido almacenados previamente (ver caso de uso *Insertar Objeto*), en un Servidor SMOP determinado. Inicialmente debe abrirse una conexión a la Base de Objetos seleccionada (ver caso de uso, Conectar a la Base de Objetos). Para seleccionar un Servidor SMOP, el mismo ha debido ser registrado con anterioridad en el Grid. Es necesario conocer los atributos de los recursos que se necesitan recuperar, para componer el criterio de la consulta.
- **Eliminar Objeto Remotamente:** A la hora de eliminar en forma remota algún Recurso persistente de la Base de Objetos, el Planificador inicia este caso de uso. Se cumplen los mismos pasos necesarios para poder realizar una operación remota: se selecciona el tipo de recurso a eliminar; Se elige el Servidor SMOP donde va a ocurrir la eliminación de Recursos; se abre una conexión a la Base de Objetos; se compone un criterio de consulta para seleccionar una Colección de objetos del tipo indicado, para su eliminación de la Base de Objetos; se ejecuta la operación de eliminación remota.
- **Modificar Objeto Remotamente:** Este caso de uso también es iniciado por el Planificador del Grid, el cual utiliza un mecanismo que le permite seleccionar de entre una lista de Servidores registrados, el Servidor SMOP al cual se le va a hacer una modificación. También en este caso de uso se deben cumplir una serie de pasos previos a la ejecución de la operación: seleccionar el tipo de recursos que se va a modificar, seleccionar el Servidor que se va a utilizar, abrir la

conexión con la Base de Objetos, crear el criterio de consulta, basado en los atributos del Recurso para seleccionar los objetos que van a ser modificados de acuerdo a ese criterio y finalmente ejecutar esa operación.

Diagramas de Interacción

Para modelar los aspectos dinámicos de los casos de uso del Sistema se han utilizado dos tipos de diagrama: Los diagramas de Colaboración y de Secuencia.

Diagramas de Colaboración

En este diagrama de interacción se destaca la organización estructural de los objetos que envían y reciben mensajes, identificados en los casos de uso del sistema. La figura 20 muestra el diagrama de colaboración del Caso de Uso: Actualizar Base de Objetos Remotamente. (UC-SMOP_05). Los diagramas de colaboración del resto de los Casos de Uso, se pueden obtener en el ANEXO C.

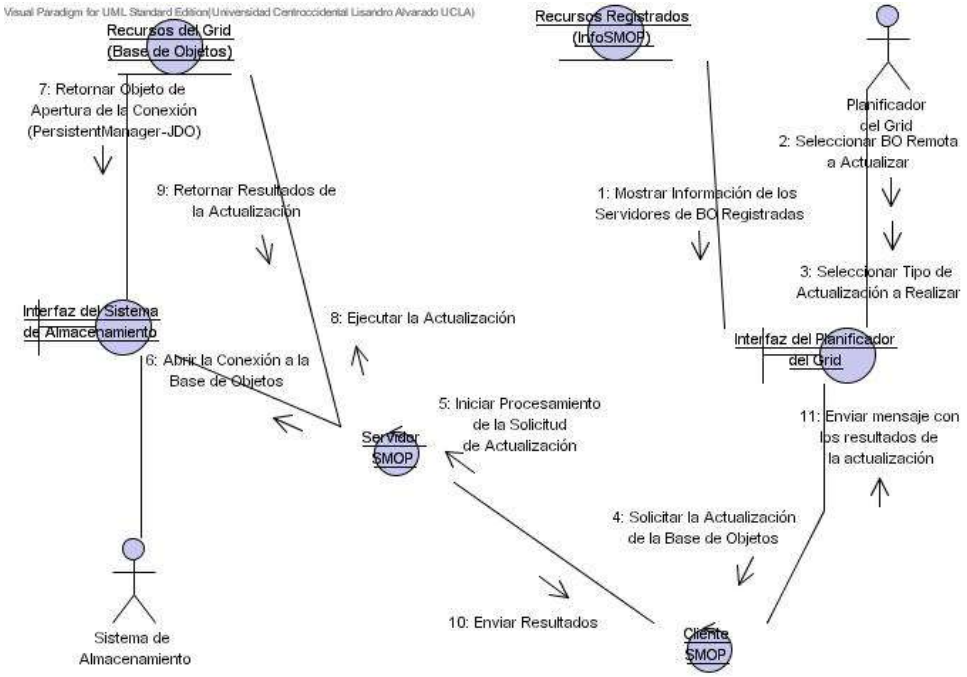


Figura 20. Diagrama de Colaboración del Caso de Uso: Actualizar Base de Objetos Remotamente. (UC-SMOP_05)

Diagramas de Secuencia

Este diagrama de interacción destaca la ordenación temporal de los mensajes entre los objetos presentes en los casos de uso. En la figura 21 este diagrama se ha utilizado para modelar un flujo de control particular del Caso de Uso: Actualizar Base de Objetos Remotamente. (UC-SMOP_05). El resto de los diagramas se pueden visualizar en el ANEXO D.

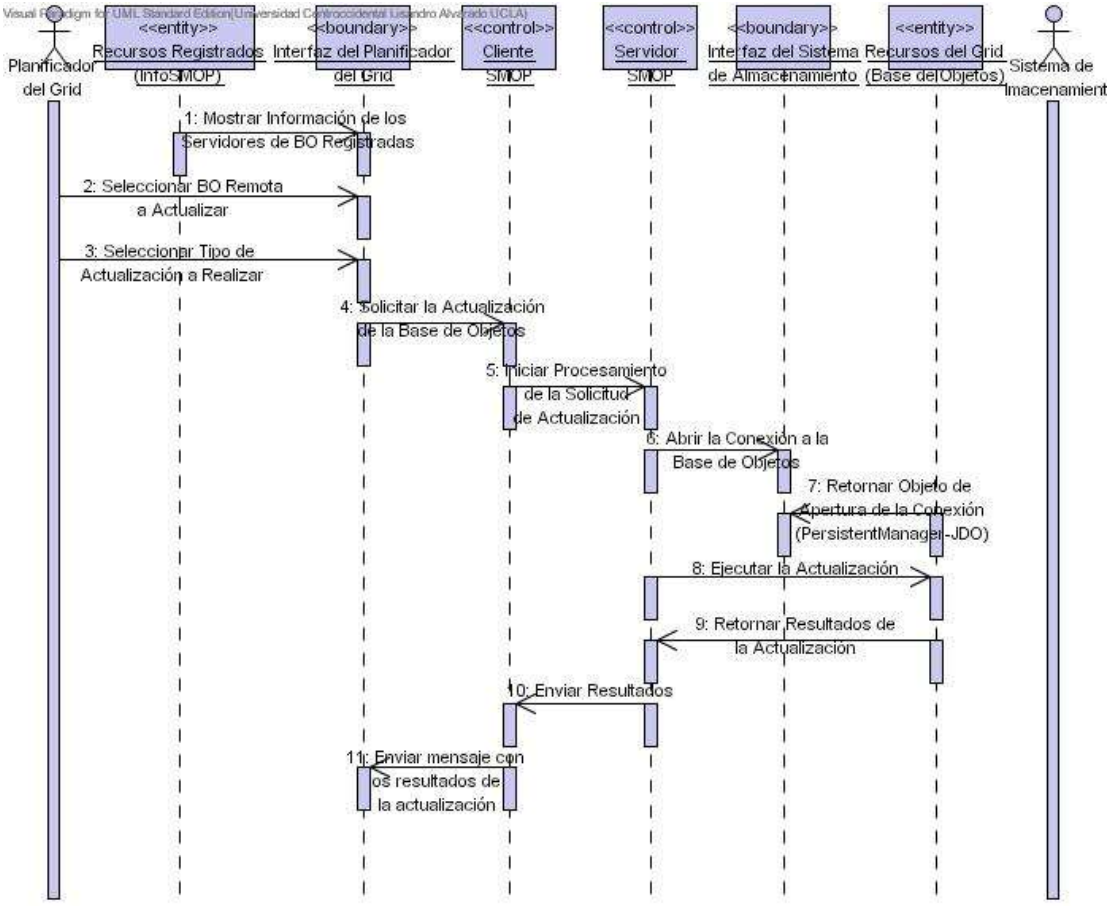


Figura 21 Diagrama de Secuencia del Caso de Uso: Actualizar Base de Objetos Remotamente (UC-SMOP_05)

Fase de Elaboración

4. Análisis del Sistema

En esta fase de desarrollo se identifican las clases del sistema de tipo Entidad, Control e Interfaz, en un elevado nivel de abstracción, la agrupación de las mismas en paquetes y la relación existente entre ellas presentada en un diagrama de clases.

Clases Entidad

Se comienza por definir los tipos de Objetos Persistentes que van a ser tratados por el SMOP; es decir, las Clases tipo Entidad más importantes, con las que funciona el Servicio y que están presentes en la gran mayoría de los casos de uso del sistema. Durante la fase de diagnóstico del sistema (Levantamiento de Información) se han encontrado ochos tipos de Objetos Persistentes que representan los recursos disponibles que pueden ser manejados por cualquier Grid Computacional, son llamados en este proyecto: Institución, Dominio, Recurso, Rol, Usuario, Almacén, Servicio y Cómputo. Estos Objetos Persistentes son instancias de las Clases Persistentes que los describen mediante sus atributos, métodos y relaciones. Seguidamente se describen, en forma breve, cada una de ellas:

Institución:

Icono

Es una clase que como su nombre los indica, representa o identifica a las instituciones u organizaciones que participan en el Grid. Es un gran contenedor de los demás recursos del mismo, los cuales son agrupados en *Dominios Administrativos*, de allí que se observe la relación de agregación entre la clase *Institución* y la Clase *Dominio*. Una Institución podrá tener entre 1 a n *Dominios Administrativos*, según su rango de direcciones IP reales, certificadas y disponibles.

Esta clase representa los Dominios Administrativos de las Instituciones u Organizaciones, en donde son agrupados los recursos de las mismas, para una administración con políticas comunes. Un Dominio podrá administrar entre 1 y n Recursos. Por esta razón existe una relación de agregación entre la clase Dominio y la clase Recurso.

Es una de las clases más importantes para este proyecto, ya que es la que representa a los recursos que están siendo otorgados por las Instituciones participantes del Grid, para ser empleados según la disponibilidad de los mismos. En este proyecto se manejan tres tipos de recursos básicos que son: El Almacén, El Cómputo y el Servicio. Por esta razón existe una relación de generalización entre esta clase y las clases Almacén, Cómputo y Servicio, para especificar la herencia entre ellas y diferenciar cada tipo de recurso de su clase base.

Es una clase muy ligada a la clase recurso, la cual es utilizada para crear grupos de usuarios, con perfiles o características comunes, mediante las cuales se permite el acceso a los recursos definidos en el Grid. La relación entre la clase Recurso y la clase Rol es de asociación. Un Recurso puede estar asociado con 1..n Roles.

Esta clase, como su nombre lo indica, representa a los usuarios que hayan sido definidos para el uso de los recursos del Grid. Esta clase guarda una relación muy estrecha entre la clase Recurso y la Clase Rol, ya que, un Usuario tiene su razón de ser por la existencia de algún Recurso que necesite utilizar. No obstante, en un Recurso en particular cada Usuario desempeña uno o más Roles específicos; es decir, presenta ciertos privilegios y restricciones que son determinados por el Grupo o Rol al cual

pertenece, para ese Recurso en particular. Un Usuario puede tener uno o más Roles en un Recurso Determinado. Por lo tanto, quien va a determinar qué Rol desempeña un Usuario para un Recurso específico, será precisamente la Clase Rol, la cual se utiliza como una especie de clase "asociativa" entre ellas.

Almacén:

Es la clase que representa a los Sistemas de Almacenamiento que como Recursos, están siendo incorporados al Grid por parte de las Instituciones participantes. Esta clase se relaciona por medio de la generalización con la clase Recurso, de la cual hereda sus atributos y comportamiento. Los atributos y operaciones propias de cada Almacén serán agregados dependiendo del interés particular de cada Grid que se quiera implementar, por ahora se incorporan entre otros, atributos como la Capacidad Máxima de Almacenamiento y el Número de Discos presentes en el Almacén.

Cómputo:

Es la clase que representa el cómputo que es compartido por el Grid. Al igual que se indicó para el Almacén, los atributos y operaciones que se tomarán en cuenta dependerán de los intereses o necesidades de la implementación del Grid con que se esté trabajando, para este caso se utilizaron atributos tales como: Número de Procesadores y Velocidad Máxima de los Procesadores, entre otros.

Servicio:

Esta clase representa otro tipo de recurso que es considerado en este proyecto para ser compartido, como servicio, en un Grid Computacional. Por lo tanto, debe ser reconocido por el Planificador para poder ubicarlo y utilizarlo apropiadamente. Por ejemplo, Servidores DNS, Servidores de Correo, Servidores Web, Servidores de Impresión, entre otros.

La figura 22, representa un Diagrama de Clases de las Entidades que se utiliza para modelar un esquema lógico de la Base de Objetos que maneja el Grid. El detalle de los elementos que componen estas clases (Atributos y Métodos), pueden ser revisados en el ANEXO A.

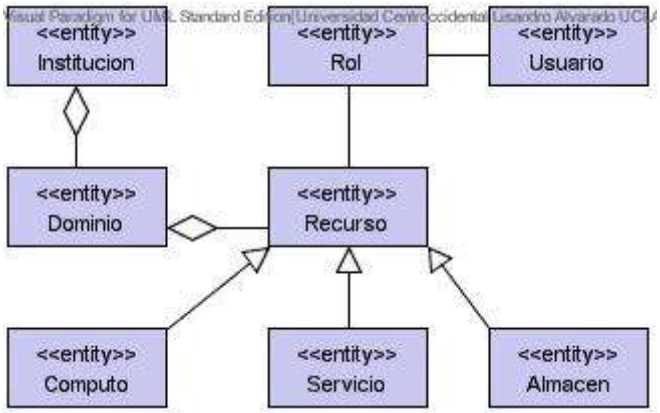


Figura 22. Esquema Lógico de la Base de Objetos del Grid

Seguidamente, una vez definido el esquema lógico de la Base de Objetos del Grid, fundamentada en las Clases y relaciones presentadas previamente en el Diagrama de Clases, se requiere establecer el mecanismo que va a proporcionar el manejo de la persistencia, primeramente de estas Clases y en consecuencia de los Objetos instanciados por ellas. De esa manera, en este proyecto se podrá hablar con mayor propiedad de Clases y Objetos Persistentes, para referirse a todas aquellas clases que hayan pasado por este mecanismo; es decir, aquellas que hayan adquirido la capacidad de ser Persistentes. Por tratarse, en este proyecto con Objetos Persistentes manejados en Grid Computacionales basados en Java, la decisión de seleccionar el mecanismo que proporcione a las clases la capacidad de ser persistentes, se basa en los mecanismos estándares que han sido diseñados para cumplir con este objetivo en Java como Lenguaje de Programación.

Para este proyecto se seleccionó el mecanismo estándar denominado “Java Data Object” (JDO). El mismo “heredó” el trabajo del Grupo Manejador de Objetos de Datos (“ODMG”) que consistió en un conjunto de especificaciones estándar identificadas como ODMG 3.0. para permitir la portabilidad entre productos de Bases de Datos Orientadas a Objetos y Bases de Datos Objeto/Relacional. “JDO” parte de una iniciativa de Sun Microsystem para modificar y adaptar el estándar ODMG 3.0 a las características peculiares de Java. Este trabajo se ha venido realizando a través de una organización abierta denominada Proceso de Comunidad Java (“JCP”), la cual trabaja con proyectos de especificaciones identificados con el nombre de Solicitudes de Especificaciones Java (“JSR”) siguiendo una codificación numérica. JDO se identifica con el “JSR” número 12 (JSR-12).

La figura 23 muestra los pasos necesarios para aplicar el mecanismo estándar JDO al Esquema Lógico de la Base de Objetos del Grid (ELBOG) y dejarla preparada para ser empleada sobre algún Sistema de Almacenamiento por la Máquina Virtual Java (“JVM”).

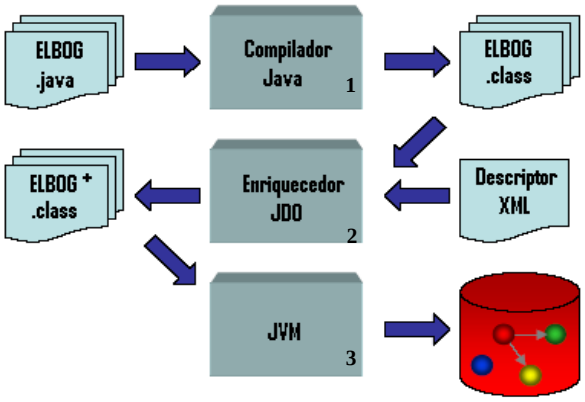


Figura 23. Proceso de Enriquecimiento del ELBOG.

Paso 1: En este primer paso se realiza la compilación de los archivos (*.java) ubicados en el paquete: SMOP.Servicio.BDOP. Estos archivos representan la implementación de las clases de identidad descritas anteriormente en lenguaje Java y

conforman el ELBOG. Esta compilación se realiza utilizando la siguiente instrucción : *javac SMOP/Servicio/BDOP/*.java.*

Paso 2: Una vez obtenido los archivos bytecode de java(*.class) como resultado de la compilación realizada en el paso 1, es necesario pasarlos por un mecanismo de "enriquecimiento" con la finalidad de incorporar los atributos y métodos de las interfaces y clases que han sido diseñadas para capacitar al ELBOG con la habilidad de implementar las especificaciones "JDO" para el manejo de la persistencia de sus objetos. Este mecanismo se puede apreciar en la figura 25. Para alcanzar este paso generalmente se realiza una post-compilación de los bytecode de Java, previamente compilados. Para ello se utiliza un "enriquecedor" o "enhancer", el cual se ha implementado siguiendo en forma rígida las especificaciones estándar "JDO". El enriquecedor utiliza un archivo XML (extensión .jdo) por medio del cual recibe la información que le indica las relaciones existentes entre las clases involucradas en el enriquecimiento. Por ejemplo, las relaciones de herencia o generalización, las relaciones de agregación o simplemente asociación entre ellas. En el caso de este proyecto se utiliza el siguiente archivo descriptor XML, identificado con el nombre de *package.jdo*, ubicado también en el paquete SMOP.Servicio.BDOP:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE jdo PUBLIC
  "-//Sun Microsystems, Inc.//DTD Java Data Objects Metadata 1.0//EN"
  "http://java.sun.com/dtd/jdo_1_0.dtd" >
<jdo>
   <package name="SMOP.Servicio.BDOP" >
      <class name="Element" />
      <class name="Category" persistence-capable-superclass="Element" />
      <class name="Institucion" persistence-capable-
superclass="Category"/>
      <class name="Dominio" persistence-capable-superclass="Category"/>
      <class name="Recurso" persistence-capable-superclass="Category">
           <field name="cast" >
               <collection element-type="Rol"/>
           </field>
      </class>
      <class name="Almacen" persistence-capable-superclass="Recurso" />
      <class name="Computo" persistence-capable-superclass="Recurso" />
      <class name="Servicio" persistence-capable-superclass="Recurso"/>
      <class name="Usuario" persistence-capable-superclass="Category">
           <field name="roles" >
              <collection element-type="Rol"/>
           </field>
      </class>
      <class name="Rol" persistence-capable-superclass="Category"/>
      <class name="Item" persistence-capable-superclass="Element"/>

