BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Mercado de Objetos de Aprendizaje"

Tesis profesional presentada por

Guillermina Sánchez Román

como requisito parcial

para obtener el título de maestro en ciencias

de la Computación

Puebla, Pue. Otoño de 2006

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Mercado de Objetos de Aprendizaje"

Tesis profesional presentada por
Guillermina Sánchez Román
como requisito parcial
para obtener el título de maestro en ciencias
de la Computación

Dr. Fabiola López y López
Asesor

Puebla, Pue. Noviembre de 2006

GRACIAS.

A Dios que me ha permitido la oportunidad de vivir.

A ti Mamá que con tu luz vas guiando mi camino día a día. Q.E.P.D.

A mis hermanos Lety, Eli, Jose y Rene que con su apoyo y cariño me ayudaron a cumplir mis objetivos.

A la Dra. Fabiola López y López por sus conocimientos, apoyo y dedicación en el tema. Al Dr. León Chavez por su paciencia y consejos para la culminación de esta trabajo.

A todos mis compañeros de maestria, que contribuyeron con su observaciones, consejos, y cariño.

Índice general

Resumen

Actualmente los cursos de aprendizaje en línea son producidos como sistemas monolíticos, que contienen grandes cantidades de información que generalmente no son reutilizables entre la comunidad académica, incrementando con esto los costos, tiempo y consecuentemente son descuidados sus aspectos de calidad.

El objetivo principal de este trabajo es facilitar el diseño de cursos en línea a través del manejo de Objetos de Aprendizaje (OA) que son unidades básicas de contenidos de información que permiten su fácil gestión, creación y distribución, lo que daría flexibilidad para crear los cursos.

Para que los objetos sean manipulados por diversos usuarios deben cumplir con un conjunto de requisitos diferentes dependiendo de cada uno de ellos, así que lo mejor es estandarizarlos lo que permitirá a su vez que estos OA sean interoperables. Además se necesita que los OA estén disponibles en un repositorio y que su acceso sea sencillo. Para proporcionar y acceder a los OA se manejará un esquema de Mercado de Objetos de Aprendizaje (MOA) que permitirá la interacción entre usuarios que necesiten los objetos y las entidades que ofrezcan el servicio, de tal manera que se tenga una gran cantidad de objetos de aprendizaje a disposición para crear los cursos de forma flexible, dinámica y vanguardista.

El objetivo de esta tesis es desarrollar un Modelo para un Sistema de Mercado de Objetos de Aprendizaje que permita a diseñadores de Objetos de Aprendizaje (OA)

ponerlos a disposición del público para que puedan ser localizados y accedidos de forma transparente a los profesores para crear sus cursos en línea. Este prototipo se implementará en Servicios Web(SW) ya que esta herramienta permite crear componentes de software que son interoperables lo que daría gran flexibilidad a la aplicación que se desarrolle. Además, la tecnología de SW se basa en la Arquitectura Orientada a Servicios(AOS) enfocada a mercados de acuerdo al esquema De Roure [12] la cual tiene tres componentes principales: comprador, proveedor y el dueño del mercado los cuales se puede modelar bajo el paradigma de agentes. Otra de las ventajas de utilizar SW es que se pueden crear aplicaciones a partir de otras que existen, ya que están basados en el principio de reusabilidad, lo que permitirá que nuevas tareas o modulos que puedan ser integrados a futuro.

Capítulo 1

Descripción General

1.1. Introducción

A principios del siglo XXI hemos vivido todo un avance continuo y dinámico en la educación, inicialmente la educación se impartió en forma tradicional donde interactuaban directamente el profesor y el alumno, después fue a distancia, en donde últimamente se incorporaron recursos tecnológicos que permiten tener cursos en línea donde

maestros y alumnos interactuan usando herramientas de Internet. En particular los cursos en línea se construyen como un todo, estos cursos contienen grandes cantidades de información que generalmente no son reutilizables entre la comunidad académica, incrementando con esto los costos de producción de contenidos y consecuentemente se descuidan los aspectos de calidad.

Hoy en día los cursos monolíticos son todavía utilizados, sin embargo, éstos ya son obsoletos, debido a que no permiten que su contenido sea reusable entre los académicos, además no se adaptan a diferentes plataformas, lo que provoca incompatibilidad ante nuevas tecnologías. Ahora los cursos en línea necesitan ser flexibles, dinámicos y dar al profesor la facilidad de adoptar un conjunto de recursos modulares para la creación de

su propio curso en línea.

En la Benemérita Universidad Autónoma de Puebla para el diseño de cursos en línea se cuenta con un Sistema de Administración de Aprendizaje (SAA, en ingles Learning Management System, LMS) el cual no es compatible con otros LMS. Esto genera un gran interés para trabajar en el tema de Educación a Distancia y ofrecer flexibilidad en la generación de cursos. Además manejar un SAA que les permita trabajar en colaboración con otras instituciones de forma gratuita y compatible.

Este problema ha sido atacado desde el punto de vista de los Objetos de Aprendizaje (OA) que son unidades básicas de información digital con contenido académico. Los OA permiten una fácil gestión, creación y distribución. Estas características que poseen los objetos permiten que los cursos en línea sean construidos de forma flexible y dinámica.

Un aspecto importante es como localizar a los OA distribuidos en una red. Con el objetivo de que los objetos se puedan compartir entre una comunidad mayor de diseñadores de cursos en línea.

De esta forma, se pretende ayudar a los profesores diseñadores de la BUAP, a crear sus propios cursos. Cada uno de los cursos puede tener componentes de diversos creadores, que les permita reutilizarlos para generar nuevos contenidos actuales, didácticos y dinámicos.

Como se pretende compartir objetos de aprendizaje, entonces habrán profesores quienes los proveen y profesores quienes requieran estos objetos. Por lo tanto, se propone diseñar el modelo para un mercado de objetos de aprendizaje en base a la Arquitectura Orientada a Servicios (AOS) donde cada uno de los componentes principales, proveedor y cliente, se pueden representar mediante agentes de software que administren los OA en un esquema de mercados [12, 10, 19]. Los sevicios web pueden ayudar a generar este tipo de aplicaciones. Los SW definen una manera sencilla de crear software donde se tienen diversos servicios de forma distribuida.

1.2. Objetivo General

- Desarrollar un modelo para que actúe como un mercado de objetos de aprendizaje, donde diseñadores de objetos de aprendizaje los pongan a disposición del público y los profesores puedan acceder a los objetos para crear sus cursos en línea.

1.3. Objetivos Específicos

1. Definir un modelo que se desarrolle de acuerdo a una arquitectura que defina los elementos que intervienen en la compra-venta de un servicio, como lo son las Arquitecturas Orientadas a Servicios.

2. Implementar un prototipo para el mercado de objetos de aprendizaje con tecnologías de vanguardia como los servicios web.

3. Modelar a proveedores, compradores y al administrador del mercado, usando el paradigma basado en agentes.

Este modelo es lo suficientemente detallado para su futura implementación, de esta forma, se plantea crear una aplicación que permita a los profesores crear sus materiales de forma sencilla y además puedan acceder a un conjunto de recursos modulares que contienen información educativa de algún tema, que se adapten a sus objetivos académicos. Por lo tanto, se evitará al profesor el volver a crear el material, ya que sólo buscará en un mercado donde se encuentran un conjunto de OA, teniendo la facilidad de elegir el elemento que más le convenga y utilizarlo para la creación de su curso.

Los recursos que se manejarán son objetos de aprendizaje, que son unidades pequeñas de aprendizaje con contenido digital los cuales están estandarizados y en repositorios, listos para ser utilizados. Gracias a sus características de modularidad y reusabilidad,

los objetos pueden ser creados, manipulados o accesibles en algún repositorio local o remoto en el que se encuentran.

La solución que se plantea está basada en las AOS, de tal manera que se identifiquen los componentes de la aplicación y se permita la interoperabilidad entre ellos. Debido a que la AOS se basa en la interacción de aplicaciones que ofrecen servicios a través de interfaces, se pretende representar a cada componente como agentes de software que darán acceso a utilizar los servicios web para realizar cada tarea específica.

1.4. Estructura de la Tesis

La tesis esta dividida en cinco capítulos más. En el siguiente capítulo se describen los fundamentos teóricos de las arquitecturas orientadas a servicios y los estándares de los servicios web. En el tercer capítulo se mencionan las características de los agentes y las ventajas de objetos de aprendizaje. En el cuarto capítulo se describe el análisis de la arquitectura y modelo del mercado de objetos de aprendizaje. Mientras que en el capítulo quinto se presenta la implementación del prototipo del sistema y sus pruebas. Por último se mencionan las conclusiones, objetivos alcanzados y perspectivas del trabajo a futuro.

Capítulo 2

Arquitectura Orientada a Servicios

A continuación definiremos la arquitectura que nos permite identificar los elementos más importantes para la localización y publicación de servicios en un ambiente distribuido. Además, describiremos los estándares de los servicios web, que son elementos importantes de esta tecnología.

2.1. Origen de la Arquitectura Orientada a Servicios

Antes de describir el concepto de Arquitecturas Orientadas a Servicios, es importante proporcionar las siguientes definiciones:

Servicio: Es la capacidad (recurso) de una entidad de software de que pueda ser accedido por otras entidades a través de una red. Un servicio es una función que está bien definida, es autocontenida y no depende del contexto o del estado de otros servicios.

Arquitectura: Describe los componentes de un sistema y la forma en que interactúan. A las interacciones entre estos componentes se llaman conectores. Nosotros vemos estos constructores como cajas negras, las cuales ocultan el detalle de su implementación. El acceso a estos componentes y el control de su conducta se hace a través

de una interfaz. La configuración de componentes y conectores proporcionan la vista estructural de la conducta de un sistema.

Hoy en día los servicios son ofrecidos por Internet a nombre de alguna organización y los consumidores, son personas que los encuentran usando un buscador. De esta forma es posible crear aplicaciones complejas en base a servicios porpiciados por diferentes proveedores.

Una AOS es una forma de conectar servicios a través de la red vía un protocolo de comunicación. La comunicación entre servicios puede involucrar desde el paso de mensajes hasta dos o más servicios coordinándose en alguna actividad. Esto permitirá crear aplicaciones a través de redes de servicio que pueden ser enlazadas para crear aplicaciones tan complejas como se requieran [12], de forma rápida. Entonces, se beneficiarán los desarrolladores, ya que sólo tendrían que imaginarse las interfaces de las aplicaciones existentes para generar otras nuevas.

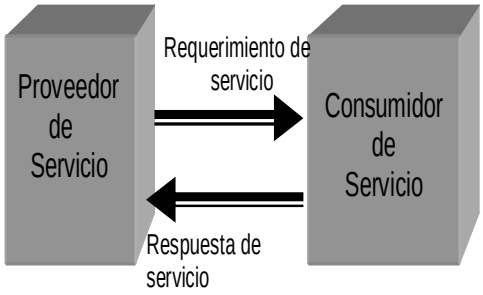

Figura 2.1: Arquitectura Orientada a Servicios Básica

En una AOS se enfatizan la interoperabilidad y la transparencia en la localización de componentes. La AOS se refiere al diseño y construcción de sistemas usando una red heterogénea de componentes direccionables de software. La figura 2.1 muestra una AOS básica[12], donde un consumidor de servicio a la izquierda que realiza una solicitud de servicio. El proveedor de servicio regresa un mensaje de respuesta al consumidor de servicio. Tanto las conexiones de requerimientos como las de respuestas subsecuentes

se definen de tal forma que son entendibles para ambos, el servidor y el proveedor.

Por ejemplo, suponga que queremos saber el precio en pesos de un libro publicado en Estados Unidos. Rápidamente podríamos construir una aplicación que usara el servicio de búsqueda de precios de un libro y un servicio que nos convirtiera el precio del libro de dólares a pesos. Esta aplicación aunque se ve sencilla si la tratáramos de construir como un sólo sistema, pero podríamos ver el hecho de que los tipos de cambio se modifican constantemente, así que dejamos la responsabilidad a un proveedor de servicios especializado en esos asuntos y resolvemos el problema.

2.1.1. Metodología Orientada a Servicios

Un servicio es una conducta proporcionada por un componente. Además, enfatiza interoperabilidad y puede ser dinámicamente descubierto y usado bajo un contrato de interfaz. Una AOS es una forma de pensar y diseñar aplicaciones que se basan en el uso de servicios disponibles en una red bajo los siguientes principios [20]:

Su uso esta basado únicamente en contratos publicados. La interfaz de un servicio funciona como un contrato, así que es indispensable diferenciar entre una interfaz pública y una interfaz publicada. La diferencia es que una interfaz pública sólo es usada por clientes conocidos, por lo cual si se realiza algún cambio se les puede notificar facilmente. En contraste los clientes de las interfaces publicadas en red no se conocen, por lo cual cambiar una interfaz puede causar serios problemas.

Interfaz direccionable en la red. Esto significa que un cliente en la red debe ser capaz de invocar un servicio. Aunque un servicio puede ser usado por componentes en la misma máquina, el servicio debe soportar una configuración de red.

Enfatiza interoperabilidad. Una AOS enfatiza interoperabilidad. Es decir cada componente debe proporcionar una interfaz que puede ser invocada a través de un

formato y un protocolo que sea entendido por todos los clientes de un servicio.

Uso y descubrimiento dinámico. Un servicio debe ser dinámicamente descubierto. Es decir, un mecanismo de una tercera parte debe ser usado para encontrar un servicio.

El hecho de estandarizar la AOS proporciona beneficios de reusabilidad entre compañías. Además, facilita la construcción de sistemas complejos de alta calidad sin necesidad de ser un experto en todo. Por otra parte, permite a los proveedores reducir costos de diseño, programación y recursos.

2.1.2. Ventajas de la AOS

- El hecho de estandarizar la AOS proporciona beneficios de reusabilidad entre compañías.
- Construcción de sistemas complejos de alta calidad sin necesidad de ser un experto en todo.
- Los proveedores pueden reducir los costos por unidad dado que pueden dividirlos entre un mercado más amplio.

2.2. Servicios Web

La Web ha revolucionado en los últimos años, lo que hace que cada vez más personas la adopten para publicar información.

2.2.1. Conceptos Generales

Existen demasiadas definiciones de SW así que mencionamos algunas de ellas que nos dan una mejor idea de lo que son:

1. Un SW es una colección de funciones que son presentadas como una sola entidad y es anunciada en la red para ser usada por otros programas. Los SW son bloques de construcción para crear sistemas distribuidos abiertos [1].

2. Un SW es un servicio con una interfaz y datos de intercambio bien definidos, sus características se anuncian en un directorio distribuido donde se puede buscar y encontrar los detalles para manipular un servicio [2].

3. Un SW es un sistema de software diseñado para soportar la interacción entre dos máquinas a través de una red. Tiene interfaz descrita en un formato que puede ser procesado por una máquina (WSDL, Web Service Description Language). Otros sistemas pueden interactuar con el SW en la manera prescrita por su descripción utilizando SOAP (Simple Object Access Protocol), típicamente transportado utilizando SOAP, y serializado utilizando XML en conjunto con otros estándares relacionados con la Web. [3].

En los SW se hace énfasis en la comunicación entre actores y se están convirtiendo en la plataforma de integración de aplicaciones gracias a las tecnologías y estándares abiertos. Esto los convierte en los elementos fundamentales en la evolución hacia la computación distribuida a través de Internet, de tal manera que no importa el hardware o software en el que operen.

Los SW son aplicaciones de XML mapeadas por programas, objetos o bases de datos o funciones de negocios. Usando un documento XML creado en forma de mensaje, un programa envía una solicitud a un servicio Web a través de una red, y opcionalmente recibe una respuesta, también en forma XML. Los SW estandarizan la definición del formato del mensaje, especifica la interface por la cual el mensaje es enviado. Describe la convención para mapear el contenido del mensaje dentro y fuera de la implementación del servicio, además define el mecanismo de publicar y descubrir interfaces de SW.

2.2.2. Componentes de Servicios Web

Los componentes de los SW son representados con rombos y las flechas denotan las operaciones que se realizan entre ellos, como se muestra en la figura 2.2 y son:

- Un servicio es una aplicación que se ofrece y es usado por solicitantes que llenan los requisitos especificados por el proveedor de servicios [1].
- Un proveedor de servicio es quien presta el servicio o la plataforma que provee el servicio.
- Un registro de servicios es el lugar donde los proveedores publican sus servicios.
- Un solicitante de servicios es la aplicación o cliente que busca e invoca un servicio.

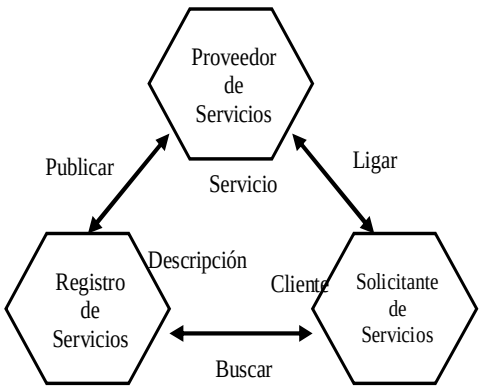

Figura 2.2: Principales componentes de los Servicios Web.

Para que los componentes interaccionen entre sí, se proporciona un conjunto de operaciones que se efectúen entre ellos, las principales operaciones de los SW [1] son:

- Publicar/Cancelar. Los proveedores de servicios publican o cancelan la disponibilidad de su servicio comercial.

- Búsqueda. Los solicitantes de servicios interactúan con uno o más registros de servicios para encontrar una solución.
- Ligar. Los solicitantes de servicios negocían con los proveedores para acceder e invocar servicios comerciales.

Gracias a las características de interoperabilidad que ofrecen los SW se permite la creación de aplicaciones a partir de otros SW de origen distinto que funcionan conjuntamente, sin importar su ubicación o la forma en que se implementaron, lenguaje o plataforma.

2.2.3. Arquitectura de Servicios Web

En la arquitectura de los SW [1] se identifican aquellos elementos globales de la red que se requieren para asegurar su interoperabilidad. Esta arquitectura da una perspectiva general de la arquitectura de servicios en red para el desarrollo de aplicaciones. La arquitectura general de SW tiene como fin mostrar el tipo de tecnologías que se pueden relacionar para ofrecer en conjunto los beneficios de los SW.

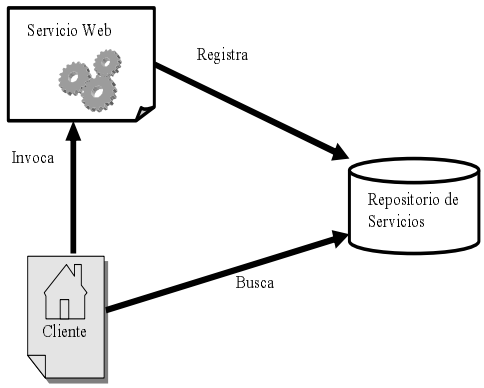

Figura 2.3: Principales tareas de los Servicios Web.

En la figura 2.3 se muestran los componentes de la arquitectura de serviciow Web,

y se denotan por medio de flechas las tareas que se efectuan entre ellos, como: buscar, invocar y registrar servicio. Los componentes de la arquitectura son[1]:

- Un *SW* que se encuentra en algún sitio, se registra y publica en un repositorio.
- El *repositorio* contiene todos los servicios públicos disponibles.
- Un *cliente* busca en el repositorio e invoca directamente al proveedor el acceso al servicio.

Como lo hemos visto la arquitectura de SW se identifica con la Arquitectura Orientada a Servicios de tal forma que combina la habilidad de invocar servicios, con herramientas para el descubrimiento dinámico de servicios, obteniendo así interoperabilidad.

En la especificación de los SW se muestra la descripción de los servicios, su implementación y su semántica. De esta manera se especifica que un SW debe tener una funcionalidad abstracta para poder realizar su implementación de forma específica. El componente que implementa el SW puede escribirse en cualquier lenguaje orientado a objetos y después, este componente puede reemplazarse de acuerdo a las necesidades del sistema, sin necesidad de cambiar la interfaz ya que está sigue siendo la misma y esta publicada para cualquier usuario.

En esta arquitectura el servicio es implementado por un agente, que se define como un software que envía y recibe mensajes. Así que el agente que se menciona en las especificaciones de la arquitectura que define la W3C se refiere a la implementación del servicio.

La Descripción de un Servicio Web (DSW) define como se lleva a cabo el mecanismo para enviar o recibir mensajes entre proveedor y consumidor. El lenguaje en que se define la descripción es WSDL, este lenguaje permite la definición de mensaje, tipos de datos

a intercambiar y protocolos en que se basará el intercambio. El documento que contiene esta información se determina como la descripción de la interfaz del servicio, ya que cada uno de estos elementos van a dar paso al acceso del servicio. Una forma general de ver los SW se muestra en la figura 2.4, donde podemos observar la secuencia en que es ejecutada la solicitud de un servicio, y como se responde a esta desde un servidor, a través de los estándares WSDL (Web Services Description Language), UDDI (Universal Description, Discovery and Integration) y SOAP (Simple Object Access Protocol), hasta completar la respuesta y ejecución de servicio.

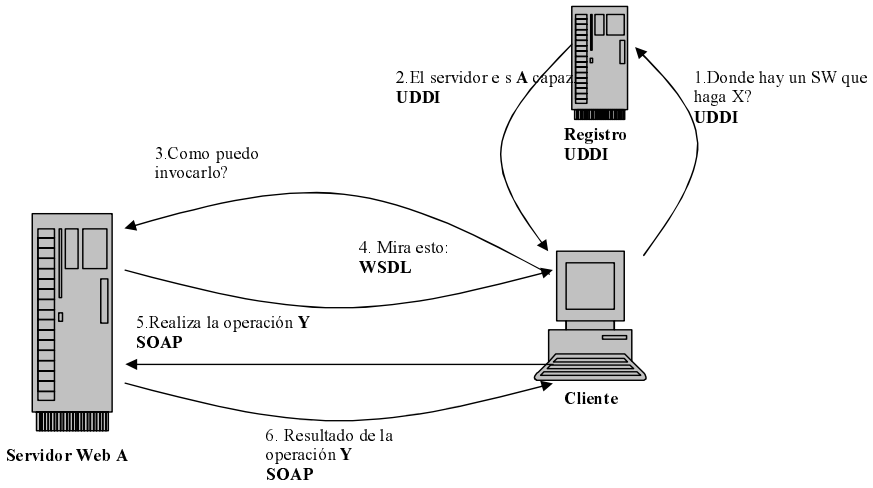

Figura 2.4: Arquitectura de Servicios Web.

En la arquitectura de servicios Web que propone la W3C [1], se define una semántica de SW. Esta semántica marca la diferencia de los SW con otras tecnologías para dar interoperabilidad, además, menciona la semántica de un SW con agentes. En este caso la semántica se da de acuerdo al comportamiento del servicio, y al contrato establecido que especifica el propósito y condiciones para ofrecer la interacción al cliente.

En la arquitectura de SW se muestra la posibilidad de que el proveedor y el cliente no se conozcan, de tal forma que los servicios pueden ser descubiertos a través de un entorno Web. Por lo tanto, pueden haber varios servicios en forma distribuida y poder

accederlos sin importar su plataforma, lenguaje o localización.

2.3. Estándares de Servicios Web

Uno de los principales problemas que se tiene es lograr que las aplicaciones entiendan la información que existe en las páginas Web, ya que la mayoría están en HTML y como bien sabemos el tipo de etiquetas que maneja sólo la podemos interpretar nosotros los humanos, así que ahora el objetivo es manejar información que esté etiquetada y estructurada de tal manera que sea entendible por los programas, para que se comuniquen e interactúen entre ellos. El lenguaje que facilita el entendimiento de la información es XML (Extensible Markup Language).

2.3.1. Lenguaje eXtensible de Marcado (XML)

La World Wide Web (W3C) generó un nuevo estándar llamado XML [16] que nace de las especificaciones de SGML (Standard Generalized Markup Language). Dándose a conocer la versión XML 1.0 en 1998, visto como un sistema para definir, validar y compartir formatos de documentos en la Web.

XML es un meta-lenguaje para representar un tipo particular de información. Se crean un conjunto de elementos y atributos, los cuales son utilizados posteriormente en otras aplicaciones dedicadas a procesar esta información [16].

XML fue desarrollado para superar las limitaciones de HTML, especialmente para soportar la creación de contenidos dinámicos y su manejo. Usando XML se pueden definir una cantidad de elementos que se asocian al significado de los datos, esto significa que puedes describir los datos y lo puedes hacer usando uno o más elementos creados con un propósito. XML es un lenguaje que permite crear su propia estructura de datos,

lo cual facilita declaraciones de contenido precisas, y ofrece la manipulación de datos en la Web.

Documentos XML

Un documento XML está formado por una estructura lógica y una física. Físicamente el documento está compuesto por unidades llamadas entidades. Lógicamente, el documento está formado de declaraciones, elementos, comentarios e instrucciones de procesamiento, los cuales están señalados con una marca explícita. Estas dos estructuras deben encajar adecuadamente para dar lugar a los documentos XML que pueden ser bien formados y válidos.