    </package>
</jdo>
```

Cuadro 4. package.jdo Descriptor XML de las clases persistentes del SMOP

Este archivo fue editado manualmente, sin embargo no se descarta la existencia de generadores automáticos del mismo. En el mismo es donde se describen las relaciones de herencia, asociación y agregación entre otras, existentes entre las clases de tipo identidad utilizadas en el proyecto. Las instrucciones contenidas en este archivo pudieran variar dependiendo de la implementación "JDO" utilizada en cada proyecto. Entre las que se destacan las siguientes: (a) < package name="SMOP.Servicio.BDOP">, se utiliza para indicar la ubicación del paquete con las clases a procesar; (b) <class name="Element">, se utiliza para indicar el nombre de la clase, en este caso la clase es "Element". (c) <class name="Almacen" persistence-capable-superclass="Recurso"/>, se utiliza para establecer una relación de herencia, indicando en este caso que la clase "Almacen" hereda de la clase "Recurso" ; (d) <field name="roles">, se utiliza para describir los atributos de las clases y (e) <collection element-type="Rol">, se utiliza para establecer una relación de agregación, en este caso el atributo "roles" se define como una colección de instancias de la clase "Rol".

En este proyecto se usan implementaciones "JDO" no comerciales, como por ejemplo: (a) La implementación referencial "JDO" 1.0.1 la cual, se basa en un sistema de archivos denominado File Object Store. La misma ha sido desarrollada directamente por la JCP (http://jcp.org/aboutJava/communityprocess/final/jsr012/index2.html), dirigido por Craig Russell (líder de Sun Microsystems, Inc.); (b) La implementación TriActive JDO o TJDO (http://tjdo.sourceforge.net), emplea un Sistema de Almacenamiento Relacional para el manejo de los Objetos Persistentes, en este caso MySQL y (c) Para emplear un Sistema de Almacenamiento basado en Objetos se ha utilizado la versión libre de un producto de Software denominado ObjectDB (http://www.objectdb.com/database/jdo/free-download.) Una instrucción típica de enriquecimiento de una clase es la siguiente:

java com.sun.jdori.enhancer.Main SMOP/Servidor/BDOP/package.jdo SMOP/Servidor/BDOP/ Institucion.class

Paso 3: A esta altura, simplemente se tiene un conjunto de clases persistentes cuyas instancias en una JVM, pueden ser manejadas de acuerdo a los estándares establecidos y según sea el tipo de Sistema de Almacenamiento que utilice la implementación de las especificaciones "JDO".

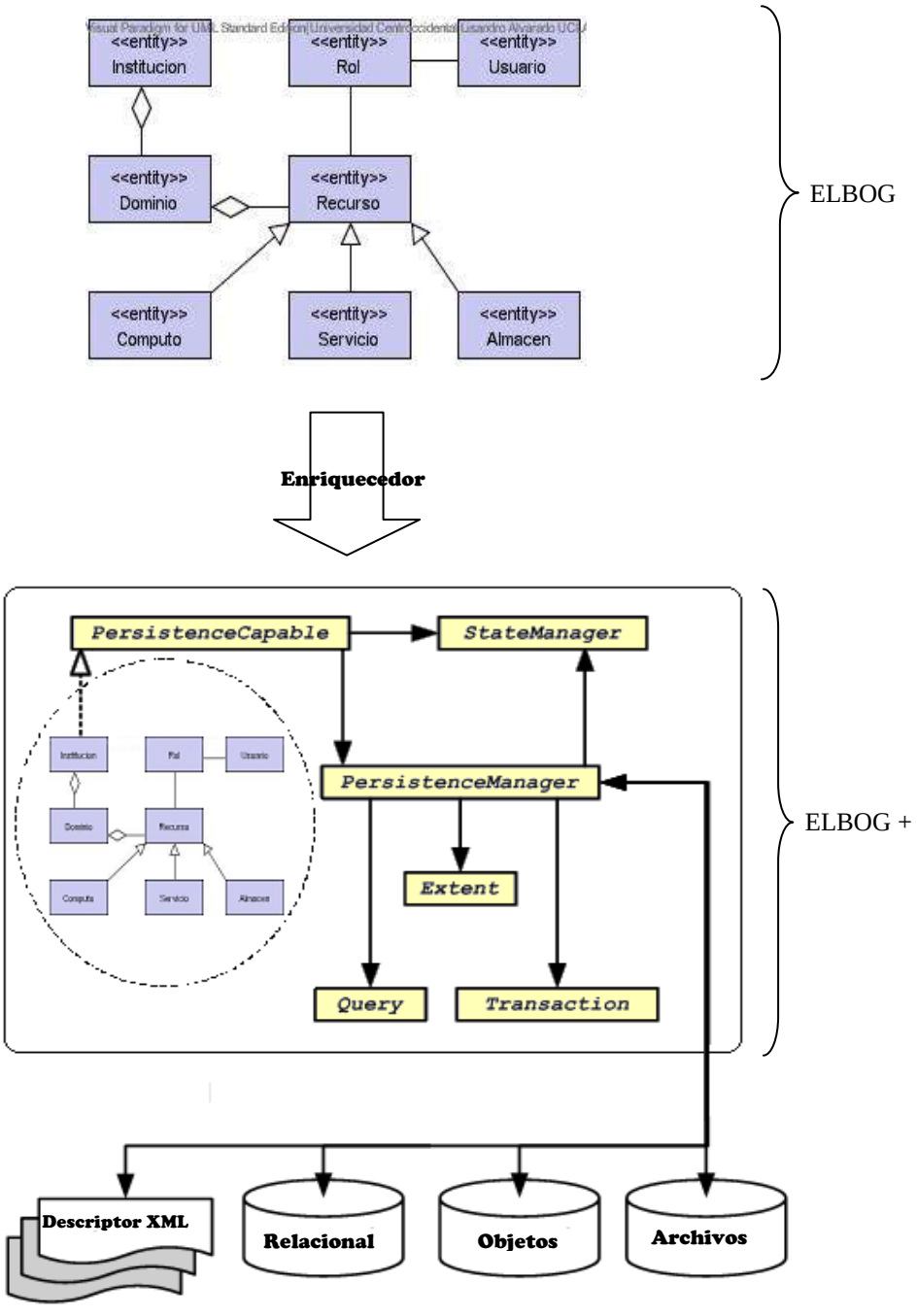


Figura 24. ELBOG Enriquecido con la capacidad de Persistencia de las Clases JDO.

En la figura 24 se muestra en forma más detallada el conjunto de clases enriquecidas como resultado del proceso descrito en la figura 23. Estas clases de tipo entidad que representan el Esquema Lógico de la Base de Objetos del Grid o ELBOG, al pasar por el proceso de enriquecimiento son integradas a un nuevo conjunto de clases que implementan las especificaciones JDO, identificadas como: *PersistenceCapable, StateManager, PersistenceManager, Query, Extent y Transaction.* Entre ellas se destaca el *PersistenceManager* como elemento central para permitir el acceso a las fuentes de almacenamiento, y poder controlar junto con el resto de las clases la persistencia de los objetos java. El ELBOG enriquecido se representa de la siguiente manera ELBOG +.

Clases Control

Estas clases han sido definidas para manejar el flujo de eventos del sistema. Se identifican con el estereotipo <<control>> y representan la lógica de secuenciación de los casos de uso creados en el análisis de los requerimientos. En esta fase, cada clase es descrita de manera muy general, sin embargo se puede obtener una descripción más detallada en la fase de diseño.

- **Planificador:** Es una de la clases más importantes del proyecto, ya que proporciona la lógica de secuenciación para la realización del caso de uso *Actualizar Base de Objetos Remotamente* y en colaboración con otras clases, interfaces y elementos permite modelar el mecanismo diseñado para la asignación de tareas a los Servidores SMOP. Se identifica con el nombre de ***Planificador.***
- **Sincronizador:** Esta es la clase por medio de la cual el Planificador del Grid accede a los servicios remotos proporcionados por el Servidor SMOP que el planificador haya seleccionado. Recibe su nombre debido a que esta clase se encarga de sincronizar los mensajes enviados entre el Servidor SMOP y el Planificador en relación a las operaciones sobre la base de objetos y a su vez,

es la que indica al Planificador qué señal enviada por los Servidores SMOP fue recibida en primer lugar. Se identifica con el mismo nombre: ***Sincronizador.***

- **Puerta de Conexión (Grupo de Sockets):** Esta clase controla un grupo de sockets que son utilizados para comunicar alguna referente a los estados tanto del Servidor SMOP como del Planificador y en general, el estado de la simulación. Por ejemplo, por medio de estos sockets se comunica información como la siguiente: "El Planificador está o no preparado", "La Simulación comienza en x segundos", "El evento número z procesará un objeto tipo y", entre otras. Es identificada como: ***PuertaConexion.***
- **Servidor SMOP:** Esta clase proporciona la lógica de secuenciación para el desarrollo de los casos de uso relacionados con la actualización de la Base de Objetos tales como: Crear Base de Objetos, Actualizar Base de Objetos entre otras. Adicionalmente, contiene la interfaz gráfica del administrador la cual permite registrar dicha base de objetos a un Planificador del Grid determinado. Esta clase es conocida como: ***ServidorSMOP***

Clases Interfaz (GUI)

Estas clases proporcionan cuadros de diálogos con los cuales los actores del sistema pueden interactuar con el mismo, para solicitar la ejecución de las operaciones que se indiquen, de acuerdo al orden expuesto en los casos de usos especificados anteriormente. Estas clases son identificadas con el prototipo <<boundary>>. Se pueden distinguir cuatro clases principales de tipo interfaz (GUI) en el SMOP, las cuales se describen brevemente a continuación:

- **Panel de Control del Planificador:** Esta clase permite iniciar la demostración de la funcionalidad del sistema, mediante una estrategia de simulación, la cual se explica con más detalles en el siguiente capítulo. En esta clase se pueden configurar los siguientes valores para controlar la

simulación: la pausa entre eventos (en segundos), el tiempo de espera antes de iniciar la demostración (en segundos), el número máximo de Instituciones que se pueden registrar y el tipo de Operación (Insertar, Consultar, Eliminar o Modificar) sobre la base de objetos, con la que se va a realizar la simulación. Se identifica con el siguiente nombre: ***GUIPanelCtrlPlanificador.***

- **Panel de Control del Servidor SMOP:** Esta clase permite al Administrador local realizar las operaciones necesarias para unir el Servidor SMOP que administra, a la demostración de la funcionalidad del Servicio. Esta clase proporciona la facilidad de configuración de los siguientes valores: Nombre de la Institución que representa la interfaz, Dirección IP o Nombre (DNS) del servidor en donde se ejecuta el Planificador del Grid para la simulación, muestra la dirección IP del Servidor SMOP y muestra el tipo de sistema de almacenamiento sobre el cual está soportada la Base de Objetos que administra el Servidor. Adicionalmente, proporciona un botón con el cual se ejecuta la operación de registro del Servidor en el Planificador y posee además, una opción para recibir los eventos del Planificador con un sonido particular. Esta clase se identifica con el siguiente nombre: ***GUIPanelCtrlServidorSMOP.***
- **Interfaz del Administrador:** Esta clase le proporciona las facilidades gráficas al Administrador Local del servidor, para mantener actualizada la base de objetos que administra. Para ello utiliza los siguientes elementos: un árbol jerárquico con el cual se puede visualizar el contenido de los objetos persistentes que se encuentran almacenados en la base de objetos. En dicho árbol se muestran los objetos persistentes, en un orden específico, los cuales representan los recursos que administra ese Servidor SMOP en particular y que a su vez debería coincidir con la realidad de cada Institución que coloca sus recursos a disposición del Grid. El árbol puede ser recorrido libremente ubicándose directamente sobre cada nodo para expandir o contraer su contenido. Cada vez que se ubica en un nodo en particular los atributos y valores de ese nodo son visualizados en un cuadro de dos columnas destinado

para tal fin. Adicionalmente, se presenta un pequeño cuadro para recoger el criterio de consulta de algún recurso determinado y finalmente, se destina un espacio para mostrar los resultados de dicha consulta. Esta clase es identificada con el nombre de ***GUIAdministrador.***

- **Eventos esperados por el Servidor:** Es una clase destinada a mostrar el conjunto de códigos de los eventos que son esperados por el Servidor SMOP para reaccionar y competir con el resto de Servidores, enviando una señal como respuesta, esperando que esa sea la primera señal en recibir el planificador, y de esa manera ganar la opción de ser el Servidor Seleccionado para actualizar su Base de Objetos. Esta clase se identifica con el nombre de ***GUIEventosSMOP.***

Paquetes

Luego de definir las principales clases que conforman el SMOP, éstas se han agrupado en tres diferentes paquetes, los cuales permiten realizar una mejor organización de los diversos elementos que componen el Sistema. La figura 25 muestra un paquete principal denominado SMOP, dentro del cual se encuentran otros tres paquetes que en concreto contienen los componentes del sistema; estos paquetes son: (a) El paquete Cliente; (b) El paquete Servidor y (c) El paquete Compartido.

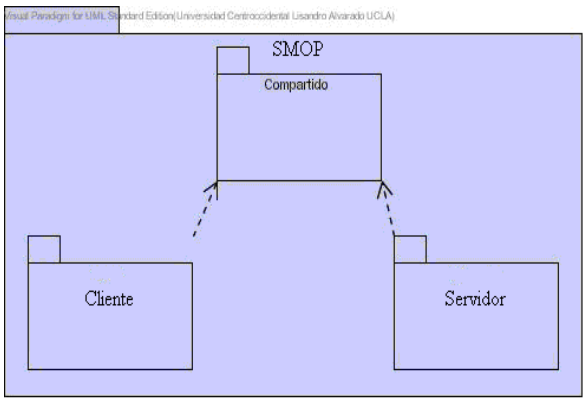


Figura 25. Paquetes UML del SMOP en la fase de Diseño.

La organización de las clases en los diferentes paquetes y sus detalles internos como atributos y métodos se indican en la fase de diseño.

Diagrama de Clases

El diagrama de clases es utilizado en esta fase del proyecto para representar las relaciones entre las clases especificadas anteriormente. En la figura 26 se pueden observar, además de las principales clases del SMOP, otro tipo de elementos tales como las Interfaces y las Relaciones de dependencia, generalización y asociación existente entre ellos. En este diagrama de clases ilustrado en la figura 26 se presenta una vista de diseño estática del sistema.

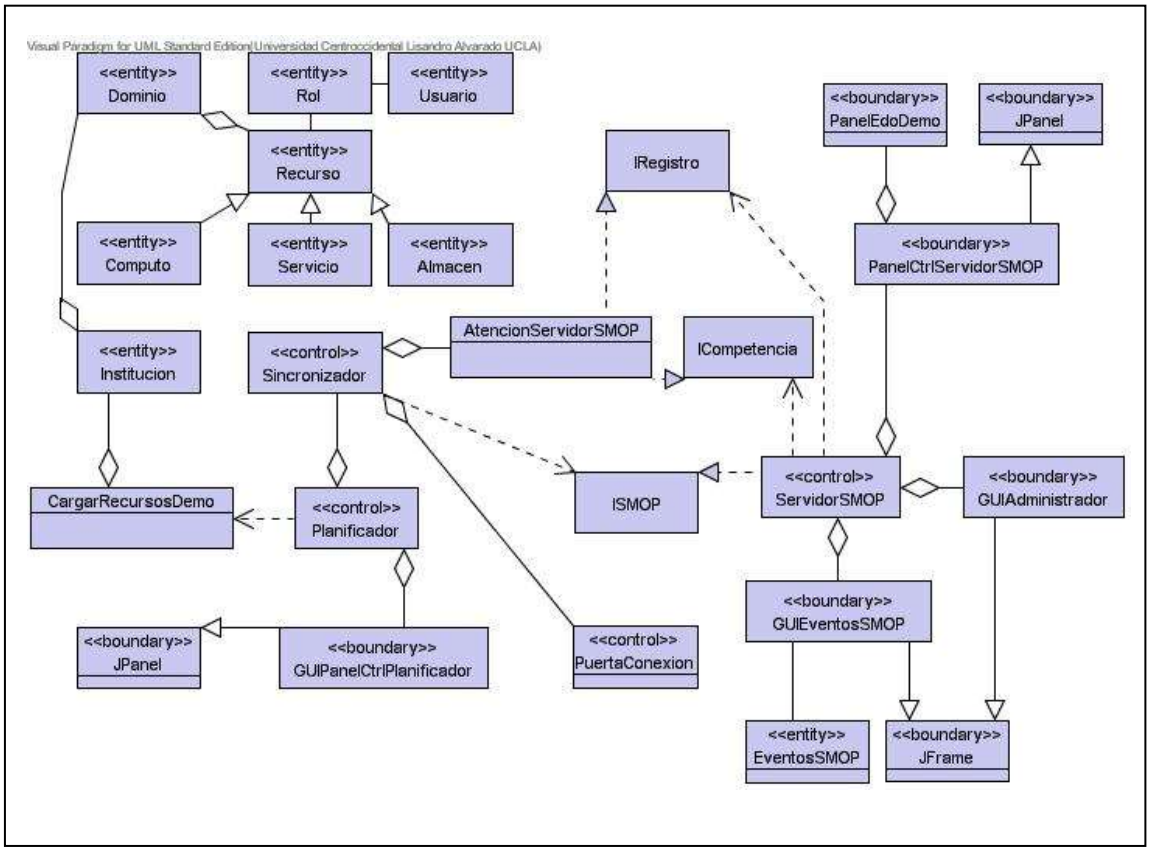


Figura 26. Diagrama de Clases del SMOP.

5. Diseño del Sistema

En esta fase de diseño se continuó con el refinamiento de los "artefactos" UML obtenidos en la fase de análisis, se expandieron los paquetes ya establecidos para organizar los elementos del software del sistema, considerando las relaciones entre estos paquetes y destacando las clases, interfaces y relaciones más importantes de cada uno; es decir, aquellas que son indispensables para la ejecución efectiva del Servicio. La relación existente entre los paquetes del sistema ha sido presentada previamente en la figura 26.

Paquete Compartido.

Este paquete contiene tres de las más importantes interfaces RMI del servicio: ISMOP, IRegistro, ICompetencia cuyas operaciones se ilustran a continuación.

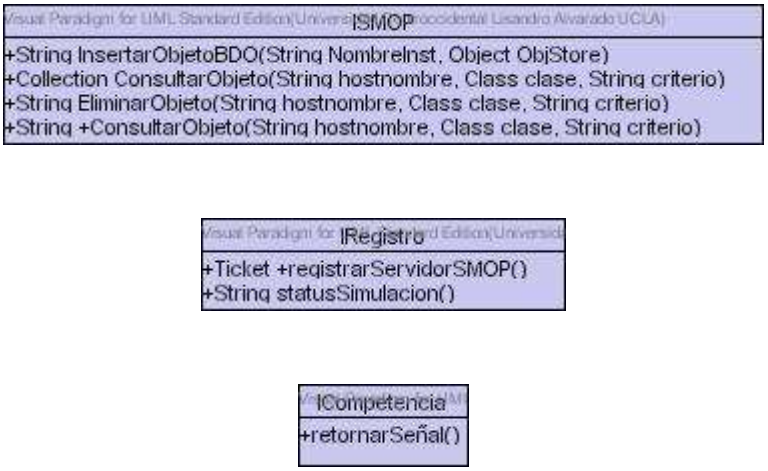


Figura 27. Interfaces Remotas RMI del SMOP.

La figura 27 ilustra las operaciones de las interfaces remotas RMI que se utilizan en este proyecto para que por un lado, el Planificador pueda acceder a los métodos remotos que han sido publicados por el Servidor SMOP, y por el otro para que el Servidor SMOP pueda utilizar el método remoto para registrarse en el Planificador y