Un ejemplo sencillo de un documento XML se muestra en la figura 2.5, en la cual se describe una persona y su área de interés:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
        <persona>
                <titulo>Datos</titulo>
                <nombre>Guillermina Sánchez R.</name>
                <edad>25</age>
                <area>
                        <especialidad> Sist. distribuidos</area>
                </area>
        </persona>
```

Figura 2.5: Un documento XML sencillo.

2.3.2. Lenguaje de Descripción de Servicios Web (WSDL)

WSDL fue creado para describir y publicar los formatos y protocolos de un SW en forma estándar [18]. Los elementos de WSDL contienen una descripción de los datos, normalmente

usando uno o más esquemas XML, para ser pasados al SW en el que ambos emisor y receptor, entienden los datos intercambiados. Los elementos WSDL también contienen

una descripción de las operaciones que serán efectuadas en los datos, también que el receptor de mensajes conoce como se procesa, y una liga para el protocolo o transporte, también que el emisor conozca como se envía. Normalmente WSDL es usado con SOAP, y la especificación WSDL incluye una conexión con SOAP.

Ambas partes que participan en la conversación de los SW, acceden al mismo WSDL que está disponible para ser entendida por cualquiera de los dos. En otras palabras, ambos el emisor y receptor de mensaje envuelven la interacción del servicio web que puede ser accedido desde el mismo esquema XML.

El emisor necesita conocer como es el formato de salida correctamente, y el receptor necesita saber como entenderlo para interpretar el mensaje de entrada correctamente. A lo largo de la interacción ambas partes tienen la interpretación del mismo archivo WSDL.

Elemento WSDL

WSDL divide la especificación de los SW en tres partes, que identifican los elementos pueden ser combinados o reusados una vez definidos. Mapeando desde las aplicaciones existentes los elementos, los cuales identifican el contenido y tipos de datos de los mensajes, las operaciones a efectuar por los mensajes y la especificación de los protocolos de interacción, o transporte, para el intercambio de mensajes con las operaciones sobre la red. Estos elementos los podemos observar en la figura 2.6 en la cual se encuentran los subelementos [18]:

- Tipos de datos: Usados en los mensajes y se encuentran en el formato de esquema XML.
- Mensaje: Es una abstracción de la definición de datos, en forma de mensaje presentados a otros como un documento completo o como argumento a ser mapeado para invocar un método.

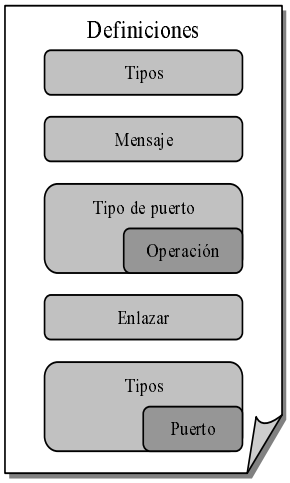

Figura 2.6: Vista de los elementos de un documento WSDL.

- Operaciones: Es una definición abstracta de las operaciones para un mensaje, tal como la llamada a un método, un mensaje encolado, o procesos de negociación, que podrían aceptar y procesar el mensaje.
- Tipo de puerto: Es una definición abstracta de las operaciones mapeadas a uno o más puntos finales, definiendo la colección de operaciones para unir.
- Enlazar: El protocolo concreto y el formato de datos para las operaciones y mensajes definidos por un tipo de puerto particular.
- Puerto: Es una combinación de una unión y una dirección de red, previendo la dirección de comunicación de un servicio.
- Servicio: Es una colección de puntos finales relacionados, que abarcan las definiciones del servicio en el archivo; los servicios mapean la unión del puerto e incluyen definiciones de extensibilidad.

Afortunadamente, estos elementos de WSDL son generadas usando herramientas que transforman los metadatos de la aplicación final en un esquema de información

XML, los cuales están combinados dentro del archivo WSDL.

Definiendo Operaciones en Mensajes

La operación es definida como que un SW que conoce como se van a interpretar los datos, si algún dato es regresado o si necesita respuesta. Las operaciones son definidas en correspondencia como un patrón común de mensaje, tales como la forma de solicitar y responder.

En la figura 2.7 se muestran los tipos de mensajes que agrupan mensajes entrada y salida para empatar el patrón de mensaje solicitud/respuesta [3].

Figura 2.7: Tipo de mensajes.

WSDL tiene cuatro tipos de operaciones sobre mensajes [13]:

- *Un-camino:* Similar a dispara y olvida, pero más simple, esto significa que el mensaje se envia sin solicitar una réplica.
- *Solicitud/Respuesta:* Similar al estilo de RPC; el emisor envía un mensaje, y el receptor envía una correspondiente réplica.
- *Solicita respuesta:* Una simple solicitud para una respuesta sin entrada de datos. Es una solicitud que obtiene un mensaje y no involucra el envío de mensaje.
- *Notificación:* Este tipo de operación define múltiples receptores para un mensaje, similar al broadcast, y frecuentemente involucra un mecanismo de suscripción,

como una publicación.

2.3.3. Descripción, Descubrimiento e Integración Universal (UDDI)

Después de que se inicializa un SW, las personas tendrían que buscar y usar el servicio. El propósito de UDDI (*Universal Description Discovery and Integration*) [1] fue establecido para crear e implementar un directorio de SW. UDDI, es una iniciativa de estándar independiente que propone la convención para crear y consultar registros de negocios y sus servicios, incluyendo a los SW [15].

Para entender mejor como funciona UDDI nos imaginamos un directorio telefónico donde nosotros podemos encontrar servicios y tomar la información como nombre de la empresa, dirección, teléfono, etc. Esta información nos sirve para ponernos en contacto con la empresa. Similarmente un programa local necesita información para que pueda interactuar con otra aplicación que esta corriendo en cualquier otra máquina en la Web, por lo tanto las aplicaciones remotas necesitan ser publicadas. A través de UDDI se facilita la publicación por medio de sus páginas blancas o amarillas, que están a disposición en tiempo de diseño y en ejecución. Por lo tanto UDDI es un recurso que puede ser considerado parte de la arquitectura e infraestructura de los SW.

UDDI tiene una prioridad en la calidad o validación de los datos, alguien tienen que asegurar que los servicios o negocios que son registrados sean un servicio real como: el nombre, el id, información de la categoría, número de teléfono, página web, y dirección sean correctos, además que la categoría del servicio e información geográfica sea correcta.

UDDI tiene dos partes principales [3]:

- Registro: Significa que los negocios pueden ingresar información.

- Publicación: Es la manera de exponer la información a disposición de otros, para que la descubran.

UDDI a través de APIs de SOAP o alguna interfaz provee las operaciones o SW a otros vendedores.

Como trabaja UDDI

La información que describe UDDI está dividida en tres categorías principales para manejar la información de repositorios [16]:

- Páginas blancas: Contienen información general sobre las organizaciones que ofertan servicios, como el nombre del negocio y dirección, información del proveedor, el nombre del sitio Web, y el identificador del negocio.
- Páginas amarillas: Incluyen información sobre la clasificación de la información de la compañía. Por ejemplo el tipo de negocio, servicio y su localización.
- Páginas verdes: Las páginas verdes contienen información técnica sobre los servicios y la forma de utilizarlos. En esta categoría se describen las funcionalidades y características de servicio, incluyendo el id del servicio, y es completamente específica para Internet.

UDDI publica los operadores de descripción de WDSL de los SW para el registro y descubrimiento, además provee de forma separada los archivos WSDL para el registro y descubrimiento de servicios, por supuesto en formato XML. Estos esquemas continúan con la extensibilidad y flexibilidad en el almacenamiento de datos para un negocio o entidad particular.

Modelo de datos de UDDI

La información de registro comprende los cinco tipos de estructuras de datos [3]:

- bussinessEntity: Es el nivel más alto de la estructura que contiene la información que se está registrando.
- businessService: Contiene el nombre y descripción del servicio publicado.
- bindingTemplate: Contiene la información acerca del servicio, incluyendo la dirección para acceder al servicio.
- tModel: Contiene una colección de información que identifica únicamente la especificación del servicio.
- publisherAssertion: Es una estructura de relación que coloca en la asociación dos o más estructuras businessEntiy de acuerdo al tipo de especificación de la relación.

En la figura 2.8 se muestra el modelo de datos básico de UDDI, en los cuales los datos son relacionados con una estructura *businessEntity* que puede ser retornada como resultado de una consulta. Entonces la relación entre las estructuras es establecida como referencia. Se muestran además los elementos que son usados para establecer las relaciones, por ejemplo, la entrada en businessServices en el esquema *businessEntity* es opcional; sin embargo si se utiliza debe contener uno o más servicios [3].

UDDI proporciona información y provee un mecanismo para describir los tipos de servicios que después se registran y publican. Por lo tanto UDDI es un registro público diseñado para almacenar de forma estructurada información sobre empresas y los servicios que éstas ofrecen. Las principales operaciones son [15]:

- Describir
- Buscar
- Consultar

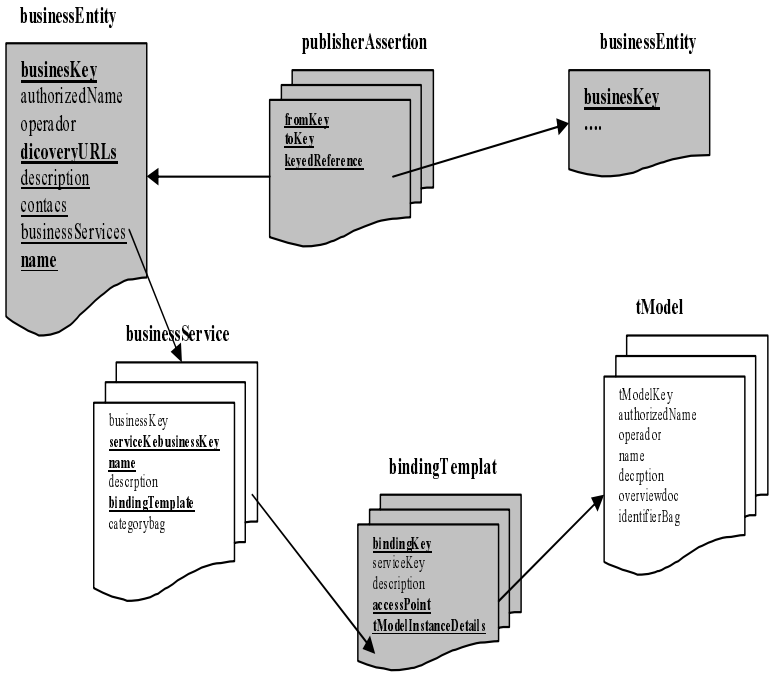

Figura 2.8: El modelo básico de datos UDDI es una jerarquía de contención

2.3.4. Protocolo de Acceso Simple a Objetos (SOAP)

SOAP es un protocolo de comunicación entre aplicaciones que permite la interacción entre diferentes Sistemas Operativos, con diferentes tecnologías y lenguajes de programación. Permite a las aplicaciones invocar métodos, objetos o funciones, que residen en sistemas remotos, es decir a través de SOAP podemos acceder al servicio de acuerdo a los métodos respectivos [17].

Este protocolo fue definido por la W3C basado en XML para invocar SW. SOAP define las operaciones de un SW como:

- Mensajes de petición
- Mensajes de respuesta.

El mecanismo más común y recomendado para transmitir los mensajes SOAP, es

el protocolo HTTP, ya que permite aprovechar la infraestructura de Internet existente. Esto da facilidad para la transmisión de mensajes ya que es el protocolo más utilizado.

SOAP se define como un esquema de codificación particular para tipos de datos que puede ser utilizado para comunicar llamadas a procedimientos remotos (RPC), así como una realización concreta de la infraestructura fundamental para los enlaces entre protocolos, definido en [17]. Este enlace permite el intercambio de mensajes SOAP, ya sean una petición y respuesta HTTP POST, o como un mensaje SOAP en respuesta a un HTTP GET.

Estructura de un mensaje SOAP

De acuerdo a la especificación, un mensaje SOAP es un documento XML que contiene 3 elementos [3] :

La envoltura del mensaje (*envelope*) es el elemento principal. La envoltura (*header*) contiene las características generales de la comunicación.

El cuerpo del mensaje (*body*) contiene los siguientes elementos:

- Nombre del método
- Argumentos del método
- Documento a enviar.

En este caso el endevelope contiene a los elementos header y body. Un mensaje tradicional [3] tiene la forma que se muestra en la figura 2.9.

El cuerpo de un mensaje en SOAP

Ahora puntualizaremos el uso de SOAP para transportar los métodos que llaman a los SW y los métodos que responden desde un Servicio Web. El cuerpo de un mensaje en SOAP es usado para transmitir el nombre del método a ser llamado, con los parámetros asociados y una respuesta desde la aplicación receptora. El cuerpo puede

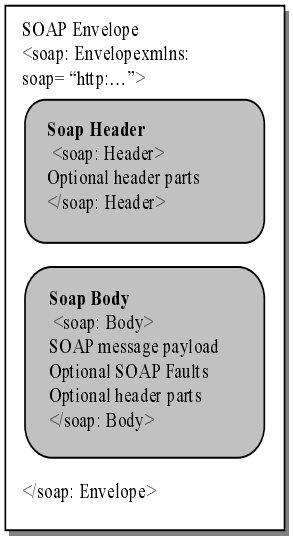

Figura 2.9: Mensaje tradicional en SOAP

además, contener información de solicitud para llamar la aplicación si la respuesta fue exitosa o si falla el mensaje por que ocurrió algún problema.

En la figura 2.10 se muestra un ejemplo de mensaje de solicitud y respuesta de SOAP, generada por un cliente

```
<?xml version = “1.0” encoding= “utf-8”?>
<soap: Envelope xmlns :SOAP-ENV=”http ://schemas.xmlsoap.org/soap/envelope/”
        xmlns:xsi=”http://www.w3c.org/2001/XMLSquema-instance”
        xmlns:xsd=”http://www.w3c.org/2001/XMLSquema”
        <soap: Body>
                <GetCalcPercent xmls=”http://www.notashop.com/wscr”>
                        <Percent> 25 </Percent>
                        <Number> 4 </Number>
                </GetCalcPercent>
        </soap: Body>
</soap:Envelope>
```

Figura 2.10: Listado de solicitud

El ejemplo mostrado representa un Servicio Web para el cálculo del porcentaje de un número. El mensaje de petición mostrado en el listado anterior indica el método que se invoca *GetCalPercent*; este método recibe como parámetro un número clave de la acción y su respuesta, se muestra en el listado anterior contiene el nombre del

mismo método, pero con el elemento *GetCalPercentResult* , que es el valor de retorno del método que representa el precio de la acción solicitada.

En la figura 2.11 se muestra la respuesta al mensaje de solicitud, un mensaje de Respuesta de SOAP, donde *GetCalcPercentResponse* es la respuesta del elemento *CalcPercent*. El ejemplo mostrado representa un SW para la consulta de precios de acciones en la bolsa de valores.

```
<?xml version = "1.0" encoding= "utf-8"?>
<soap: Envelope xmlns :soap="http ://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsi="http://www.w3c.org/2001/XMLSquema-instance"
        xmlns:xsd="http://www.w3c.org/2001/XMLSquema"
        <soap: Body>
                <GetCalcPercentResponse xmls="http://www.notashop.com/wscr">
                        <CalcPercentResult> 1</CalPercentResult>
                </GetCalcPercentResponse>
        </soap: Body>
</soap:Envelope>
```

Figura 2.11: Listado de respuesta

2.3.5. Escenario de los Servicios Web

Los SW podrían no regresar algún resultado en la misma forma, y por eso la solicitud es importante. Similarmente, un servicio puede ser usado internamente por una aplicación o publicado en Internet para ser incluido en otras aplicaciones. Por lo que es importante la naturaleza de una solicitud y su manejo. En la práctica, estos son tres principales tipos de SW:

Servicios simples: Es la premisa básica de los SW para proveer una pieza simple de funcionalidad a un cliente accesible desde Internet. Por ejemplo: una empresa de transporte podría exponer una función a la que se le pasen como parámetros el lugar de

origen, el lugar de destino, la hora deseada de entrega, y el peso del paquete, y obtendríamos como respuesta el precio del envío.

Integración de aplicaciones. Como las conexiones de redes e Internet son muy rápidas, las compañías toman ventajas del ancho de banda para separarse de sus aplicaciones y recursos en red o locales. Varios servidores pueden hablar con otros, u otras aplicaciones autónomamente. Esta integración es muy buena cuando una compañía usa el mismo tipo de sistema, pero, que pasa si las compañías quieren compartir recursos con otras compañías?. Entonces, se diría que no son compatibles. La ventaja de la integración en los SW es que la integración de aplicación o middlemen, trabaja como intermediario entre las aplicaciones que normalmente no deberían estar disponibles para hablar con otras que están escritas en otra plataforma y que tienen otros componentes de arquitectura. De esta forma podemos integrar software muy diverso, incluso aplicaciones que realicen transacciones de empresa a empresa.

Ambientes de servicios: La idea de SW como aplicaciones agrega la posibilidad de no sorprender a nadie. El concepto de un componente que provee funciones adicionales a una aplicación no es nuevo, pero cabe la forma en que los SW trabajen como un soporte. Considerando por un momento las características generales de los SW, se puede ver como estos pueden usarse en la misma forma como en el host más que en el cliente. Existen problemas para los que su resolución exige tener muchos procesadores u ordenadores funcionando de forma coordinada. Lo que se hace es dividir el problema en subproblemas y resolver cada uno de estos subproblemas en un nodo, finalmente con el resultado devuelto por todos estos nodos se obtiene la solución. Los sistemas de Grid Computing también están siendo sometidos a un examen concienzudo.

Capítulo 3

Agentes y Objetos de Aprendizaje

Inicialmente se trabajaba con aplicaciones independientes, después debido a la necesidad de compartir recursos de información crearon aplicaciones en redes de cómputo, pero aún hay más retos en esta área, ya que se necesitan sistemas que corran en ambientes distribuidos, abiertos y dinámicos y que no actúen de forma aislada.

Se requiere que las aplicaciones se comuniquen entre sí, y además que interactuen con otras aplicaciones para participar en la solución de problemas. Las aplicaciones pueden necesitar ayuda debido a que la solución de su problema va más allá de sus capacidades o bien porporcionar apoyo a otras aplicaciones.

3.1. Agentes

Hoy en en día requerimos sistemas de cómputo que respondan de forma similar como lo hacemos los humanos. Este tipo de situaciones a implementar comprende una complejidad de tareas que pretenden ser automatizadas, y es un área de investigación que ha tenido mucho interés. Ahora lo que se necesita es que los sistemas se adapten a requerimientos de los humanos.

Si se desea delegar una tarea a sistemas de cómputo, es necesario que estas tengan cierto grado de inteligencia para poder realizar su tareas, o bien la capacidad de socializar para alcanzar las metas que sus limitaciones no les permita resolver por si sólos. Este tipo de escenarios hizo que la tendencia en el campo de la computación naciera el paradigma basado en agentes, que inició siendo un área de interés dentro de la Inteligencia Artificial, y ahora debido a la investigación en diferentes ciencias han aportado sus teorías para dar origen a la tecnología de agentes.

La característica principal de los agentes es que son autónomos: es decir que son capaces de actuar independientemente, exhibiendo el control sobre sus estados internos. Un ejemplo de un agente trivial es un termostato, o un demonio de UNIX, los cuales cada determinado tiempo efectúan un cambio para ejecutar su tareas, claro censando su medio ambiente.

Según Nwana [13] un agente se define como un componente de hardware y/o software que es capaz de actuar exactamente en un orden para cumplir sus metas, beneficiando al usuario.

Un agente es una entidad autónoma de software que puede resolver problemas y es capaz de operar efectivamente en un medio ambiente dinámico y abierto. [Luck, et al. 03].

Un agente inteligente es un sistema (hardware o software) situado en un determinado entorno, capaz de actuar de forma autónoma y razonada en dicho entorno para llevar a cabo sus objetivos.[10]

Una forma en que trabajan los agentes se muestra en la figura 3.1. [10]. Por lo que se puede decir que un agente es un sistema de cómputo capaz de realizar acciones autónomas en algún medio ambiente con el fin de alcanzar sus objetivos.

Por lo tanto a los agentes los podemos ver como entidades autónomas capaces de interactuar sin la intervención directa de un usuario y capaces de reaccionar a cambios

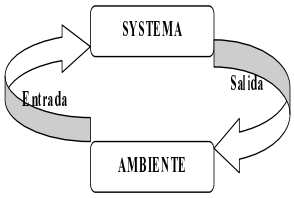

Figura 3.1: Esquema de funcionamiento de los Agentes

en su medio ambiente ambiente, así como de modificarlo para alcanzar sus metas.

Según Wooldridge y Jennings [10] los agentes tienen las siguientes propiedades:

Autónomos: Agente que opera sin la necesidad directa del humano u otros, y tiene algún tipo de control sobre sus acciones y estados internos.

Reactivos: Los agentes perciben su medio ambiente y responden en un tiempo considerado de acuerdo a los cambios que han ocurrido en él.

Sociables: Los agentes interactúan con otros agentes (y posiblemente humanos) a través de algún lenguaje de comunicación de agentes.

Proactivos: Agentes que son capaces de cambiar su comportamiento de acuerdo a su medio ambiente para alcanzar sus metas tomando sus propias iniciativas.

Otras propiedades de los agentes [10]:

Movilidad: La habilidad de un agente para moverse en una red.

Veracidad: Los agentes nunca se comunican información falsa.

Racionalidad: Los agentes siempre actúan buscando la satisfacción de sus metas y nunca harán algo que les impida alcanzarlas.

Aprendizaje / adaptación: Los agentes mejoran su desempeño con el tiempo.

Además existen otros tipos de agentes según Nwana [13] que muestra la topología, de acuerdo a sus metas, beneficios, roles, desafíos, entre otros.

Heterogéneos: Son la combinación de dos o más características de agentes simples.

Inteligentes: Son aquellos que aprenden, cooperan y son autónomos.

Movil: Tiene la habilidad de moverse en una red.

Información: Son aquellos que tienen la capacidad de manejar información a través de un medio ambiente dinámico.

Reactivos: Tienen un modelo interno de su ambiente y reaccionan a cambios que ocurran en el mismo.

Colaborativos: Tienen la característica de ser autónomos y cooperar.

Existe una clasificación de agentes de acuerdo a sus características antes mencionadas:

De acuerdo a la Movilidad existen agentes: estático o móvil.

Con respecto al Modelo de Razonamiento se clasifican en: reactivo o deliberativo.

Las funciones que puede hacer, van de acuerdo a la tarea que tenga de: información, gestión, etc.

De acuerdo a sus atributos se clasifican en: autónomo, adaptable, cooperativo,etc. Por último se clasifican como híbridos si contienen combinación de las anteriores características.

Entonces si a los agentes los comparamos con los objetos, los agentes son activos. Sus actividades incluyen objetivos y condiciones que guían la ejecución de las tareas definidas, y toman responsabilidades de sus necesidades. Los agentes pueden actuar de forma individual o con otros agentes, es decir, pueden tener la capacidad de formar una comunidad social de miembros independientes que actúan de forma autónoma.

"Los agentes constituyen el próximo avance más significativo en el desarrollo de sistemas y pueden ser considerados como la nueva revolución en el software". Esta frase fue pronunciada por el Dr. Nicholas Jennings en su discurso al recoger el premio al mejor investigador novel del congreso internacional de Inteligencia Artificial celebrado en Estocolmo [11]. Resulta mucho más impactante aún cuando tal afirmación se ve refrendada por numerosos indicadores, como por ejemplo el gran interés despertado

tanto en el ámbito académico como industrial. Este es el nuevo paradigma a seguir para generación de aplicaciones.

3.1.1. Sistemas Multiagentes

Un Sistema Multiagente (SMA) es un sitema distribuido en el cual los elementos o nodos donde la conducta combinada de los elementos produce un resultado inteligente, y satisfactorio de la tarea global.

Además, los SMA permiten a los desarrolladores enfocarse a tareas y actividades que han de seguirse para la solución de un problema. Estas actividades se asignan a un programa especializado (agente) el cual es capaz de trabajar utilizando información y tomando desiciones en nombre de un usuario u otro programa.

Los Sistemas Multiagentes buscan lograr la cooperación de un conjunto de agentes autónomos para la realización de una tarea. La cooperación depende de las interacciones entre los agentes e incorpora tres elementos: la colaboración, la coordinación y la resolución de conflictos.

Los SMA tienen un amplio rango de aplicaciones, uno de los campos principales lo constituyen los agentes de software, los cuales se focalizan principalmente en la explotación cooperativa de recursos accesibles por Internet.

3.1.2. Arquitecturas Orientadas a Servicios Basadas en Agentes

Dada la naturaleza dinámica de los servicios (la creación de nuevos servicios, la desaparición de otros, la suspensión de algunos, etc.) y la heterogeneidad de los consumidores y proveedores, De Roure [12] propone un nuevo modelo para ofrecerlos y encontrarlos. Este modelo está basado en el esquema de mercados donde clientes y proveedores convergen de forma regulada y establecen una relación que les permita satisfacer sus necesidades. Así, clientes entrarían al mercado en busca de proveedores

de servicios y los proveedores entrarían con la finalidad de encontrar quien consuma sus servicios, ambos buscarían obtener el máximo beneficio. En esta estructura no es necesario que los proveedores y clientes se conozcan entre sí con anterioridad, los dueños de mercados actuarían no sólo como introductores sino como reguladores de las relaciones que puedan establecerse entre ellos.

En esta estructura social, es posible que clientes y proveedores tengan intereses conflictivos, por lo cual antes de que un servicio pueda ser proporcionado, se deben negociar los términos y las condiciones de uso. Los acuerdos finales deben ser legalizados en un contrato donde no sólo se especifiquen los términos y condiciones del servicio, sino también las obligaciones, los beneficios y las posibles penalizaciones (en caso de incumplimiento de alguno de los términos) de los participantes. Los agentes de software son un paradigma ideal para representar proveedores, consumidores y propietarios de servicios y los sistemas multi-agentes podrían usarse para modelar los mercados [12] .

En este modelo de mercados, los servicios tendrían un ciclo de vida consistiendo de tres etapas a saber: creación, obtención y cumplimiento del contrato, los cuales son ilustrados en la figura 3.2.

- *Creación de un servicio.* En este paso, el propietario del servicio define el servicio que quiere poner a la disposición de los demás. Todo el tiempo se están creando y destruyendo servicios lo cual hace que el sistema sea dinámico. El proceso de creación de un servicio involucra tres actividades algunas de las cuales podrían automatizarse:

- La especificación de como se realizará el servicio en un lenguaje apropiado. Esta parte en general no está disponible a la vista externa. En términos de agentes ésta sería la parte correspondiente a la creación de un agente.

- La especificación de la meta-información asociada con el servicio (ejemplo: la

especificación de quien puede usar el servicio y las posibles formas en que puede satisfacerse).

- Finalmente, el hacer disponible el servicio en el mercado apropiado. Esto requiere de propaganda para el servicio y facilidades de registro en el mercado.

- *Obtención del servicio.* La obtención de un servicio involucra un proveedor de servicio y un consumidor estableciendo un contrato para legalizar que el servicio acordado se proporcione en los términos y condiciones aprobados.

Aquí hay que considerar tres cosas importantes:

- El proveedor puede fallar (por voluntad o por inhabilidad).

- Dado que los agentes proveedores y consumidores pueden representar a usuarios con intereses no totalmente compatibles, el contrato se firma después de un proceso de interacción, por lo tanto hasta que ambas partes estén de acuerdo en los términos y las condiciones.

- Los procesos de negociación pueden ser hechos fuera de línea entre usuarios propietarios de los agentes, o bien en línea por los propios agentes. Este proceso se dejará como trabajo a futuro por su complejidad y por que no es el objetivo de la tesis.

- *Cumplimiento del contrato.* El punto final debe ser el cumplimiento del contrato donde el proveedor del servicio lleva a cabo las acciones necesarias para cumplir con sus obligaciones. Al finalizar, el proveedor debe informar al consumidor su cumplimiento del contrato. Cuando el proveedor se vea imposibilitado de cumplir con el contrato, este puede ser renegociado y modificar los términos y condiciones iniciales.

En resumen, los elementos identificados como claves para la implementación de una arquitectura orientada a servicios son: los propietarios y proveedores del servicio,

los consumidores del servicio y los mercados donde ambos confluyen. Las actividades realizadas por cada uno de ellos están resumidas en la figura 3.2 tomada de [12].

Propietario del Servicio	**Consumidor del Servicio**	**Mercados**
Creación del servicio	Descubrimiento del servicio	Registro de propietarios y consumidores
Publicitar el servicio		Registro del Servicio
Creación del contrato de servicio	Creación del contrato de servicio	Especificación de políticas
Entrega del servicio	Recepción de los resultados del servicio	Políticas de monitoreo y forzamiento de contratos

Figura 3.2: Elementos y sus roles en el Mercado

Un enfoque de esta arquitectura se muestra al contemplar la arquitectura orientada a servicios como un modelo de mercado [12] como se muestra en la figura 3.3 que se menciona a continuación:

- El proveedor de servicio (vendedor)
- El solicitante del servicio (comprador)
- El registro del servicio (propietario del mercado)

Cada uno de estos elementos juega un papel muy importante para llegar a un acuerdo y así obtener la compra-venta de un servicio.

3.1.3. Agentes en el Futuro

Las tecnologías en agentes han sido catalogadas como el elemento clave para desarrollar aplicaciones tales como las requeridas en:

- Inteligencia Ambiental: Computación ubicua e interfaces inteligentes para dispositivos industriales y domésticos.

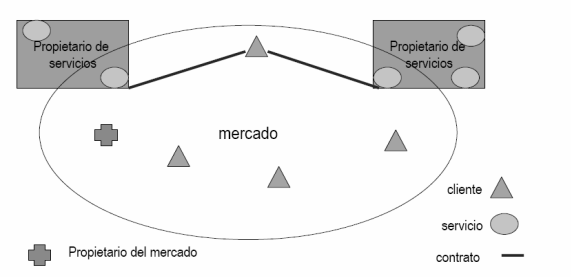

Figura 3.3: Arquitectura Orientada a Servicios visto en un esquema de mercado

- Computo en la Grid: Se requiere el uso eficiente de los recursos para aplicaciones de alto desempeño.
- Comercio Electrónico: Para transacciones de tipo comercial en cualquier organización a través de Internet.
- La Web Semántica: En esta caso se requieren agentes proveedores de algún tipo de servicio, que se publican y solicitan por parte de alguna organización, particular o aplicación.

3.2. Objetos de Aprendizaje

Actualmente, en la educación a distancia los cursos en línea son producidos como sistemas monolíticos que tienen contenidos indivisibles, no reutilizables entre la comunidad académica, incrementando los costos de producción de contenidos y consecuentemente son descuidados sus aspectos de calidad. De acuerdo a los costos que genera la creación de cursos, determinamos manejar unidades pequeñas que se puedan intercambiar, esto permitirá que sean útiles en diversos cursos en línea, es decir que sean flexibles y reutilizables en el ámbito educativo, lo anterior, nos lleva al manejo de objetos de aprendizaje.

3.2.1. Origen de los Objetos de Aprendizaje

El desarrollo de la metodología de objetos de aprendizaje ha permitido plantear una nueva forma de pensar la estructura del aprendizaje electrónico (e-learning) y, principalmente en el diseño de los cursos en línea, de tal manera que proporcione flexibilidad para el desarrollo de contenidos, disminución de costos y menos inversión de tiempo para crear el material. Podríamos definir a los objetos de aprendizaje como las unidades básicas de contenido educativo con información tal como un texto, sonidos, videos, etc. Por lo tanto la estructura de un objeto de aprendizaje puede ser tan sencilla como una definición (digital) o hasta una simulación, es decir, cualquier recurso digital y pedagógico.

Existen una gran cantidad de problemas asociados al origen de los OA, los principales son:

- La gran diversidad de criterios para elaborar cursos de aprendizaje electrónico, ya que cada quien crea sus cursos según su conveniencia, sin un estándar de enseñanza que permita a todos seguir el mismo esquema para generar el material.
- Los cursos o los programas no son interoperables (capacidad para poder integrarse en estructuras y plataformas diferentes) y su diseño no es transportable.
- La administración de los programas y los cursos no es interoperable, por lo tanto no permiten esfuerzos interinstitucionales. Debido a que no hay una forma de establecer como compartir cursos que están diseñados bajo distintas herramientas de diseño.
- Los contenidos no están estandarizados, por lo tanto no son intercambiables, modificables o reutilizables.

- Los programas o cursos son unidades demasiado grandes y enfrentan problemas de reconocimiento, revalidación, acreditación y certificación. Además, como son demasiado grandes, si alguien requiere sólo una parte de un curso este no se puede prestar ya que es un curso indivisible.

Por lo tanto, en el área de la investigación nació un nuevo paradigma basado en los objetos de aprendizaje. Donde un conjunto de objetos de aprendizaje conforman un curso en línea, tal como se ilustra en la figura 3.4.

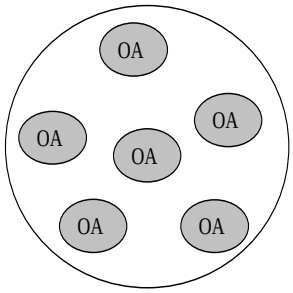

Figura 3.4: Curso en línea:Uno o más objetos de aprendizaje

3.2.2. Características de Objetos de Aprendizaje

Existen muchas definiciones y formas de caracterizar a los OA, aqui presentamos algunas:

- Un OA es cualquier recurso digital que se puede utilizar como apoyo para el aprendizaje [19].(todo tipo de archivo digital como texto, video, artículo, página web, etc).

- De acuerdo con Simón [14] un OA es un recurso digital que está encapsulado en una lección o un conjunto de lecciones que conforman unidades, módulos, cursos e incluso programas.

- Según González [6] un OA es una entidad digital con características de diseño instruccional, que puede ser usada, reutilizada o referenciada durante el aprendizaje soportado por computadora con el objetivo de generar conocimientos, habilidades y actitudes en función de las necesidades del alumno.

Para nosotros en lo que resta de este documento un *objeto de aprendizaje* será y está formado por cualquier tipo de recurso digitalizado como texto, video, artículo, etc., con una caracterización pedagógica, y que está basado en estándares internacionales.

Wiley, González y Simón describen las características mínimas de los objetos de aprendizaje de la siguiente forma:

- Autocontenidos, es decir, los OA deben tener la capacidad de auto explicarse y posibilitar experiencias de aprendizaje integral.
- Interoperables. Esta propiedad se refiere a la capacidad de integración con otros OA.
- Durables. La información de los OA tienen una vigencia a fin de eliminar obsolescencia.
- Accesibles: Es el acceso a los OA debe ser fácil.
- Reusables: La reusabilidad se refiere a la capacidad de los OA para combinarse dentro de nuevos cursos.
- Interactivos: A la capacidad de generar actividad y comunicación entre sujetos involucrados, se le conoce como interactividad.

A continuación se mencionan las *principales ventajas de usar OA*:

- Estimular el estudio autogestivo. Es decir dada la flexibilidad en el contenido estos objetos permitiran una fácil comprensión del material por parte del alumno.
- Posibilitar el acceso remoto a la información y a contenidos de aprendizaje. Permitir que cualquier usuario pueda acceder al contenido desde otra máquina.
- Posibilitar la integración de diferentes elementos multimedia a través de una interfaz gráfica.
- Contribuir a la actualización permanente de profesores y alumnos. Estructuración de la información en formato hipertextual.
- Facilitar la interacción entre diferentes niveles de usuarios (administrador, diseñador y alumno).

Los OA están basados en la innovación tecnológica así como en la programación orientada a objetos, pues hacen uso de sus elementos y propiedades para poder ser reutilizados e interoperables con otros. Además, la reutilización se permite a través de Internet, ya que es un medio que está disponible y se tiene la facilidad de obtener información en cualquier lugar y en cualquier momento. Esta visión de cambio permite que el diseño y distribución de objetos de aprendizaje innoven la docencia en el contexto educativo. A continuación podemos ver la relación entre componentes de un OA, denotadas por círculos en la figura 3.5, además nos define como un OA está basado en la programación orientada a objetos. Un objeto de aprendizaje está definido por estándares basados en metadatos, y empaquetados, con el fin de ser catalogados e indexados.

La reutilización de los objetos de aprendizaje entre sistemas distintos requiere que estén estandarizados, por lo que muchas organizaciones se dedican al desarrollo de

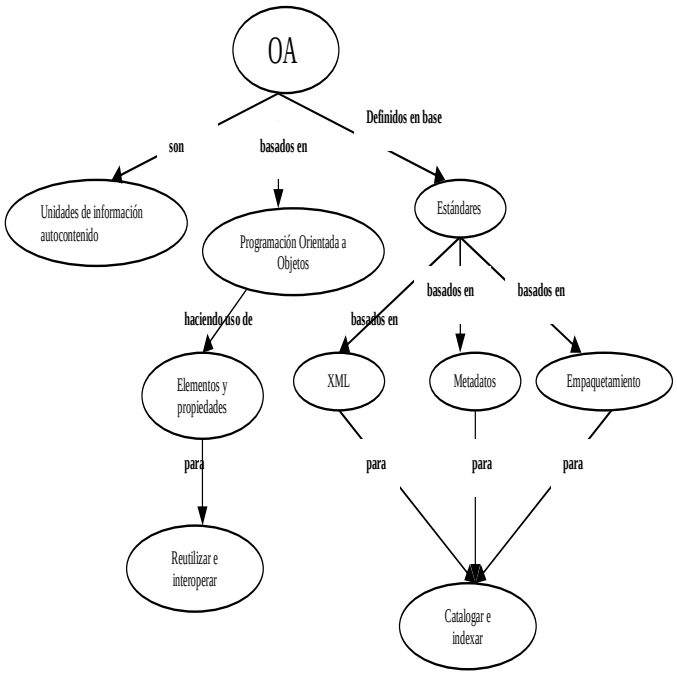

Figura 3.5: Panorama general de la interacción de componentes OA

estándares, especificaciones y modelos de referencia. El estándar en el que están basados para generar su estructura es XML permite definir a través de metadatos el contenido del objeto, de tal manera que se mantiene "empaquetado"para su utilización e interoperabilidad, lo que facilita la forma de presentar, manipular y transmitir documentos. En el caso de los objetos de aprendizaje se requieren para el empaquetamiento de contenidos. Esto facilita la forma de agregar, indexar y localizarlos fácilmente de acuerdo al contenido.

En la figura 3.6 se muestra un ejemplo de algunos elementos definidos como metadatos de un objeto de aprendizaje, como lo denota cada etiqueta: el objetivo, tipo de recurso, el nivel educativo , contexto, etc.

Para que los objetos de aprendizaje sean manipulados por cualquier persona es necesario que cumplan con un estándar, a fin de poder reutilizarlos. Existen diversas organizaciones que se dedican a ello, una de las principales es SCORM (Sharable Con-

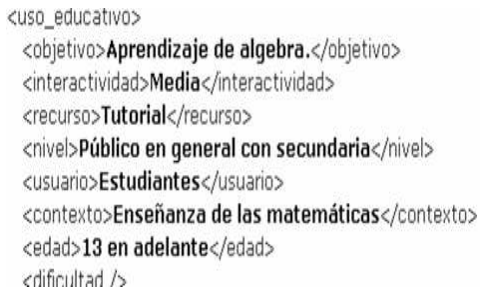

Figura 3.6: Metadatos de un objeto de aprendizaje

tent Object Reference Model). Por otra parte, para que los objetos sean reutilizables es necesario que estén en un repositorio, ya que solo asi podrán realizarse las operaciones de catalogar, indexar y localizar.

Los aportes en investigación se han centrado en generar nomenclaturas para los objetos de aprendizaje, optimizar los procesos de diseño, el estudio de las combinaciones de elementos nucleares en la construcción de objetos de aprendizaje, y finalmente, su relación con las teorías del diseño instruccional.

La investigación realizada está enfocada en sentar las bases para crear un repositorio distribuido de OA que facilite su localización.

3.3. Metodología

El problema planteado en el capítulo 1, se piensa resolver en base a la AOS, donde cada uno de los componentes de la arquitectura estará representado por administradores de los OA. Cada uno de estos componentes tiene tareas específicas.

Para el desarrollo del sistema hemos decidido usar el Proceso Unificado de Desarrollo de Software para identificar los requerimientos del sistema, así como la forma de interacción entre los agentes. A continuación, mencionaremos una breve explicación de este método.

3.3.1. Proceso Unificado de Desarrollo de Software (PUDS)

El Proceso Unificado de Desarrollo de Software se centra en un carácter rígido por casos de uso, centrado en la arquitectura, iterativo e incremental. El proceso utiliza el Lenguaje Unificado de Modelado (UML). El proceso se basa principalmente en el proyecto de desarrollo de componentes reutilizables, es decir en piezas de software con interfaces bien definidas.

Otro punto importante es que se requiere crear sistemas en menos tiempo, pero esto puede resultar difícil, ya que los métodos de desarrollo que se utilizan son muy obsoletos, lo que genera un gran problema para el desarrollador. Por lo tanto los desarrolladores necesitan apoyarse en un método que les permita a varios desarrolladores crear sistemas grandes que se dividan para trabajar de forma coordinada. De tal manera este proceso que utilizamos es el Proceso Unificado de Desarrollo de Software [7] que se basa en:

- Proporcionar una guía para ordenar las actividades de equipo.
- Permitir dirigir las tareas de cada desarrollador por separado y del equipo como un todo.
- Especificar los artefactos que deben desarrollarse.
- Ofrecer criterios para el control y la medición de los productos y actividades del proyecto.

Un proceso bien definido y gestionado es la diferencia entre proyectos hiperproductivos y otros que fracasan. Es así como nace el PUDS.

Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de usuario en un sistema de software. Sin embargo el proceso unificado es mas que un proceso, en sí es un marco de trabajo que se puede aplicar a una

gran gama de sistemas de software, diferentes áreas de aplicación, diferentes tamaños de proyectos, como se muestra en la figura 3.7.

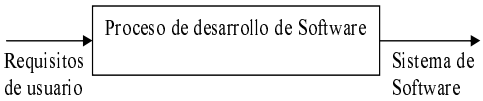

Figura 3.7: Un proceso de desarrollo de Software

El PUDS se basa en componentes, es decir que el sistema de software esta se esta construyendo en base a otros componentes de software, a través de interfaces. En general los principales aspectos del proceso son [7]:

Está dirigido a casos de uso: Un sistema de software por lo general tienen como propósito dar un servicio a sus usuarios, por lo tanto se necesita saber cuales son los requisitos y lo que desean los futuros usuarios del sistema. Los casos de uso representan los requisitos funcionales del sistema, entonces todos los casos de uso generan el modelo de casos de uso, en el cual vemos todas las funcionalidades del sistema. Por lo tanto podemos identificar los casos de uso con las siguientes preguntas. Que debe hacer el sistema para cada usuario?

Centrado en la arquitectura: Este punto es similar al término arquitectura que conocemos, ya que nos da diferentes vistas del sistema en construcción. Por lo tanto, la arquitectura es una vista de diseño completo, con sus características más importantes resaltadas. Además, se ve influida por otros factores como la plataforma en que deben funcionar el software, los bloques de construcción reutilizables, consideraciones de implantación y requisitos no funcionales.

Iterativo e incremental: Debido a que un sistema de software puede tardar varios

meses, hasta más en ocasiones. Es muy útil dividir el trabajo en varias partes más pequeñas como en el paradigma de divide y vencerás, en la que cada parte es una interacción y resulta en un incremento. En este caso las iteraciones hacen referencia a pasos de flujo de trabajo y los incrementos, al crecimiento del producto.

El producto que se obtiene empleando PUDS incluye los requisitos, los casos de uso, especificaciones no funcionales y casos de prueba. Además el modelo de la arquitectura y el modelo visual, todos estos son modelados con UML, estos elementos son los que permiten a los usuarios utilizar y modificar el sistema en cada etapa. Como se muestra en la figura 3.8, existen dependencias entre modelos, cada uno de ellos denotados a través de las flechas puntedas.

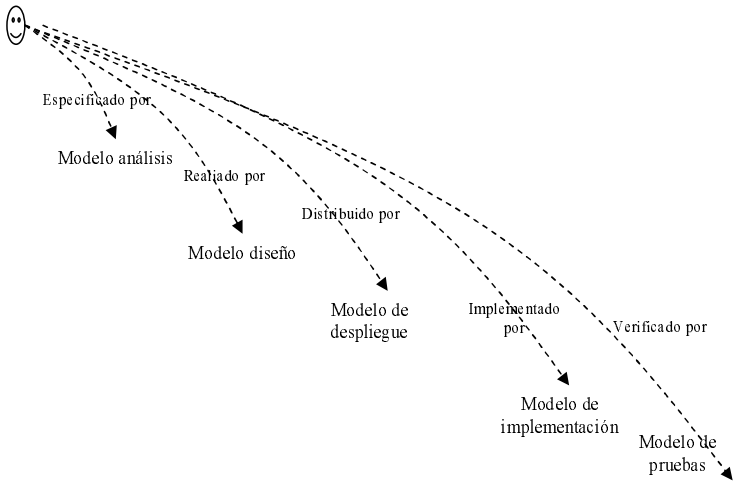

Figura 3.8: Modelo del PUDS.

Objetivo de utilizar PUDS

Hoy en día al tratar de modelar sistemas reales complejos y con características visiblemente distribuidas, es necesario identificar los diferentes subsistemas que forman parte del sistema global. Así será más fácil identificar las interacciones y dependencias entre estos componentes. Como hemos mencionado PUDS nos permite tener la base de un modelo bien fundamentado para realizar la especificación de un sistema de software

a construir.

Para comenzar a realizar el primer análisis hay que tener en cuenta un punto de vista interno del agente y un punto de vista externo del agente para poder identificar los roles de cada uno en el sistema. Para comenzar con el análisis es necesario tener una visión general de lo que es UML.

3.3.2. Lenguaje Unificado de Modelado (UML)

Desde los inicios de la informática se han estado utilizando distintas formas de representar los diseños de una manera más bien personal o con algún modelo gráfico. La falta de estandarización en la representación gráfica de un modelo impedía que los diseños gráficos realizados se pudieran compartir fácilmente entre distintos diseñadores, con este objetivo se creo el Lenguaje Unificado de Modelado (UML: Unified Modeling Language). UML contempla los métodos de Booch, Rumbaugh y Jacobson, UML no es un método, más bien es un lenguaje basado en el diseño orientado a objetos, ya que se tiene un método en al menos un lenguaje y un proceso para modelar [9].

Notaciones y modelos.

En su condición actual, UML define una notación y un metamodelo, en donde la notación es el material gráfico que se ve en los modelos. UML es una notación para modelar los procesos, teniendo en cuenta que un proceso es la orientación que nos dan sobre los pasos a seguir para realizar el diseño, es decir es la sintaxis del lenguaje de modelado. Por ejemplo, la denominación de un diagrama de clases define como se representan conceptos y temas como clase, asociación y multiplicidad. Los metamodelos buscan hacer más rigurosos los métodos orientados a objetos, a través de un diagrama, usualmente un diagrama de clases, que defina la notación.

Para la construcción de un sistema se necesitan una serie de iteraciones, donde cada

iteración sea un miniproyecto. Por lo tanto se hace un análisis, diseño, codificación, pruebas e integración de casos de uso asignados a cada iteración. En el caso de crear un sistema se necesita un modelo que contemple estas etapas, y además los tipos de diagramas que manejan para desarrollar el sistema de acuerdo a [9] son:

Modelo de casos de uso

Modelo de diagramas de clases

Modelo de diagramas de interacción

Modelo de diagramas de estado

Capítulo 4

Mercado de Objetos de Aprendizaje (MOA)

Este proyecto desarrollará un prototipo de mercado de objetos de aprendizaje, de acuerdo a la AOS basada en agentes,. En este mercado se ponen a disposición objetos de aprendizaje para que profesores puedan crear en sus cursos en línea. Además se pueden localizar objetos de aprendizaje mediante una entidad encargada del mercado.

Si proveedores y consumidores se conocieran, no habría gran dificultad, pero como están en un ambiente distribuido es muy probable que los proveedores ofrezcan, suspendan o cancelen su participación para publicar sus objetos de aprendizaje. Por lo tanto se propone manejar una estructura de mercado en la cual existe una entidad administrador del mercado quien garantiza que se cumpla cualquier tipo de contrato entre miembros del mercado.

En este capítulo se describiran las tareas que tiene cada uno de los participantes, proveedor, consumidor y dueño del mercado. Se mostraran las tareas que realiza cada integrante en el mercado a través de los diagramas de secuencia, los cuales son representados en el Lenguaje Unificado de Modelado (UML)[9], en el que se denotan los mensajes mediante los cuales se comunican los agentes y la interacción que ocurre entre ellos.

4.1. Arquitectura General del Mercado

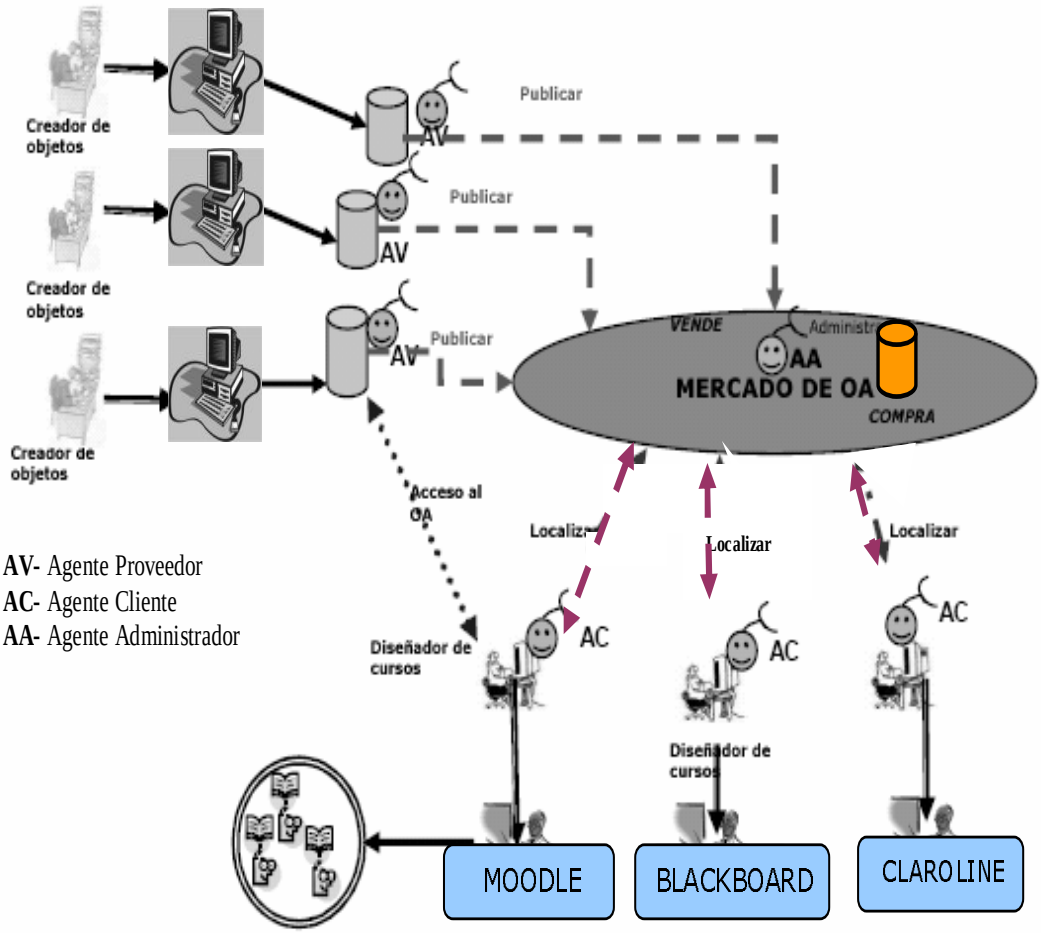

Figura 4.1: Esquema General del Mercado de Objetos de Aprendizaje