posteriormente, en su debida oportunidad, pueda usar el método remoto para retornar una señal al Planificador y poder competir por el procesamiento de las operaciones del Planificador con el resto de los Servidores SMOP.

Paquete Servidor

El paquete Servidor contiene las clases de Entidad, Interfaz y Control necesarias para iniciar las diferentes instancias del SMOP que se necesiten. Entre las clases más sobresalientes para cumplir con este objetivo se presentan: PanelCtrlServidorSMOP, GUIEventosSMOP y GUIAdministrador.

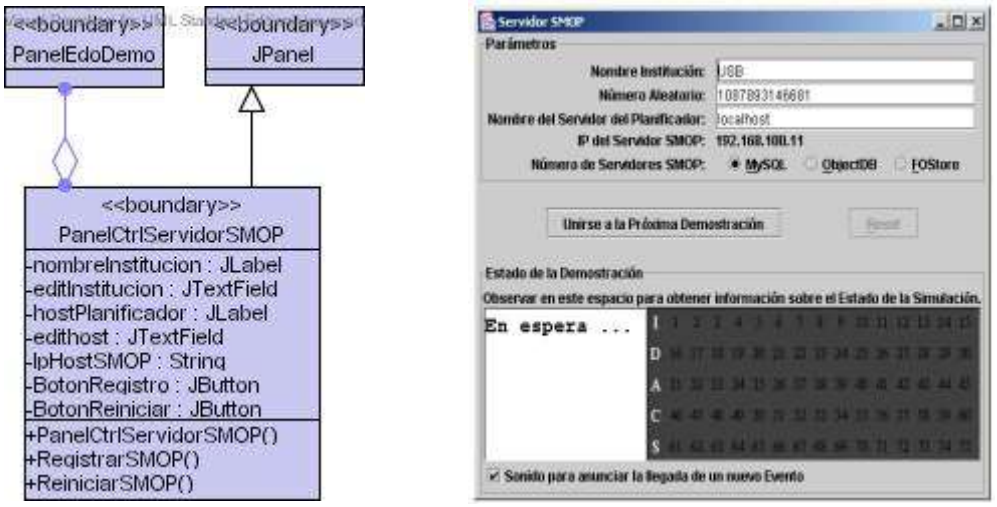


Figura 28. Atributos y Métodos de la Clase <<boundary>> PanelCtrlServidorSMOP.

La Clase tipo interfaz que se ilustra en la figura 28, constituye la interfaz Gráfica del Usuario utilizada por el Actor “Administrador Local”. En el lado izquierdo de la figura se pueden detallar los atributos y métodos más importantes de esta clase y del lado derecho se muestra un ejemplo de la ejecución de una instancia de esta clase.

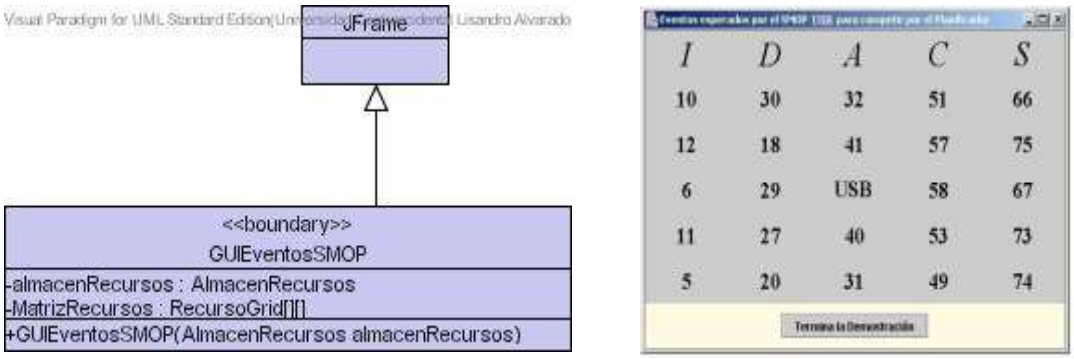


Figura 29. Atributos y Métodos de la Clase <<boundary>> GUIEventosSMOP.

Esta clase ilustrada en la figura 29, es utilizada por el SMOP para dar a conocer los códigos de los eventos que han sido generados aleatoriamente para el Servidor SMOP, los cuales representan las características que identifican el tipo de Servidor y mediante las cuales va a poder reaccionar, para competir por atender las solicitudes del Planificador. Del lado izquierdo de la ilustración se especifican los atributos y métodos más importantes de esta clase y del lado derecho, se tiene una visión gráfica de la ejecución de una instancia de esta clase.

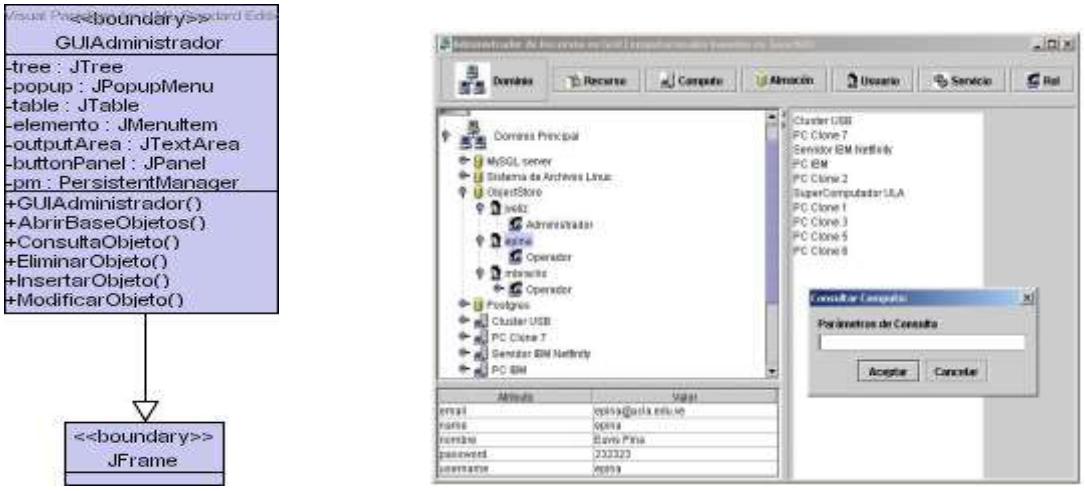


Figura 30. Atributos y Métodos de la Clase <<boundary>> GUIAdministrador.

La figura 30 muestra la interfaz gráfica del Administrador local, la cual le permite realizar todas las operaciones necesarias para mantener actualizada la base de objetos. A la izquierda de la figura se presentan sus atributos y métodos, y a la derecha se

muestra la ejecución de una instancia de esta clase, con la finalidad de apreciar su apariencia exterior.

Figura 31. Atributos y Métodos de la Clase <<Control>> ServidorSMOP .

Se culmina el recorrido por el contenido del paquete Servidor, con la descripción de la Clase ServidorSMOP, cuyos atributos y métodos se ilustran el la figura 31. Esta clase control es muy importante para proporcionar la lógica de secuenciación necesaria para completar la simulación del lado de la administración de las operaciones sobre la Base de Objetos.

Paquete Cliente.

En el paquete Cliente pueden ser halladas las Clases destinadas a proporcionar la funcionalidad del lado del Planificador del Grid.

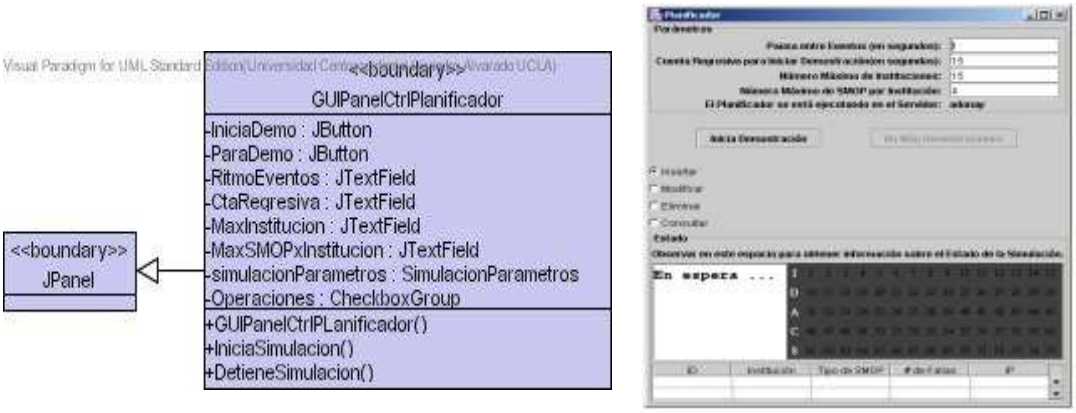


Figura 32. Atributos y Métodos de la Clase <<boundary>> GUIPanelCtrlPlanificador.

La figura 32, muestra la interfaz del Planificador del Grid, desde el punto de vista de los atributos y métodos que la componen (lado izquierdo) y a su vez, se puede apreciar la ejecución de una instancia de la misma, para apreciar con mayor claridad la apariencia final que se obtiene al combinar los diferentes elementos de interacción.

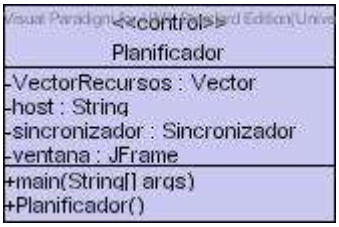


Figura 33. Atributos y Métodos de la Clase <<Control>> Planificador.

La clase de tipo control identificada con el nombre de Planificador proporciona la lógica de secuenciación para desarrollar el componente que se incorpora al servicio, con el propósito de simular su integración a un Grid computacional para obtener solicitud de operaciones sobre la Base de Objetos en forma remota. La figura 33, muestra los detalles de la clase en cuanto a los atributos y métodos se refiere.

Figura 34. Atributos y Métodos de la Clase <<Control>> Sincronizador.

Finalmente, se ilustran gráficamente los detalles internos de la Clase Sincronizador a través de la figura 34, en donde se aprecian sus atributos y métodos. Esta clase es la encargada de establecer contacto con los Servidores SMOP mediante la capa de conectividad del Servicio. Esta capa combina el uso de sockets para monitorear el estado de una simulación en curso y también utiliza la Invocación de

métodos remotos de java (RMI) para acceder a las operaciones remotas que proporciona el Servicio.

Diagrama de Componentes

Una vista muy cercana a la fase de implementación del Servicio es el Diagrama de componentes, el cual se muestra en la figura 35, destacando los componentes centrales del Servicio; es decir, los componentes que conforman el núcleo del SMOP. Estos componentes modelan los aspectos físicos en el sistema; es decir, los mismos pueden ser hallados en los nodos en donde se decida instalar el servicio, en forma de archivos de ejecutables y además, los archivos de configuración.

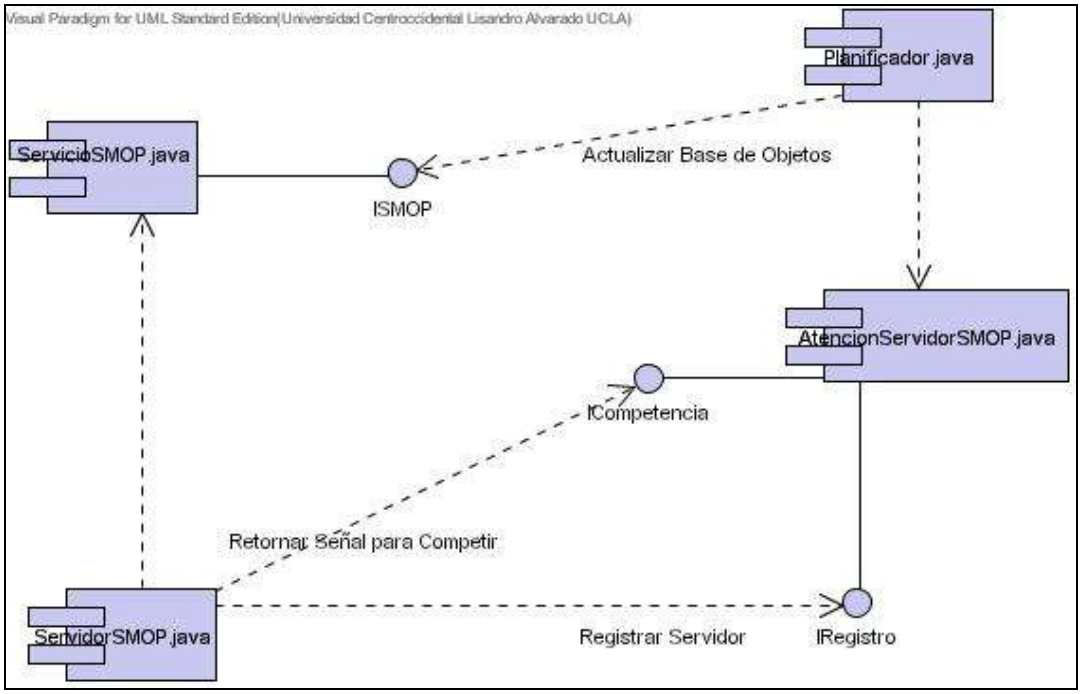


Figura 35. Diagrama de Componentes.

En esta figura se destaca la forma como estos componentes se comunican por medio de las interfaces: ISMOP, Icompetencia e IRegistro, con lo cual se garantiza que puedan ser reemplazados fácilmente en un futuro.

Diagrama de Despliegue

El diagrama de despliegue del SMOP se muestra en la figura 36, con la finalidad de reforzar un poco más la visión física del sistema. Se pueden apreciar los recursos computacional o nodos en donde se despliegan los componentes del SMOP. Los nodos son representados por cubos y se distingue en la ilustración una relación entre ellos que corresponde a una conexión física por cualquier medio en la red y finalmente, se observa la cardinalidad entre ellos. En este caso un Planificador del Grid puede administrar a n nodos o Servidores SMOP y un Servidor SMOP puede estar registrado en más de un Planificador, sin embargo en este trabajo sólo se utilizará una sola instancia del Planificador del Grid, a la vez.

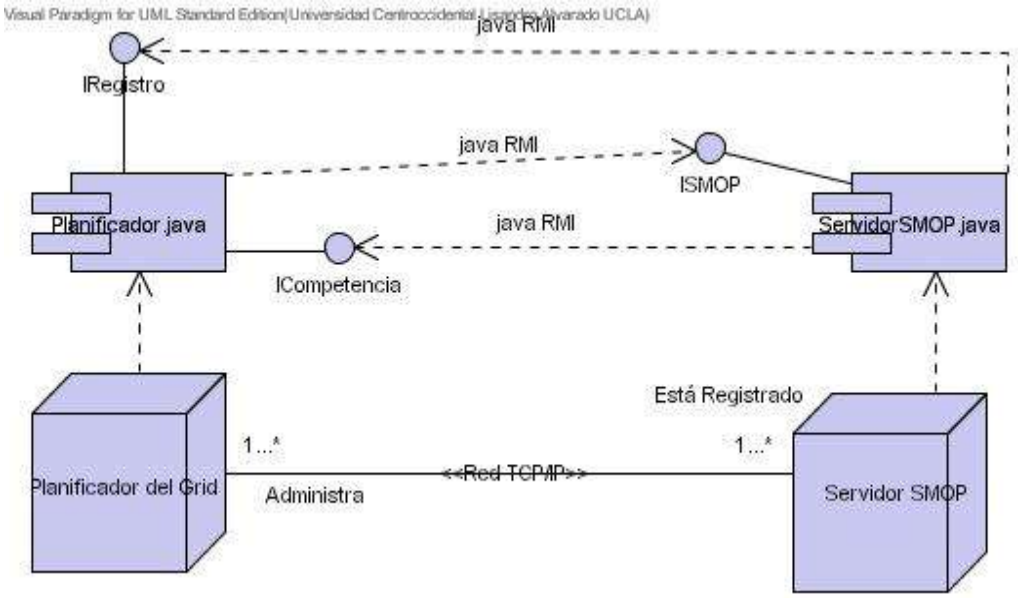


Figura 36. Diagrama de Despliegue.

CAPÍTULO V

IMPLEMENTACIÓN DEL MODELO SMOP

En esta fase se presenta la implementación de un prototipo funcional del Servicio. Este prototipo fue utilizado para la ejecución de las primeras pruebas del SMOP, con la finalidad de validar el diseño propuesto.

Prototipo Funcional

Se comienza la descripción del prototipo funcional con la explicación del mecanismo utilizado como estrategia de simulación, luego se muestran los pasos necesarios para iniciar una simulación del Servicio, más adelante se presenta el estado de las interfaces en la ejecución de una simulación típica y finalmente se obtienen algunos datos estadísticos que reflejan el comportamiento del Servicio en un ambiente de simulación. El SMOP puede trabajar con varios tipos de almacenamiento, sin embargo se realizará la demostración con un solo tipo, no usando el tipo relacional sino el orientado a objetos.

Estrategia de Simulación

Para los efectos de observar la funcionalidad del Servicio desde la perspectiva del Grid, en este proyecto se contempló la elaboración de un mecanismo que muestra una forma de incorporar este Servicio en un Grid determinado, basado en Java. Para ello se ha implementado, a pequeña escala, la programación de un Pseudo-Planificador de un Grid.

El mecanismo que se ha ideado para simular la ejecución de las operaciones de selección de recursos se debe basar en algún factor de decisión. En este caso, el factor es el tiempo de respuesta de las instancias del Servidor SMOP en una institución dada. Esto funciona de la siguiente forma: (a) Se producen, en forma aleatoria, una serie de eventos que representan las necesidades del Planificador de utilizar el SMOP, ya sea para el almacenamiento o recuperación de algunos objetos específicos. (b) Luego se determina cuál es el Servidor SMOP que está en la capacidad de atender con mayor rapidez la solicitud de almacenamiento o recuperación del Planificador. Esto se obtiene al enviar una señal de comunicación (tipo radar), hacia todos los Servidores registrados en el Planificador, con la finalidad de medir el tiempo de respuesta de la señal de retorno. (c) Se selecciona el Servidor SMOP con la señal de retorno más veloz, para ejecutar en él la operación requerida por el Planificador. Se ignoran el resto de los Servidores SMOP con tiempo de respuesta, en sus señales de retorno, superiores al seleccionado.

Es importante señalar, que lo ideal es que el Planificador de un Grid conozca las características particulares de los Servidores SMOP que tiene disponibles en sus registros. Esto es debido a que las mismas pueden conforman otro factor de decisión determinante a la hora de seleccionar algún Servidor SMOP para el procesamiento de una tarea específica; es decir, el Planificador va a tomar la decisión considerando sólo el grupo de Servidores SMOP que cumplen con las características deseadas, en un momento determinado. Esto quiere decir, que el Planificador no trabaja, en todo momento, con la totalidad (el 100%) de los Servidores SMOP sino con un subconjunto de ellos, que son quienes compiten para atender los requerimientos del Planificador.

Concluyendo con la descripción de este mecanismo y para los efectos de esta demostración de la funcionalidad del SMOP, se explica que cada vez que un Servidor es registrado en el Planificador del Grid, se genera automáticamente un conjunto de valores numéricos asociados al mismo. Estos valores representan los códigos de los eventos ante los cuales cada Servidor SMOP retornará una señal al

Planificador, para poder competir por la atención de las necesidades de almacenamiento o recuperación de objetos del Planificador. Finalmente, la interpretación que se da cada vez que el Planificador selecciona un Servidor SMOP debería ser la siguiente: (a) El Servidor SMOP seleccionado cumplió con las características que requiere el Planificador en ese momento determinado y (b) Entre todos los Servidores que cumplen con esas características ese es quien presenta un mejor desempeño.

Demostración de la Funcionalidad del SMOP.

La demostración de la funcionalidad del SMOP se presenta en dos partes: La Demostración Local y Demostración Remota. La Demostración Local, consiste en la presentación de las operaciones básicas para la persistencia de los objetos (Incluir, Modificar, Eliminar y Consultar), en forma local. Son empleadas generalmente por el Administrador local del SMOP, para la Actualización de la Base de Objetos. Estas operaciones se han descrito detalladamente, con anterioridad, en el diseño de los Casos de Uso del sistema. La Demostración Remota tiene que ver también con la presentación de las operaciones básicas de la persistencia de objetos pero utilizando un ambiente de simulación, en donde se atienden los requerimientos de persistencia de objetos (operaciones básicas remotas) de un supuesto Planificador de un Grid, por medio de los Servidores SMOP que se tengan disponibles.

Demostración Local:

En la figura 37 se muestra la interfaz que utiliza el Administrador local, con la finalidad de permitirle la gestión de los recursos que forman parte de su Dominio Administrativo y que colocará a disposición del Grid computacional. Esta gestión o control de recursos la efectúa por medio de las operaciones básicas, que a nivel

local puede realizar con el SMOP; es decir, Incluir, Modificar, Eliminar y Consultar objetos:

Consultar: Cuando se ejecuta esta interfaz, el Administrador entra en modo de consulta de todos los Recursos (objetos) disponibles en la Base de Objetos del Servidor local. Esto se puede apreciar en la *Parte Central*, opción **a** de la figura 37.

Por otro lado JDO maneja un lenguaje de consulta denominado JDOQL, que es utilizado por el SMOP para permitirle al administrador local mostrar una lista de los objetos que cumplan con el criterio seleccionado por el mismo. Para ello se realiza una comparación de las características contenidas en los atributos de los objetos con este criterio.

Incluir: Antes de realizar la inclusión de un nuevo Recurso es necesario tener presente, cuáles son los tipos de Recursos disponibles para ser incluidos en la Base de Objetos (ver la *Parte Superior* de la figura 37, opción **b**.), y cuál es la relación existente entre ellos (ver la Figura 22, en la Fase de Elaboración del Diseño del SMOP. Capítulo IV). Por esta razón, si se quiere indicar la relación de pertenencia de un Recurso dentro de otro, lo primero que se debe hacer es seleccionar con el ratón (mouse) del computador el tipo de Recurso al cual se le desea hacer la inclusión de uno nuevo. Por ejemplo, en la Parte Central de la figura 37, se ha seleccionado un Recurso de tipo *Institución*, identificado como UC (haciendo referencia a la Universidad de Carabobo). Luego para indicar el tipo de Recurso que se va a agregar a esa *Institución* seleccionada, simplemente se escoge por medio del ratón, el nuevo tipo de Recurso, en la *Parte Superior* de la Interfaz y el mismo aparecerá automáticamente, en la *Parte Central* de la misma, debajo de la *Institución* (UC, en este caso).

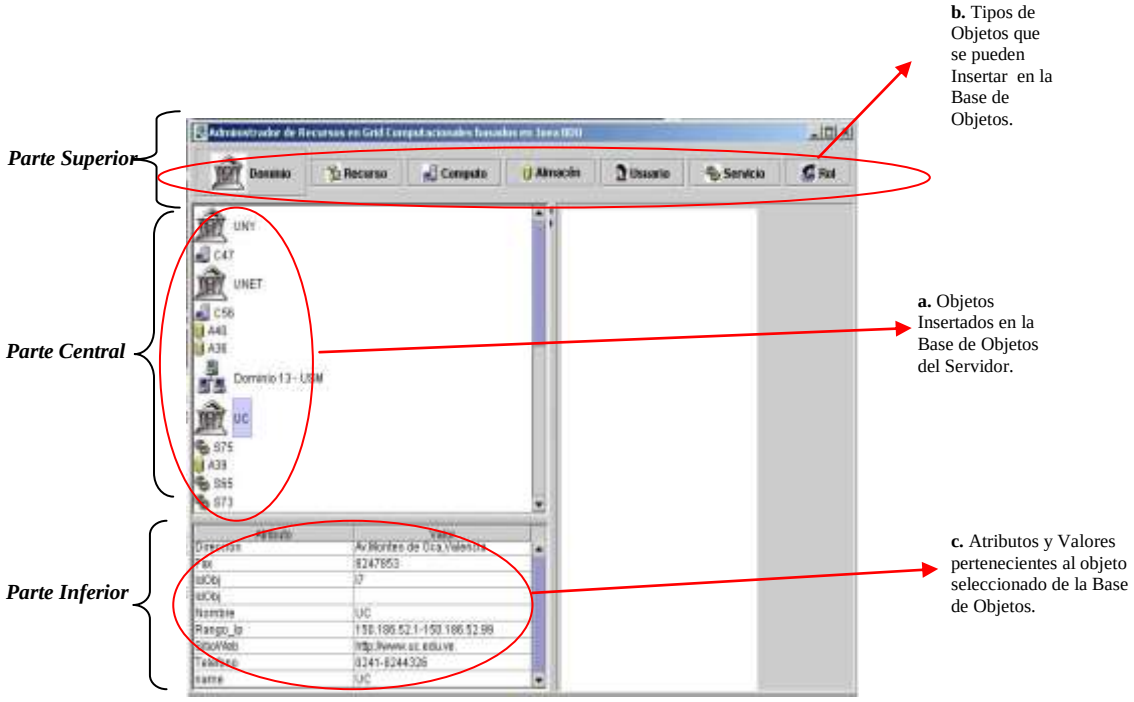


Figura 37. Interfaz Gráfica para la Actualización local de la Base de Objetos del SMOP.

Eliminar: Para la eliminación de los objetos persistentes o recursos disponibles utilizando esta herramienta, el administrador local deberá primero seleccionar el recurso que se dará de baja (Ver Figura 37, Parte Central); es decir que ya no estará disponible para los componentes del Grid. Luego, simplemente presionará la tecla destinada para suprimir desde el teclado de su equipo y automáticamente el recurso será eliminado.

Modificar: Para la operación de modificación, de igual manera se debe seleccionar con el ratón (mouse) del equipo, el objeto al que se le aplicará la modificación (Ver Figura 37, Parte Central). Cada vez que se realiza una selección de objetos como ésta, todos sus atributos son mostrados en la Parte Inferior de la interfaz gráfica, como se aprecia en la Figura 37. De esta manera si se requiere realizar alguna modificación de estos atributos, solo deben ser seleccionados con el ratón del equipo y automáticamente se entra en modo de Edición del atributo, para ser cambiado por medio del teclado del equipo.

Demostración Remota:

Paso 1: Se instancian los Servidores SMOP. La demostración remota del Servicio, en esta segunda parte, se inicia con la instrucción de ejecución de una instancia del Servidor SMOP, en la cual se indica: El tipo de Sistema de Almacenamiento que manejará el Servidor; el nombre de la Base de Objetos (se creará en caso de no existir previamente) y El nombre de la Institución a la cual pertenece el Servidor, tal como se aprecia en la figura 38:

Figura 38. Instrucción para Iniciar un Servidor SMOP

Como resultado de la instrucción anterior se obtiene una interfaz del Servidor SMOP como la desplegada en la figura 39.

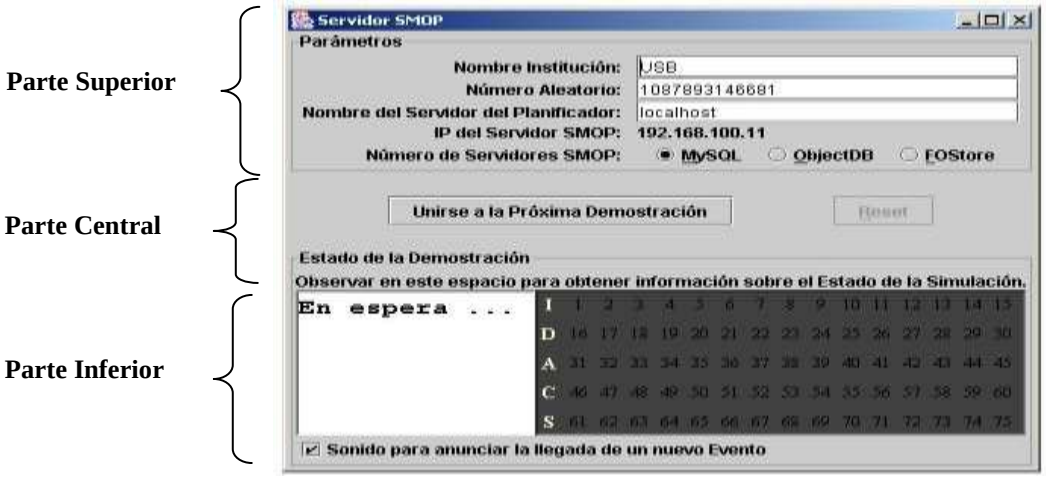


Figura 39. Interfaz Gráfica de un Servidor SMOP

Esta interfaz consiste en una ventana de diálogo, en donde se distinguen claramente tres partes:

Parte Superior: Se encuentran los parámetros del Servidor, entre los que se resaltan el nombre de la institución a la cual pertenece el Servidor SMOP que se está ejecutando, el nombre o la dirección IP del Servidor en donde se está ejecutando el Planificador del Grid, la dirección IP del Servidor SMOP y el tipo de almacenamiento que maneja el Servidor

Parte Central: Se observan dos botones, uno para unirse a una demostración o simulación que se inicia usando un pseudo Planificador y otro botón para reiniciar el Servicio, dejándolo listo para comenzar un nuevo proceso. Además se encuentra una línea para mensajes de seguimiento, que refleja constantemente el estado de la simulación.

Parte Inferior: Se distingue un cuadro de fondo negro a la derecha, que contiene una serie de números, que representan los códigos de los eventos que recibe el Servidor SMOP desde el Planificador, cada cierto intervalo de tiempo (en milisegundos) configurado por el administrador. Estos eventos codificados caracterizan el tipo de requerimiento (recurso de almacenamiento) solicitado por el Planificador. El Servidor SMOP debe comparar cada evento generado con una tabla que contiene un conjunto de eventos específicos, creada en forma aleatoria y asignada a cada Servidor SMOP instanciado. De esa manera se puede identificar si le corresponde o no reaccionar ante el evento para competir con el resto de Servidores SMOP registrados, con la finalidad de atender la solicitud del Planificador, para ese evento en particular. A la izquierda y con fondo blanco se encuentra un recuadro utilizado para mostrar la recepción de un evento generado, en forma remota, por el Simulador, cuyo significado está en que el Planificador anda en la búsqueda de un recurso de almacenamiento específico, para procesar un tipo de objetos particular (**I**nstitución, **D**ominio, **A**lmacenamiento, **C**ómputo o **S**ervicio). Finalmente, se presenta una opción para elegir si se quiere que la interfaz SMOP emita un sonido cada vez que recibe un nuevo evento.

Paso 2: Se ejecuta la Capa de Conectividad: Para desplegar efectivamente el SMOP y permitir el acceso remoto a los métodos que proporciona el mismo, ha sido implementada la capa de conectividad del Servicio, por medio de la Invocación de Métodos Remotos de java RMI. El servicio de nombres RMI es usado para obtener referencias a los objetos remotos y poder aprovechar sus operaciones. Este servicio es iniciado con la siguiente instrucción: *start rmiregistry*. La figura 40 muestra una pequeña ventana, indicando la correcta ejecución del servicio de nombres RMI.

Figura 40. Ejecución del Servicio de Nombres RMI.

Paso 3: Se instancia el Planificador. Otro elemento importante para la realización de esta demostración es el componente de software que permite simular el funcionamiento del Planificador del Grid. En la figura 41 se aprecia la ejecución de un archivo en lotes (o *script*) que contiene las instrucciones necesarias para ejecutar el pseudo Planificador del Grid que utiliza este Servicio.

Figura 41. Ejecución del Planificador del Grid de la Simulación.

Como resultado de la instrucción anterior se obtiene una interfaz gráfica que permite iniciar la ejecución de un ambiente de simulación, en donde el

Planificador del Grid selecciona en forma aleatoria un conjunto de objetos para procesar su persistencia por medio del SMOP propuesto en esta investigación. Se utilizan las diferentes Bases de Objetos que administran los Servidores SMOP activos; es decir, que se estén ejecutando y se hayan registrado previamente en el Planificador. En la figura 42 que se muestra a continuación, se describen los detalles de esta interfaz:

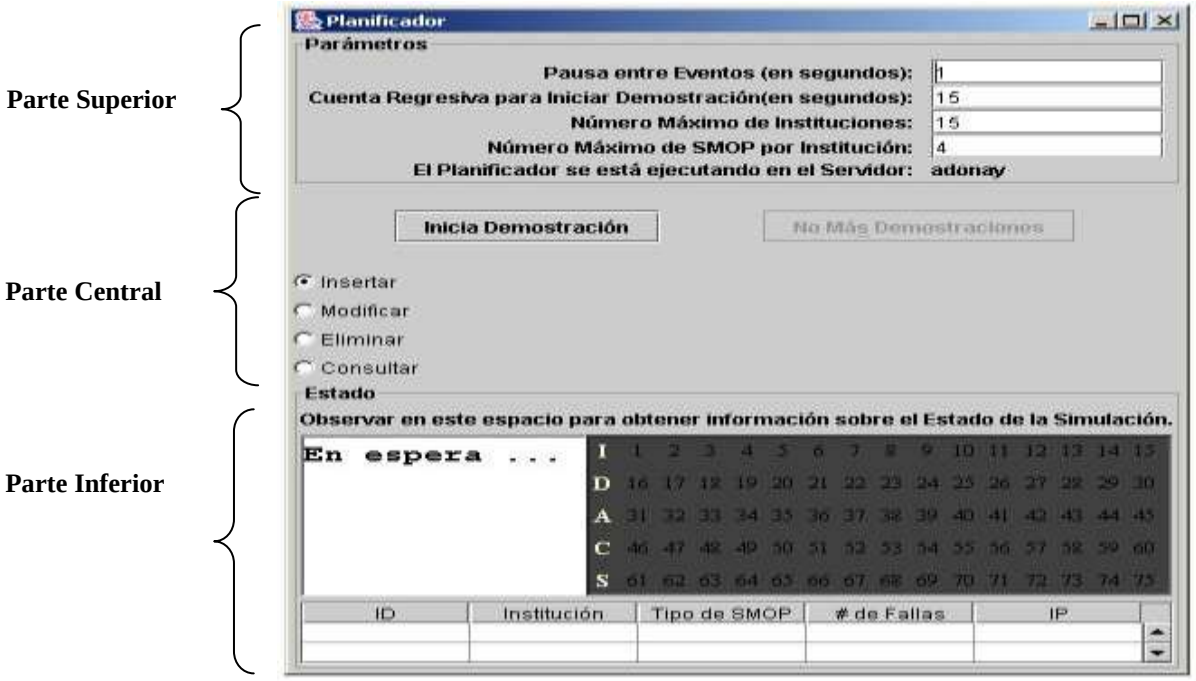


Figura 42. Interfaz Gráfica para iniciar la Simulación del Planificador del Grid.

En la interfaz gráfica se distinguen tres partes:

Parte Superior: Contiene algunos parámetros de la simulación, como por ejemplo la indicación de pausa entre los eventos que se generan del lado del Planificador hacia los diferentes Servidores SMOP. Además, se tiene una cuenta regresiva (en segundos) antes de iniciar la simulación; esto es para establecer el tiempo de que disponen los Servidores SMOP, para decidir unirse o no a la simulación que se está próxima a iniciarse. Adicionalmente, esta interfaz permite restringir el número máximo de instituciones que se pueden registrar en la simulación y el número máximo de Servidores que se

pueden registrar por cada institución. Esta primera parte culmina con una línea que muestra el nombre o dirección IP del Servidor en donde se está ejecutando esta interfaz.

Parte Central: Se presentan dos botones, uno de ellos está a la izquierda y permite dar inicio a una simulación y el otro botón, el de la derecha, indica que al terminarse la simulación que está en marcha en ese momento, no se permitirá iniciar ninguna otra, hasta no volver a ejecutar la interfaz. Además, en esta parte se permite elegir las operaciones básicas (Insertar, Modificar, Consultar y Eliminar) para procesar los objetos que sean seleccionados por el Planificador.

Parte Inferior: Inicia con una línea de información, que indica el estado en que se encuentra la simulación. Seguidamente en la parte inferior se distinguen dos cuadros, uno a la derecha en fondo negro que contiene una serie de valores numéricos que representan los eventos que son generados aleatoriamente para indicar la necesidad del Planificador de localizar a un recurso que coincida con algunas características particulares establecidas. Ese código es precisamente el que se envía como señal a todos los Servidores SMOP conectados al Planificador para determinar cuál de ellos cumple con esas características especificadas, y que están representadas por ese valor numérico. El otro cuadro se encuentra ubicado del lado izquierdo de la interfaz, y en él se muestra el tipo de Objeto Persistente que será procesado por cada evento aleatorio asociado. Finalmente, en la parte inferior se puede apreciar un recuadro que muestra la información general de los Servidores SMOP registrados en ese Planificador: código de identificación, nombre de la institución, tipo de sistema de almacenamiento utilizado y dirección IP.

Paso 4: Se inicia la Simulación. La figura 43 muestra una instancia de la interfaz del planificador, que se ha obtenido al iniciar el proceso de simulación. Se distinguen claramente algunos elementos importantes tales como: Código de

eventos generados, número de eventos, tipo de operación, tipo de objeto que está siendo procesado, e información sobre un Servidor SMOP que ha sido registrado, entre otros.

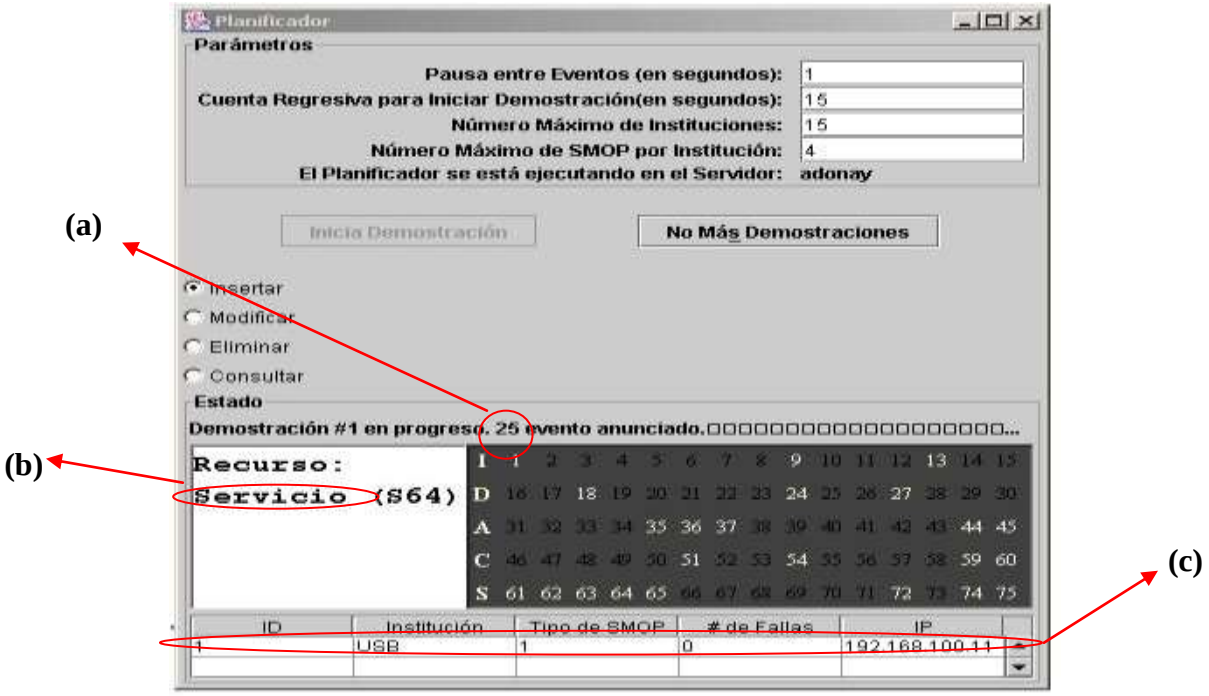


Figura 43. Vista de la Interfaz del Planificador durante la ejecución del simulador.

En el caso de la figura 43, se puede asegurar, por simple observación, que: **(a)** Ya se han anunciado, para ese instante, una cantidad de 25 eventos; **(b)** El tipo de objetos a procesar, en ese momento, es el tipo de objeto denominado *Servicio* y **(c)** que existe una Institución registrada con el nombre de USB, cuyo servidor puede ser ubicado en la dirección IP 192.168.100.11

Se muestra el estado de la Interfaz del Servidor SMOP, en la figura 44, como contraparte (porque ambas se ejecutan simultáneamente) de la Interfaz del Planificador presentada anteriormente (figura 43). Estas interfaces fueron obtenidas durante la ejecución de una simulación iniciada para demostrar la funcionalidad del SMOP propuesto en esta investigación.

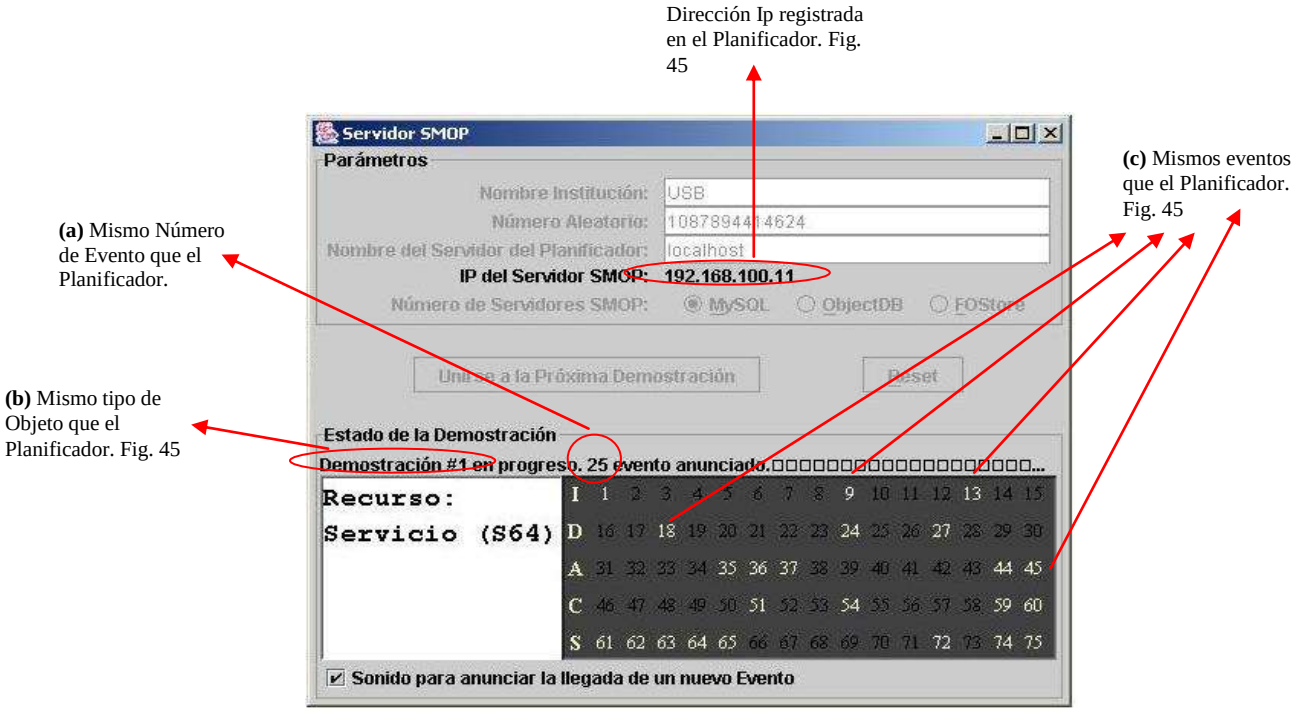


Figura 44. Vista de la Interfaz de un Servidor SMOP durante la ejecución del simulador.

Se quiere resaltar de manera muy especial la relación observada en ambas imágenes con respecto a: **(a)** La coincidencia en el número de eventos anunciados, **(b)** El tipo de objeto que se está procesando (*Servicio*) en un instante determinado y el **(c)** código de los eventos generados. Estos se presentan en el recuadro de fondo oscuro, distinguiendo el código numérico de los eventos en un color más claro, para indicar que los eventos que han sido generados en forma remota por el Planificador del Grid simulado, también han sido recibidos exitosamente en el Servidor SMOP.

Una vez efectuada la recepción de cada uno de estos eventos, el Servidor SMOP debe revisar, el código del evento, con el conjunto de códigos numéricos creado durante su instanciación. Ya que, como se ha explicado anteriormente, estos números caracterizan al Servidor y le permiten reaccionar ante la señal de localización enviada

por parte del Planificador simulado, para tratar de ubicar un Servidor SMOP que cumpla con las características que representan estas señales. El conjunto de eventos esperados por el Servidor SMOP es generado aleatoriamente, y un ejemplo de ello se puede apreciar en la ilustración presentada en la figura 45.

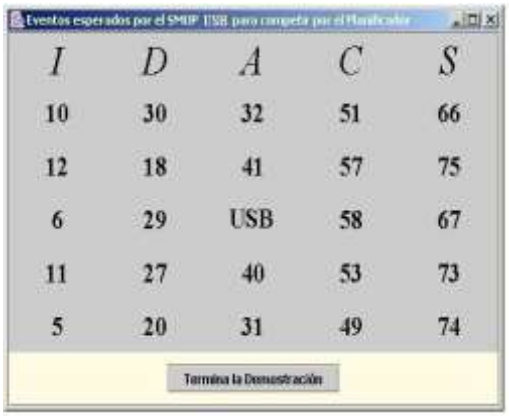