Se necesita implementar un sistema de mercado de OA que permita al usuario que tiene objetos de aprendizaje publicarlos en un sitio para que estén a disposición de profesores para crear cursos en línea. Además, debe permitir localizar de forma sencilla y garantizar que se respeten los elementos del contrato sobre el que se rige la transacción. De esta forma, el administrador del mercado dará seguimiento a las obligaciones de cada entidad participante, proveedor o consumidor.

La figura 4.1 [5] muestra los componentes del mercado y la forma en como interactúan entre ellos. Cada agente está representado mediante caritas y estos se comunican mediante mensajes denotados por líneas punteadas. Es importante mencionar que cada agente representa a un usuario distinto, por lo cual los intereses de cada uno de ellos puede diferir.

Ahora se describirán las tareas principales en la arquitectura de mercado de objetos de aprendizaje para los tres tipos de agentes: agente proveededor (AV), agente consumidor (AC) y agente administrador del mercado (AA).

1. *El agente propietario de objetos de aprendizaje es representado por el agente proveedor (AV) y es quien los pone a disposición en el mercado. Es importante mencionar que cada agente representa a un propietario que publica tipos de objetos de aprendizaje.* Sus principales funciones son:

   - Actualizar base de datos local de OA.
   - Registrarse en el mercado.
   - Darse de baja del mercado.
   - Publicitar tipos de OA en el mercado.
   - Proporciona lista de OA.
   - Negociar contrato.
   - Proporcionar OA.
   - Cumplir contrato.

   El agente proveededor reacciona a los eventos que realiza a través de la interfaz de usuario (IU) y la comunicación se denota mediante líneas continuas. Además, el agente vendedor tienen acceso a un repositorio local donde están los objetos

de aprendizaje y está representado por un cilindro. También, se comunica con el agente administrador del mercado para darse de alta y publicitar el servicio, esta comunicación se representa por líneas punteadas entre ambos agentes. Por otra parte, el agente proveedor establece una comunicación con un agente cliente cuando tiene que negociar alguna disposición de objeto, y proporcionarlo.

2. *El agente cliente de objetos de aprendizaje* trabaja para el diseñador de cursos en línea y es quien solicita un OA. Sus tareas principales son:

   - Registrarse en el mercado.
   - Darse de baja en el mercado.
   - Solicitar lista de proveedores (al administrador).
   - Seleccionar proveedor.
   - Proporciona lista de OA.
   - Comunicarse con proveedor.
   - Negociar contrato.
   - Obtener OA.

   La comunicacion entre el agente cliente y el usuario se da por medio de eventos en la IU los cuales determinan la siguiente tarea a realizar. Además, el agente consumidor AC se comunica con el agente administrador para registrarse y localizar objetos de aprendizaje en el mercado. Otra comunicación que se establece es cuando el agente consumidor pretende acceder a un OA de un proveedor. Por último, cuando el OA lo obtiene el usuario/diseñador de cursos, el OA está listo para desplegarse en algún Sistema Administrador de Aprendizaje (SAA).

3. *El propietario del mercado es quien obtiene información del proveedor y cliente.* Sus principales funciones se mencionan a continuación.

- Definir políticas de ingreso.
- Registro de proveedores.
- Registro de consumidores.
- Localizar proveedores de un tipo específico de OA.
- Supervisar cumplimiento de contratos y aplicar sanciones.
- Baja de proveedores.
- Baja de consumidores.

Este agente se muestra en el interior del semi-circulo central que representa el mercado al cual convergen los participantes. El agente administrador realiza el registro de todos los agentes que cumplan ciertas políticas de entrada, además, tiene acceso a una base de datos local, en la cual lleva el registro de proveedores y clientes que se han dado de alta. El agente administrador establece una la comunicación con un agente cliente para permitir la localización de objetos de aprendizaje. Este agente registra las transacciones que se realizen entre un agente cliente y proveedor, además de revisar el cumplimiento del contrato entre ambos.

4.2. Casos de Uso

Para tener una perspectiva global de las tareas del sistema a continuación se muestra en la figura 4.2 el diagrama de casos de usos general de forma puntual

usando la notación de UML.

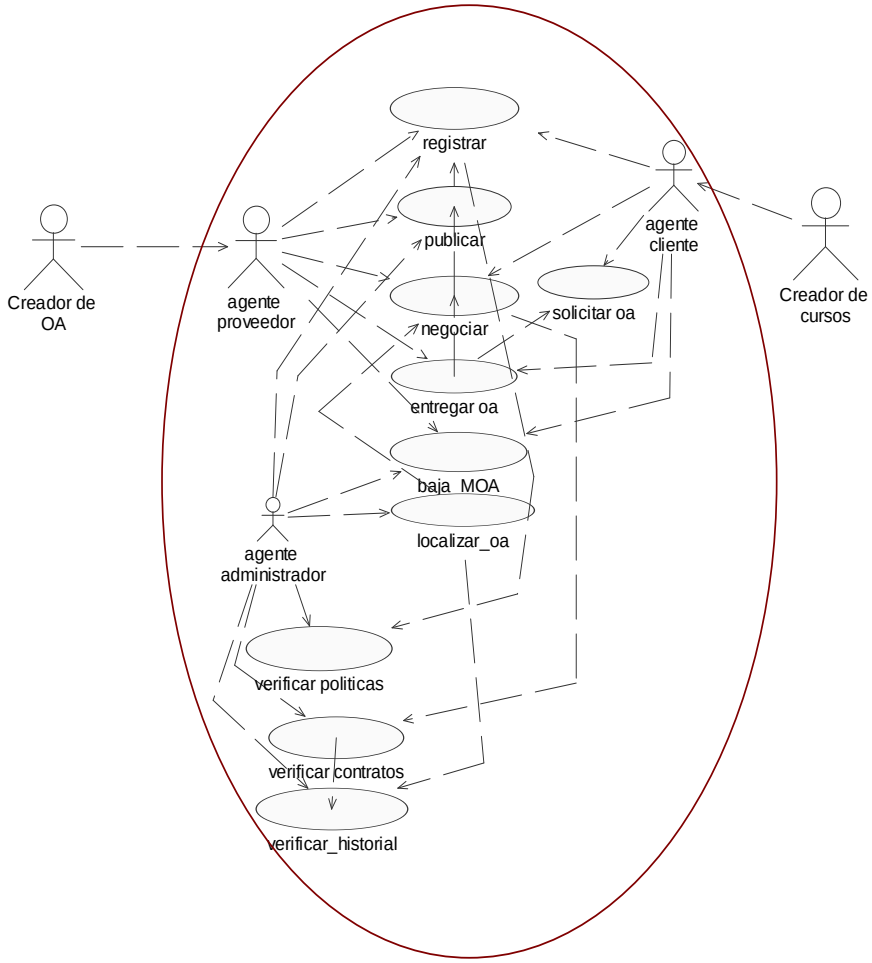

Figura 4.2: Casos de uso general del Mercado de Objetos de Aprendizaje.

Los casos de uso para el mercado de objetos de aprendizaje son los siguientes:

Caso de uso: Registrar

Actor:agente proveedor, agente administrador, agente cliente

Descripción: Este caso de uso lo requieren los actores una vez que determinen hacer una busqueda o publicación, ya que deben solicitarán ingresar al mercado, pasando por un conjunto de políticas para ser aceptados. En el caso del agente administrador

es quien va a generar un registro de ambos (consumidor y proveedor) en el mercado, asignandoles una clave única para ser identificados en cualqueir momento. Después de que se registran pueden realizar la tarea de publicitar o solicitar objetos de aprendizaje de forma correspondiente.

Caso de uso: Publicar

Actor: agente proveedor, agente administrador

Descripción: Este caso de uso se tiene en dos ámbitos, una publicacion local donde al agente vendedor determina en su repositorio cuales son los objetos que se deben publicar y los tendrá a disposición cuando un cliente se los solicite. Por otra parte el agente administrador utiliza el mismo caso de uso, pero para tener la relación de los proveedores que se publicitan el mercado, de los cuales solo tenemos referencia a los tipos de objetos que maneja cada proveedor.

Caso de uso: Negociar

Actor: agente proveedor, agente cliente

Descripción: Este servicio se habilita una vez que el agente cliente solicite un objeto de aprendizaje, entonces el agente proveedor recibe la solicitud y comienza a negociar con el cliente. Como hemos mencionado sólo lo implementaremos como una decisión *si* o *no* del cliente. (La negociación entre agentes será un trabajo futuro a implementar). Después de tomar la determinación de la negociación ambos agentes deben notificar la respuesta al agente administrador.

Caso de uso: Entregar OA

Actor: agente proveedor, agente cliente

Descripción: Se ejecuta una vez que el agente cliente firma el contrato, aceptando

todas las condiciones. Entonces el agente proveedor es quien entrega el objeto de aprendizaje con los permisos que se hayan determinado en la negociación. También el agente cliente hace uso de este servicio pero en la forma de recepción del objeto de aprendizaje, con los derechos que se le hayan delegado para utilizar el OA.

Caso de uso: Baja MOA

Actor: agente proveedor, agente cliente

Descripción: Este servicio se ejecuta cuando alguno de los actores determina darse de baja del mercado, es decir suspender su servicio de publicitar o solicitar objetos de aprendizaje, lo cual permite tener el mercado de forma actualizada acerca de los integrantes del mercado.

Caso de uso: Localizar OA

Actor: agente administrador, agente cliente

Descripción: Esta disponible cuando el agente cliente envía una solicitud de búsqueda al agente administrador para entrar el mercado. El adminstrador es quien procesa la búsqueda en el mercado y le devuelve la lista de proveedores que tienen de acuerdo al tipo de objeto de aprendizaje especificado .

Caso de uso: Verificar politicas

Actor: agente administrador

Descripción: Este servicio esta disponible cuando un agente cliente o proveedor quiere ingresar al mercado, ya que el administrador necesita saber si este es un miembro que cumpla con cierta especificaciones, como la institución, la clave de acceso que tiene, la facultad, etc. Una vez que el agente cliente o proveedor cumpla con estas políticas entonces puede ingresar al mercado. Es importante mencionar que estas políticas se

van a definir de forma más completa para que los agentes puedan determinar de forma automatica como y quien ingresa al mercado, este tema ya esta integrado como otro trabajo de tesis.

Caso de uso: Verificar contratos

Actor: agente administrador

Descripción: Este caso se dispondrá una vez que ambos hayan firmado el contrato. El agente administrador se encargará de verificar si el cliente paga el servicio y si el proveedor obtiene el objeto de aprendizaje de acuerdo a las especificaciones del contrato, es decir el administrador verifica si ambos participantes han cumplido con sus contratos en el mercado.

Caso de uso: Solicitar OA

Actor: agente consumidor

Descripción: Se llamará a este caso de uso cuando un agente cliente solicite un objeto de aprendizaje de forma directa con el agente proveedor, es decir el cliente ya sabe con quien se va a comunicar para obtener un OA. Este proceso se da después del caso de uso de localizar OA.

Caso de uso: Verificar historial

Actor: agente administrador

Descripción: Este caso de uso se ejecuta cuando se le pide información al agente administrador sobre algún cliente o proveedor. También se activa antes de verificar contratos, ya que primero tiene que revisar el historial de ambos participantes ejecutando el método *verifica historial()*, es decir si ambos son responsables en el cumplimiento de las transacciones que realizaron en el mercado.

4.3. Diagramas de Clases General

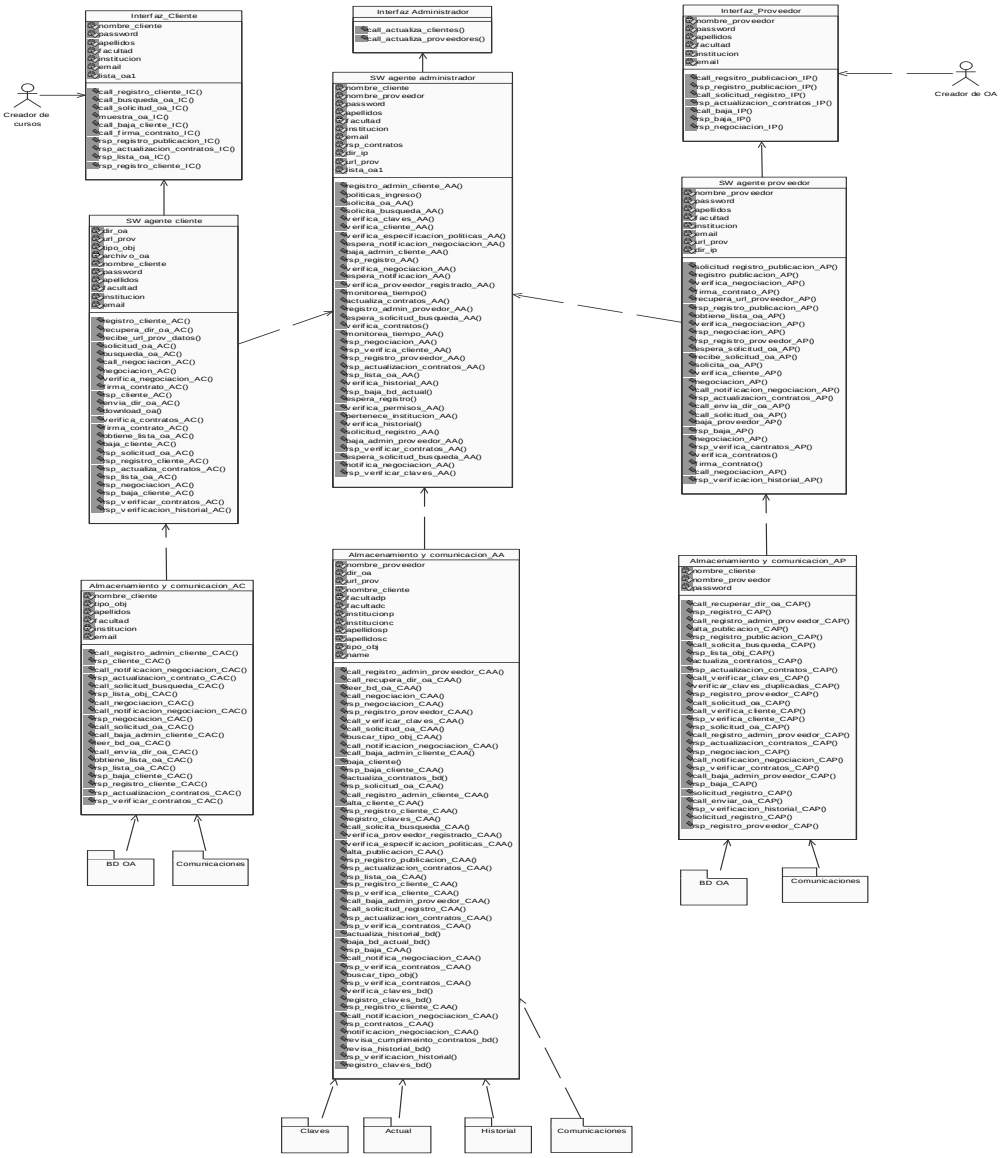

Figura 4.3: Esquema General del Mercado de Objetos de Aprendizaje

A continuación se presentan las clases que definen las tareas que se plantearon inicialmente para cada agente, además de las bases de datos que manejan para poder manipular la información de los agentes, usuarios y objetos de aprendizaje. El diagrama de clases general se muestra en la figura 4.3.

A continuación describiremos los tres componentes principales del sistema: Diagrama general del agente proveedor, diagrama general del agente cliente y el diagrama general del agente administrador.

En este diagrama se muestran las clases principales de acuerdo al modelo de capas en la que se muestran los niveles de interfaz, lógica y comunicaciones. En la parte superior se muestran las clases que representan la interfaz de los agentes, en la parte intermedia se tienen las clases encapsuladas en servicios web que contienen todas las operaciones que realiza cada agente, y por ultimo las bases de datos que manejan los diferentes componentes de software. Es importante mencionar que las comunicaciones que se muestran en este diagrama permite visualizar como se realiza la interacción entre las clases que se encuentran de forma distribuida, que es el planteamiento inicial de nuestro mercado de objetos de aprendizaje.

Descripción del diagrama de clases general del agente proveedor.

El creador de objetos de aprendizaje desea publicitar objetos de aprendizaje, esta tarea se realiza por medio de la clase interfaz que tiene la opción de registrase, publicarse y darse de baja en el mercado a través de los métodos *call registro publicacion()*, *call solicitud registro()*, *call baja IP()*.

La clase SW agente provedor es un servicio web en el que se encuentran encapsulados los métodos que realizan las tareas del agente el método *registro cliente AP()* tiene como atributos el *nombre*, *apellidos*, *facultad* e *institucion* del proveedor. El método *recupera url proveedor()* se encarga de identificar el URL del servidor del proveedor y lo guarda para que puedan comunicarse con él. Para solicitar la publicación de un *tipo de objeto* de aprendizaje por ejemplo de matemáticas, fisica, biología, etc. tiene que utilizar el método *solicita registro proveedor AP()* que se encarga de dar de alta la publicidad del proveedor y su correspondiente

tipo de OA lo cual se envía al agente administrador. Después, el agente proveedor queda en espera de la actualización de la lista local de objetos de aprendizaje y lo hace mediante el método *obtiene lista oa AP()*. Entonces el agente proveedor queda en espera de una solicitud de OA el cual se realiza por medio del método *espera solicitud oa AP()* que esta monitoreando cada vez que le llegue una petición. Después de recibir la solicitud el agente proveedor tiene que ejecutar el método *verificar cliente AP()* que le permita determinar por medio del agente administrador si es un cliente válido en el mercado, este método tiene como atributos el nombre del cliente que envía la solicitud. Para negociar se ejecuta el método *verifica negociacion()* que se encarga de recibir la respuesta que le envía el cliente para compararla y aceptarla o rechazarla. Después se ejecuta el método de *firma contrato AP()* en el cual acepta el contrato que determinó el cliente. El método de *envíar dir oa AP()* se ejecuta para obtener la direccion del OA de forma remota y envíarla al cliente para que sea accesado de forma remota. En el caso de que el usuario quiera darse de baja del mercado tiene que ejecutar el método *baja proveedor AP()* y que tiene como atributo el nombre del proveedor y el tipo de objeto de aprendizaje que publicito, el agente se da de baja de la base de datos actual en la que se encuentran todos los proveedores.

Después de realizar estas tareas, efectua a su vez transacciones que envían las peticiones y reciben los datos por medio del protocolo TCP/IP y permite dirigir la comunicación a través de Internet. También efetua el acceso a las bases de datos de objetos de aprendizaje que pertenece al proveedor.

Descripción del diagrama de clases general del agente cliente.

El creador de cursos necesita un conjunto de objetos de aprendizaje, de tal manera que para lograr su propósito tiene que solicitar un objeto de aprendizaje en el

mercado de objetos de aprendizaje. El usuario realiza la solicitud mediante la interfaz del componente del cliente del mercado por medio de los métodos *call registro cliente IC()*, *call busqueda oa IC()*, *call solicitud oa IC()*, *muestra oa IC()*, por otra parte el cliente puede darse de baja mediante el método *call baja cliente IC()*.

La clase del agente cliente se encapsula en un servicio web y en nuestro diagrama se denota como SW agente cliente que contiene los diferentes métodos que debe realizar el agente. El primer método *registro cliente()* permite envíar los atributos con los cuales se va a dar de alta como nombre del usuario, *apellidos, facultad, institucion*, entre otros. El método *recupera dir oa IC()* atributo el URL del proveedor y retorna la dirección del objeto de aprendizaje de forma remota. El método *recibe url prov datos AP()* se encarga de obtener el URL del proveedor que le envía el agente administrador. El método *busqueda oa AC()* tiene como atributos el nombre del usuario, el tipo de objeto de aprendizaje que solicita, además se encarga de envíar la solicitud al agente administrador. El método *solicitud oa AC()* tiene como atributos el nombre del agente cliente y el tipo de objeto que solicita, esto con el fin de interactuar con el agente proveedor para obtener la lista de objetos que tiene. Después de hacer la solicitud se tiene que hacer la negociacion mediante el método *negociacion AC()* en este caso el cliente determina aceptar un contrato escrito con un simple *si* o *no* que permite por el momento realizar la negociación, y envía esta respuesta al agente proveedor. Después el metodo *verifica negociacion AC()* se encarga de tomar la respuesta que envía el agente proveedor y la compara con la que él realizó, si ambas son positivas continua con el método *firma contrato AC()* que retorna un mensaje de negociación exitosa al usuario. El método *download oa()* tiene como atributos el

nombre del objeto de aprendizaje, la dirección remota del objeto, y el URL del proveedor.

Además, en el componente del agente cliente existe una base de datos de objetos de aprendizaje que va adquiriendo en el mercado, así como el paquete de comunicaciones TCP/IP para las interacciones de forma remota y distribuida entre los agentes.

Descripción del diagrama de clases general del agente administrador.

Para realizar el componente que permita ver los cambios que se representen en el mercado de objetos de aprendizaje, se creo el agente administrador de mercado que tiene como tarea el monitorear los cambios que se realizen dentro del mercado. Esta tarea se muestra por medio de una interfaz de usuario que tiene tres tablas en las que se despliegan cuando se ejecutan los métodos *call actualiza clientes()* y *call actualiza proveedores()*, de esta manera presenta lás transacciones que se realizan en el mercado.

Para efectuar la lógica del agente administrador se tiene la clase SW agente administrador que contiene una gran cantidad de tareas que van a permitir la administración y monitoreo del mercado. Iniciamos con el método *registro admin cliente AC()* que tiene los atributos de identificación del cliente como *nombre, apellidos, institucion, facultad*, etc., otro método es *politicas ingreso()* que tiene como atributos el *nombre* y *contraseña* del cliente que determina si el cliente es válido en el mercado. Otro aspecto que se toma encuenta es la institucion de la que proviene, para verificar si pertenece a las instituciones miembros ya determinados previamente por los interesados del proyecto.

El método *espera notificacion negociacion()* se encarga de monitorear alguna solicitud de un cliente o proveedor. El método *verifica negociacion AA()* se en-

carga de comparar las respuestas de ambos cliente y proveedor, para esto tiene como parametros de entrada los nombres de los agentes que representan a los usuarios. Para el método *espera solicitud busqueda AA()* el agente administrador ser encarga de monitorear las solicitudes que lleguen de cualquier cliente con el fin de ejecutarlas en su base de datos de proveedores por medio del campo *tipo de objetos*, los atributos son el nombre del cliente y el tipo de objeto que busca. En el método *verifica historial()* el administrador se encarga de revisar si el par cliente-proveedor que realizan una negociación, son participantes que han cumplido en el mercado, entonces los atributos del método anterior son el nombre del cliente y el nombre del proveedor. Después de que se notifique una negociación, el agente administrador tiene que revisar el cumplimiento de los contratos, y lo hace mediante el método *monitorea tiempo()*, después de verificar si se cumple o no los contratos se ejecuta *actualiza historial AA()* y tiene como atributos el nombre del cliente y proveedor, y las respuestas de ambos para registrarlas en la base de datos historial. Por último, el administrador se encarga de eliminar clientes y proveedores, en el caso de eliminar un proveedor se ejecuta el método *baja admin proveedor AA()* que tiene como atributos el nombre del proveedor y lo elimina de la base de datos actual que contiene todos los miembros del mercado. Además, se tienen dos tablas anexas que contienen información de forma separada a clientes y proveedores para realizar las búsquedas de forma rápida.

4.4. Diagramas de Secuencia

Las tareas que tiene cada uno de los participantes, proveedor, consumidor y dueño del mercado que se mencionaron anteriormente, ahora las mostraremos a través de los diagramas de secuencia. Estos diagramas son representados en UML, y denotan los

mensajes mediante los cuales se comunican los agentes, y las tareas que realizan.

Los diagramas se describen en el siguiente orden:

1. Diagrama de secuencia registro (cliente)
2. Diagrama de secuencia verificar contratos
3. Diagrama de secuencia verifica historial
4. Diagrama de secuencia baja MOA (cliente)
5. Diagrama de secuencia localizar oa
6. Diagrama de secuencia negociar

Realizando una mapeo de los elementos en la AOS, cada proveedor, comprador y administrador del mercado, será representado por medio de un agente, donde cada agente tiene tareas específicas como se mencionaron en la sección 4.1.

1. El *diagrama de secuencia de registrar para clientes* se muestra en la figura 4.4. Este diagrama representa como un cliente puede registrarse en el mercado, donde el usuario da su *nombre*, *apellidos*, *facultad*, *instituto*, y *contraseña* a través de la clase *Interfaz Cliente()*, después el agente cliente envía la solicitud de registro mediante el método *registro cliente AC()*, cuando el administrador recibe la solicitud de

   registro la procesa, es decir al agente administrador verifica si el cliente cumple con las políticas del mercado mediante el método *politicas ingreso()*, despues verifica si el cliente no tiene historial previo deficiente ejecutando el método *verifica historial()* y por último el agente administrador verifica si el nombre y contraseña

son válidos, si se obtiene una respuesta afirmativa, entonces se da de alta al cliente en la base de datos claves donde solo se tiene los datos generales de todos los integrantes de mercado, y por otra parte se anexan los datos del cliente en una tabla exclusiva para clientes del mercado de objetos de aprendizaje. Finalmente el agente administrador envía la respuesta del registro de regreso mediante el método *rsp registro cliente CAA()* al agente cliente quien la recibe y la muestra en la interfaz del cliente con el método *rsp registro cliente IC()*, entonces el cliente puede ver si su solicitud de registro fue aceptada por el administrador del mercado.

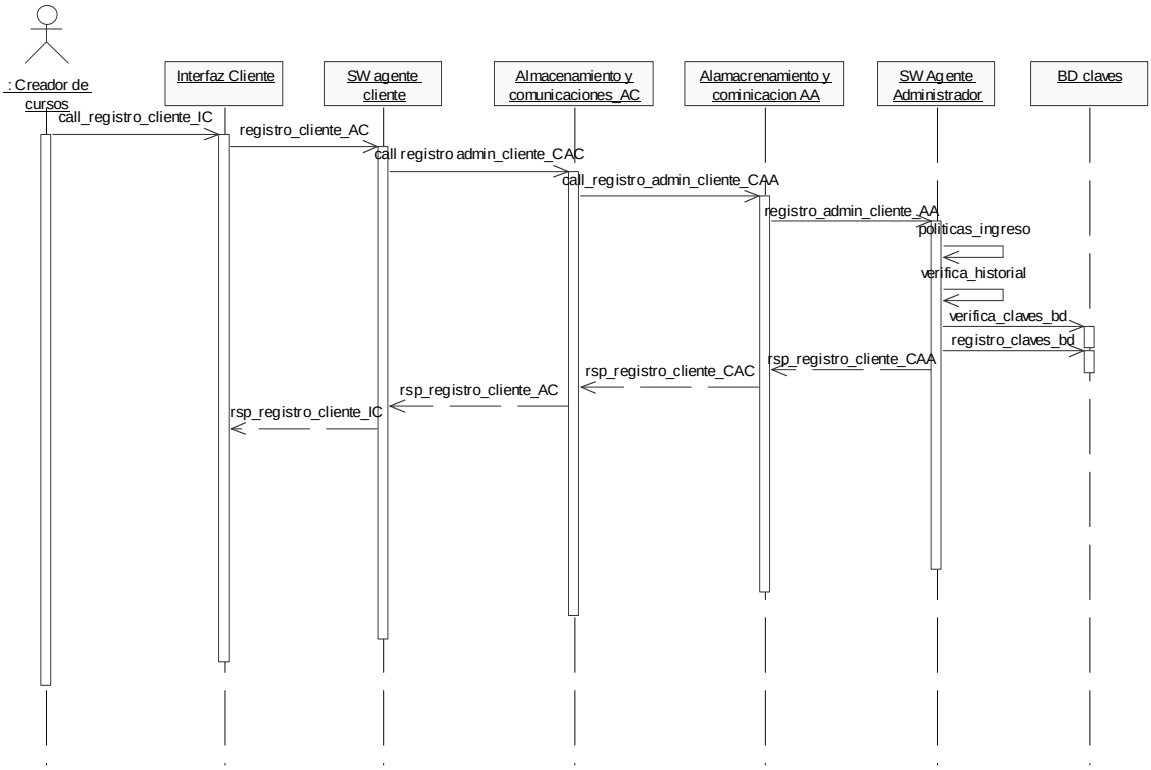

Figura 4.4: Diagrama de Secuencia de registro de clientes en el MOA

2. En el *diagrama de secuencia verificar contratos* se muestra en la figura 4.5, donde tanto el agente cliente o el agente proveedor pueden solicitar este caso de uso. Inicia

el agente administrador con el método *espera notificacion negociacion()* con la espera de una notificación del termino de una negociación por cualquier integrante ya sea cliente o proveedor. Cuando le llegan las notificaciones al administrador las revisa y se verifican los tipos de contratos que cada participante (cliente o proveedor) acepta. Después, el administrador monitorea el cumplimiento de los contratos por ambas partes cada cierto tiempo mediante el método *monitorea tiempo()* donde se tiene que verificar que ambos esten cumpliendo con su contrato de permiso sobre los objetos de aprendizaje. Si existe alguna nueva negociación el agente administrador actualiza la base de datos del historial mediante el método *actualizacion contratos bd()* donde se anexan los tipos de contratos aceptados por cada usuario, por último el agente administrador envía la respuesta de actualizar contratos al agente cliente o proveedor de forma correspondiente ejecutandose *rsp actualizacion contratos CAA()*.

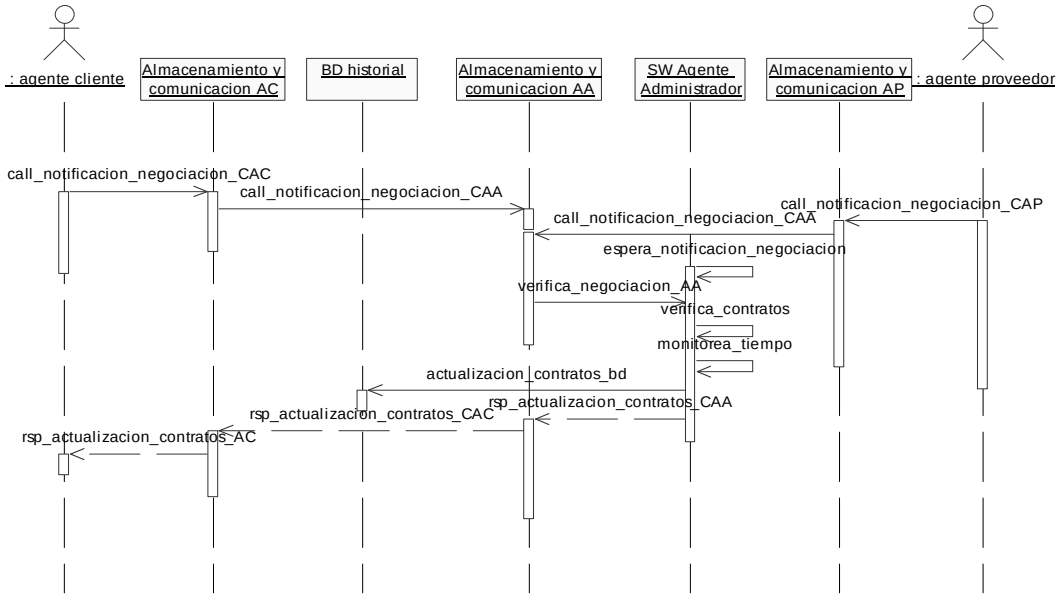

Figura 4.5: Diagrama de Secuencia de verificar contratos en el MOA

3. El *diagrama de secuencia verificar historial* se muestra en la figura 4.6 donde

quien inicia la secuencia es el agente adminstrador quien espera una notificación de negociación por un agente cliente y un agente proveedor con el método *espera notificacion AA()*. Después de que llegan ambas notificaciones el agente administrador se encarga de revisar el cumplimiento de contratos por medio del método *revisar cumplimiento contratos()* de acuerdo a lo que se haya acordado entre ambas partes. Además, el administrador verifica el historial de cada usuario con respecto a las transacciones que se han realizado en el mercado *ejecutando revisa contratos bd()*. Por último el agente administrador envía una respuesta sobre el historial de cada usuario (proveedor o cliente) respectivamente ejecutando *rsp verificacion historial()*.

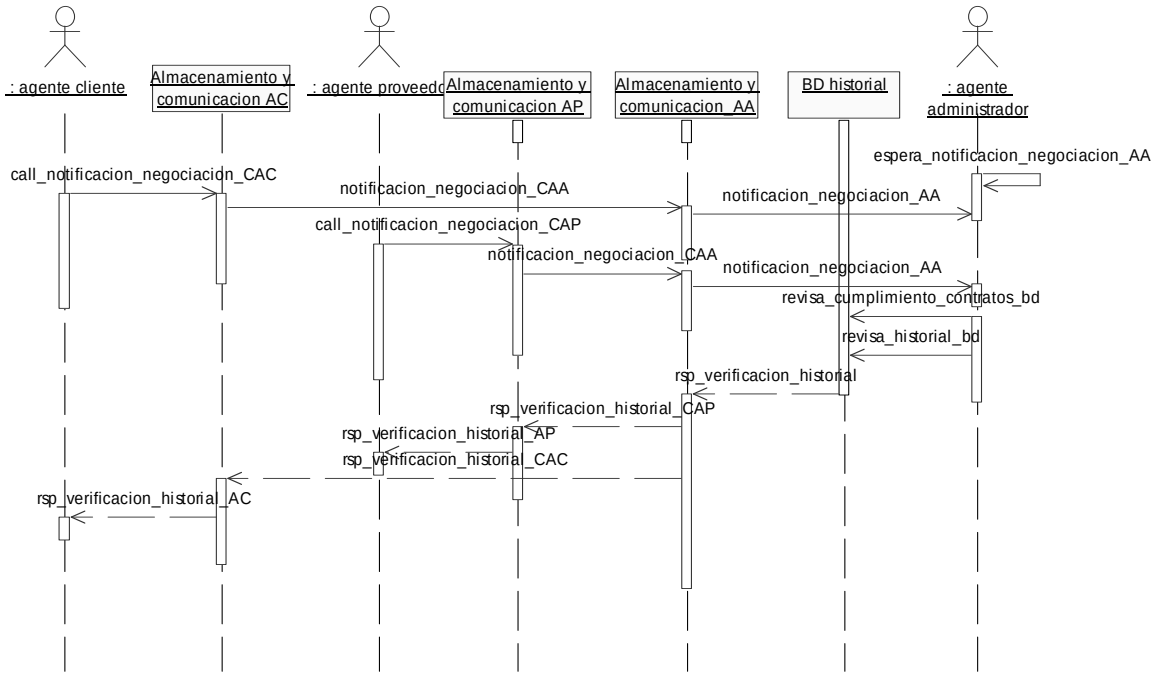

Figura 4.6: Diagrama de Secuencia verificar historial de clientes y proveedores en el MOA

4. El *diagrama de secuencia baja MOA (cliente)* se muestra en la figura 4.7 la interacción se inicia cuando el usuario determina darse de baja del mercado de objetos

de aprendizaje e inicia dando sus datos como nombre y contraseña en la interfaz de usuario con el método *call baja cliente IC()*. Después el agente cliente se encarga de procesar esta solicitud ejecutando *call baja admin cliente AA()* y envía la petición al agente administrador. Por otro lado, el agente administrador se encarga de realizar la eliminación del cliente de la base de datos actual en la tabla correspondiente de clientes mediante el método *baja cliente bd actual()*, también actualiza la base de datos historial donde se realizan los cambios que se hasta el momento ha realizado el cliente ejecutando el método *actualiza historial bd actual()*. Terminando este proceso con la respuesta de la baja del cliente de forma exitosa, o si existe algún problema o inconveniente con algún contrato que no se haya terminado hasta el momento mediante el método rsp baja cliente CAA().

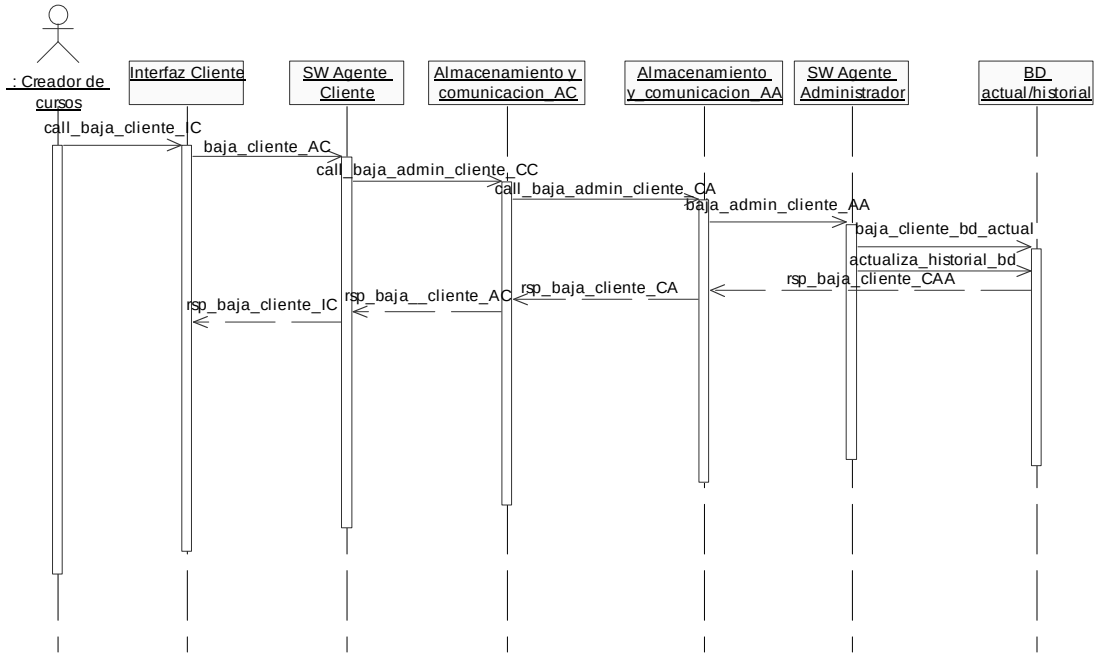

Figura 4.7: Diagrama de Secuencia de baja de clientes en el MOA

5. El *diagrama de secuencia localizar oa* se muestra en la figura 4.8 donde el creador de cursos inicializa la petición de búsqueda de un tipo de objeto de aprendizaje

mediante el método *call busqueda oa AC()* que tiene como atributos el nombre del cliente y el tipo de objeto de aprendizaje. Después el agente cliente estructura la solicitud para envíarla al agente administrador mediante el método *busqueda oa AC()* que tiene los mismos atributos mencionados anteriormente. El agente administrador esta en espera de una solicitud de tipo de objeto de aprendizaje por medio del método *espera solicitud busqueda()*, después de que llega la solicitud al agente administrador se ejecuta el método *busca tipo obj()* que se encarga de realizar la busqueda en la base de datos actual que contiene la tabla de proveedores con sus respectivos tipos de objetos que publicitan. Después de obtener el conjunto de proveedores que tienen los tipos de objetos de acuerdo con la solicitud, el agente administrador retorna un *DataSet* al agente cliente a través del método *rsp lista oa AA()*.

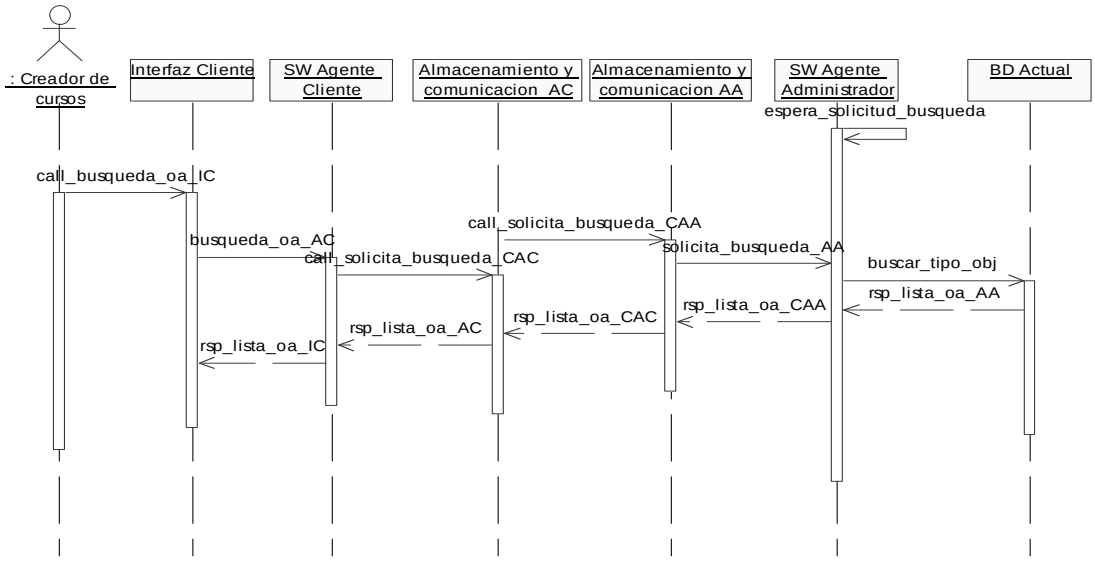

Figura 4.8: Diagrama de Secuencia para localizar un objeto de aprendizaje (cliente-administrador)

6. El *diagrama de secuencia de negociacion* que se muestra en la figura 4.9, como

puede ver el agente cliente es quien inicia la solicitud de negociación a través del método *call negociación CAC()* que tiene como atributos el nombre del cliente y URL del proveedor con el que va a comenzar la negociación. Después el agente cliente envía al agente proveedor a través del método *negociacion()* su petición para comenzar la negociación.

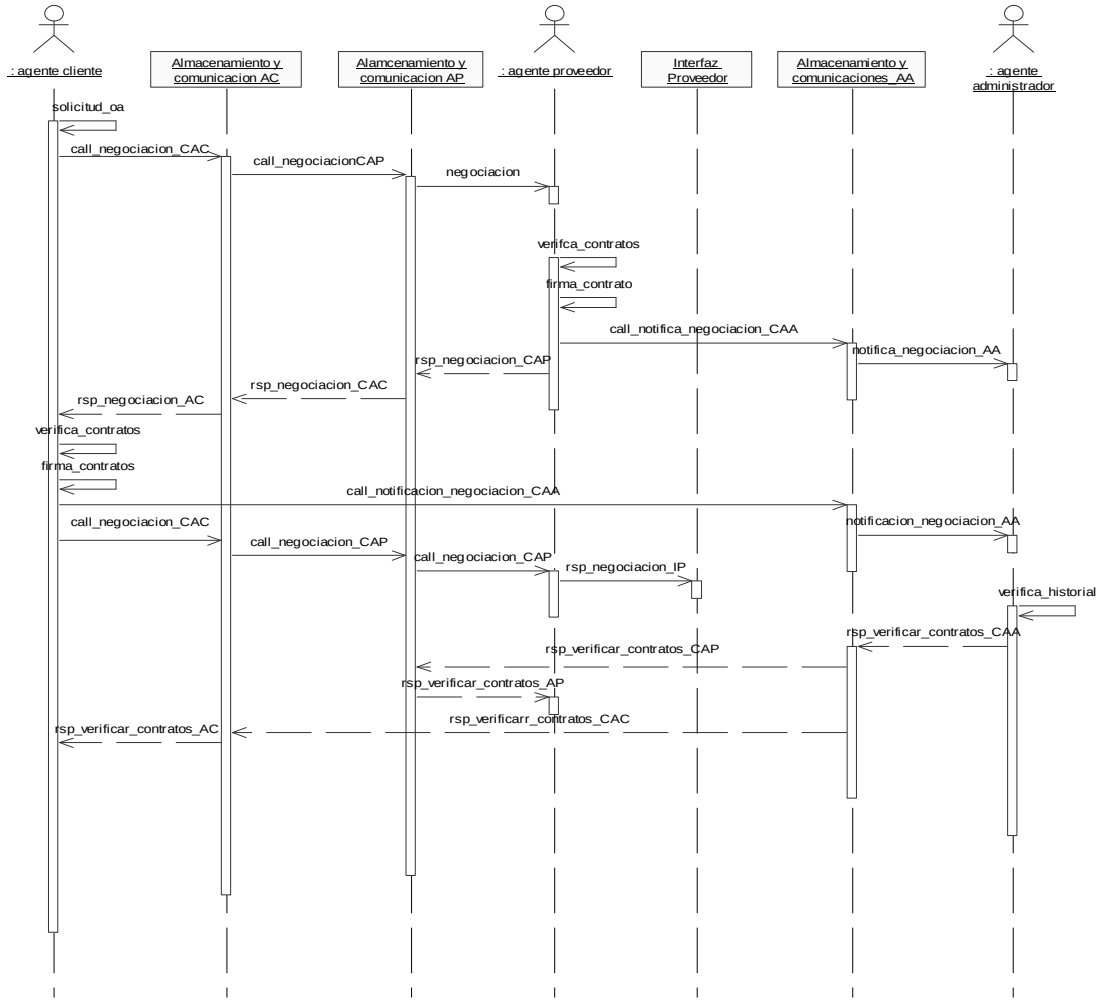

Figura 4.9: Diagrama de Secuencia para la negociacion entre agentes

El agente proveedor se encarga de recibir el contrato que envía el cliente mediante el método *verifica contrato()* que tiene como atributos el nombre del

cliente, el URL, y el contrato que se desea manejar, si este es viable entonces acepta el contrato ejecutando el método *firma contrato()* donde se da una respuesta positiva o el rechazo del mismo. Después de que el agente proveedor acepta el contrato tiene que avisarle al agente administrador del mercado la nueva negociación establecida mediante el método *call notifica negociacion CAA()*.

Por otra parte el agente cliente recibe la respuesta de la negociacion que le envía el agente proveedor mediante el método *rsp negociacion CAP()* teniendo como atributos el contrato, la respuesta del contrato y el URL del cliente a quien va dirigido, después el agente cliente revisa el contrato y decide aceptarlo a rechazarlo mediante el método *firma contrato()*. Al terminar el proceso de negociación el agente cliente le envía al agente administrador la respuesta de la negociación a través del método *call notificación negociacion()* que tiene un resultado verdadero o falso. Después el agente administrador revisa el historial de cumplimiento de contratos de ambos usuarios a través del método *verifica historial()* y envía la respuesta de los antecedentes de cada usuario al agente correspondiente mediante el método *rsp verificar contratos()*. Cuando los agentes reciben la respuesta pueden determinar volver a negociar de acuerdo al historial obtenido.

Se enfatiza que el manejar el esquema de mercado, de alguna forma permite garantizar las transacciones para la adquisición de objetos de aprendizaje, teniendo una lista de compradores y vendedores que entren al mercado y realizando un seguimiento del cumpliento del contrato. En base a la AOS se puede ver que es posible generar el mercado de forma que se tengan varios repositorios de objetos de aprendizaje de forma distribuida. Como trabajo futuro se pueden realizar el proceso de negociación, además la inferencia entre agentes permite realizar las transacciones de forma directa para la

negociación entre agentes tomando en cuenta los intereses particulares de cada quien.

Capítulo 5

Prototipo del MOA

El objetivo de esta tesis es desarrollar un prototipo de un Mercado de Objetos de Aprendizaje (MOA). En base al modelo y prototipo se podrá en un futuro implementar un sistema distribuido donde se tienen solicitantes y proveedores de objetos de aprendizaje. Estos clientes deben pasar por una entidad administrador de mercado que se encarga del registro de quienes requieran publicar o bien de localizar OA. Otro de los objetivos del administrador del mercado es tener un seguimiento de contratos que se realicen en el mercado de OA.

5.1. Herramientas para la implementación

Requisitos necesarios para el desarrollo del sistema:

- Internet Information Server(IIS)
- Visual Studio .NET 2003
- Un manejador de base de datos (Access, SQL, MySQL)
- WinXP

- Internet

En este caso se trabajó con C# de Visual Studio .NET. Se implementó la arquitectura general de mercado en base a la AOS de tal foma que los integrantes del mercado: proveedores, clientes y administrador se encuentren en diferentes sitios y permitir la funcionalidad de la aplicación de forma distribuida.

Agentes y Servicios Web

Con el fin crear el prototipo de Mercado de Objetos de Aprendizaje, se utilizaron Servicios Web que permiten una forma de comunicación dinámica a través de la red, gracias a la arquitectura en que se basan. Los servicios web alojan las tareas de correspondientes agentes proveedor, cliente y administrador. Cada uno de los agentes define de forma concreta las tareas que realizan mediante métodos web como se describieron en el Capítulo 4. El encapsular a los agentes en los SW, permitirá que se puedan acceder a sus funciones, compartir información y realizar interacciones de forma remota entre los agentes.

5.2. Desarrollo de la aplicación

Hasta hoy, se ha realizado un prototipo del mercado de objetos de aprendizaje conformado por tres componentes principales, cada uno de ellos representan a un proveedor, consumidor o administrador respectivamente. El objetivo es que el mercado de OA permita el ingreso dinámico a los usuarios, a través de los agentes que representan sus intereses particulares asignados y desde cualquier lugar. Las principales tareas del mercado son: *publicar, localizar y obtener Objetos de Aprendizaje.*

El siguiente paso es colocar los agentes encapsulados en los servicios web para interactuar a través de los estándares que facilitan el acceso a los metodos web.

Implementación del Proveedor: La figura 5.1 representa la interfaz del proveedor la cual contiene tres opciones disponibles al usuario a través de tres ligas para registrarse,ingresar y darse de baja del mercado. El agente proveedor envía la solicitud elegida y es procesada por el administrador del mercado, ya que es quien va a permitir el ingreso de algún usuario.

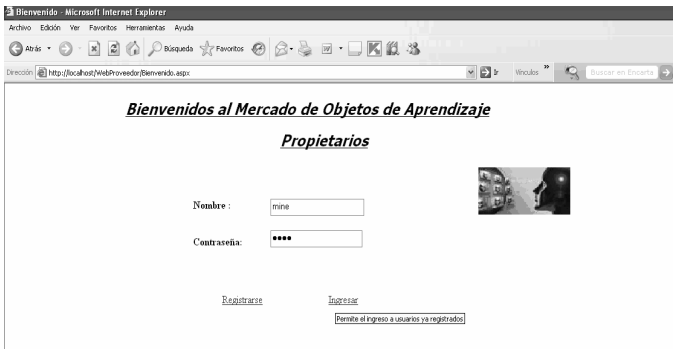

Figura 5.1: Interfaz de usuario del profesor para registrarse o ingresar en el MOA

Por otra parte, el agente proveedor es quien se encarga de procesar las actualizaciones en su repositorio local (directorio), es decir los nuevos OA que se ingresen al directorio, su modificación o eliminación. Para realizar esta verificación de cambios se utilizó el espacio de nombres *System.IO* para manipular los eventos de entrada y salida. Las actualizaciones que realize el agente de acuerdo a los cambios que se presenten en el repositorio se activarán a su vez en la base de datos de objetos de aprendizaje, después de analizar el *imsmanifest.xml* (archivo que contiene información acerca del objeto de aprendizaje) de cada OA. Es importante mencionar que el archivo xml es generado a través de una herramienta de verificación [8] acuerdo a un conjunto de estándares determinado SCORM (Sharable Content Object Reference Model). Al recorrer el árbol del archivo *imsmanifest.xml* se recolectan los datos necesarios para mostrar la especificación de cada OA como nombre, descripción, costo, y url local del OA.

Se utilizó el espacio de nombres *System.Xml*, el cual nos define clases para manipular

los elementos de un archivo .xml. El pseudocódigo se presenta en la figura 5.2, donde se realiza el recorrido del archivo de especificaciones del OA, y solo tomamos los nodos y atributos donde se tiene el nombre del objeto, su descripción y costo. Los datos que se van recolectando se almacenan en un arreglo ya que al final esta información se pasa a la tabla de objetos locales. Este proceso es muy importante ya que nos va a definir todas las características que tiene cada uno de los OA locales para cada agente proveedor.