Figura 45. Conjunto de eventos que caracterizan al Servidor SMOP para reaccionar ante las señales enviadas por el Planificador simulado.

En la figura 46, se visualiza del lado del Servidor SMOP la reacción producida en el mismo como consecuencia de la localización de un evento enviado por el Planificador, dentro del conjunto de eventos esperados. Básicamente esta reacción consiste en imprimir un mensaje indicando la localización exitosa del evento, identificando el tipo de objeto que se espera sea procesado y en enviar una señal de retorno al Planificador con la finalidad de competir con el resto de servidores para optar por la selección del mismo. Se reitera que esta es la forma en que se determina qué Servidor SMOP va a procesar las operaciones de persistencia de objetos, solicitadas por el Planificador en su Base de Objetos.

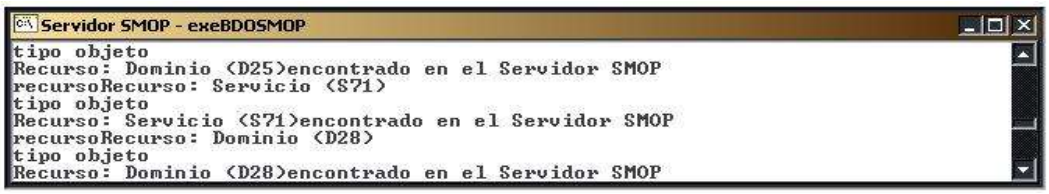


Figura 46. Reacción del Servidor SMOP ante la localización de un evento generado por el Planificador.

Figura 47. Resultado de las operaciones de actualización de Base de Objetos solicitadas por el Planificador.

Finalmente se muestra en la figura 47 el resultado de las operaciones de procesamiento de los objetos persistentes, las cuales han sido solicitadas a los Servidores SMOP elegidos de acuerdo al mecanismo utilizado para tal fin. En ella se puede apreciar como dato importante el tiempo en milisegundos (ms) empleado para el procesamiento de dichas solicitudes. Esto proporciona un valor de gran interés, que puede ser considerado a la hora de evaluar el rendimiento del servicio en general.

Pruebas

Es importante señalar que estas pruebas no tienen ninguna finalidad de comprobar el rendimiento del SMOP. Para ello sería necesario realizar otro trabajo de investigación por separado. Sin embargo, las pruebas se han querido realizar con la idea de introducir algunos datos e indicadores preliminares, que puedan ser de apoyo, como punto de partida para investigaciones futuras.

A continuación se describe la experiencia obtenida como resultado de una serie de experimentos realizados con el Simulador en una pequeña plataforma de red.

La plataforma consistió en la interconexión de 6 equipos de computación a través de una red Ethernet 10/100 MB con las siguientes características: (a) Un equipo portátil toshiba con 128 MB de RAM, 30 GB de disco duro y un procesador Intel celeron de 700 Mhz; este equipo se identificó con el nombre de la Institución UCLA. Sistema operativo windows 2000 Server; (b) Un equipo IBM pc 300 GL con 64 MB de RAM , un disco duro de 2 GB y un procesador Intel Pentium II de 350 Mhz; este equipo se identificó con el nombre de la Institución USB. Sistema operativo NT

WorkStation 4.0; (c) Un equipo IBM pc 300 GL con 64 MB de RAM , un disco duro de 2 GB y un procesador Intel Pentium II de 350 Mhz; este equipo se identificó con el nombre de la Institución ULA. Sistema operativo windows NT workstation 4.0 (d) Un equipo IBM Netfinity 3500 con 128 MB de RAM, 9 GB de disco duro y un procesador Intel Pentium III de 550 Mhz; este equipo se identificó con el nombre de la Institución UCAB. Sistema operativo windows 2000 Server; (e) Un equipo IBM Netvista con 128 MB de RAM, 20 GB con disco duro y un procesador Intel Pentium III de 1000 Mhz; este equipo se identificó con el nombre de la Institución LUZ. Sistema operativo windows 2000 Server; (f) Un equipo IBM pc 300 GL con 64 MB de RAM , un disco duro de 40 GB y un procesador Intel Pentium II de 350 Mhz; en este equipo se ejecutó el Planificador del simulador; Sistema operativo windows 2000 Profesional.

Se realizaron diez (10) experimentos de simulación sobre esta plataforma, utilizando el ObjectDB como sistema de almacenamiento para el soporte de la base de objetos, el cual es un sistema de base de datos orientado a objetos. Los resultados de los experimentos se han resumido en los siguientes recuadros:

Experimento 1:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	3	12	170	56,67	50	60
172.17.8.138	USB	10	40	391	39,10	20	60
172.17.8.156	ULA	13	52	541	41,62	20	60
172.17.8.29	UCAB	15	60	1444	96,27	10	812
172.17.8.157	LUZ	21	84	1560	74,29	20	360

Cuadro 5: Resultados del Experimento 1 de la Simulación.

Experimento 2:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	12	48	640	53,33	30	140
172.17.8.138	USB	15	60	651	43,40	21	110
172.17.8.156	ULA	12	48	2383	198,58	20	1730
172.17.8.29	UCAB	11	44	752	68,36	20	271
172.17.8.157	LUZ	18	72	1570	87,22	20	400

Cuadro 6: Resultados del Experimento 2 de la Simulación.

Experimento 3:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	5	20	392	78,40	10	291
172.17.8.138	USB	8	32	280	35,00	20	50
172.17.8.156	ULA	15	60	1314	87,60	10	291
172.17.8.29	UCAB	19	76	1691	89,00	10	901
172.17.8.157	LUZ	19	76	752	39,58	20	101

Cuadro 7: Resultados del Experimento 3 de la Simulación.

Experimento 4:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	5	20	611	122,20	20	420
172.17.8.138	USB	14	56	2344	167,43	30	1341
172.17.8.156	ULA	11	44	600	54,55	30	140
172.17.8.29	UCAB	15	60	931	62,07	30	120
172.17.8.157	LUZ	20	80	1643	82,15	30	381

Cuadro 8: Resultados del Experimento 4 de la Simulación.

Experimento 5:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	9	36	792	88,00	30	421
172.17.8.138	USB	14	56	762	54,43	20	121
172.17.8.156	ULA	11	44	2303	209,36	30	1672
172.17.8.29	UCAB	12	48	581	48,42	20	110
172.17.8.157	LUZ	19	76	903	47,53	20	110

Cuadro 9: Resultados del Experimento 5 de la Simulación.

Experimento 6:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	6	24	452	75,33	20	191
172.17.8.138	USB	16	64	901	56,31	30	190
172.17.8.156	ULA	12	48	1964	163,67	20	1503
172.17.8.29	UCAB	16	64	1231	76,94	30	310
172.17.8.157	LUZ	17	68	823	48,41	20	110

Cuadro 10: Resultados del Experimento 6 de la Simulación.

Experimento 7:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	7	28	351	50,14	40	70
172.17.8.138	USB	19	76	2372	124,84	40	1241
172.17.8.156	ULA	13	52	750	57,69	40	120
172.17.8.29	UCAB	15	60	1132	75,47	30	371
172.17.8.157	LUZ	12	48	1022	85,17	20	451

Cuadro 11: Resultados del Experimento 7 de la Simulación.

Experimento 8:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	7	28	180	25,71	10	40
172.17.8.138	USB	12	48	370	30,83	10	70
172.17.8.156	ULA	12	48	280	23,33	10	60
172.17.8.29	UCAB	14	56	340	24,29	10	50
172.17.8.157	LUZ	19	76	1182	62,21	10	541

Cuadro 12: Resultados del Experimento 8 de la Simulación.

Experimento 9:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	6	24	180	30,00	10	40
172.17.8.138	USB	15	60	570	38,00	10	70
172.17.8.156	ULA	12	48	491	40,92	20	80
172.17.8.29	UCAB	16	64	2173	135,81	30	961
172.17.8.157	LUZ	16	64	1134	70,88	21	270

Cuadro 13: Resultados del Experimento 9 de la Simulación.

Experimento 10:

Ubicación IP Servidor	Servidor SMOP	Nro. Objetos Atendidos	% Atención	Tiempo Acumulado	Tiempo Promedio (ms)	Tiempo Mínimo (ms)	Tiempo Máximo (ms)
172.17.8.21	UCLA	8	32	241	30,13	10	61
172.17.8.138	USB	13	52	1082	83,23	10	341
172.17.8.156	ULA	8	32	310	38,75	10	90
172.17.8.29	UCAB	20	80	2052	102,60	20	931
172.17.8.157	LUZ	16	64	792	49,50	20	170

Cuadro 14: Resultados del Experimento 10 de la Simulación.

Análisis de Resultados

Actualmente, se presentan indicadores como el denominado porcentaje de atención *(% Atención)*. Se calculó tomando en consideración la capacidad máxima de atención de cada Servidor SMOP en cada uno de los experimentos, que en este caso fue de 25 objetos. Este número coincide con el conjunto de eventos generados aleatoriamente, a partir de los cuales reaccionan los servidores registrados en el planificador. Ahora bien si se toma el *número de objetos atendidos* efectivamente y se compara con la capacidad máxima de atención se puede obtener una tasa o porcentaje de atención usando la siguiente fórmula:

*% de Atención = Nro. de Objetos Atendidos * 100 / Capacidad Máxima de Atención.*

De esta misma forma se pueden obtener otros indicadores que se requieran de acuerdo a los aspectos del servicio que se deseen observar. En esta investigación simplemente se han recopilado estos datos experimentales para tener una idea muy general del funcionamiento del servicio.

Para finalizar con esta sección, se presentan algunos indicadores adicionales que pueden ser considerados como parte de la evaluación preliminar del Servicio SMOP, entre otros están : (a) El *tiempo promedio* de duración de las operaciones realizadas en el servidor. Este indicador se calcula sumando la duración de cada transacción realizada en el Servidor dividida entre el número de operaciones efectuadas en el mismo.(b) Otro indicador que pudiera ser de interés en esta observación del servicio sería la *demanda insatisfecha* del Planificador del Grid simulado. Este indicador se obtiene al tomar, por un lado, el número total de eventos generados por el Planificador y por el otro, al verificar el número de objetos que efectivamente se atendieron. Este número es exactamente igual al número de eventos que fueron generados por el Planificador y que por lo menos hallaron algún Servidor SMOP que reaccionara ante la señal recibida, que está asociada al evento generado. En concreto, se toma el número total de eventos generados por el Grid para cada experimento. Esto es 75 (eventos) y se compara con la suma de todos los objetos atendidos por los Servidores registrados en el Planificador y se obtiene un porcentaje de *demanda satisfecha e insatisfecha* del Planificador, mediante esta fórmula:

% demanda insatisfecha = (Sumatoria de todos los objetos atendidos por los Servidores) * 100 / Número total de eventos generados (75).

El siguiente gráfico (Gráfico 1) muestra la demanda satisfecha e insatisfecha del Planificador obtenida en la simulación para cada experimento realizado:

Indicador	Experimento									
	Nº1	Nº2	Nº3	Nº4	Nº5	Nº6	Nº7	Nº8	Nº9	Nº10
% Demanda Insatisfecha	17,33	9,33	12	13,33	13,33	10,67	12	14,67	13,33	13,33
% Demanda Satisfecha	82,67	90,67	88	86,67	86,67	89,33	88	85,33	86,67	86,67

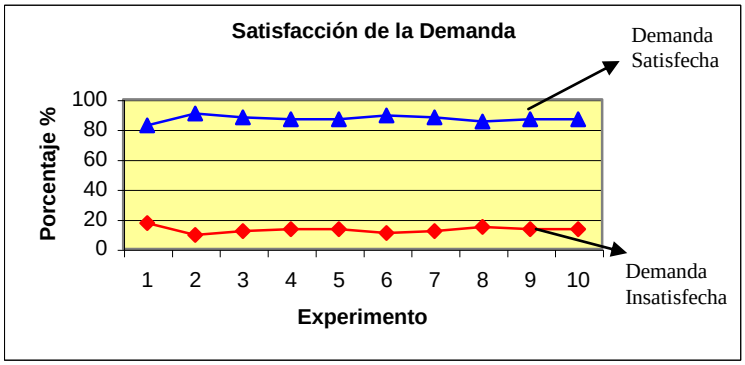
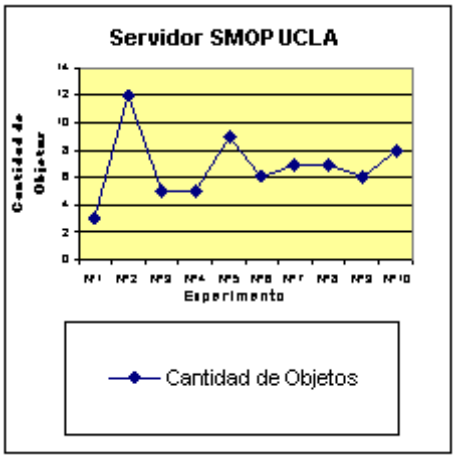
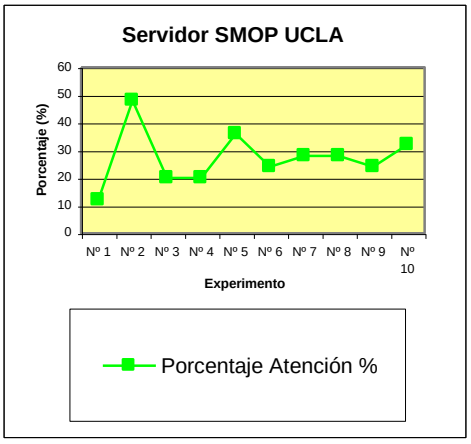


Gráfico 1: % Demanda Satisfecha e Insatisfecha del Planificador en la Simulación.

El resto de este capítulo muestra los datos obtenidos en la experimentación con el simulador del Servicio, agrupados por cada Servidor SMOP participante, identificado con el nombre de la Institución otorgado para efectos de la simulación. El análisis del rendimiento del SMOP basado en estos resultados, está fuera del alcance de esta investigación. Se presentan acá simplemente como un tendencia que refleja el comportamiento del Servicio en una plataforma experimental determinada, que pudiera servir como punto de partida de futuras investigaciones en donde se realicen de manera más exhaustiva la experimentación del servicio, tomando en consideración otras variables y adaptando la simulación a una realidad de un Grid en particular que se establezca como caso de estudio, por ejemplo el Grid SUMA.

UCLA	Cantidad de Objetos	Porcentaje Atención %	Tiempo Promedio (ms)
Nº 1	3	12	56,67
Nº 2	12	48	53,33
Nº 3	5	20	78,40
Nº 4	5	20	122,20
Nº 5	9	36	88,00
Nº 6	6	24	75,33
Nº 7	7	28	50,14
Nº 8	7	28	25,71
Nº 9	6	24	30,00
Nº 10	8	32	30,13





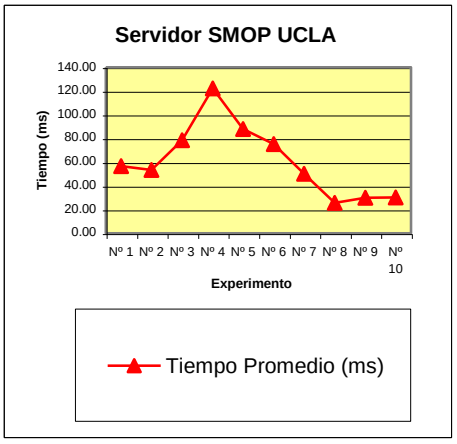
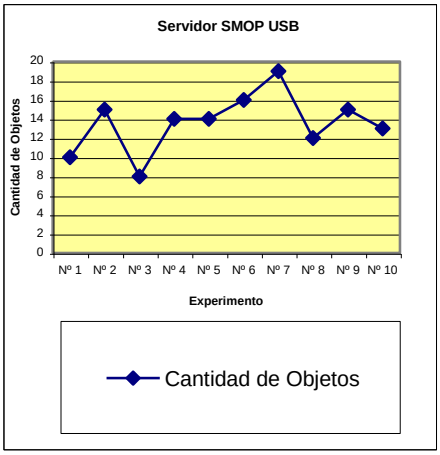
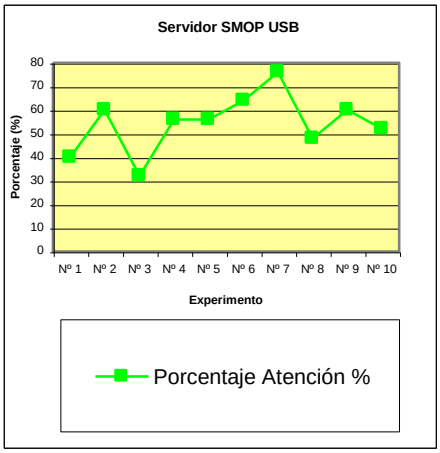


Gráfico 2. Resultados de la Experimentación del Servidor SMOP denominado UCLA en la simulación.

USB	Cantidad de Objetos	Porcentaje Atención %	Tiempo Promedio (ms)
Nº 1	10	40	39,10
Nº 2	15	60	43,40
Nº 3	8	32	35,00
Nº 4	14	56	167,43
Nº 5	14	56	54,43
Nº 6	16	64	56,31
Nº 7	19	76	124,84
Nº 8	12	48	30,83
Nº 9	15	60	38,00
Nº 10	13	52	83,23





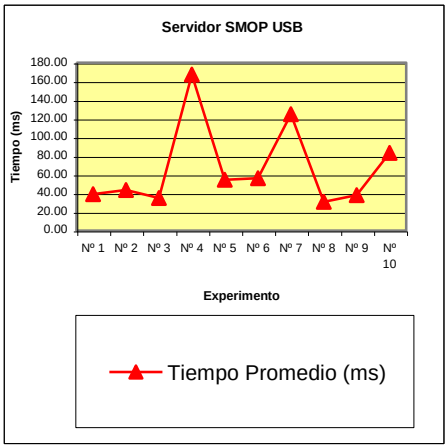
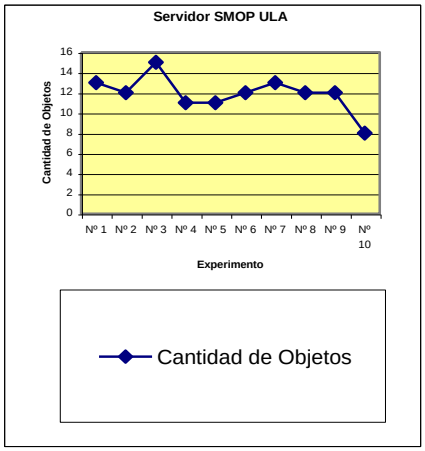
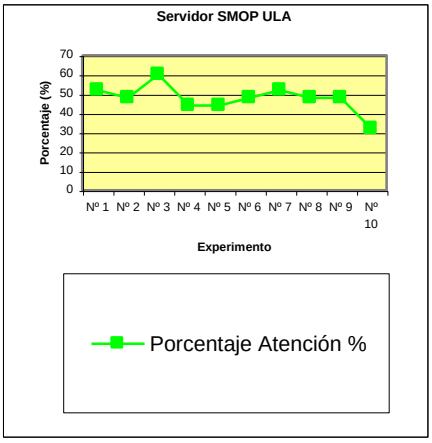


Gráfico 3. Resultados de la Experimentación del Servidor SMOP denominado USB en la simulación.

ULA	Cantidad de Objetos	Porcentaje Atención %	Tiempo Promedio (ms)
Nº 1	13	52	41,62
Nº 2	12	48	198,58
Nº 3	15	60	87,60
Nº 4	11	44	54,55
Nº 5	11	44	209,36
Nº 6	12	48	163,67
Nº 7	13	52	57,69
Nº 8	12	48	23,33
Nº 9	12	48	40,92
Nº 10	8	32	38,75





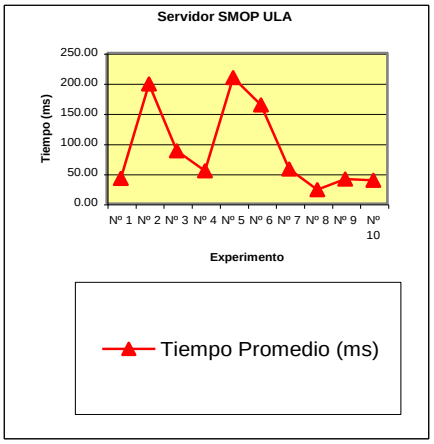
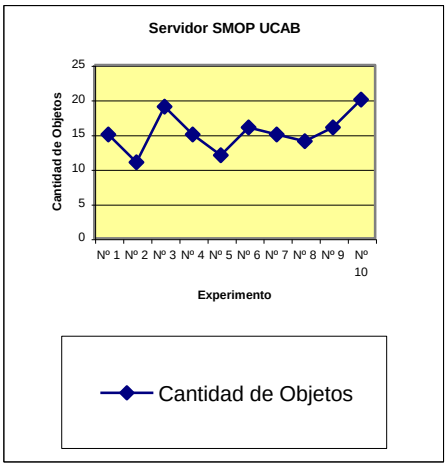
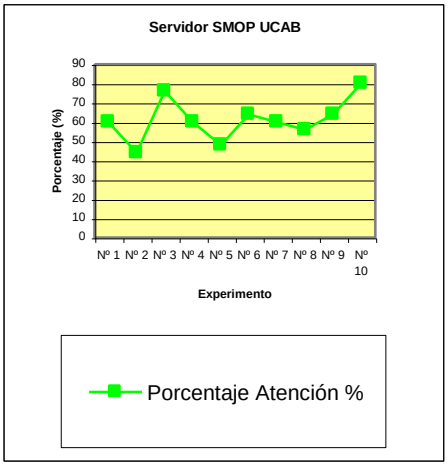


Gráfico 4. Resultados de la Experimentación del Servidor SMOP denominado ULA en la simulación.

UCAB	Cantidad de Objetos	Porcentaje Atención %	Tiempo Promedio (ms)
Nº 1	15	60	96,27
Nº 2	11	44	68,36
Nº 3	19	76	89,00
Nº 4	15	60	62,07
Nº 5	12	48	48,42
Nº 6	16	64	76,94
Nº 7	15	60	75,47
Nº 8	14	56	24,29
Nº 9	16	64	135,81
Nº 10	20	80	102,60





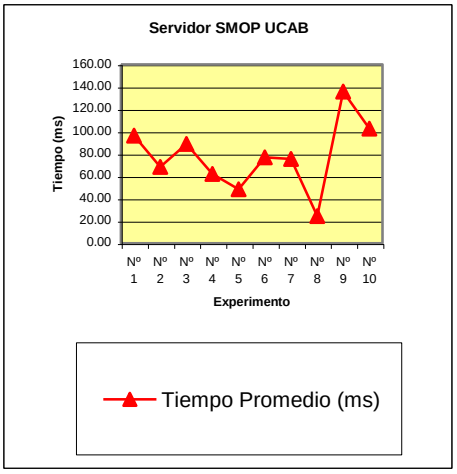
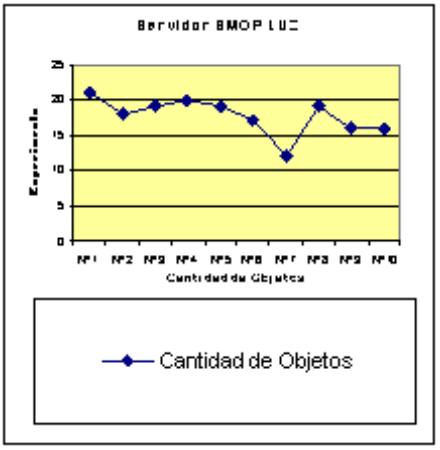
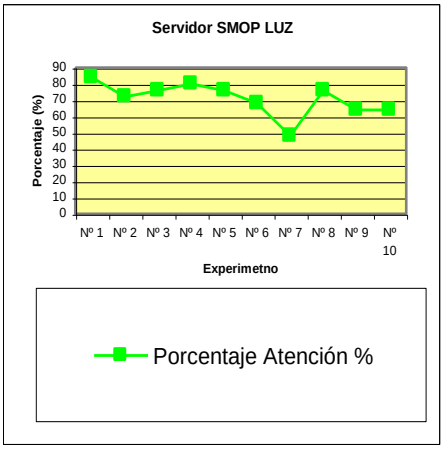


Gráfico 5. Resultados de la Experimentación del Servidor SMOP denominado UCAB en la simulación.

LUZ	Cantidad de Objetos	Porcentaje Atención %	Tiempo Promedio (ms)
Nº 1	21	84	74,29
Nº 2	18	72	87,22
Nº 3	19	76	39,58
Nº 4	20	80	82,15
Nº 5	19	76	47,53
Nº 6	17	68	48,41
Nº 7	12	48	85,17
Nº 8	19	76	62,21
Nº 9	16	64	70,88
Nº 10	16	64	49,50





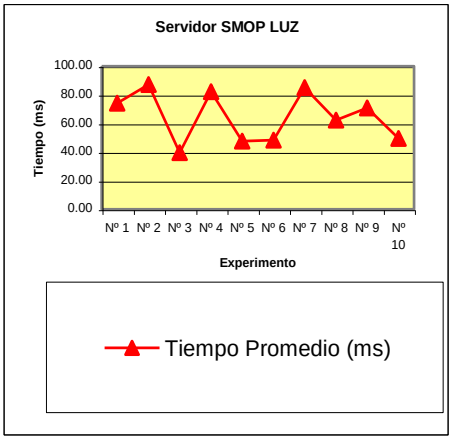


Gráfico 6. Resultados de la Experimentación del Servidor SMOP denominado LUZ en la simulación.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

En la presente investigación se ha obtenido como resultado final, el diseño del modelo del Servicio SMOP (Servicio Manejador de Objetos Persistentes). Este servicio permite el manejo de la persistencia de objetos java en sistemas de objetos distribuidos, como lo son aquellos utilizados para las implementaciones de los Grids Computacionales. Se tomó como un punto referencial, el Grid Computacional basado en java desarrollado en el ámbito nacional, denominado SUMA, con el cual se realizó un levantamiento de la información necesaria para elaboración de esta investigación.

Los diversos recursos heterogéneos disponibles en el Grid fueron representados, en esta investigación, por los objetos persistentes procesados por el Servicio SMOP propuesto. Este servicio fue diseñado para administrar el almacenamiento y recuperación de objetos, sobre un Repositorio o Base de Objetos también heterogénea soportada, para esta versión del Servicio, básicamente por Sistemas de Almacenamiento de tres tipos: Sistemas de Archivos, Base de Datos Relacional y Base de Datos Orientada a Objetos.

Se realizó un mecanismo de simulación que permitió demostrar la funcionalidad del Servicio mediante la ejecución de un prototipo funcional y se obtuvieron algunos datos estadísticos, los cuales proporcionan una idea del comportamiento que presenta el Servicio SMOP bajo con las condiciones impuestas en la simulación y permiten tener valores de referencia con la finalidad de evaluar posteriormente y de manera más exhaustiva el rendimiento del Servicio.

Entre los beneficios que se pueden destacar al utilizar el Servicio SMOP en los Grid Computacionales basados en java, se encuentran: (a) El Planificador del Grid

puede delegar completamente a este servicio, el manejo de la persistencia de los objetos (Recursos) que administra; (b) El Servicio pudiera ser utilizado por otros componentes del Grid , como por ejemplo el Controlador de Usuarios, debido a que éste también maneja información persistente (datos personales, privilegios, entre otros) de los usuario del Grid; (c) El Servicio permite al Grid manejar tipos de datos persistentes más complejos como lo son los objetos; (d) Al estar soportado el Servicio por un tipo de Sistema de Almacenamiento como las Bases de Datos o de Objetos, entre otros, se incorporan las ventajas presentes en las mismas, entre las cuales se nombran las siguientes: Control sobre la redundancia de los datos, Consistencia de los Datos, Datos Compartidos, Integridad de los datos, Mejora la Seguridad, Accesibilidad a los Datos, Manejo de la Concurrencia; (e) El Servicio maneja los estándares para la persistencia en java, favoreciendo la interoperabilidad y portabilidad del mismo.

Entre los hallazgos encontrados en el desarrollo de esta investigación se destacan:

- El gran esfuerzo que se está realizando a nivel nacional para el mejoramiento de la Red Académica de Centros de Investigación y Universidades Nacionales, a través del proyecto Reacciun2, mediante el cual se tiene previsto, en una primera etapa, la interconexión de algunas Universidades e Instituciones de Investigación con las redes Internet2 de alta velocidad. De esta manera se garantiza una plataforma tecnológica adecuada para la implantación de Grids Computacionales que proporcionen un valor agregado a la misma, conjuntamente con el Servicio propuesto en esta investigación.
- El Grid computacional SUMA ha surgido como una implementación de un proyecto Grid basado en java, destacado en el ámbito nacional.

Entre los aportes de este trabajo de investigación se resaltan:

- El Modelo del Esquema lógico de la Base de Objetos del Grid (Ver).
- El Mecanismo de simulación.
- Los “artefactos” resultantes de la aplicación de RUP en el área de los Grids Computacionales basados en Java.
- El Prototipo funcional del SMOP.

Recomendaciones

- Completar un esquema de simulación en forma más general que describa con mayor fidelidad las condiciones reales presente en una implementación de un Grid en particular.
- Implementar el Servicio en una plataforma de un Grid en funcionamiento como por ejemplo SUMA.
- Realizar otra serie de experimentos de simulación del servicio, integrando una mayor cantidad de recursos e incorporando nuevos indicadores matemáticos que permitan un análisis, más completo, de los resultados.
- Rediseñar el mecanismo de Simulación para la localización y asignación de Recursos por parte del Planificador, de tal manera que se incorporen técnicas inteligentes (inteligencia artificial) al mismo.

Adicionalmente, se considera adecuado recomendar estudios futuros que se puedan realizar a partir de esta investigación:

- Para darle continuidad a esta investigación se pudiera intentar la integración de este servicio con SUMA tomándolo, exclusivamente, como caso de estudio de un Grid computacional.
- También se pueden realizar estudios relacionados con la búsqueda de aplicaciones de este Servicio en la Red UCLA, por ejemplo explorando áreas de investigación tales como la Data Warehouse, Minería de Datos o Descubrimiento del conocimiento (KDD) entre otros, en los sistemas de información de la Universidad.
- Incorporar tecnología de agentes inteligentes e Ingeniería de Software basada en Componentes reutilizables en el desarrollo de futuras versiones del Servicio.
- Elaborar proyectos y líneas de investigación en el Decanato de Ciencias y Tecnología de la UCLA dirigidas a proporcionar productos de software que permitan generar un valor agregado a las plataformas de alta velocidad (Internet2) que se encuentran en proceso de instalación.

Se finalizan las recomendaciones con una cita textual tomada como parte de las conclusiones del taller sobre Grids / E-Ciencia, que se realizó durante el primer

seminario de “Aplicaciones científicas y educativas en Redes de Alto Rendimiento (Internet2)” realizado el 29 y 30 de Abril del presente año, en donde se invita a :

“Asociarse y participar en grupos técnicos de trabajo en el desarrollo de aplicaciones Grid”.

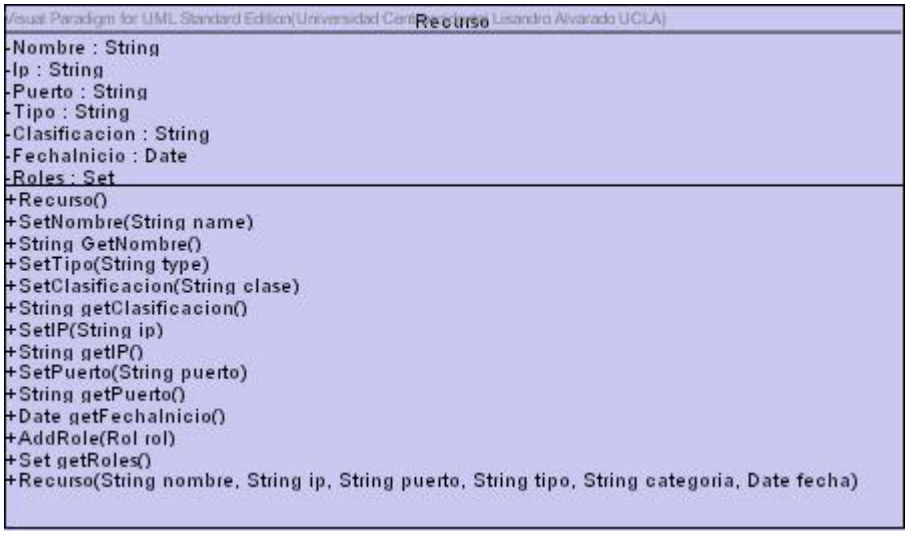
ANEXOS

ANEXO A

CLASES ENTIDAD

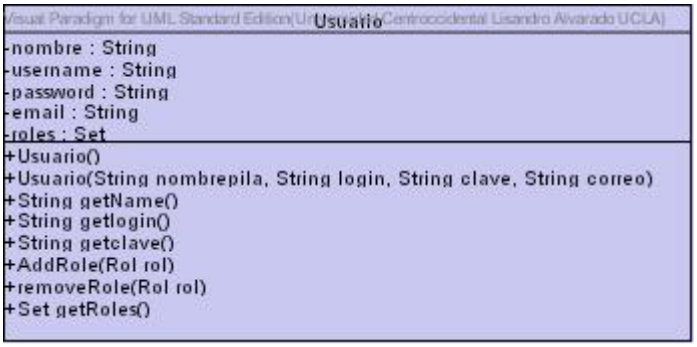
Atributos y Métodos de la Clase Entidad: Institución

Atributos y Métodos de la Clase Entidad: Dominio



Atributos y Métodos de la Clase Entidad: Recurso

Atributos y Métodos de la Clase Entidad: Rol

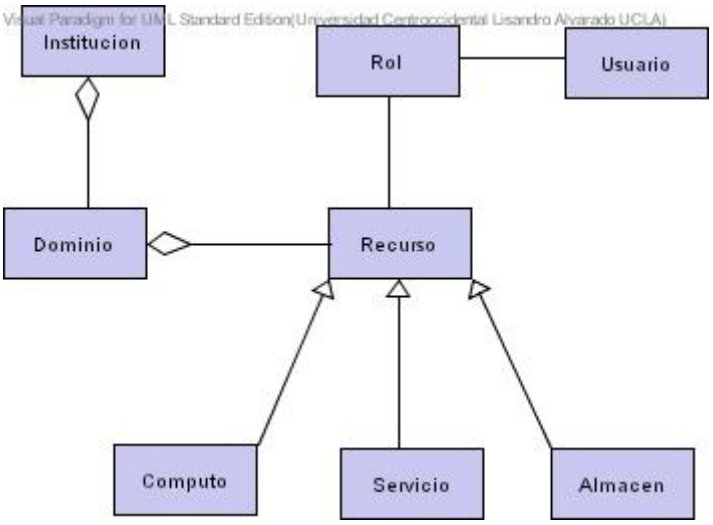


Atributos y Métodos de la Clase Entidad: Usuario

Atributos y Métodos de la Clase Entidad: Almacén

Atributos y Métodos de la Clase Entidad: Cómputo

Atributos y Métodos de la Clase Entidad: Servicio



Relaciones entre las Clases Tipo Entidad (Diagrama de Clases)

ANEXO B

DESCRIPCIÓN TEXTUAL DE LOS CASOS DE USO POR MEDIO DE PLANTILLAS

<table>
<tr><td colspan="2">Caso de Uso: Crear Base de Objetos</td><td>Referencia: UC-SMOP_01</td></tr>
<tr><td colspan="3">Actor(es): Administrador Local</td></tr>
<tr><td colspan="3">RESTRICCIONES</td></tr>
<tr><td colspan="2">Pre-Condiciones: Disponer de un Sistema de Almacenamiento local, ya sea Orientado a Objetos, Relacional o basado en un Sistema de Archivos.</td><td>Post-Condiciones: Base de Objetos Creada.</td></tr>
<tr><td colspan="3">FLUJO DE EVENTOS</td></tr>
<tr><td colspan="2">Camino Básico</td><td>Caminos Alternativos</td></tr>
<tr><td colspan="2">1. El Administrador local selecciona e introduce al sistema un nombre para la Base de Objetos.
2. El Administrador local selecciona el tipo de sistema de almacenamiento sobre el cual funcionará el Servicio.
3. El Administrador local indica al sistema la ejecución de la orden de Creación de la Base de Objetos.
4. Fin del caso de uso.</td><td>1. Si existe algún error en el paso 3, se notifica al Administrador local, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 4, para terminar el caso de uso.</td></tr>
</table>

Caso de Uso: Registrar la Base de Objetos en el Grid.	**Referencia:** UC- SMOP_02
Actor(es): Administrador Local, Planificador del Grid.	
RESTRICCIONES	
Pre-Condiciones: UC-SMOP_01, Conocer la ubicación (Dirección IP) del Planificador del Grid y Por lo menos una instancia del Planificador debe estar en ejecución.	**Post-Condiciones:** Base de Objetos Persistentes (Recursos) registrada en el Grid.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Administrador local inicia este caso de uso indicando al Sistema la dirección IP donde se encuentra ubicada y en ejecución una instancia del Planificador del Grid, en la cual se intentan registrar los Recursos. 2. El Administrador local envía una solicitud de registro de la Base de Objetos al Planificador del Grid, ubicado en la Dirección IP indicada en el paso 1. Dicha solicitud incluye el tipo de sistema de almacenamiento utilizado por el Servidor SMOP que administra y la propia dirección IP de este Servidor local, que es quien gestiona en realidad la Base de Objetos que será registrada. 3. El Planificador del Grid recibe la solicitud del Administrador del Servidor SMOP para el registro de su Base de Objetos en el Grid. 4. El Planificador del Grid verifica que tenga la capacidad disponible para registrar un nuevo Servidor SMOP. 5. El Planificador del Grid hace efectivo el Registro del Servidor SMOP, asignando un código de identificación y a su vez, emite un mensaje al Servidor SMOP indicando el resultado de su solicitud. 6. Fin del caso de uso.	1. Si en la verificación del paso 4 el Planificador del Grid determina que no tiene capacidad de atención para más Servidores, entonces se envía un mensaje al Servidor SMOP indicando el resultado de la solicitud. Ir al paso 6.

Caso de Uso: Conectar a Base de Objetos	**Referencia:** UC- SMOP_03
Actor(es): Administrador Local, Planificador	
RESTRICCIONES	
Pre-Condiciones: UC- SMOP_01	**Post-Condiciones:** Conexión Abierta a la Base de Objetos según el estándar "JDO".
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. Este caso de uso se activa cuando el sistema detecta una solicitud de actualización de la Base de Objetos, ya sea, localmente, por parte del Administrador Local o en forma remota por parte del Planificador del Grid. 2. El sistema lee las propiedades presentes en los archivos de configuración de la Base de Objetos, que le brindan información sobre el tipo de Almacén o Repositorio de Objetos, al que se intenta acceder y algunos otros datos como por ejemplo: Nombre de la Base de Objetos, Ubicación local en el Servidor, Identificación del Usuario, entre otros. 3. Abrir la conexión con la Base de Objetos de acuerdo a las propiedades obtenidas en el paso 2. El sistema retornará un objeto, el cual proporcionará una vía de acceso directo al almacén de objetos, para realizar las operaciones de recuperación y actualización sobre la Base de Objetos, de acuerdo a los estándares de la tecnología empleada para ello. En este caso se utilizan los estándares "JDO" para el acceso al repositorio de objetos persistentes solicitado. 4. Fin del caso de uso.	1. Si existe algún error en el paso 2, el sistema devuelve un mensaje de error al actor que inicio la transacción, para luego terminar el caso de uso en el paso 4.