```
XmlTextReader reader = new XmlTextReader (reader1);
while (reader.Read())
{
   switch (reader.NodeType)
    {
      case XmlNodeType.Element: // El nodo es un elemento.
            if (reader.Name=="resource")
                  while (reader.MoveToNextAttribute())
                    if(reader.Name=="href")
                     {
                        datos.Insert(1,reader.Value);
                                                              }
                        if(reader.Name=="description" & !(ban_des))
                                    { reader.Read();reader.Read();
                                      if (reader.Name=="string")
                                          {
                                                reader.Read();
                                          aux_descrip=reader.Value;
                                          datos.Insert(2,reader.Value);
                                          ban_des=true;
                                          }
                                    }
                                    if(reader.Name=="cost")
                                    {
                                    while (reader.Read())
                                          if (reader.Name=="value")
                                          {
                                                      reader.Read();
                                          datos.Insert(3,reader.Value);
                                          break;
                                          }
                                          reader.Read();
                                    }
                                    break;
                              case XmlNodeType.Text:
                                    //texto en cada elemento.
                                    break;

                              case XmlNodeType.EndElement:
                                    //Muestra el final del elemento.
                                    break;
                        }
```

Figura 5.2: Función que lee los elementos del archivo *imsmanifest.xml* de cada OA

Otro aspecto importante es saber la localización que tiene el proveedor, entonces se necesita conocer el IP del agente que esta publicando, para saber donde están los objetos de aprendizaje disponibles. El pseudocódigo para conocer el IP se muestra en la figura 5.3,donde las funciones *GetHostName* y de acuerdo al hostname se obtiene la IP de la máquina. Este información se necesita para saber la localización de cada agente y poder realizar la conexión de forma dinámica entre agentes.

Inicialmente, el agente proveedor se comunica con el administrador para registrarse

```
public string ip_address()
    {       string strHostName =  "";

            //Obteniendo el hostname de la maquina local
            strHostName = Dns.GetHostName ();
            //En base al hostname se obtiene la IP de la maquina local
            IPHostEntry ipEntry = Dns.GetHostByName (strHostName);
            IPAddress [] addr = ipEntry.AddressList;
            return IPAddress  [0];
    }
```

Figura 5.3: Método que obtiene la IP y hostname local del proveedor

haciendo una llamada al método registro que tiene el administrador del mercado.

Es importante mencionar que Visual Studio .NET, através de ADO.NET maneja el objeto *DataSet*, que es independiente las bases de datos, teniendo encuenta la escalabilidad e independencia, además del estándar XML. Por ello, sólo necesita tener acceso mediante el *DataAdapter* para poder obtener la información de la base de datos. Se puede considerar el objeto *DataSet* como un conjunto de registros que siempre está desconectado y que no sabe nada sobre el origen y el destino de los datos que contiene. Dentro de un objeto *DataSet*, al igual que dentro de una base de datos, hay tablas, columnas, relaciones, restricciones, vistas, etc.

Para entender como el *DataSet* puede obtener la información, se necesita del objeto *DataAdapter*, este objeto es quien se conecta a la base de datos para llenar el objeto *DataSet*. Aunque el objeto *DataSet* no tiene conocimiento del origen de sus datos, el proveedor tiene información necesaria y específica. La función del proveedor administrado es conectar, llenar y almacenar el objeto *DataSet* desde bases de datos. Las actualizaciones, o solicitudes se realizan mediante comandos SQL específicos por medio del comando *DataAdapte*r que proporciona el puente entre un objeto *DataSet* y la base de datos para recuperar y guardar datos. Los proveedores de datos *OLE DB* de .NET (System.Data.OleDb y System.Data.ODBC) proporcionan cuatro objetos básicos: *Command, Connection, DataReader y DataAdapter*, que se utilizaron para los

propositos siguientes:

Objetos Connection. Para conectar con una base de datos y administrar las transacciones en una base de datos.

Objetos Command. Para emitir comandos SQL a una base de datos.

Objetos DataReader. Proporcionan una forma de leer una secuencia de registros de datos, sólo hacia delante desde un origen de datos SQL Server.

Objetos DataSet. Para almacenar datos sin formato, datos XML y datos relacionales, así como para configurar el acceso remoto y programar sobre datos de este tipo.

Objetos DataAdapter. Para insertar datos en un objeto DataSet y reconciliar datos de la base de datos.