Caso de Uso: Actualizar Base de Objetos Localmente	**Referencia:** UC- SMOP_04
Actor(es): Administrador Local	
RESTRICCIONES	
Pre-Condiciones: UC- SMOP_01	**Post-Condiciones:** Base de Objetos Actualizada Localmente.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Administrador Local inicia este caso de uso indicando al sistema por medio de una Interfaz Gráfica de Usuario, su deseo de Actualizar la Base de Objetos, la cual representa el repositorio de Recursos del Grid, y lo hace indicando el tipo de actualización que va a realizar. Los tipos de actualizaciones se describen en detalle por medio de los siguientes casos de uso: Insertar Objeto Localmente (UC-SMOP_05), Consultar Objetos Localmente (UC-SMOP_06), Eliminar Objetos Localmente (UC-SMOP_07) y Modificar Objetos Localmente (UC-SMOP_08). *2.* El Sistema abre una conexión a la Base de Objetos, <<include>> (UC-SMOP_02). 3. El Sistema ejecuta la operación indicada por el Administrador Local en el paso 1. 4. El Administrador Local recibe los resultados de la operación según sea el caso. 5. Fin del caso de uso.	1. Si existe algún error en el paso 3, se notifica al Administrador local, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 5, para terminar el caso de uso.

Caso de Uso: Actualizar Base de Objetos Remotamente	**Referencia:** UC- SMOP_05
Actor(es): Planificador del Grid	
RESTRICCIONES	
Pre-Condiciones: UC- SMOP_01, UC- SMOP_02	**Post-Condiciones:** Base de Objetos Actualizada

	Remotamente.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Planificador del Grid inicia este caso de uso seleccionando la Base de Objetos remota que se vaya a actualizar, de acuerdo a algún mecanismo establecido por el Grid para tal fin. *2.* El Planificador del Grid selecciona el tipo de actualización a realizar sobre la base de objetos remota. Los tipos de actualizaciones se describen en detalle por medio de los siguientes casos de uso: Insertar Objeto Remotamente (UC-SMOP_10), Consultar Objetos Remotamente (UC-SMOP_11), Eliminar Objetos Remotamente (UC-SMOP_12) y Modificar Objetos remotamente (UC-SMOP_13). *3.* El Sistema abre una conexión a la Base de Objetos, <<include>> (UC-SMOP_02). 4. El Sistema ejecuta la operación indicada por el Planificador del Grid en el paso 1. 5. El Planificador del Grid recibe los resultados de la operación según sea el caso. 6. Fin del caso de uso.	1. Si existe algún error en el paso 4, se notifica al Planificador del Grid, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 5, para terminar el caso de uso.

Caso de Uso: Insertar Objeto Localmente	**Referencia:** UC- SMOP_06
Actor(es): Administrador Local	
RESTRICCIONES	
Pre-Condiciones: UC- SMOP_01	**Post-Condiciones:** Objeto Persistente (Recurso del Grid) Almacenado en el Repositorio Local de Objetos.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Administrador Local escoge el tipo de	1. Si existe algún error en el paso 4, se notifica al

recursos que desea insertar o almacenar en la Base de Objetos. *2.* El Administrador local instancia un Objeto según el tipo de Recurso seleccionado en el paso 1, puede ser: *Institución, Dominio, Almacén, Computo, Servicio, Usuario, Rol.* 3. El Administrador Local también escoge la ubicación del Recurso seleccionado dentro del repositorio de Objetos. 4. El Administrador local ordena al sistema la ejecución de la operación de Inserción del Objeto Persistente en la Base de Objetos. 5. Fin del caso de uso.	Administrador local, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 5, para terminar el caso de uso.

Caso de Uso: Consultar Objeto Localmente	**Referencia:** UC- SMOP_07
Actor(es): Administrador Local	
RESTRICCIONES	
Pre-Condiciones: UC-SMOP_01. Además el Administrador local deberá estar familiarizado con los atributos de los Recursos a Recuperar.	**Post-Condiciones:** Colección de Objetos Persistentes (Recursos del Grid) Recuperados de la Base de Objetos local.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Administrador Local escoge el tipo de recursos que desea consultar o recuperar de la Base de Objetos. 2. El Administrador Local indica al sistema el criterio de consulta a utilizar para la recuperación de los objetos requeridos. La construcción de estos criterios se basan en los atributos de los Recursos a Recuperar. 3. El Administrador local ordena al sistema la ejecución de la operación de Consulta de la Base de Objetos, según el criterio de consulta especificado en el paso 3.	1. Si existe algún error en el paso 3, se notifica al Administrador local, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 6, para terminar el caso de uso. 2. Si en el paso 4 se determina que no existe ningún objeto que satisfaga el criterio de consulta, entonces el Administrador Local recibirá un mensaje indicando está situación, en lugar de la Colección de Objetos recuperada.

4. El Administrador local recibe una Colección de Objetos recuperados de la Base de Datos. 5. El Administrador Local muestra alguno de los atributos que identifican a los Objetos recuperados para comprobar el éxito de la operación. 6. Fin del caso de uso.	

Caso de Uso: Eliminar Objeto Localmente	**Referencia:** UC- SMOP_08
Actor(es): Administrador Local	
RESTRICCIONES	
Pre-Condiciones: UC-SMOP_01. Además el Administrador local deberá estar familiarizado con los atributos de los Recursos a Recuperar.	**Post-Condiciones:** Colección de Objetos Persistentes (Recursos del Grid) Eliminados de la Base de Objetos local.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Administrador Local escoge el tipo de recursos que desea eliminar de la Base de Objetos. 2. El Administrador Local le indica al sistema, al igual que en el Caso de Uso anterior <<include>>(UC-SMOP_07), el criterio de consulta que le permita al Servicio de Eliminación de Objetos, seleccionar solamente a los Objetos que cumplan con el criterio señalado para su eliminación de la Base de Objetos. 3. El Administrador local ordena al sistema la ejecución de la operación de Eliminación de los Objetos Persistentes en la Base de Objetos. 4. El Administrador local recibe un mensaje del resultado de la eliminación de objetos. 5. Fin del caso de uso.	1. Si en el Paso 2 no se encuentran Objetos que satisfagan el criterio de selección, se devuelve un mensaje al Administrador Local y se termina el Caso de Uso en el paso 5. 2. Si existe algún error en el paso 3, se notifica al Administrador local, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 5, para terminar el caso de uso.

Caso de Uso: Modificar Objeto Localmente	**Referencia:** UC- SMOP_09

Actor(es): Administrador Local	
RESTRICCIONES	
Pre-Condiciones: UC-SMOP_01. Además el Administrador local deberá estar familiarizado con los atributos de los Recursos a Modificar.	**Post-Condiciones:** Colección de Objetos Persistentes (Recursos del Grid) Modificados de la Base de Objetos local.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Administrador Local escoge el tipo de recursos que desea Modificar de la Base de Objetos. 2. El Administrador Local le indica al sistema, un criterio de consulta <<include>> (UC-SMOP_07) que le permita al Servicio de Modificación de Objetos, seleccionar de la Base de Objetos, solamente aquellos que cumplan con el criterio señalado para la modificación de sus atributos. 3. El Administrador local indica al sistema cuáles son los atributos del Recurso que va a modificar, junto con el correspondiente nuevo valor que se le va a colocar a cada uno. 4. El Administrador local ordena al sistema la ejecución de la operación de Modificación de los atributos de los Objetos Persistentes en la Base de Objetos. 5. Fin del caso de uso.	1. Si existe algún error en el paso 4, se notifica al Administrador local, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 5, para terminar el caso de uso.

Caso de Uso: Insertar Objeto Remotamente	**Referencia:** UC- SMOP_10
Actor(es): Planificador del Grid	
RESTRICCIONES	
Pre-Condiciones: UC- SMOP_01 y UC-SMOP_02.	**Post-Condiciones:** Objeto Persistente (Recurso del Grid) Almacenado en el Repositorio Remoto de Objetos.

FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Planificador del Grid obtiene una instancia del Recurso que requiere insertar en la base de objetos, ya sea produciéndola él mismo directamente, o recibiéndola de los otros componentes del Grid. *2.* El Planificador del Grid identifica el tipo de recursos que representa esa instancia: *Institución, Dominio, Almacén, Computo, Servicio, Usuario, Rol.* 3. El Planificador del Grid determina el orden en que debe ser almacenado el recurso, en relación al resto de los objetos que se encuentran almacenados en el Repositorio Remoto. 4. El Planificador del Grid selecciona uno de los Servidores SMOP disponibles, entre los que hayan sido registrados previamente, de acuerdo a las políticas que emplee el Grid para tal fin. 5. El Planificador del Grid ordena al Servidor SMOP seleccionado, la ejecución del servicio de Inserción Remota de Recursos (Objetos Persistentes) en la Base de Objetos que administra dicho Servidor. 6. Fin del caso de uso.	1. Si existe algún error en el paso 5 se notifica al El Planificador del Grid el cual decidirá si vuelve al paso 4 para reintentar la operación en otro Servidor SMOP o si va directamente al paso 6, para terminar el caso de uso.

Caso de Uso: Consultar Objeto Remotamente	**Referencia:** UC- SMOP_11
Actor(es): Planificador del Grid.	
RESTRICCIONES	
Pre-Condiciones: UC-SMOP_01 y UC-SMOP_02. Además el Planificador del Grid deberá conocer de antemano los atributos de los Recursos a Recuperar.	**Post-Condiciones:** Colección de Objetos Persistentes (Recursos del Grid) Recuperados de la Base de Objetos remota.

FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Planificador del Grid escoge el tipo de recursos que desea consultar o recuperar de la Base de Objetos remota. 2. El Planificador del Grid selecciona uno de los Servidores SMOP disponibles, entre los que hayan sido registrados previamente, de acuerdo a las políticas que emplee el Grid para tal fin. 3. El Planificador del Grid indica al sistema el criterio de consulta a utilizar para la recuperación de los objetos requeridos. La construcción de estos criterios se basan en los atributos de los Recursos a Recuperar. 4. El Planificador del Grid local ordena al sistema la ejecución de la operación de Consulta de la Base de Objetos, según el criterio de consulta especificado en el paso 3. 5. El Planificador del Grid recibe una Colección de Objetos recuperados de la Base de Objetos Remota. 6. Fin del caso de uso.	1. Si existe algún error en el paso 4, se notifica al Planificador del Grid, el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 6, para terminar el caso de uso.

Caso de Uso: Eliminar Objeto Remotamente	**Referencia:** UC- SMOP_12
Actor(es): Planificador del Grid	
RESTRICCIONES	
Pre-Condiciones: UC-SMOP_01 y UC-SMOP_02. Además el Planificador del Grid deberá conocer de antemano los atributos de los Recursos a Eliminar.	**Post-Condiciones:** Colección de Objetos Persistentes (Recursos del Grid) Eliminados de la Base de Objetos remota.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Planificador del Grid escoge el tipo de recursos que desea eliminar de la Base de Objetos remota. 2. El Planificador del Grid selecciona uno de los	1. Si existe algún error en el paso 4 se notifica al El Planificador del Grid el cual decidirá si vuelve al paso 2 para reintentar la operación en otro Servidor SMOP o si va directamente al