```
[WebMethod]
public bool verifica_registro_ag_prov( string nombre,string pass)
{
      if (mwadm.politicas_ingreso(nombre, pass)==true)
            return true;
      else
            return false;

[WebMethod]
public bool registro_ag_prov( string nombre,string pass)
{
      if (mwadm.politicas_ingreso(nombre, pass)==false)
            {
                        mwadm.registra_claves(nombre,pass);
                        return true;
                  }
                  else
                        return false;
            }
```

Figura 5.4: Método que registra al proveedor en el mercado

Como se puede observar en la figura 5.4 se van tomando del nombre y password que introduce el usuario. Primero verifica si cumple con las políticas de entrada, comparando el resultado que envió el administrador. Si la respuesta es verdadero entonces envía los valores obtenidos de la IU (Interfaz de Usuario), a través de la llamada al método de

registro del agente administrador para darse de alta en el mercado, por el momento el alta de proveedor que realiza el administrador se refleja en su base de datos, la cual es manejada en la programación mediante un DataSet por las ventajas mencionadas anteriormente, aunque utilizan parte de memoria residente nos conviene en tiempos de respuesta, ya que sólo accederíamos a la base de datos cuando se necesitaran realizar cambios.

En la segunda página para clientes que despliega un *listbox*, en la cual se muestran un conjunto de áreas de la educación, en las cuales se debe determinar que se encuentra el tipo de OA que desea publicar, y dar click en la liga *Publicar* para ejecutar la solicitud al administrador del mercado quien le va a permitir estar visible para cualquier cliente. Después de haber publicado, muestra los OA que tiene en ese momento. Entonces el proveedor queda en espera de una solicitud para mostrar sus objetos de aprendizaje y dar permiso de elegir uno de ellos. Para lo cual verifica mediante el agente administrador si el comprador esta registrado en el mercado, si es así, entonces envía el conjunto de objetos que tiene mediante un DataSet. Entonces el agente proveedor espera la selección de alguno de sus OA y da el permiso de acceso al OA, si previamente el cliente ha aceptado el contrato. Esto se muestra en el pseudocódigo de la figura 5.5.

```
////////////////////////////////////////////////////////////////////////////
//Método que obtiene la dirección del objeto de aprendizaje, para poder permitir el
acceso
[WebMethod]
public DataSet url_OA(string nom_objeto)
{
      OleDbConnection con =new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;"
                  + "Data Source=C:\\base_datos\\bd_local.mdb");
      con.Open();
      OleDbDataAdapter daCust = new OleDbDataAdapter("Select * From objetos_local WHERE
      nom_obj ='" + nom_objeto+"'", con);
      DataSet Dt=new DataSet();
      daCust.Fill(Dt);
      con.Close();
      return Dt;
}

//Método que verifica si el cliente tiene permiso para enviarle la //lista de sus OA
[WebMethod]
public DataSet permiso_acceso(string nom_cli)
{
      if (mwadm.verificar_cliente(nom_cli))
      {
            return proporciona_listaOA();
      }
      else
      {
            return null;
      }
}
```

Figura 5.5: Código de interacción entre la IU y el agente proveedor

Otro aspecto importante que toma en cuenta el agente proveedor es poder solicitar una lista de los clientes que se hayan contactado para realizar alguna transacción, para lo cual el administrador le muestra los contratos que ha llevado acabo.

Implementación del Cliente: A continuación se muestra en la figura 5.6 la página de la interfaz de cliente de OA. En el caso del cliente que requiere objetos de aprendizaje, tenemos en su interfaz de inicio tres ligas la primera de *Registrarse* en el caso de que el usuario no es miembro del mercado. Después de haberse registrado en el mercado, el usuario dara su nombre y contraseña para ingresar al mercado, mediante la liga *Ingresar*. Por último se encuentra la liga *Eliminar* que permite dar de baja del mercado al cliente.

Figura 5.6: Sección de la interfaz de usuario del agente consumidor, para ingresar al MOA.

En este caso tenemos un servicio web que representa al agente, donde cada uno de los métodos que se definen como WebMethod son aquellos que encapsulan las tareas del agente cliente y a su vez permite realizar las llamadas al agente administrador de forma dinámica. Algunas de las principales tareas del agente se muestran en la

figura 5.7, donde tenemos las funciones *registro ag prov()* que realiza la conexión con el administrador remotamente para poder envíar sus datos y el método *elimina ag prov()* que se comunica con el agente administrador para envíar los datos del propietario de OA que se va a suspender, y por último se muestra el método *proporciona lista OA()* que obtiene la información que tiene en su base de datos local sobre sus objetos de aprendizaje.

```
Solo registra al agente cliente con el nombre y contraseña dadas por el usuario, en
caso de que el login ya este registrado se avisa al usuario, para que introduzca otro
diferente
[WebMethod]
public bool registro_ag_cliente( string nombre,string pass)
{
  if (mwadm.politicas_ingreso(nombre, pass)==false)
  {
      mwadm.registra_claves(nombre,pass);
      return true;
  }
  else
      return false;
}


//Método web que elimina al cliente del mercado
[WebMethod]
public bool elimina_ag_cliente(string nomx)
{
//Llama al método eliminar del administrador, regresa el valor true si es
satisfactorio
      mwadm.elimina_admin_agcliente(nomx);
      return true;
}

[WebMethod]
public DataSet solicita_busqueda(string tipo)
{//Llama al metodo de buscar del administrador
      return(sw_adm.localizar_tiposOA(tipo));
}

//Recupera o actualiza la conexion con el sw del proveedor a interactuar
[WebMethod]
public DataSet recupera_url_prov_datos(string temp, string nom_cli)
{
      prov_dinamico.Url=temp;
      return conecta_prov_datos(nom_cli);
}
```

Figura 5.7: Métodos Web del agente cliente que interactúan con el agente administrador del MOA.

Para la opción de ingresar, permite pasar a otra página donde se define el tipo de OA a buscar. Entonces se realiza una solicitud de algún tipo de OA que se especifica de acuerdo a las áreas de educación por el momento definidas previamente. La solicitud es envíada al administrador para que realice la búsqueda en su repositorio de tipos de OA que tiene actualmente. Por lo tanto, el agente cliente recibe el conjunto de resultados por medio del método ya mencionado *proporciona lista OA()* que se apegan

a la solicitud como un *DataSet* que es un tipo de datos que maneja Visual Studio para manipular la información de una base de datos sin necesidad de estar accediendo a cada momento, con el inconveniente de tráfico en la red y tiempo que esto provoca. En este conjunto de datos que se le envian al cliente se encuentran el nombre del proveedor, el tipo de objetos que publica, su URL de localizacion (en este caso este URL es visible solo al agente cliente, para poder hacer la conexión con el agente proveedor (SW) de forma dinámica e interactuar con él), los otros campos son visibles para el usuario permitiendo elegir al proveedor más adecuado de acuerdo a la descripción que defina mejor sus especificaciones de búsqueda sobre el OA.

Se despliega la información a través de dos *DataGrid* que son la tablas donde se van a mostrar los resultados de la búsqueda de proveedores. La primera tabla contiene una columna que permite realizar la conexión con el proveedor que decida el usuario, dando click en la columna *comunicar*. El agente lee el URL del agente proveedor y por medio de un Proxy dinámico se conecta de forma remota con el proveedor. El Proxy que se genera permite cambiar el URL de forma dinámica dependiendo de la localización del SW, es decir redireccionando el llamado de los métodos web dependiendo de los diferentes agentes que se encuentren en cualquier lugar, parte de esta tarea se realiza en el método *recupera url prov datos()*, donde se obtiene el url del servicio web donde queremos comunicarnos y en la instrucción *prov dinamico.Url=temp;* se hace la comunicación dinámicamente a otro servicio web (agente encapsulado). Es aquí donde comienza la transacción de forma directa entre cliente y proveedor (ambos ya registrados en el mercado de OA). Para que ésta comunicación se realice de forma transparente, se tienen que definir los servicios web que se deben utilizar en tiempo de diseño, en este caso se define un servicio web para tener comunicación con el agente administrador del que sabemos su localización. Pero en el caso del agente proveedor no sabemos donde se va a localizar en el momento de diseñar la aplicación, lo que se hace es un servicio web

dinámico que permita enviar el URL en tiempo de ejecución. La forma en que se crea el servicio web permitirá comunicarse con cualquier proveedor que se encuentre dado de alta en el mercado, siempre que sea definido su URL de servicio web (esta información solo es vista y manipulada entre agente, no es disponible al usuario). En la figura 5.8 se muestra como se agregaban los servicios web del administrador y del proveedor en tiempo de diseño.

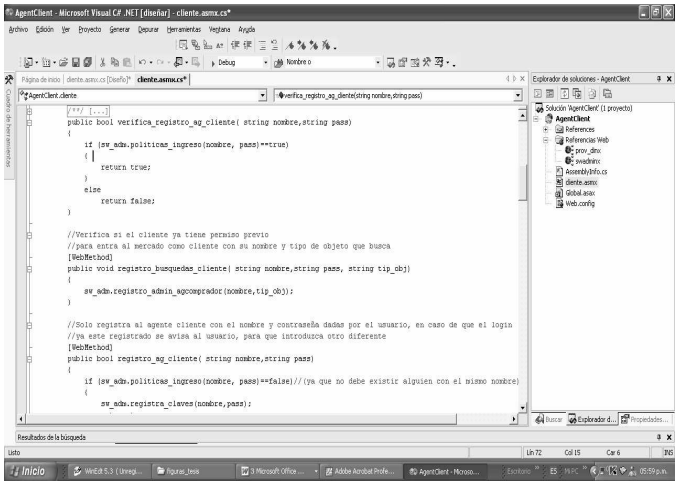

Figura 5.8: Muestra la forma como se tienen definidos los SW en tiempo de diseño para el cliente.

De forma predeterminada se baja al directorio c:/objetos/. A demás los objetos que son importados son .zip ya que son el formato que se requiere para ser integrados en el sistema administrador de aprendizaje (MOODLE por el momento), y crear sus cursos en línea, pero pueden ser transportados cualquier tipo de archivos.

Como se muestra en la figura 5.8, el agente cliente tiene que recibir información de la interfaz del cliente, del administrador y del proveedor, en donde los dos últimos se encuentran de forma remota y deben ser de cierta forma anexados para su interacción en tiempo de diseño. Parte del código de interacción entre la IU y el agente se muestra la

figura 5.9. Como se puede observar, se despliega primero un *DataSet* que es el objeto que nos va a permitir desplegar el resultado de los proveedores que retorna el administrador en caso de que haya un conjunto de resultados, en caso contrario no se despliegan resultados en la tabla. Una vez que el usuario hace un evento de selección sobre el botón comunicar perteneciente a un proveedor, este evento es atendido por el agente cliente que se comunica con el agente proveedor mediante el método de *permiso de acceso()*, devolviendo la información de los objetos de aprendizaje.

```
if (( cliente_bus1.solicita_busqueda(c_tipo.Text) )!= null)
            {
            //Solicita la busqueda y retorna el conjunto de datos de respuesta
                  DataSet vista_sw=cliente_bus1.solicita_busqueda(c_tipo.Text);
                  DataView vista= new DataView(vista_sw.Tables[0]);
                  //se muestra la informacion que regresa el SW de consultar
                  DG_obj.DataSource=vista_sw;
                  DG_obj.DataBind();
            }
            else
            {
                  Respuesta.Text="No se encontraron objetos";
            }

//Selecciona un proveedor de objetos
private void DG_obj_SelectedIndexChanged(object sender, System.EventArgs e)
      {
            nom_bd=DG_obj.DataKeys[DG_obj.SelectedIndex].ToString();
            muestra_objetos_proveedor1(nom_bd);
      }

//Muestra los objetos en la IU
private void muestra_objetos_proveedor1()
            {    //Retorna los OA del proveedor elegido
                  DataSet vista_obj=prov_obj1.permiso_acceso(c_nombre.Text);
                  DataView vista1= new DataView(vista_obj.Tables[0]);
                  DTLista.DataSource=vista_obj;
                  DTLista.DataBind();
            }
```

Figura 5.9: Código de interacción entre el agente cliente y la IU para búsqueda o selección de un OA

Después, el usuario lee el contrato y determina aceptarlo o no, si el usuario lo acepta, el agente proveedor accesa al OA, mediante el código de la figura 5.10.

Una vez efectuado esto se notifica al agente de la transacción realizada y se envía la información necesaria de ambos, para ser registrada por el agente administrador.

Implementación del Administrador: Por último, se creó el agente administrador con una página ASP como IU, que contiene la información que se va actualizando en el mercado, mostrando tres tablas con los datos de proveedores, compradores y las transacciones que se han efectuado. El agente administrador tiene que actualizar la

```
//Si selecciona el botón aceptar contrato entonces accede al OA
private void acepto_Click(object sender, System.EventArgs e)
{
      download_file(url_prov,aux_ach);
      estado_contrato=cliente_bus1.proceso_contrato("Terminado");
      cliente_bus1.registra_CV(c_nombre.Text,nom_vend,estado_contrato);
      Label4.Visible=true;
}


//Si selecciona no, entonces no puede acceder al OA
private void no_Click(object sender, System.EventArgs e)
{
      Label4.Text="No tienes permiso de Descargar el archivo";
}

//Método que obtiene el OA
public void download_file(string url,string arch1)
      {
                  string dir_local=@"C:\objetos\"+arch1;
                  try
                  {
                        cliente_bus1.negociacion();

                        WebClient client_web = new WebClient();
                        client_web.DownloadFile(url,dir_local);

                  }
            }
```

Figura 5.10: Código de accesar objeto, y registrar compra en al agente consumidor

información cada que llegue un nuevo agente al mercado, o bien cada que se este realizando una transacción entre un par proveedor-cliente para ver los cambios que se vayan dando se puede dar click en la liga *Actualizar*.

El agente administrador es quien se encarga de verificar si un agente es válido en el mercado, es decir, si tiene permiso de ingresar al mercado, por el momento este permiso depende si ya existe el nombre de usuario y la clave. También, el agente se dedica a dar de alta los agentes, para lo cual utiliza una base de datos en la cual tiene una tabla para los clientes, y una para los proveedores, en las cuales se reflejan las altas y bajas de ambos actores. Entonces, los datos a publicitar a los proveedores y los clientes, son ingresados en la tabla correspondiente de la base del agente administrador. Para mostrar los cambios en su IU, tiene que conectarse a su base de datos para mostrar los datos, esto lo hace mediante un código similar al de la figura 5.11.

Por lo tanto, todas las acciones que se realicen en el mercado, el agente administrador las va guardando en una base de datos, la cual es accesible desde los métodos web que contienen las tareas del agente para insertar, modificar o eliminar algún elemento de la base de datos. En el caso de la búsqueda, es importante mencionar que por el momento

```
//Se conecta a su base de datos donde obtiene los datos registrados en sus
tablas de vendedores, compradores y compra-venta, para mostrarlas en los
DataSet corespondientes.

Private void actualiza_bd_Click(object sender, System.EventArgs e)
{
string origen_datos="Data Source=C:\\base_datos\\bd_prueba.mdb";
string tabla="ag_vendedores";

   /**VENDEDORES*/
   DataSet x_ven=admin_grl.Leer_datos_bd_local_admin(origen_datos,tabla);
   DataView vista_v= new DataView(x_ven.Tables[0]);
   dg_vendedore.DataSource= x_ven;
   dg_vendedore.DataBind();


   //*COMPRA_VENTA**/
   tabla="compra_venta";
   DataSet x_cv=admin_grl.Leer_datos_bd_local_admin(origen_datos,tabla);
   DataView vista_cv= new DataView(x_cv.Tables[0]);
   dt_com_ven.DataSource= x_cv;
   dt_com_ven.DataBind();

   //**COMPRADORES**/
   tabla="ag_compradores";
   DataSet x_set=admin_grl.Leer_datos_bd_local_admin(origen_datos,tabla);
   DataView vista_ven= new DataView(x_set.Tables[0]);
   dt_clientes.DataSource= x_set;
   }
```

Figura 5.11: Métodos del interacción entre la IU del mercado y el agente administrador

solo se genera a través de una instrucción SELECT de SQL para obtener el resultado de acuerdo a la especificación del campo *tipo de objeto*. Para realizar la eliminación de un agente ya sea proveedor o cliente solo se utiliza la instrucción DELETE en la cual se especifica el nombre y contraseña para dar de baja al agente del mercado.

A continuación sólo mostraremos algunos métodos que presta el administrador a través de su servicio web principal. Estos métodos de búsqueda, publicación y registro de transacción se muestran en las figuras 5.12 y 5.13. Otros más, por espacio se anexarán posteriormente.

```
/*********************************************************************/
 METODO QUE DEVUELVE UNA CONSULTA/BUSQUEDA ESPECIFICADA EN: obj_bus
/*********************************************************************/
public DataSet Consultar_Click(string obj_bus)
{     OleDbConnection con =new
                  OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;"
                        + "Data Source=C:\\base_datos\\bd_prueba.mdb");
      con.Open();
      OleDbDataAdapter daCust = new OleDbDataAdapter("Select * From
                  ag_vendedores WHERE tipo_obj = '" + obj_bus + "'
                  ORDER BY ID " , con);

      DataSet Dt=new DataSet();
      daCust.Fill(Dt,"tipos");
      con.Close();
      return Dt;
}
```

Figura 5.12: Método buscar del agente administrador del MOA

```
/*********************************************************************/
//METODO QUE PERMITE EL REGISTRO DE LOS CLIENTES Y COMPRADORES EN LA
TABLA DE COMPRA-VENTA
/*********************************************************************/
public DataSet registra_compra_venta(string id, string ag_ven, string
ag_comp)
{     DataSet Dt=new DataSet();
      OleDbConnection con =new
                  OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;"
                        + "Data Source=C:\\base_datos\\bd_prueba.mdb");
      con.Open();
      OleDbDataAdapter daCust = new OleDbDataAdapter("INSERT INTO
                  compra_venta (ID,AG_VEN,AG_CLIENTE) VALUES ('" +id+
                  "','"+ag_ven+"','"+ag_comp+"' );",con);
      OleDbCommandBuilder cbCust = new OleDbCommandBuilder(daCust);
      DataTable dtTest = new DataTable();
      daCust.Fill(dtTest);

      daCust.Update(dtTest);
      con.Close();
      return Dt;
}
```

Figura 5.13: Métodos registrar del agente administrador del MOA

5.3. Pruebas de MOA

5.3.1. Introducción

A continuación se describen las etapas de prueba que determinan la funcionalidad del prototipo, así como la forma en que se pueden utilizar cada uno de los módulos para propietarios que desean publicar y clientes que solicitan algún tipo de objetos de aprendizaje. Para hacer las pruebas se necesitaron tres máquinas que tienen alojados al propietario, cliente y administrador del mercado con el fin de probar la funcionalidad del sistema de forma distribuida, de tal manera que diferentes personas puedan manipular la aplicación desde diferentes puntos en la red. Por lo tanto, las máquinas que alojan a los módulos tienen IP fija lo que permitirá acceder a ellos desde cualquier PC a través de Internet.

Definición de objetivos

Objetivo General

- Probar el prototipo que actúe como un mercado de objetos de aprendizaje donde

diseñadores de objetos de aprendizaje los pongan a disposición del público y los profesores puedan acceder a los objetos para crear sus cursos en línea.

Objetivos Específicos

- Verificar que cada módulo represente las tareas correspondientes a los proveedores, compradores y al Administrador del mercado.
- Que el modelo mercado interactúe de acuerdo a una AOS que defina los elementos que intervienen en la transacción de un servicio.

De acuerdo al prototipo de mercado de objetos de aprendizaje que se realizó, este se divide en tres capas: Capa Web (IU), intermedio (lógica), datos (base de datos), lo que nos define la forma en que se deben hacer las pruebas del sistema. A continuación describimos algunas de las pruebas principales.

Capa Web

- Pruebas de contenido del sitio Web
- Pruebas del sitio Web
- Compatibilidad del explorador

En el caso de contenido del sitio web, se hicieron pruebas sobre errores gramaticales y en los elementos de interacción de las páginas web que se presentan al usuario, sin notar problemas. Se realizaron pruebas en los browser de Internet Explorer, Nestcape y Mozila sin ningún problema al cargar las páginas.

En este caso la IU para el usuario que requiere publicar se necesita su *nombre*, y *contraseña*, los cuales no pueden ser espacios en blancos, ya que tiene que ser una clave alfanumérica que se debe de registrar. Además, el tipo de objetos es un tipo de datos

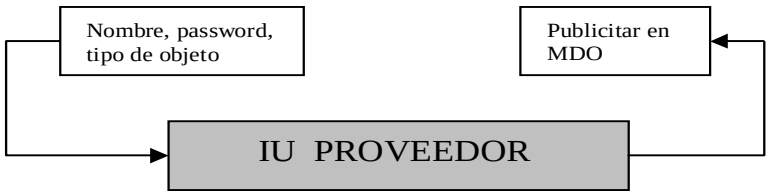

Figura 5.14: Caja negra de la IU del propietario de OA

string donde los valores están predefinidos, y el usuario selecciona algún tipo. Una vez dados estos datos, el usuario obtiene la respuesta de que ha sido publicitado en el MOA, como se muestra en la figura 5.14.

Para el caso de la interfaz de los clientes se puede observar en la figura 5.15 que se requiere dos cadenas alfanuméricas, que no deben estar vacías, y en caso de que estén vacías, el programa le envía un aviso a usuario de que no son aceptadas, hasta que se ingresen nuevas cadenas no vacías. Por otra parte, si el nombre ya esta ocupado, es decir ya han sido dadas de alta anteriormente, se le envía un mensaje para que las cadenas sean ingresadas nuevamente.

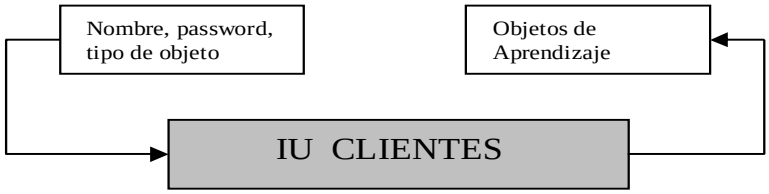

Figura 5.15: Caja negra de la IU del cliente de OA

Como lo muestra la figura 5.16 para el administrador del mercado no hay parámetros que se reciban directamente por parte del usuario humano, pero hay dos tipos de usuarios (programas), el agente proveedor, y el agente cliente, envían el nombre de quien representan, su password y el tipo de datos, además de una solicitud (publicitar, búsqueda), de tal manera que la IU(Interfaz de Usuario) muestra cada que exista un cambio en proveedores, clientes y transacciones que se realicen, por lo que

solo mostrará un *DataGrid* para que el usuario vea las actualizaciones.

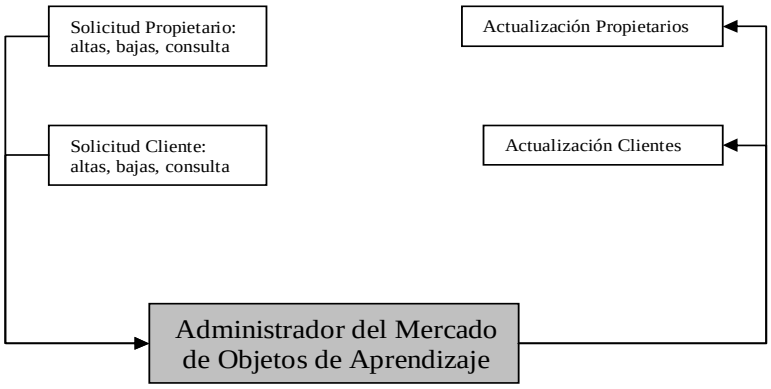

Figura 5.16: Caja negra de la IU del administrador del MOA

Capa intermedia

- Pruebas de funcionamiento del software (lógica empresarial)

Definiremos de forma general el patrón que se esta llevando acabo para la lógica de la aplicación distribuida. Debido a que se basa en la AOS, se definen tres componentes que van a estar comunicándose: clientes, proveedores,y administrador. Para que clientes y proveedores se comuniquen debe existir un intermediario que proporcione localizaciones transparentes que encapsulen los detalles de la comunicación entre clientes y proveedores, es decir estos componentes pueden estar distribuidos ya sea localmente o en una red. Los canales de comunicación que son los servicios Web en que esta encapsulados los proveedores, y se permite acceder a ellos a través de Internet, para realizar las interacciones desde cualquier lugar.

Ahora vamos a detallar algunas pruebas que se hicieron en la lógica de cada módulo, de acuerdo a las principales tareas en cada uno de ellos.

Los componentes *proveedores* son aquellos que ofrecen servicios a los clientes, registrándose previamente con el componente administrador, y al momento de registrarse proporcionan el url de servicio Web donde están. Este proveedor puede estar

localizado en la misma computadora que el cliente o bien en otro lugar de la red. El disponer del URL del proveedor permite que cualquier cliente pueda comunicarse a través de un Proxy generado por el cliente, y dando respuesta a su petición de acuerdo al servicio solicitado.

Para los procedimientos que permiten el registro e ingreso de proveedores al mercado se muestran en la figura 5.17, donde *nombre y pass* son cadenas validadas desde la interfaz de usuario de tal manera que no sean vacías, y lo que estamos mostrando son los métodos web del agente quien se comunica con el administrador del mercado, y de acuerdo a la respuesta que le envía, entonces retorna verdadero o falso, es decir si el usuario ya esta registrado y tiene permiso de ingresar. En el caso de que el usuario sea nuevo, envía los valores, esperando la respuesta de que estos no hayan sido dado de alta anteriormente.

```
[WebMethod]
public bool verifica_registro_ag_prov( string nombre,string
pass)
{
      if (mwadm.politicas_ingreso(nombre, pass)==true)

            return true;
      else
            return false;
}

//////////////////////////////////////////////////////////////
////
[WebMethod]
public bool registro_ag_prov( string nombre,string pass)
{
      if (mwadm.politicas_ingreso(nombre, pass)==false)

      {
            mwadm.registra_claves(nombre,pass);
            return true;
      }
      else
            return false;
}
```

Figura 5.17: Métodos Web del agente proveedor para comunicarse con el administrador

Otra de las funciones que tiene el proveedor es proporcionar una lista de los OA que tenga. Para ello necesita saber si el cliente que le esta pidiendo la lista pertenece al MOA, entonces solicita información del cliente y procede a leer el resultado para procesar y facilitar la lista de OA.

```
//Verifica si el cliente es valido en MOA
public bool cliente_valido(string nombre)
{
      return mwadm.verificar_cliente(nombre);
}

////////////////////////////////////////////////////////
[WebMethod]
public DataSet permiso_acceso(string nom_cli)
{
      if (mwadm.verificar_cliente(nom_cli))
      {     //Si el cliente es valido, entonces lo ingresa como cliente,
que se empieza la transaccion con el
            anexa_cliente(nom_cli);
            return proporciona_listaOA();
      }
      else
      {
            return null;
      }
}
```

Figura 5.18: Métodos Web del agente proveedor verificar datos con el administrador

En el método *cliente valido()* el parámetro de entrada es un string que no es un espacio blanco, ya que eso se valida en la entrada de la interfaz, y el método solicita información al administrador el cual procesa en su base de datos para verificar si existe este string en su lista de clientes, entonces regresa un valor verdadero si es válido, y falso en caso contrario 5.18.

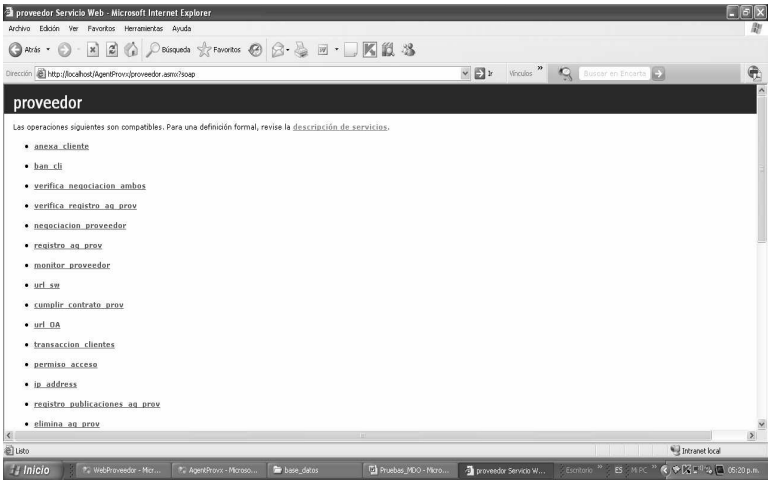

Figura 5.19: Métodos Web del agente proveedor que pueden ser accesados mediante SOAP

La forma en que se probaron la mayoría de los métodos y la facilidad que se dió es

que estaban dentro del servicio web, se prestaba para ejecutar los métodos con diferentes valores e independientemente de la interfaz, lo cual contribuyó a probar los resultados que podrían arrojar al ingresar valores inesperados. Para las pruebas que se generaban de forma funcional se podían ejecutar a través de la interfaz del servicio web del proveedor, donde los métodos que se generaron se muestran en la figura 5.19, y donde estos se pueden acceder por medio del protocolo SOAP.

Las siguientes pantallas 5.20 , 5.21 muestran los elementos que genera VS.NET para la comunicación entre sistemas remotos, a través de los protocolos mencionados como WSDL y el descubrimiento a través de DISCO que contiene la dirección del servicio web que se esta creando.

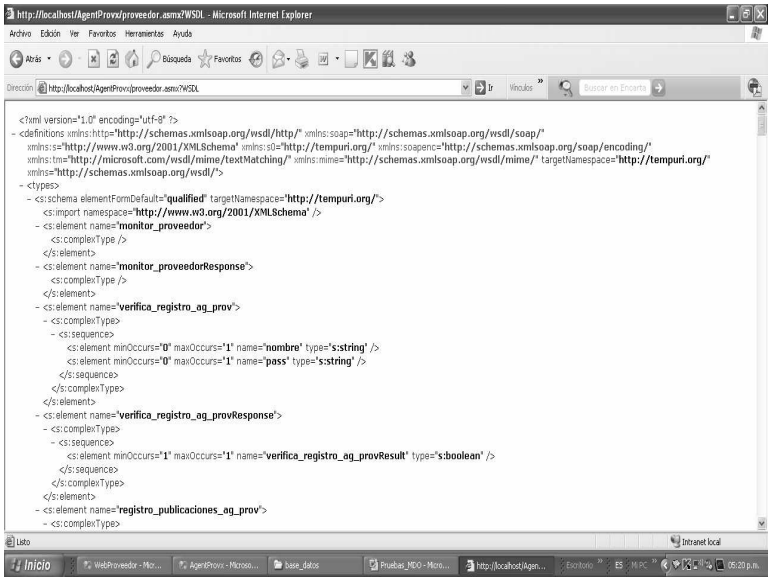

Figura 5.20: Archivo WSDL que contienen información sobre los métodos, para efectuar las comunicaciones

Los componentes clientes, se registran inicialmente con el administrador, a continuación llevan acabo tareas específicas de acuerdo a los servicios ofrecidos por el proveedor (tomando en cuenta que cada proveedor es un servidor independiente) cada vez que un cliente requiere un servicio de un proveedor, le pregunta al administrador un

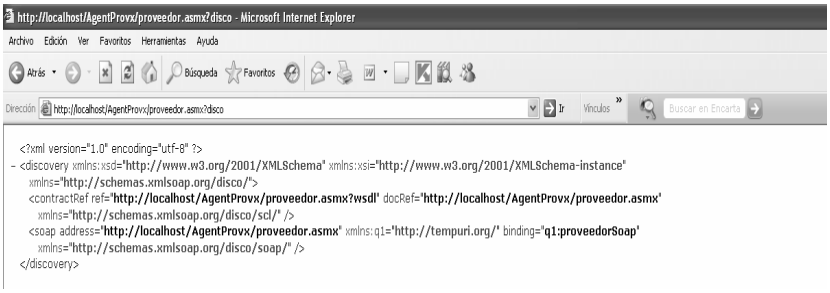

Figura 5.21: Archivo DISCO que contienen información para efectuar el descubrimiento de un servicio web

canal de comunicación, el cual es proporcionado y manipulado como un Proxy dinámico que se actualiza en tiempo de ejecución, de tal manera que se proporciona transparencia ya que el proveedor remoto aparece como un componente local para el cliente. Por lo tanto existe flexibilidad y bajo acoplamiento entre componentes.

No se puede solicitar un proveedor que no este dado de alta con el administrador, ya que solo pueden ser seleccionados quienes ya son miembros del mercado. En caso de que el proveedor no este disponible o ya se este eliminado, se mandará un mensaje de error, ya que no hay proveedor con el cual comunicarse.

En el caso del cliente, los principales métodos son la solicitud de búsqueda, la comunicación con el agente proveedor, y por ultimo la recepción del OA. A continuación mostramos su código en la figura 5.22 .

```
//Verifica si el cliente ya tiene permiso previo para entra al
//mercado como cliente con su nombre y tipo de objeto que busca
[WebMethod]
public bool registro_busquedas_cliente( string nombre,string pass,
string tip_obj)
{ return sw_adm.registro_admin_agcomprador(nombre,tip_obj);

}
//Metodo que envia la solicitud de busqueda al mercado, y //obtiene
el resultados en un DataSet
[WebMethod]
public DataSet solicita_busqueda(string tipo)
{//Llama al metodo de buscar del administrador
      return (sw_adm.localizar_tiposOA(tipo));
}
```

Figura 5.22: Metodos Web de comunicación para solicitar una búsqueda

El primer método *registro busquedas cliente()* recibe tres parámetros tipo string, y con estos parámetros realiza el registro del cliente enviando al administrador una solicitud. De acuerdo a la respuesta del administrador, devuelve un tipo *bool*, es decir verdadero si ya esta registrado, o falso de lo contrario. El método web *solicita busqueda()* se tiene como dato de entrada el string *tipo*, en este caso como ya vienen definidos los valores de forma predeterminada en la interfaz de un *listbox.* En este caso se solicita la información al administrador, retornando un conjunto de datos como un DataSet que contiene el tipo de objetos que cumplen con los requisitos de búsqueda.

```
//Recupera o actualiza la conexion con el sw del proveedor a
interactuar
[WebMethod]
public DataSet recupera_url_prov_datos(string temp, string nom_cli)
{
      prov_dinamico.Url=temp;
      return conecta_prov_datos(nom_cli);
}
```

Figura 5.23: Metodos Web para redireccionar dinámicamente

En el método *recupera url prov datos()* que se muestra en la figura 5.23 tiene dos parámetros de entrada *temp* que es el URL que se recupera de la selección del proveedor, y el nombre de cliente *nom cli*, los cuales son de tipo string. Se redirecciona el Proxy la comunicación a través de *prov dinamico.Url*, una vez que se ha comunicado con el proveedor, retorna un tipo de datos DataSet que contiene el conjunto de objetos de aprendizaje del proveedor con el que se esta comunicando.

Para el método web *download file()* 5.24, se obtienen el *url, y arch1*, los cuales son datos que se obtienen de otros programas como *leer ip*, y el nombre del archivo se obtiene de leer-xml donde se analizan datos del objeto de aprendizaje en el proveedor, por lo tanto es información que ya debe estar validada cuando la obtiene el proveedor. Para ejecutar el procedimiento previamente ya se verificó que el cliente pertenece al mercado, y de acuerdo a los datos URL y nombre del archivo que el proveedor ha

```
//Metodo que obtiene el archivo de acuerdo al URL especificado por el
proveedor y lo coloca en la ruta especificada
[WebMethod]
public void download_file(string url,string arch1)
{
      string dir_local=@"C:\objetos_down\"+arch1;
      try
      {
      WebClient client_web = new WebClient();

      client_web.DownloadFile(url,dir_local);//http://acer/objeto1.zip",
      }
      catch (WebException ex)
      {
                  Handle error.
      }
}
```

Figura 5.24: Metodos Web de comunicación para bajar un archivo al cliente solicitante

concedido, realiza la trasferencia del objeto del objeto de aprendizaje, a una dirección local por el momento predefinida.

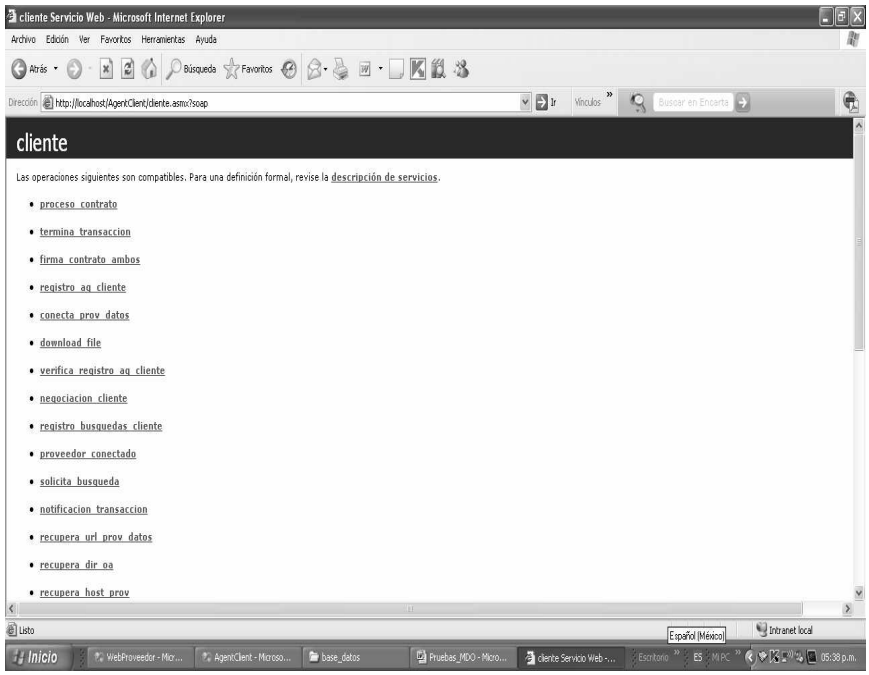

Figura 5.25: Metodos Web de comunicación para el cliente mediante SOAP, generados por VS.NET

Para la generación del agente cliente se crearon los métodos que se observan en la siguiente figura 5.25, por medio de los cuales se comunican a través del protocolo SOAP, y permitirán efectuar la tarea de comunicación con proveedores que se encuentran en cualquier lugar.

La funcionalidad de cada tarea del agente se encuentra en los métodos y envían mensajes a través de ellos. Se muestra la figura 5.26 el archivo WSDL que genera VS.NET para permitir la interacción entre aplicación de forma transparente. Esto representa una ventaja al desarrollador, ya que permite enfocarse a la lógica de la aplicación y no en la construcción de los archivos de descripción de métodos para la comunicación. De la misma forma se genera un archivo *.disco* que contiene la dirección URL para poder realizar el descubrimiento y comunicación con el servicio web del cliente.

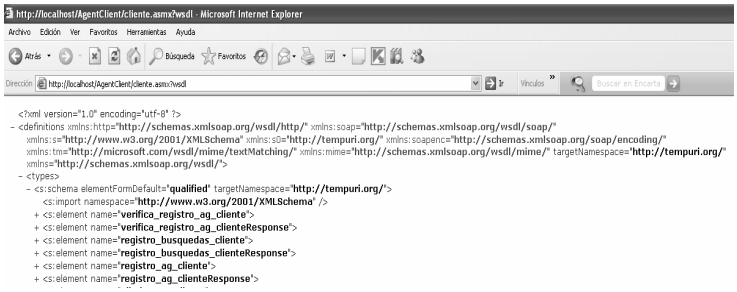

Figura 5.26: Descripción de los métodos, parámetros y protocolos SOAP para el SW del cliente

El método *localizar tipos OA()* 5.27 del administrador establece un vínculo entre el nombre del proveedor y su localización física (URL) en la red, de forma que el cliente solo pregunta por el nombre del proveedor, y se le devuelve la localización física. De tal manera, queda como responsabilidad de los clientes la conexiones establecidas. Además el administrador es el responsable de registrar proveedores y los servicios que ofrezcan.

Para que el administrador pueda determinar los proveedores que va a enviar, tiene que esperar una solicitud de algún cliente a través de string *obj bus*, la cual es tomada de un conjunto de tipos predefinidos en la interfaz en un *listbox*, por lo cual el parámetro esta validado, solo el caso en que no haya proveedores con ese tipo de objetos, se devolverá un mensaje avisando el resultado no satisfactorio. Para ello se realiza una consulta con este criterio y se devuelve el resultado obtenido en un *DataSet*.

```
 METODO QUE DEVUELVE UNA CONSULTA/BUSQUEDA ESPECIFICADA EN:
[WebMethod]
public DataSet localizar_tiposOA(string obj_bus)
{
      OleDbConnection con =new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;"
            + "Data Source=C:\\base_datos\\bd_prueba.mdb");
      con.Open();
      OleDbDataAdapter daCust = new OleDbDataAdapter("Select *
From ag_vendedores WHERE tipo_obj = '" + obj_bus + "'" , con);
      DataSet Dt=new DataSet();
      daCust.Fill(Dt,"tipos");
      con.Close();
return Dt;
}
```

Figura 5.27: Método que efectúa la búsqueda en la base de datos del Administrador

En el método *politicas ingreso()* 5.28, valida los datos de el ingreso de un usuario al mercado proveedor o cliente, para lo cual se verifica si estos datos son válidos dentro de su lista de claves (ya predeterminadas con anterioridad), si el resultado es satisfactorio, entonces pueden proseguir con sus tareas correspondientes, para ello se les devuelve un valor verdadero, en caso contrario se devuelve falso y se manda un mensaje al usuario, para que revise sus datos.

```
Metodo que verifica si un usuario (proveedor/cliente) ya esta registrado
para participar en el Mercado de Objetos de Aprendizaje
[WebMethod]
public bool politicas_ingreso(string login1, string password1)
{
      ADODB.Connection cn = new ADODB.Connection();
      ADODB.Recordset rs = new ADODB.Recordset();
      string cnStr;
      string query;

      cnStr = "Provider=Microsoft.Jet.OLEDB.4.0;"
                  + "Data Source=C:\\base_datos\\bd_prueba.mdb";
                  query = "Select * From identificacion WHERE usuario= '"
+ login1 + "' and clave='"+password1+"'";
      cn.Open(cnStr, null, null, 0);
            cn.ConnectionString = cnStr;
      cn.Open(null, null, null, 0);


      rs.Open(query, cnStr, ADODB.CursorTypeEnum.adOpenKeyset,
ADODB.LockTypeEnum.adLockOptimistic, -1);
      while (!(rs.EOF))
      {
            ccc=rs.Fields["usuario"].Value.ToString();
            c=rs.Fields["clave"].Value.ToString();
            if (ccc==login1 && c==password1)
                  return true;
            rs.MoveNext();
      }
      rs.Close();
      cn.Close();
      return false;
}
```

Figura 5.28: Método que verifica la validez de un usuario en la BD del administrador

Una vez que se ha validado el usuario, el administrador lo da de alta en la base de

datos correspondiente de proveedores o clientes. Para ello utiliza la instrucción *INSERT INTO* de SQL para anexar el nuevo integrante al mercado. Por ejemplo, la información envíada por un proveedor es: nombre de cliente, el tipo de objeto que esta publicitando y el URL que corresponde al servicio Web donde se encuentra el proveedor, es la que se anexa a la base de datos correspondiente al proveedor, y el mismo proceso se realiza para ingresar un cliente.

```
//METODO QUE PERMITE EL REGISTRO DE LOS CLIENTES Y COMPRADORES EN LA
TABLA DE
[WebMethod]
public DataSet registra_transacciones( string ag_ven, string ag_comp,
string estado)
{
      DataSet Dt=new DataSet();
      OleDbConnection con =new
OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;"
                  + "Data Source=C:\\base_datos\\bd_prueba.mdb");
      con.Open();
      OleDbDataAdapter daCust = new OleDbDataAdapter("INSERT INTO
transacciones (AG_VEN,AG_CLIENTE, ESTADO) VALUES
('"+ag_ven+"','"+ag_comp+"','"+estado+ "');",con);
      OleDbCommandBuilder cbCust = new OleDbCommandBuilder(daCust);
      DataTable dtTest = new DataTable();
      daCust.Fill(dtTest);