Servidores SMOP disponibles, entre los que hayan sido registrados previamente, de acuerdo a las políticas que emplee el Grid para tal fin. 3. El Planificador del Grid indica al sistema el criterio de consulta <<include>> (UC-SMOP_11) a utilizar para la recuperación de los objetos requeridos. La construcción de estos criterios se basan en los atributos de los Recursos a Eliminar. 4. El Planificador del Grid ordena al sistema la ejecución de la operación de Eliminación de la Base de Objetos, según el criterio de consulta especificado en el paso 3. 5. El Planificador del Grid recibe un mensaje con el resultado de la eliminación de Objetos de la Base de Objetos Remota. 6. Fin del caso de uso.	paso 6, para terminar el caso de uso.

Caso de Uso: Modificar Objeto Remotamente	**Referencia:** UC- SMOP_13
Actor(es): Planificador del Grid.	
RESTRICCIONES	
Pre-Condiciones: UC-SMOP_01 y UC-SMOP_02. Además el Planificador del Grid deberá conocer de antemano los atributos de los Recursos a Modificar.	**Post-Condiciones:** Colección de Objetos Persistentes (Recursos del Grid) Modificados de la Base de Objetos Remota.
FLUJO DE EVENTOS	
Camino Básico	**Caminos Alternativos**
1. El Planificador del Grid escoge el tipo de recursos que desea eliminar de la Base de Objetos remota. 2. El Planificador del Grid selecciona uno de los Servidores SMOP disponibles, entre los que hayan sido registrados previamente, de acuerdo a las políticas que emplee el Grid para tal fin. 3. El Planificador del Grid indica al sistema el	1. Si existe algún error en el paso 3, se notifica al Administrador local y el cual decidirá si vuelve al paso 1 para reintentar la operación o si va directamente al paso 4, para terminar el caso de uso.

criterio de consulta <<include>> (UC-SMOP_11) a utilizar para la recuperación de los objetos requeridos. La construcción de estos criterios se basan en los atributos de los Recursos a Eliminar. 4. El Planificador del Grid indica al sistema cuáles son los atributos del Recurso que va a modificar, junto con el correspondiente nuevo valor que se le va a colocar a cada uno. 5. El Planificador del Grid ordena al sistema la ejecución de la operación de Eliminación de la Base de Objetos, según el criterio de consulta especificado en el paso 3. 6. El Planificador del Grid recibe un mensaje con el resultado de la eliminación de Objetos de la Base de Objetos Remota. 7. Fin del caso de uso.	

ANEXO C

DIAGRAMAS DE COLABORACIÓN

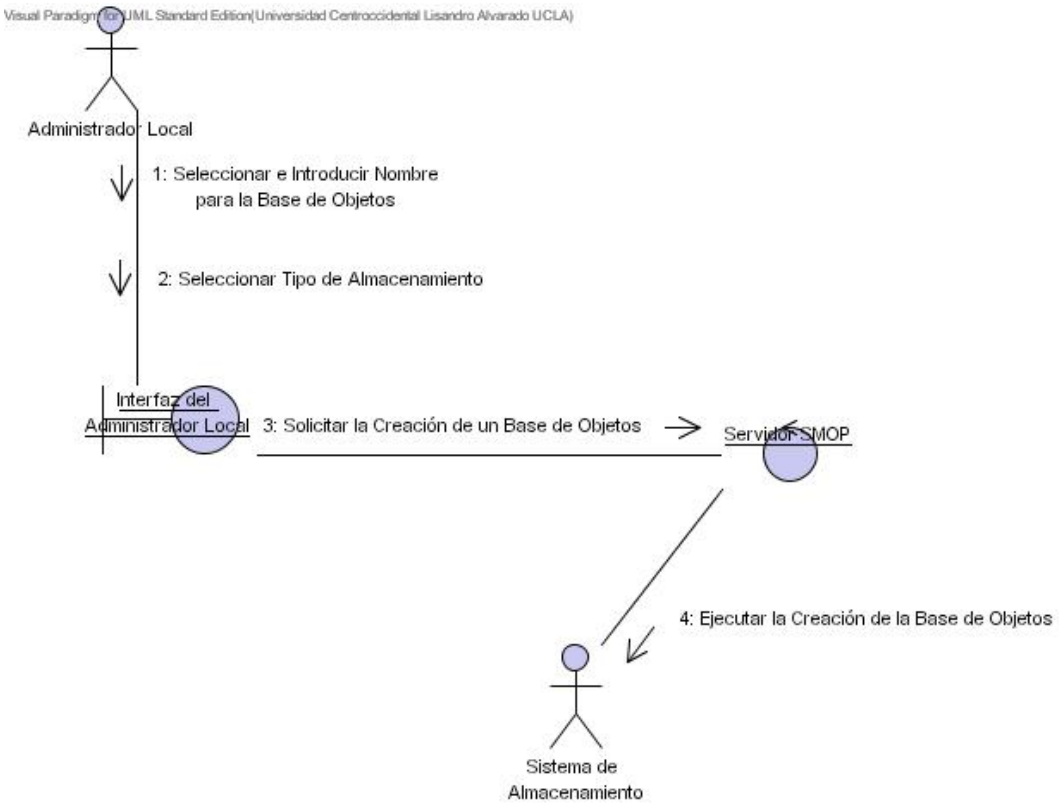


Diagrama de Colaboración del Caso de Uso: Crear Base de Objetos (UC-SMOP_01)

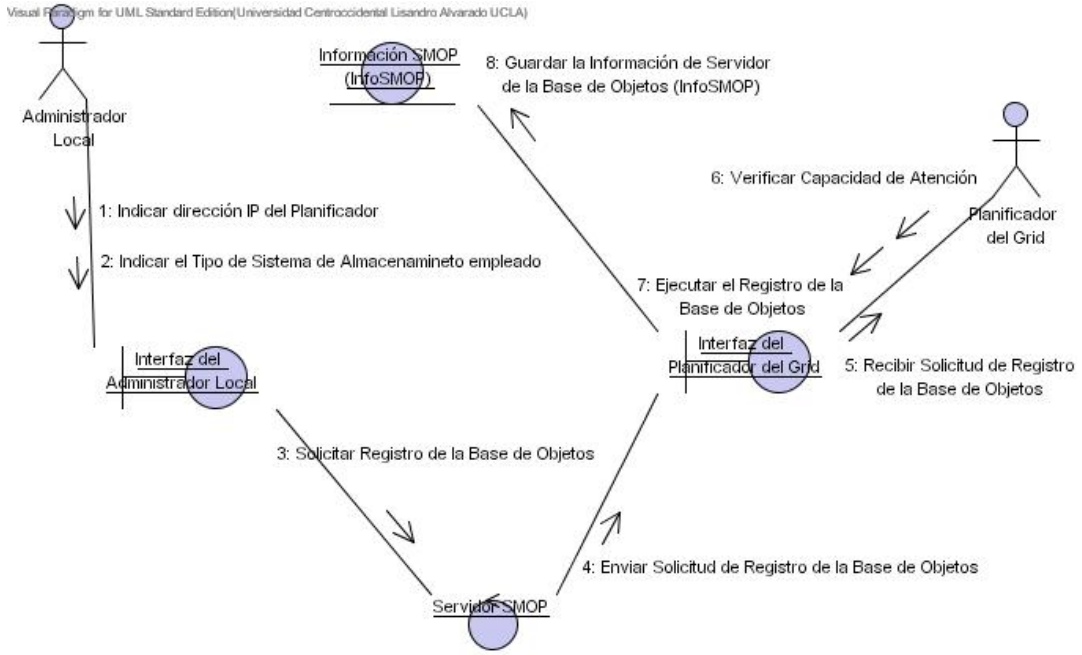


Diagrama de Colaboración del Caso de Uso: Registrar la Base de Objetos en el Grid
(UC-SMOP_02)

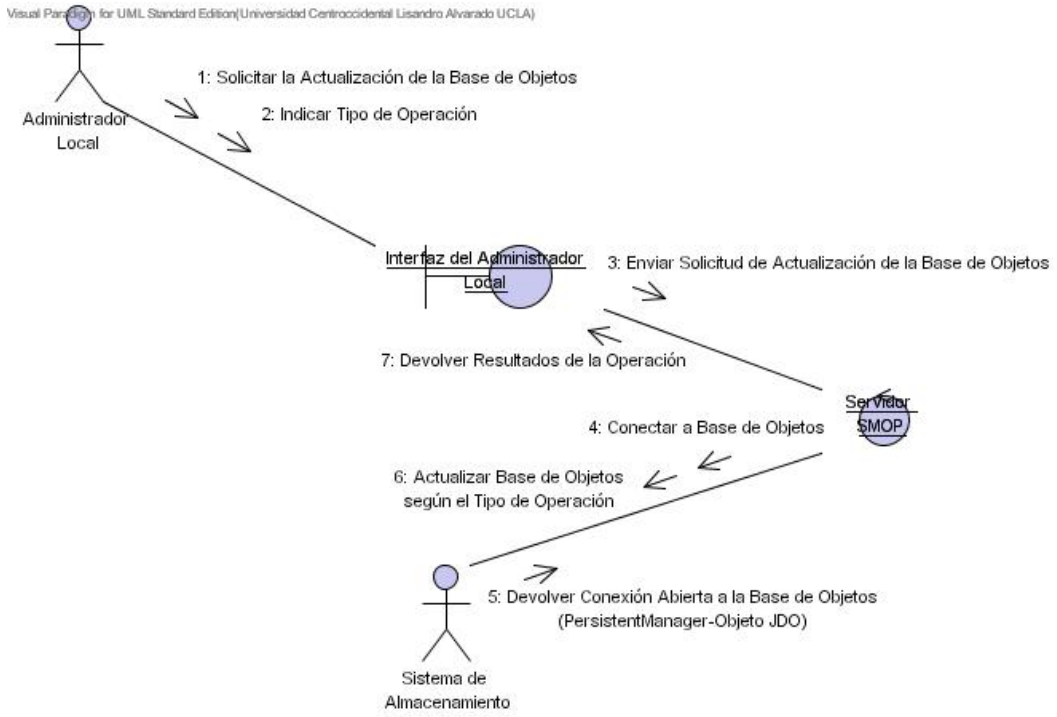


Diagrama de Colaboración del Caso de Uso: Actualizar Base de Objetos Localmente
(UC-SMOP_04)

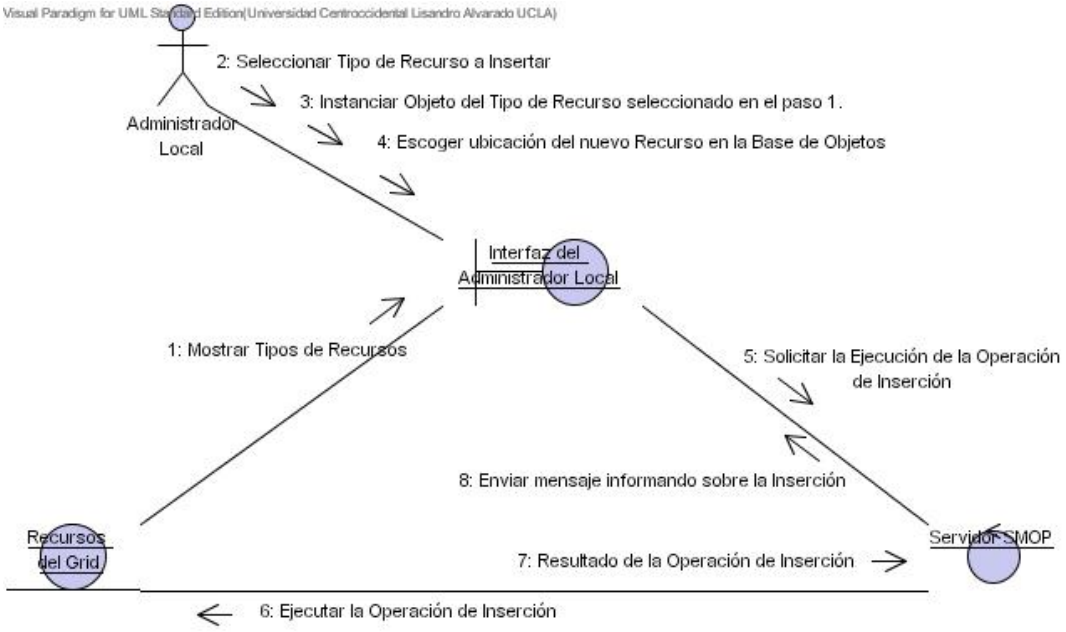


Diagrama de Colaboración del Caso de Uso: Insertar Objeto Localmente
(UC-SMOP_06)

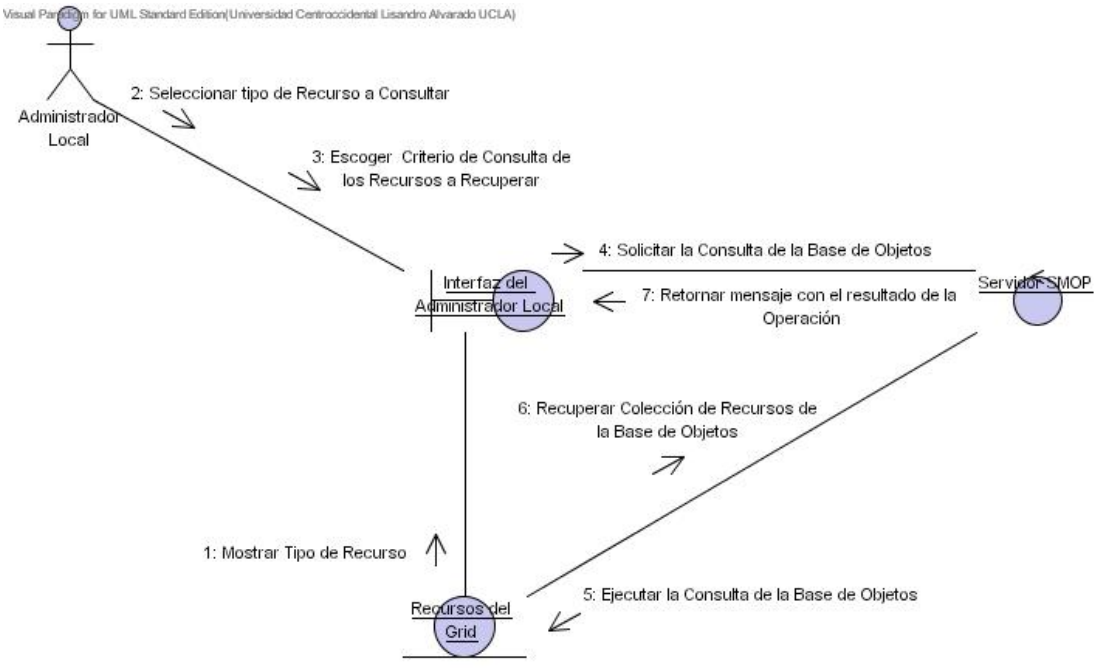


Diagrama de Colaboración del Caso de Uso: Consultar Objeto Localmente
(UC-SMOP_07)

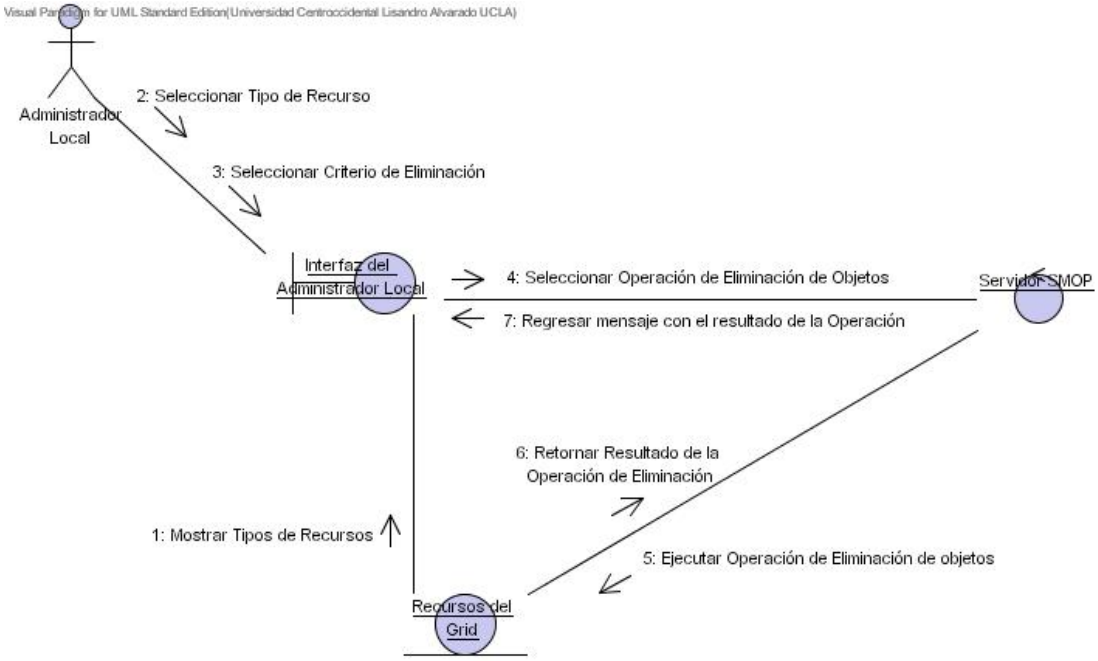


Diagrama de Colaboración del Caso de Uso: Eliminar Objeto Localmente
(UC-SMOP_08)

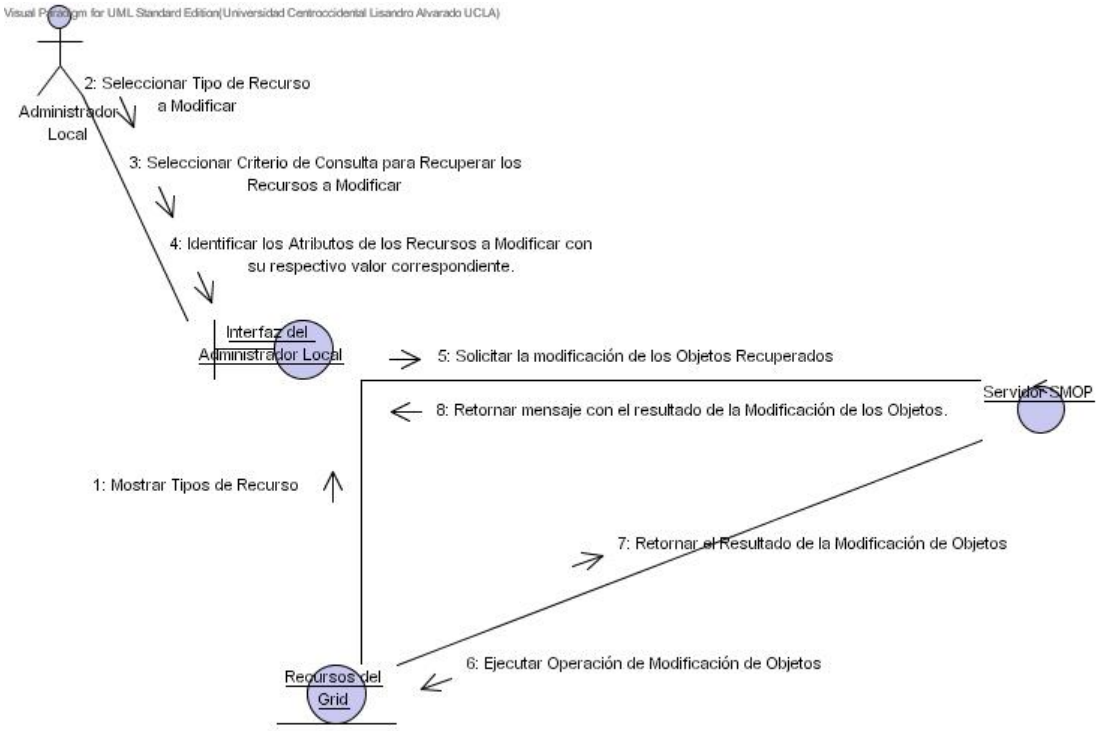


Diagrama de Colaboración del Caso de Uso: Modificar Objeto Localmente
(UC-SMOP_09)

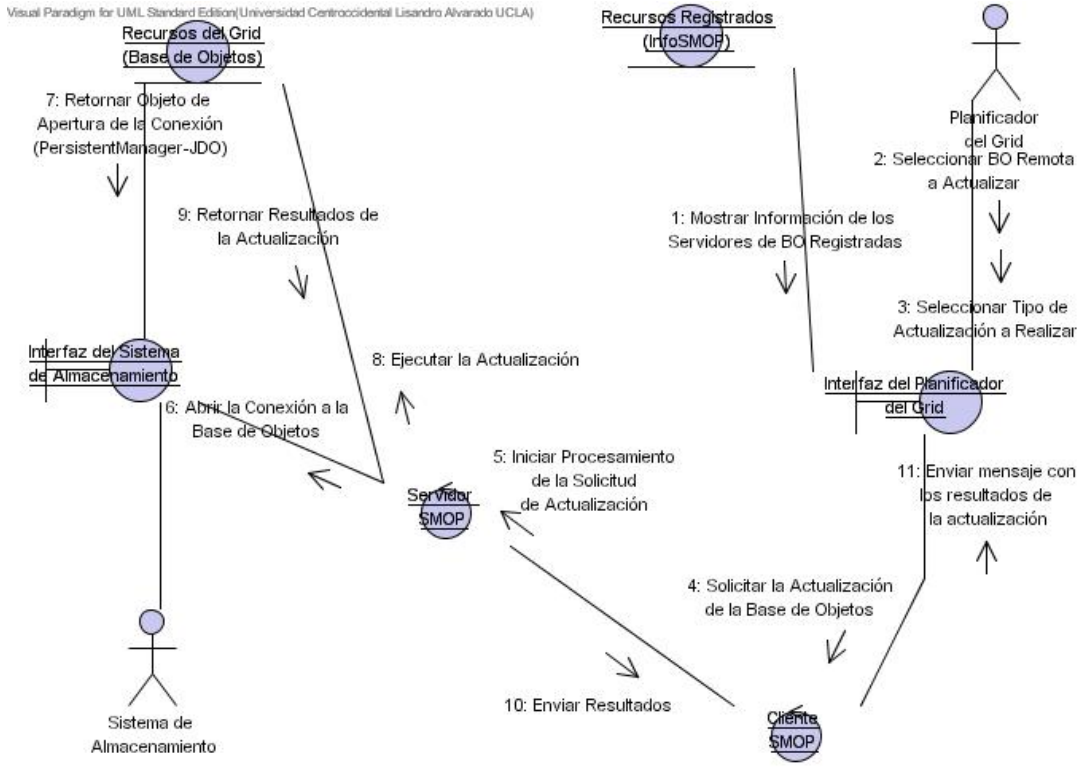


Diagrama de Colaboración del Caso de Uso: Actualizar Base de Objetos Remotamente (UC-SMOP_05)

ANEXO D

DIAGRAMAS DE SECUENCIA

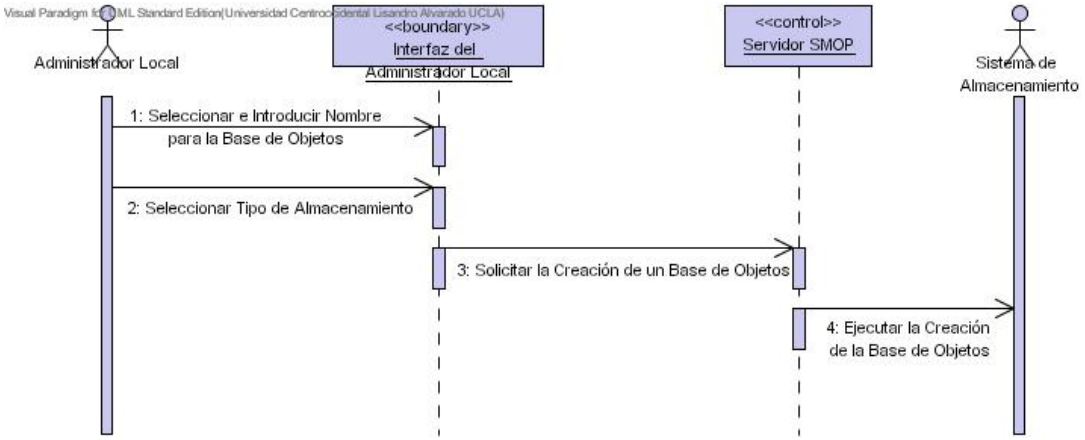


Diagrama de Secuencia del Caso de Uso: Crear Base de Objetos (UC-SMOP_01)

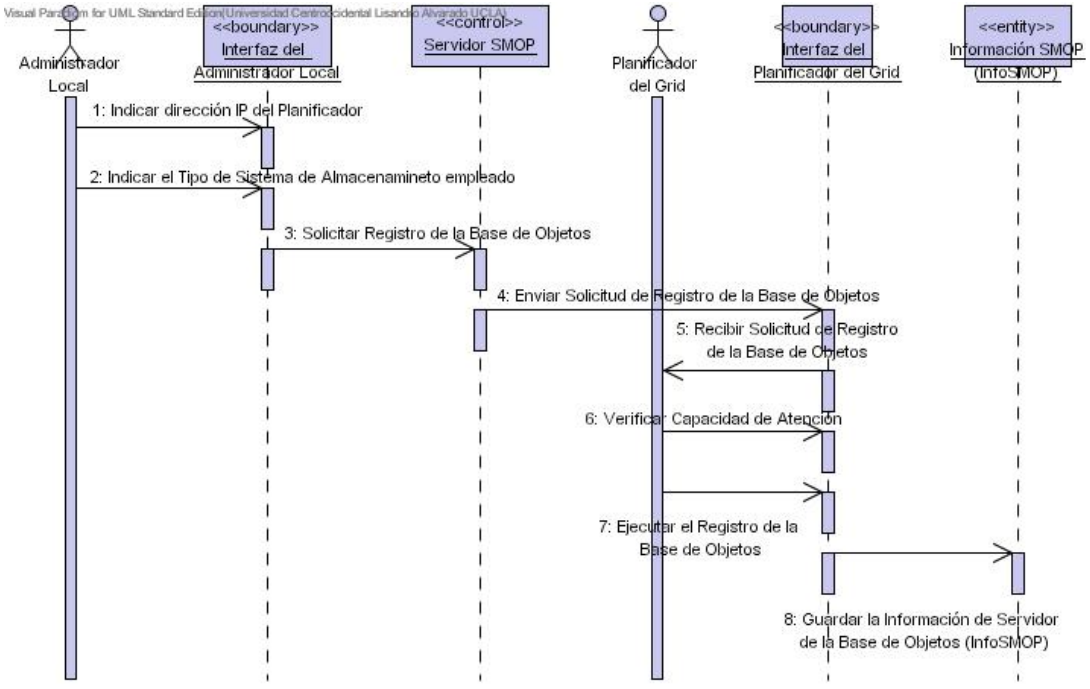


Diagrama de Secuencia del Caso de Uso: Registrar la Base de Objetos en el Grid (UC-SMOP_02)

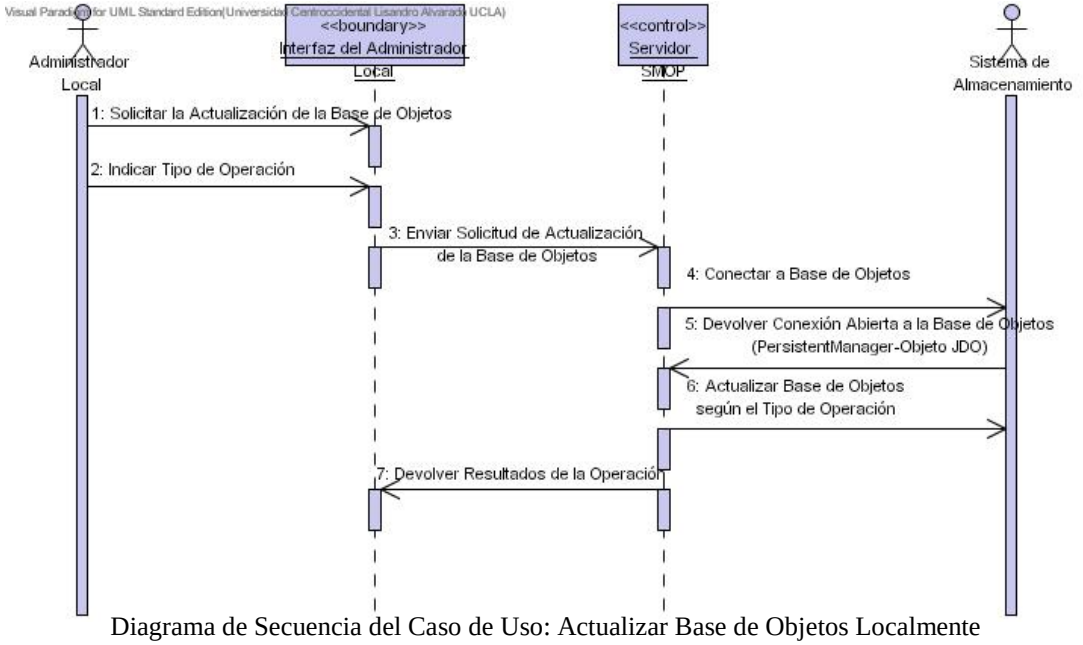


Diagrama de Secuencia del Caso de Uso: Actualizar Base de Objetos Localmente (UC-SMOP_04)

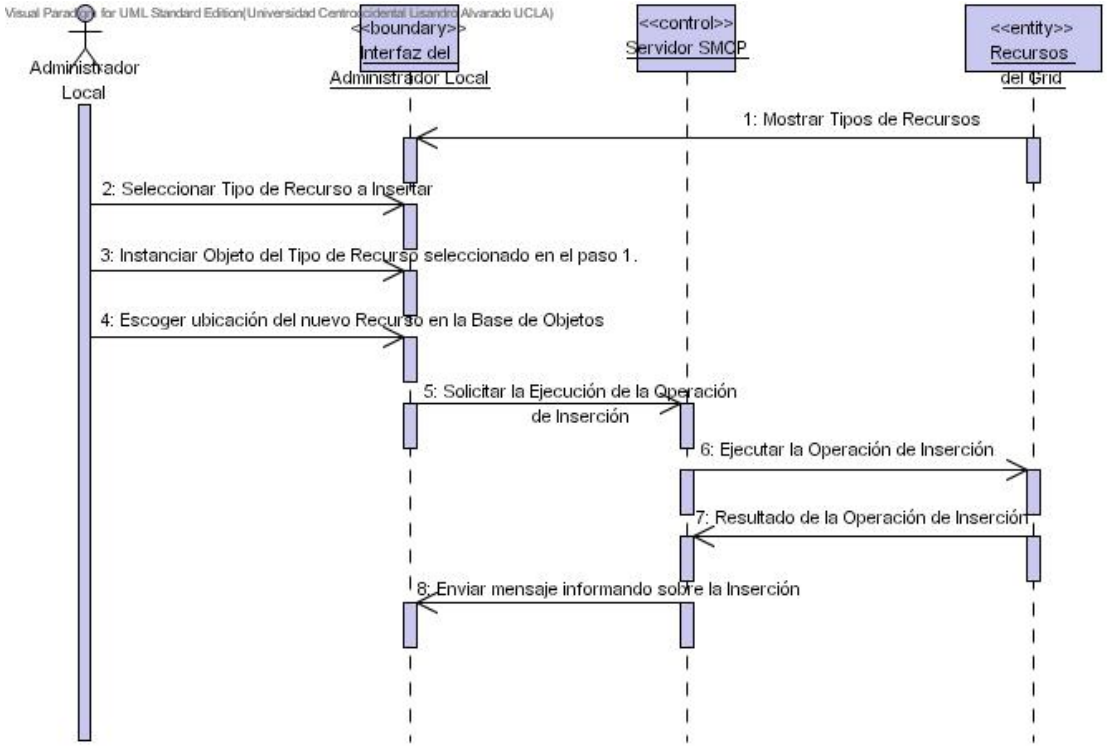


Diagrama de Secuencia del Caso de Uso: Insertar Objeto Localmente (UC-SMOP_06)

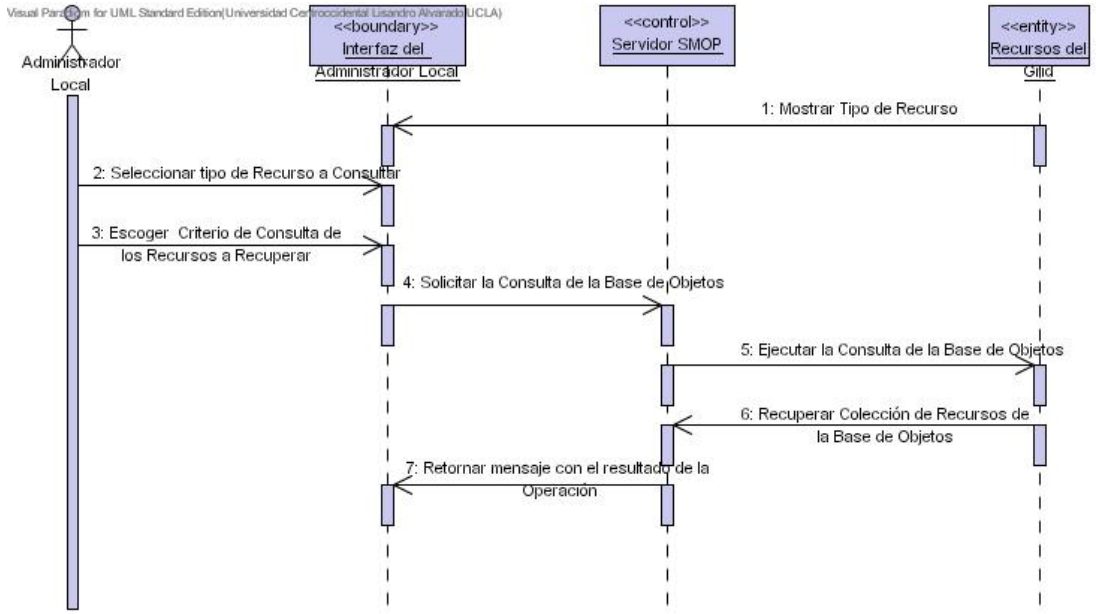


Diagrama de Secuencia del Caso de Uso: Consultar Objeto Localmente
(UC-SMOP_07)

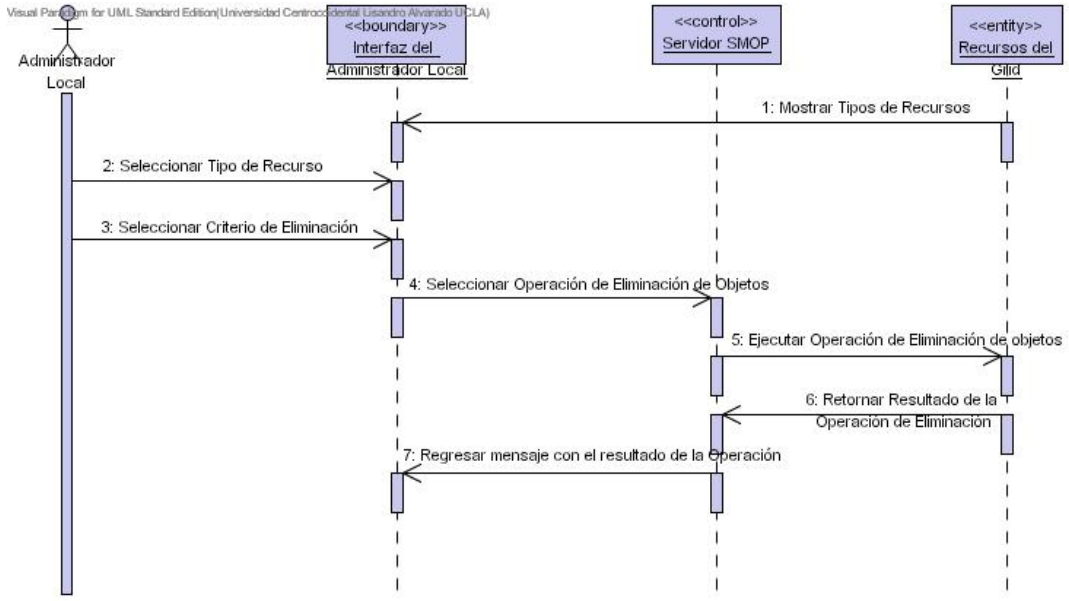


Diagrama de Secuencia del Caso de Uso: Eliminar Objeto Localmente
(UC-SMOP_08)

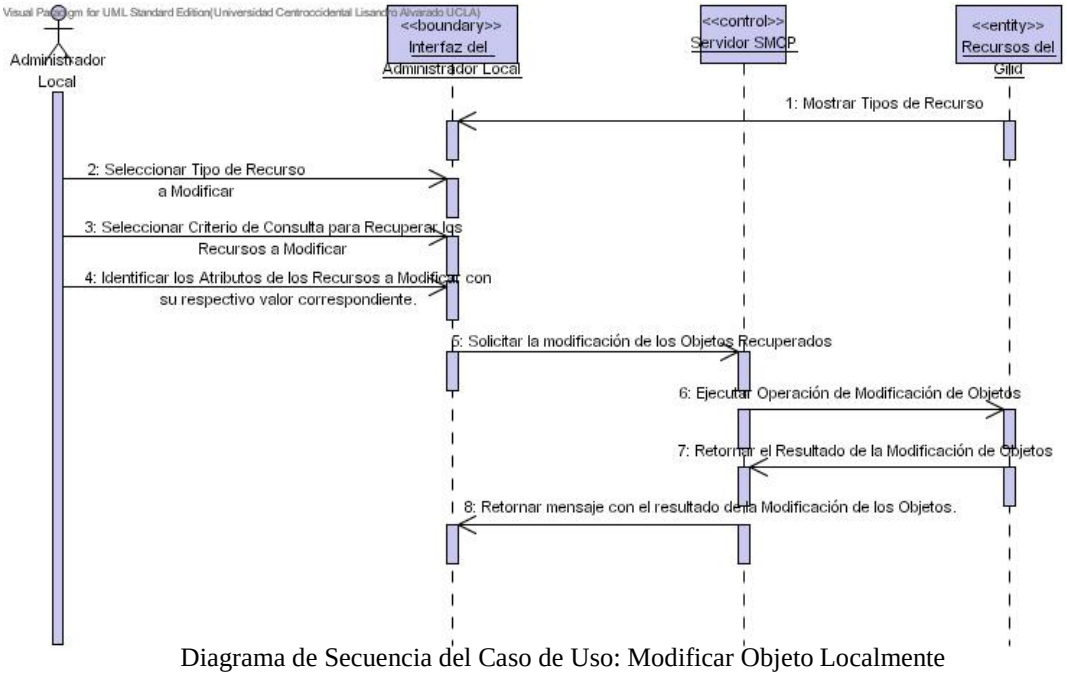


Diagrama de Secuencia del Caso de Uso: Modificar Objeto Localmente
(UC-SMOP_09)

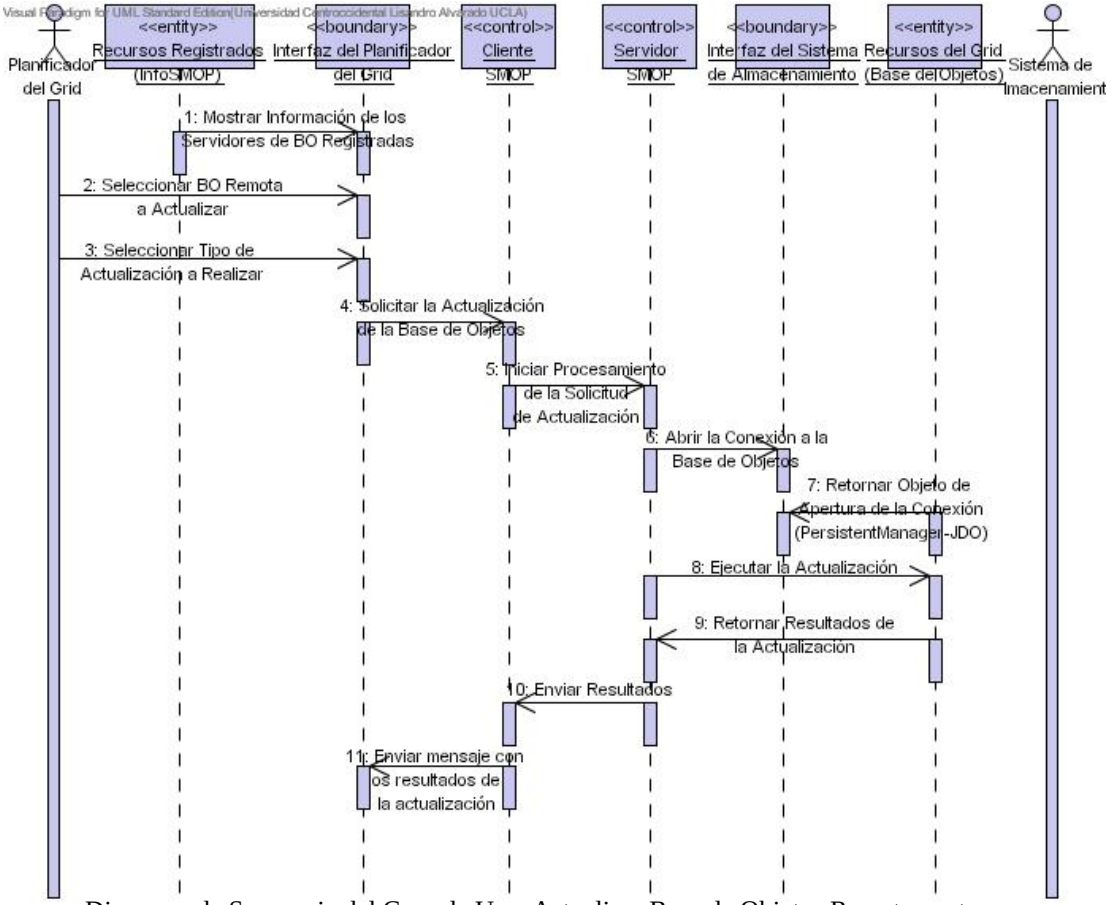


Diagrama de Secuencia del Caso de Uso: Actualizar Base de Objetos Remotamente
(UC-SMOP_05)

ANEXO E

GLOSARIO

Base de Objetos: Las bases de objetos son el soporte natural para la persistencia en tecnologías de objetos. El término proviene de la combinación del nombre tradicional otorgado al mecanismo que provee la persistencia de *datos* relacionales, denominado Bases de Datos, con la unidad básica manejada en la tecnología de *objetos*, el objeto mismo.

Controlador de Usuario (User Control): Es un componente de software que forma parte del núcleo del Grid SUMA. Tiene la finalidad de manejar el registro y autenticación de los usuarios del Grid.

Impedancia: Es un término proveniente de la Ingeniería Eléctrica, y es una medida del grado de resistencia de un componente al flujo de electricidad cuando un voltaje es aplicado.

Desadaptación de Impedancias ("Impedance mismatch"): Es un problema en Ingeniería Eléctrica que ocurre cuando dos líneas o circuitos de transmisión con diferentes impedancias son conectados. Esto puede ocasionar ruido y algún tipo de pérdida. En la Ingeniería del Software, la terminología se refiere a un problema ocasionado al tratar de conectar dos sistemas que presentan bases conceptuales muy diferentes. Esto ocurre comúnmente, a nivel de programación, cuando se trata de usar una base de datos relacional (SQL) desde un programa orientado por objetos.

Dominio Administrativo: Es una colección de recursos: redes, computadoras, y bases de datos, que comparten una administración común; como por ejemplo, en una Intranet empresarial. Los dispositivos que son operados en un singular Dominio Administrativo comparten características de seguridad comunes, que son administradas a través de la red y las entidades que se encuentran asociadas a él.

Dominio del Modelo de Objetos: Es el contexto o área del conocimiento específica en la cual se operará.

Metadatos XML: Es un término que se refiere a datos sobre los propios datos. Los metadatos han cobrado importancia en Internet por la necesidad de utilizarlos para la clasificación de grandes cantidades de datos. En el contexto JDO se utiliza un archivo que contiene toda la información relacionada con los objetos persistentes que se van a manejar, tales como: Definición de Objetos (atributos y métodos) y las Relaciones entre ellos (Herencia, Contenedor, entre otras). Esta información se estructura en formato XML y a este archivo también se denomina *Descriptor XML*.

Marshaling / UnMarshaling : Son algunos de los procesos por lo que debe pasar toda información para que ésta sea utilizable en ambientes heterogéneos (diferentes sistemas operativos).

Modelo de Objeto: Es una abstracción que identifica entidades del "mundo real" y sus relaciones en un contexto determinado. El modelo de objetos de una aplicación, es la colección de todas las definiciones de clases de la aplicación.

Orientación a Objetos: Paradigma de desarrollo del software.

Objeto: Es una instancia de una *Clase*, en donde su *estado* y *comportamiento* son determinados por el valor de sus *atributos* y *métodos* respectivamente.

Planificador (Scheduler): Es un componente de software que forma parte del núcleo del Grid SUMA. Es el encargado de proporcionar la asignación de los recursos necesarios (Representante y Agente de Ejecución, entre otros), para permitir la ejecución distribuida de aplicaciones java sobre ésta plataforma.

Persistencia de Objeto (Java): La persistencia de un objeto en Java, consiste en el almacenamiento del objeto para que perdure, más allá del tiempo de vida de la Máquina Virtual Java (JVM, "Java Virtual Machine"), en la cual el objeto fue instanciado. La persistencia requiere el almacenamiento del estado de un objeto para su futura recuperación.

OO: Se le puede señalar de diferentes maneras dependiendo del contexto en donde se encuentre. Significa Orientado a Objetos u Orientación a Objetos. No obstante, algunos profesionales de la Tecnología de la Información, lo señalan como Orientado por Objetos u Orientación por Objetos.

Representante (Proxy): Es un componente de software que forma parte del núcleo del Grid SUMA. Tiene la finalidad de conectar a un Cliente de SUMA con su Agente de Ejecución, de tal forma que estos componentes se comuniquen directamente.

RMI (Remote Meted Invocation): Es un Mecanismo ofrecido en Java que permite a un método de un objeto, poder ser invocado remotamente.

SQL("Structured Query Languaje"): Lenguaje estructurado de consultas, usado en las bases de datos relacionales.

Stubs y Skeletons. Son elementos necesarios para la ejecución e invocación de funciones remotas. Estos elementos permiten que al momento de ser invocada la función remota esta pueda ser simulada localmente. En el ambiente Cliente-servidor, el Stub funciona como un simulador local para todas las funciones que están siendo invocadas y pertenecen a la implementación del Servidor. El Skeleton funciona como simulador para recibir parámetros de la implementación del Cliente.

VO (“Virtual Organization”): La Organización Virtual es un concepto fundamental en los sistemas Grids y consiste en la agrupación de recursos de varias organizaciones distintas que colaboran para alcanzar un objetivo común. La agrupación en Organizaciones Virtuales facilita la gestión de recursos y la seguridad. La pertenencia a una Organización Virtual no es permanente sino que puede cambiar según las necesidades.

ANEXO F

Resumen Curricular del Autor

Julio César Véliz Sira, portador de la cédula de identidad número: V-7.422.543, nació en la ciudad de Barquisimeto el 03 de Julio de 1970. Realizó estudios de educación primaria y ciclo básico en el Colegio "La Presentación" de Barquisimeto. Obtuvo el título de bachiller en ciencias en el Liceo "Lisandro Alvarado" de esta misma ciudad, en el año 1986. Realizó sus estudios de pregrado en la Universidad Centroccidental "Lisandro Alvarado" de la cual recibió el título de Ingeniero en Informática en el año 1997, ocupando el noveno lugar de su promoción (XXIII) conformada por un total de 98 estudiantes. Actualmente cursa estudios de postrado en la maestría de Ciencias en Computación, mención Ingeniería de Software, del Decanato de Ciencias y Tecnología, cuya escolaridad y Tesis de Grado ha finalizado, y se encuentra en espera por la defensa de la misma, para la obtención de su título.

Incursionó en el campo laboral en el área de desarrollo de Software, trabajando en proyectos de software de varias empresas entre las que se destacan: CIDESA y MARNA. En el año 1998 ingresa como trabajador administrativo de la UCLA, con el cargo de Analista de Procesamiento de Datos I, en el cual se desempeña como soporte técnico de la Oficina de Registro Académico del Decanato de Ciencias y Tecnología de esta casa de Estudios, labor que continúa realizando en la actualidad. Inició su experiencia docente en el año 2002, cuando proporcionó apoyo de asesoría técnica en las asignaturas de S5 - Base de Datos (Tema: Structured Query Language - SQL) y en O1 - Cibernética y Sociedad (Tema: Tópicos para la construcción de Aplicaciones Web) pertenecientes al Departamento de Sistemas del Decanato de Ciencias y Tecnología, para el programa de Análisis de Sistemas que administra dicho Decanato. En la actualidad, se desempeña como Docente (Ad-honoren) de la Asignatura Cibernética y Sociedad (O1).