      daCust.Update(dtTest);
      con.Close();
      return Dt;
}
```

Figura 5.29: Método que registra un contrato que se ha establecido entre un cliente y un propietario

Una vez que el administrador ya envió la dirección solicitada y el cliente se comunica con un proveedor que comienza una transacción la cual es notificada al administrador del mercado, esta tarea se realiza en el método *registra transacciones* 5.29 en la cual se reciben como parámetros el nombre de los agentes (que teclearon los clientes), y el estado en el que se encuentra la transacción (proceso/terminado). La información es envíada a *registra transacciones()* y es integrada a la tabla transacciones.

La siguiente figura 5.30 muestra cuales son las tareas del agente administrador y se exponen por medio del servicio web que encapsula para facilitar la comunicación con agentes proveedores y clientes.

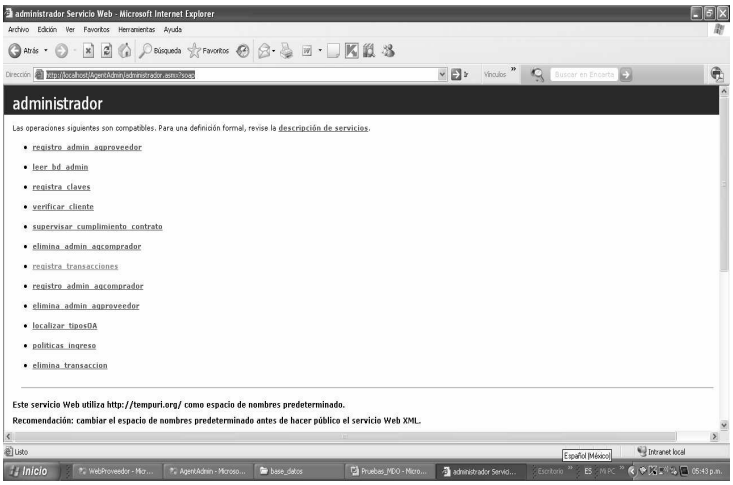

Figura 5.30: Interfaz del servicio Web que contiene las tareas de agente Administrador

Capa Datos

Se muestra la base de datos que tienen las tablas donde se guarda la información relevante de cada uno de los participantes. Las tablas que comprende la base de datos 5.31 son: *identificación* que contiene los datos *nombre* y *password* para cada participante del MOA y el cual es dado por cada usuario correspondiente. La tabla *ag vendedores*

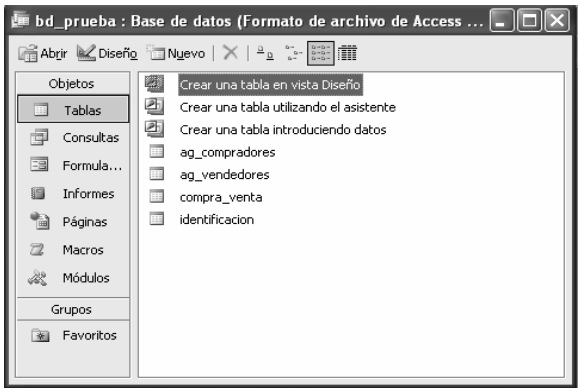

Figura 5.31: Base de datos del Administrador

corresponde a los propietarios que se publiquen en el MOA contiene tres campos con el

nombre del usuario (representado por el agente propietario), *password, y url* del servicio web que encapsula las tareas del agente para publicar. La tabla *ag compradores* que contiene información de los clientes que ingresen al MOA con los campos nombre y password. En la tabla *compra venta* se guarda las transacciones que se efectúan entre clientes y propietarios de objetos de aprendizaje, teniendo como campos nombre del propietario, nombre del cliente y estado de los contratos (transacción de acceso a OA).

Otra de las bases de datos que se maneja es la correspondiente a cada propietario, la cual contiene los datos principales de cada grupo de objetos de aprendizaje que tenga y que va a estar actualizándose de acuerdo a cambios que se lleven acabo en el repositorio local de cada propietario.

5.3.2. Ejecución del Mercado de Objetos de Aprendizaje

Ahora describiremos la forma en que actua cada componente del mercado y visualizaremos la interacción que se da por medio del adminsitrador del mercado.

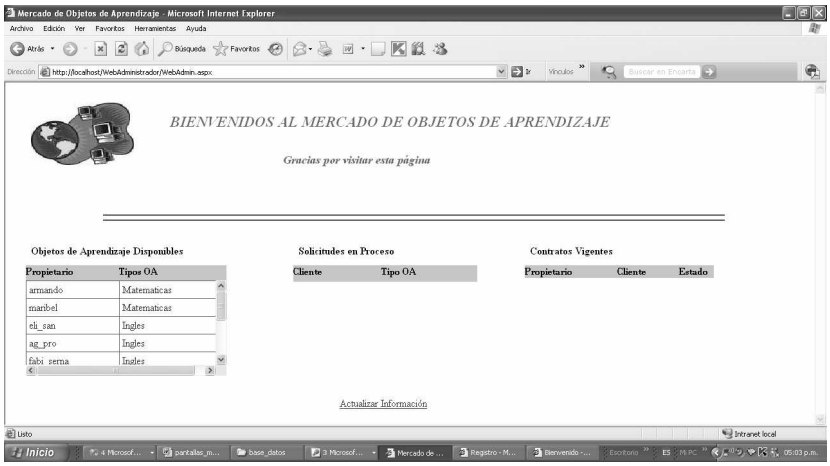

Figura 5.32: Interfaz del administrador al iniciar operaciones

Se muestra el administrador del mercado en la figura 5.32 al inicio de la prueba, no hay clientes que soliciten OA, por lo tanto, no hay contratos.

Probando el sistema para los propietarios de OA.

Resultados esperados:

- Que el sistema permita dar de alta a un propietario de OA.
- Que permita publicitarse en el MOA.
- Que pueda ver los clientes que se comunican para obtener OA.
- Que muestre sus OA disponibles.
- Que se pueda dar de baja del MOA.

Como es muestra en la figura 5.33 se inicializa la página para propietarios, el propietario *guid* ingresa mal sus datos, a lo cual el sistema devuelve un aviso para que ingrese los datos nuevamente, de lo contrario no se dará acceso al MOA.

Figura 5.33: Interfaz de inicio para profesores propietarios de OA, ingresando datos erróneos.

Otra vez, el propietario *guid* agrega datos correctos, a lo cual el sistema le permite seguir a la página en la que pueda publicitar un tipo de OA.

Una vez que esta en la página para publicitar 5.34, el usuario elige el tipo de objetos de aprendizaje a publicitar, y dando click en la liga *Publicar*, y muestra los OA que tiene por el momento en su base de datos.

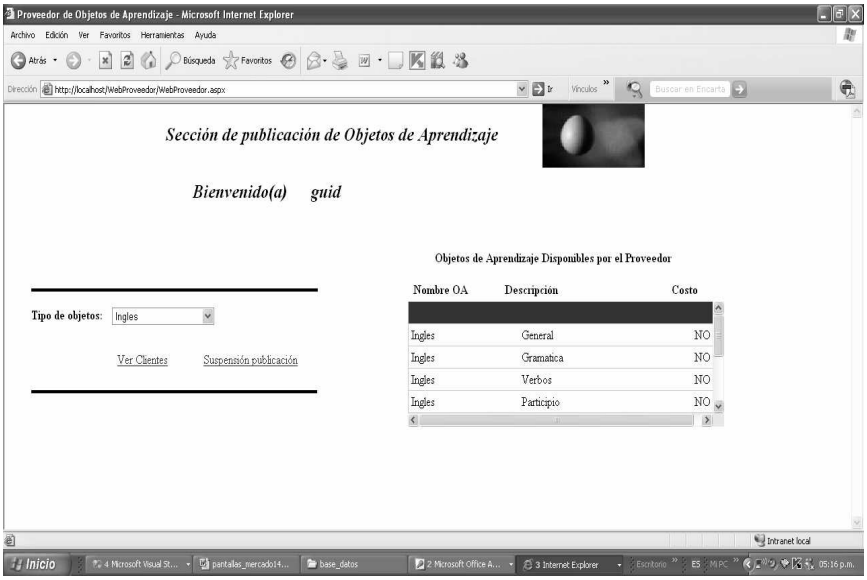

Figura 5.34: Interfaz donde se publicitan los propietarios, de acuerdo a un tipo de OA.

La liga *Ver Clientes* despliega una tabla con datos de los proveedores a los que ha solicitado OA, la primera vez no hay información ya que se acaba de publicitar.

Después de que un propietario de OA, ha ingresado al MOA. Lo que se espera es que el propietario aparezca en la lista de vendedores para poner a disposición algún tipo de OA en particular y entonces pueda prestar sus OA, y además, estar al tanto de los clientes que se comuniquen para acceder a un OA particular.

Probando el sistema para los clientes del MOA.

Resultados esperados:

- Que el sistema permita dar de alta a un cliente (solicitante) en el MOA.
- Que permita la localización de tipos de OA en el MOA.

- Que obtenga los tipos de OA disponibles en el MOA.
- Que pueda obtener algún OA del propietario elegido.
- Que pueda ver los propietarios con quien se ha realizado algún contrato para obtener los OA.
- Que se pueda dar de baja del MOA.

A continuación intenta ingresar un usuario *lerys25* en la sección de Clientes como se muestra en la figura 5.35 ingresando sus datos de *nombre* y *contraseña* que son verificados por el administrador, y le permite continuar en su proceso de búsqueda.

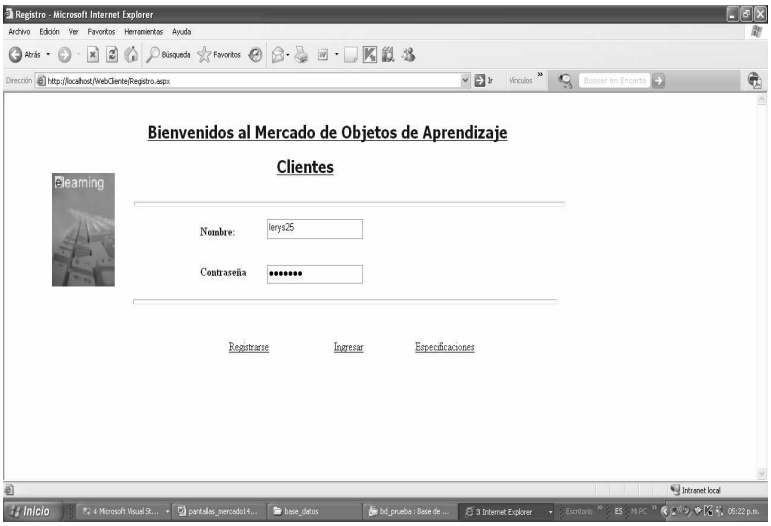

Figura 5.35: Interfaz de ingreso al MOA como solicitante de objetos de aprendizaje.

A continuación se muestra una nueva pantalla para el cliente que acaba de ingresar al MOA y puede determinar el *tipo de OA* que desea localizar, seleccionándolo en el *listbox*, y dando click en el liga *Buscar* tipos de OA, entonces despliega una tabla con los propietarios que tiene los objetos que corresponden con la solicitud. A continuación podemos observar que se despliegan los propietarios que existen en el MOA, en este

caso ya podemos ver al proveedor *guid*, como uno de los postulantes ya que ingreso anteriormente (ejemplo parcial del módulo anterior).

Al seleccionar el proveedor *guid*, muestra los OA que tiene este propietario como se observa en la figura 5.36 (previamente el proveedor pregunta al administrador si es un cliente registrado en el MOA).

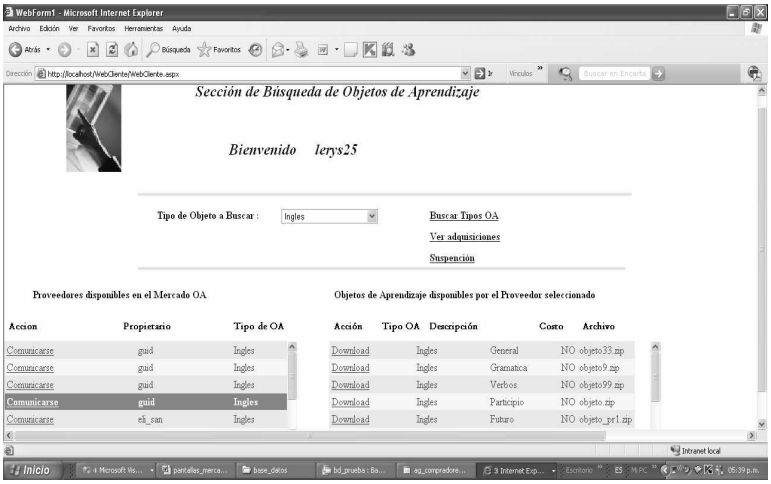

Figura 5.36: Interfaz que muestra los OA del propietario seleccionado.

Una vez que despliega los OA que tiene el propietario, selecciona un OA para bajarlo, previamente tiene que aceptar el contrato de uso de OA, y entonces puede obtener el OA. Si no acepta el contrato, entonces se le avisará que no puede tener acceso al OA.

Administrador del mercado de OA.

La vista de la interfaz del administrador del mercado se muestra en la figura 5.37 después de las transacciones correspondientes del cliente y proveedor, donde podemos apreciar los clientes y propietarios que se han dado de alta y las transacciones de OA que se han realizado a través de contratos.

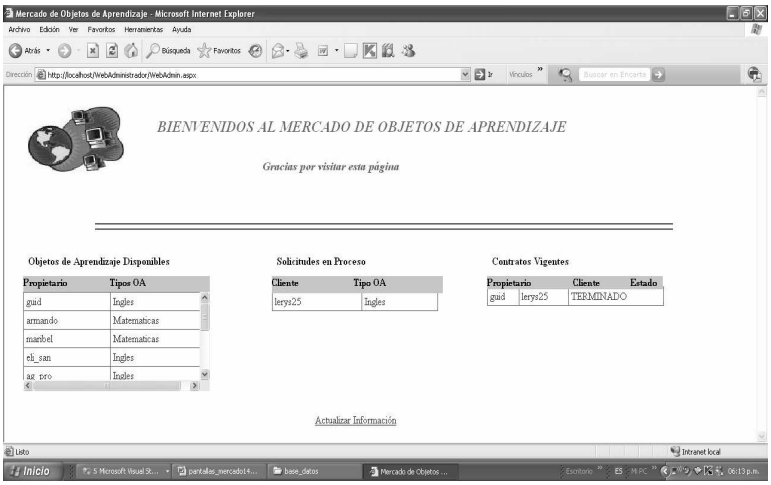

Figura 5.37: El administrador presenta los cambios efectuados en el MOA

Segunda prueba

Ahora vemos en la figura 5.38 como ingresa el usuario *mine* como propietario que ya fue registrado previamente.

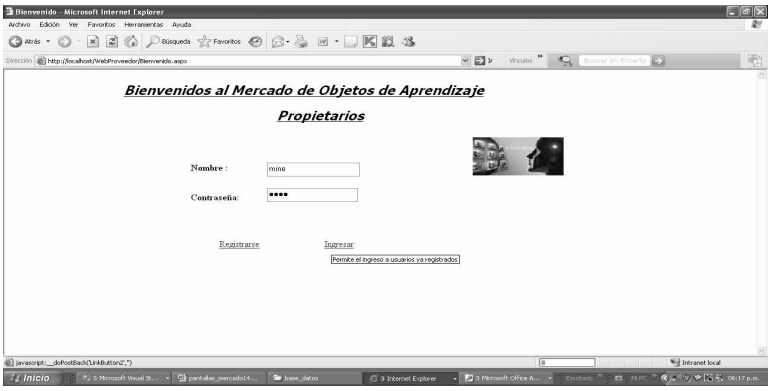

Figura 5.38: Ingreso de un usuario propietario en MOA

Una vez que ha ingresado, puede publicar y entonces ahora solo puede darse de baja o ver los clientes que se comuniquen con él, en este caso el usuario no puede publicar dos veces ya que solo puede publicar un tipo de objetos de aprendizaje.

Ahora un cliente intenta ingresar al MOA con datos erróneos, a lo cual el sistema le

notifica con un mensaje para ingresar nuevamente los datos, entonces el usuario teclea datos válidos dandose de alta como cliente y puede buscar un tipo de OA en el MOA. Cada vez que registra un usuario el administrador debe verificar si tiene permisos para entrar al MOA, si no tienen permiso, entonces no le permite el ingreso.

Cuando el usuario elige un tipo de OA que desea buscar y el administrador le proporciona una lista con los propietarios que cumplen con la solicitud de búsqueda, muestra los proveedores disponibles como se muestra en la figura 5.39 donde vemos los proveedores que hemos dado de alta anteriormente. Entonces el usuario puede elegir algún propietario y ver los OA que tiene disponibles. En este caso, selecciona *mine* como proveedor, y le despliega la información que tiene este propietario.

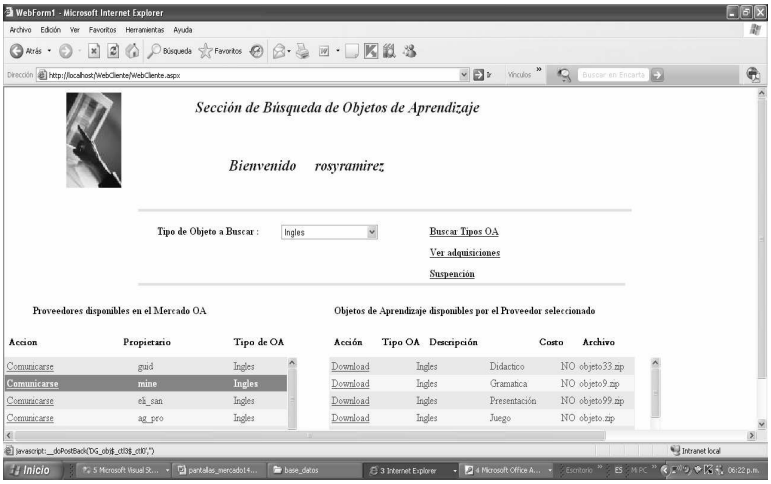

Figura 5.39: Se despliegan los propietarios que tienen el tipo de OA especificado

Se muestra la interfaz para aceptar el contrato en la figura 5.40, al seleccionar un OA y aceptar el contrato, queda en espera de que se le envíe el OA que ha solicitado. Hay que recalcar que la notificación del proceso del contrato lo va observando el administrador del mercado ya que la información de las transacciones de los participantes la va resguardando.

En el caso de la baja de un cliente, solo nos regresa a la página principal para poder

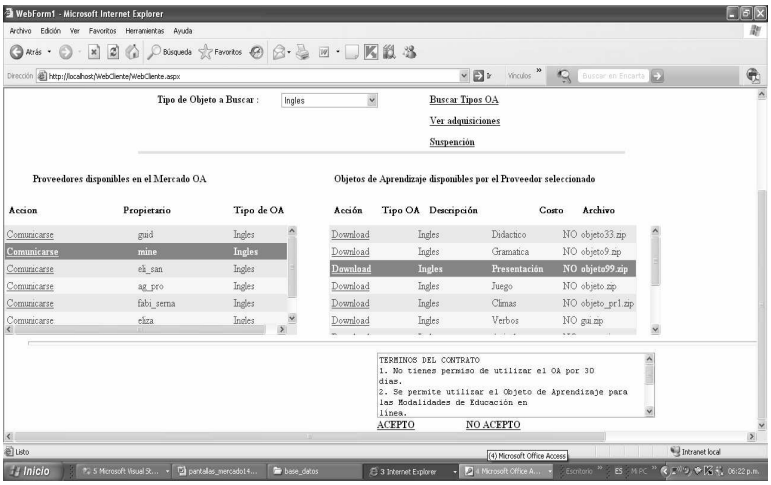

Figura 5.40: Se muestra el contrato al cliente

darse de alta otra vez, y en la base de datos del administrador se le da de baja como cliente, ahora vemos los cambios en la figura 5.41 reflejado en el administrador después de una eliminación del MOA.

Figura 5.41: Se muestran las actualizaciones efectuadas en el administrador después de la eliminación.

Continuando con el otro proveedor activo, se elimina ahora el primer proveedor *guid* que se dio de alta en el primer paso de la prueba 1. Ahora podemos ver en la interfaz del administrador como ya no esta *guid* en la lista de propietarios, es decir ya fue dado de baja. Sin embargo los datos de contratos son guardados para ver todas las interacciones

que se ha realizado entre clientes y propietarios.

Al estar probando el sistema en la segunda versión se obtuvieron diversas propuestas para mejorar el prototipo, sin embargo en la tercera versión ya conformada con las tareas de los agentes de forma sencilla encapsulados en los servicios web como simple métodos web, se probo el sistema de forma distribuida con el fin de ver como se comportaba el sistema de forma remota. Una de las desventajas es que al comunicarse a través de la red el tiempo de transmisión puede variar, pero en la prueba de funcionalidad, el sistema realizó la mayoria de las tareas que se propusieron en un principio.

Además se comprobó que el esquema de mercados basado en la AOS, es un modelo que permite establecer una comunicación a través de la red para acceder a un servicio que esta disponible en algún lugar. Otro punto importante es que la tecnología de servicios Web nos dio una gran ventaja para poder realizar el sistema de forma remota a través de los estándares que maneja, y sobre http que es un protocolo que facilita la comunicación en Internet, que es el medio por el cual la mayoría de los usuarios tiene acceso.

Como hemos visto el sistema cumple con los objetivos de que se plantearon inicialmente de forma satisfactoria, verificando que la tecnología de servicios Web es factible para realizar un sistema basado en el esquema de mercado donde los participantes se encuentran de forma distribuida y con repositorios locales teniendo a cada propietario como un servidor que ofrece sus objetos de aprendizaje sin la necesidad de envíar toda esa información en un repositorio central. Otro punto importante es que los agentes que se desarrollaran a futuro, pero por el momento se identificaron y realizaron las tareas que tíenen los agentes de forma sencilla para realizar el prototipo y dar la visión general del proyecto.

Cada quien tiene una tarea especial, publicitar o solicitar un OA, pero en conjunto la tareas de los OA es funcionar como un MOA el cual esta distribuido. Este proyecto

es un punto de partida que se puede ir complementando anexando otras tareas de forma más detallada con el fin de automatizar el sistema de acuerdo a criterios más específicos, y la nueva tendencia para incluir de forma amplia el tema de negociación e inferencia entre agentes. Este proyecto tiene gran impacto en el ámbito educativo, ya que los profesores necesitan estar a la vanguardia y tener nuevas herramientas que les permitan compartir y adquirir material educativo para sus cursos en línea y además que puedan ser capaces de integrarse en un grupo que puede proponer y disponer de nuevos materiales como lo son los objetos de aprendizaje.

Capítulo 6

Resultados, Conclusiones y Perspectivas

Debido al crecimiento que hay en la Educación en línea, cada vez se exigen cursos con contenidos pedagógicos y vanguardistas, para obtener un mejor aprovechamiento y retroalimentación por parte del estudiante. Por lo tanto, los profesores requieren crear materiales que sean reutilizables y que se puedan compartir. Como bien sabemos, el hecho de compartir información facilita la construcción de nuevos cursos de forma sencilla, minimizando tiempos e incrementando calidad.

Resultados

Por las necesidades ya planteadas en el Capítulo 4, en ésta tesis se pretende que los diseñadores de Objetos de Aprendizaje puedan ponerlos a disposición para que otros profesores los utilizan y crear sus cursos en línea, con las ventajas en reducción de costos y tiempo. Para lograr este fin se llevaron a cabo las siguientes tareas:

- Modelo de diseño y de secuencia para realizar la implementación del sistema de mercado de objetos de aprendizaje.
- Identificación de las interacciones entre componentes.
- Definición de la tareas de los agentes.

- Identificación de agentes encapsulados en Servicios Web, para la transmisión de los OA.
- Desarrollo del prototipo del mercado de OA de forma dinámica.
- Pruebas del prototipo con diferentes usuarios de forma distribuida.

De acuerdo a las necesidades y recursos que se van teniendo a través de la infraestructura de Internet, es muy importante minimizar el tráfico en la red. Este problema se presenta cuando se tiene un repositorio central al que todos llegan e interactuan para depositar y obtener información. Por otra parte, la generación de aplicaciones más complejas determina el no tener que empezar desde cero, si no el proporcionar y obtener funcionalidades o servicios que minimizen en cierta forma el desarrollo de nuevas aplicaciones.

Es importante manejar objetos de aprendizaje para dar mayor flexibilidad en la reutilización de los recursos digitales pedagógicos entre la comunidad académica, ya que estos pueden ser transportables y reutilizables, pues es uno de los objetivos del nuevo paradigma de objetos de aprendizaje. Debido a las necesidades que se tienen en la generación de cursos en línea y después del planteamiento de generar una aplicación en base a la AOS como se muestra en el el Capítulo 4 donde se describe el modelo de mercado, se confirma que es factible crear una aplicación a través de esta estructura, con el fin de tener un conjunto de repositorios de objetos de aprendizaje de forma distribuida.

El representar a los proveedores y consumidores a través de agentes facilita la tarea general de la aplicación, pues cada uno tiene ciertas tareas específicas, que se mencionaron en el Capítulo 4. De esta forma podemos determinar la comunicación de los agentes por medio de Servicios Web publicados y localizados en cualquier parte y con disponibilidad en cualquier momento y por supuesto que estos SW pueden ponerse a

disposición de otras aplicaciones que requieran tareas similares u otras instituciones que deseen crear sus propios clientes (claro que éste sería un trabajo futuro muy interesante) de acuerdo a políticas que se determinen. La interacción a futuro será en un sistema multiagente con el fin de prestar y compartir OA, entre un conjunto de proveedores y consumidores que pueden ingresar en cualquier momento o bien darse de baja del mercado de OA.

Ahora tenemos el prototipo en el cual se tiene el esquema de mercado que permite registrarse, ingresar o darse de baja como proveedor o cliente. Cada componente que represents a los agentes fue encapsulado en un servicio web por objetivos de comunicación entre los diversos componentes que se encuentran distribuidos. Además, las ventajas que representa el tener cada tarea como un método web permite la interacción entre componentes independientemente de la localización que tengan.

Conclusiones

El haber implementado este prototipo nos permite ver que la AOS es una arquitectura que permite gran flexibilidad para la creación de aplicaciones distribuidas y de acuerdo al esquema de mercado de De Roure [12]. Además, en este esquema donde los proveedores y clientes son representados mediante agentes, permite definir tareas concretas para los intereses de cada uno, donde la inferencia de los agentes serán agregadas más adelante. Por ahora, el prototipo del MOA (Mercado de Objetos de Aprendizaje) nos da una visión general de como trabaja cada agente en el mercado de forma que las tareas que se mencionaron en el Capítulo 4 para cada uno se pueden ir refinando para tener un funcionamiento más completo. Sin embargo la versión que se tiene hasta ahora nos permite tener el bosquejo general de un gran proyecto que ha comenzado con grandes perspectivas de crecimiento a futuro.

Ventajas de trabajar con SW:

- Flexibilidad para anexar nuevos componentes de software(agentes), para crear el sistema de MOA.
- Descubrimiento dinámico de los componentes, gracias al URL de los agentes que se registran.
- Redireccionamiento dinámico para la comunicación entre agentes.

Ventajas del esquema de mercado en base a la AOS:

- El registro de los clientes y proveedores para poner a disponibilidad y accesar a los OA en cualquier momento.
- La notificación al administrador de su suspensión en cualquier momento, con el fin de evitar problemas en el MOA.
- Tener el mayor control de los clientes y proveedores de acuerdo a las transacciones realzadas en el MOA.

Perspectivas y trabajo Futuro

- Se implementará el manejo de inferencia en los agentes (trabajo de tesis en curso).
- Realizar una búsqueda más específica de acuerdo a criterio de búsqueda del cliente, tal vez empleando motores de búsqueda, como por ejemplo en base a palabras que se encuentren en la descripcón del objeto.
- Crear nuevas políticas de ingreso en base a otro criterios de permiso o características del agente.

- Se implementará como otro tema de tesis, la tarea de negociación, ya que es un tema demasiado extenso y tiene que determinarse con que protocolo de negociación entre agentes es viable trabajar (trabajo de tesis en curso).

- La integración de otros proyectos de validación de OA [8] y administración de derechos digitales [4], los cuales están en curso.

En el caso de la seguridad es necesario que se trabaje ya que este tema es muy importante para aplicaciones distribuidas, pero es un tema que será implementado a futuro. Por el momento se realizó la primera versión con el fin de tener el prototipo general del funcionamiento del mercado dinámico, en base al modelo de mercado que se planteo inicialmente bajo la AOS.

Es importante mencionar que el prototipo sobre un mercado de objetos de aprendizaje es un esquema básico como punto de partida para poder implementar el sistema de forma completa. El proyecto se llevará a cabo entre miembros de CENIDET, la Universidad de Hidalgo y la BUAP, el proyecto esta financiado por el fideicomiso SEP-UNAM el proyecto 159/2006.

Bibliografía

[1] Austin, Grainger, Barbir, Netxorks, and et. al. Web Services Architecture Requirements. Technical report, W3C, Working Group Note, Febrero 2004.

[2] Curbera F., Meredith G., and Weerawarana S. Web Services Description Language(WSDL 1.1). Technical report, W3C, Working Group Note, Marzo 2001.

[3] Newcomer E. *Understanding Web Services, XML, WSDL, SOAP and UDDI*. Addison Wesley, Madrid, 2002.

[4] Serna Hernandez F. Administración de derechos digitales (drm) para objetos de aprendizaje(oa). *Avances en la Ciencia de la Computación,ISBN 968 5733 06 6*, 1:6, 2006.

[5] Sánchez Román G.. Un modelo de mercado para acceder a objetos de aprendizaje. *Avances en la Ciencia de la Computación,ISBN 968 5733 06 6*, 1:6, 2006.

[6] Alvarez González L. and Gallardo González M. Diseño de un Repositorio de Objetos de Apoyo al Aprendizaje Colaborativo. Technical report, CISCI, Julio 2004.

[7] Jacobson I., Booch G., and Rumbaugh J. *El Proceso Unificado de Desarrollo de Software*. Addison Wesley, Madrid, 2000.

[8] Priego Melendez L. Una herramienta para la estandarización y verificación de objetos de aprendizaje. *XIX Congreso Nacional y V Congreso Internacional de Informática y Computación,ISBN 970-31-0751-6*, 1:8, 2006.

[9] Fowler M. and Kendall S. *UML gota a gota*. Addison Wesley, New York, 1999.

[10] Wooldridge M. and Jennings N. Intelligent agents: Theory to practice. *The Knowledge Engineering Rewiew*, 10(2):115–152, 1995.

[11] Jennings N. Workshop on agent communication languages. 1999.

[12] De Roure, Jennings N., and Shadbolt N. The Semantic Grid: A future e-science infraestructure. *Grid Computing*, pages 437–470, 2003.

[13] Nwana Hyacint S. Software agents: An overview. *The Knowledge Engineering Rewiew*, 11(3):205–244, 1996.

[14] B. Simon, Dolog. P., Miklós Z., Olmedilla D., and Sintek M. Conceptualising Smart Spaces for Learning. Technical report, Interactive Media in Education, Septiembre 2004.

[15] UDDI. Universal description, discovery and integration of web services,UDDI. UDDI Organization, 2005.

[16] W3C. Extensible markup language XML. World Wide Web Consortium, 2005.

[17] W3C. Simple object access protocol, SOAP. World Wide Web Consortium, 2005.

[18] W3C. Web services description languages, WSDL. World Wide Web Consortium, 2005.

[19] D. A. Wiley. Connecting Learning Objects to Instructional Design, Theory: A definition, a metaphor, a taxonomy. Technical report, Utah State University, Mayo 2001.

[20] López y López F. Arquitecturas Orientadas a Servicios y Agentes. Technical report, FCC, Benemérita Universidad Autónoma de Puebla, 2004.