Capítulo 1

Introducción

En este primer capítulo se realiza la presentación del *Proyecto Fin de Carrera.* Se explica cuáles son los motivos, objetivos y propósitos del mismo. Se presenta su planificación temporal y se expone cómo está estructura esta memoria.

1.1 .Motivación y propósito del proyecto

Los sistemas de información basados en agentes son una tecnología usada para la resolución de problemas complejos que son difíciles de resolver para el hombre. De esta manera, mediante el uso de agentes que siguiendo unas determinadas pautas establecidas logran obtener la solución a esos problemas.

El radio de actuación de estos sistemas es muy amplio y en este proyecto se centra en el desarrollo de un prototipo para el *Comercio Electrónico,* el cual se encuentra en alza en nuestros días debido al incremento del uso de Internet.

La idea de realizar este proyecto está basada en "Kasbah" [8], un sitio Web donde los usuarios crean agentes autónomos para comprar y vender productos en el Mercado. Es un sistema multiagente donde los agentes son totalmente autónomos. De esta manera se ayuda a los usuarios a realizar el proceso de negociación, proporcionando los agentes que actúan como negociadores intentando obtener el mejor acuerdo posible.

Es por ello por lo que se lleva a cabo el diseño de agentes de negociación (agentes compradores y agentes vendedores) y de una agente mediador (Mercado) que sirva de control y de conexión entre los agentes, para realizar de forma exitosa un proceso de negociación que es

el objetivo de este proyecto. En concreto, planteamos un prototipo de un *Mercado Virtual de Libros* en el cual los agentes negocien por sí solos sin la intervención del usuario llegando a obtener acuerdos en un tiempo finito que satisfagan a ambas partes.

Para realizarlo se usará la plataforma JADE *(Java Agent Development Framework)* que posee las herramientas y estándares necesarios para establecer el funcionamiento de los agentes. En esta plataforma se desarrollará el prototipo para el cual se diseñará y pondrá en funcionamiento una interfaz de JADE. Esta interfaz habrá de ser amigable y fácil de usar. Además, se planteará cómo debería ser la *interfaz Web* y cómo conectarse con el prototipo creado en JADE.

1.2. Objetivos del sistema

El objetivo del sistema es poner en funcionamiento un Mercado Virtual donde negocien agentes creados por los usuarios que puedan realizar la compra-venta de productos (en el caso de este prototipo los productos serán libros) sin necesidad de llevar a cabo por ellos mismos el proceso de negociación. Los usuarios delegan en unos agentes (comprados o vendedores según el rol que vaya a desempeñarse) y estos controlados por un agente mediador, que será el Mercado, realizarán el proceso de negociación.

De un modo más detallado, se puede decir que el desarrollo de este sistema pretende alcanzar las siguientes metas:

- Conseguir poner en funcionamiento el prototipo diseñado.
- Agilizar el proceso de negociación en la compra-venta de libros.
- El desentendimiento parcial por parte del usuario del proceso de negociación evitando molestas relaciones con el resto de usuarios.
- Dar confianza a los clientes realizando los procesos de negociación atendiendo a los parámetros y estrategia establecidos por el usuario.
- Controlar por medio del mercado el proceso de negociación y los agentes creados por los usuarios para que cumplan con su objetivo manteniendo su estrategia y no realicen prácticas fraudulentas.
- Obtener acuerdos de negociación que sean satisfactorios para ambas partes en un tiempo finito.
- Permitir al usuario controlar la conformidad o no con los acuerdos realizados por sus agentes.
- Ser un prototipo para comprobar el funcionamiento de los agentes y el proceso de negociación y servir como planteamiento para trabajos futuros.

1.3. Planificación del proyecto fin de carrera

Para la correcta planificación del desarrollo del proyecto se ha dividido en una serie de tareas que tienen establecida una duración especificada en semanas. El *Proyecto Fin de Carrera* son 7,5 créditos lo que supone unas 38 semanas.

ACTIVIDAD	DURACIÓN ESTIMADA *(semanas)*
Búsqueda Bibliográfica	3
Revisión Bibliográfica	4
Estudio de la plataforma JADE, SMA y conexiones Web	4
Análisis, Diseño y Desarrollo del Prototipo	18
Obtención de Resultados de Experimentación	2
Análisis y Evaluación	2
Completar la memoria	5
DURACIÓN TOTAL	**38**

Las primeras actividades a realizar son la "*Búsqueda y Revisión de la información bibliográfica*" que será útil en el proyecto y que serán objeto de estudio en la siguiente fase de "*Estudio de la plataforma JADE, SMA y conexiones Web*". La etapa de "*Análisis, Diseño y Desarrollo del Prototipo*" es la que posee una mayor duración ya que engloba a su vez un conjunto de subtareas. En la fase de "*Obtención de Resultados de Experimentación*" se realizarán pruebas sobre el prototipo para comprobar su funcionamiento óptimo y posteriormente en "*Análisis y Evaluación*" se estudiarán los resultados experimentales obtenidos. Por último, se completará la memoria del proyecto.

A través del siguiente diagrama se pueden observar las distintas actividades y las dependencias entre las mismas. En rojo se han indicado los hitos que indican el comienzo y fin del desarrollo del proyecto; en amarillo dos hitos intermedios y el resto de actividades en azul.

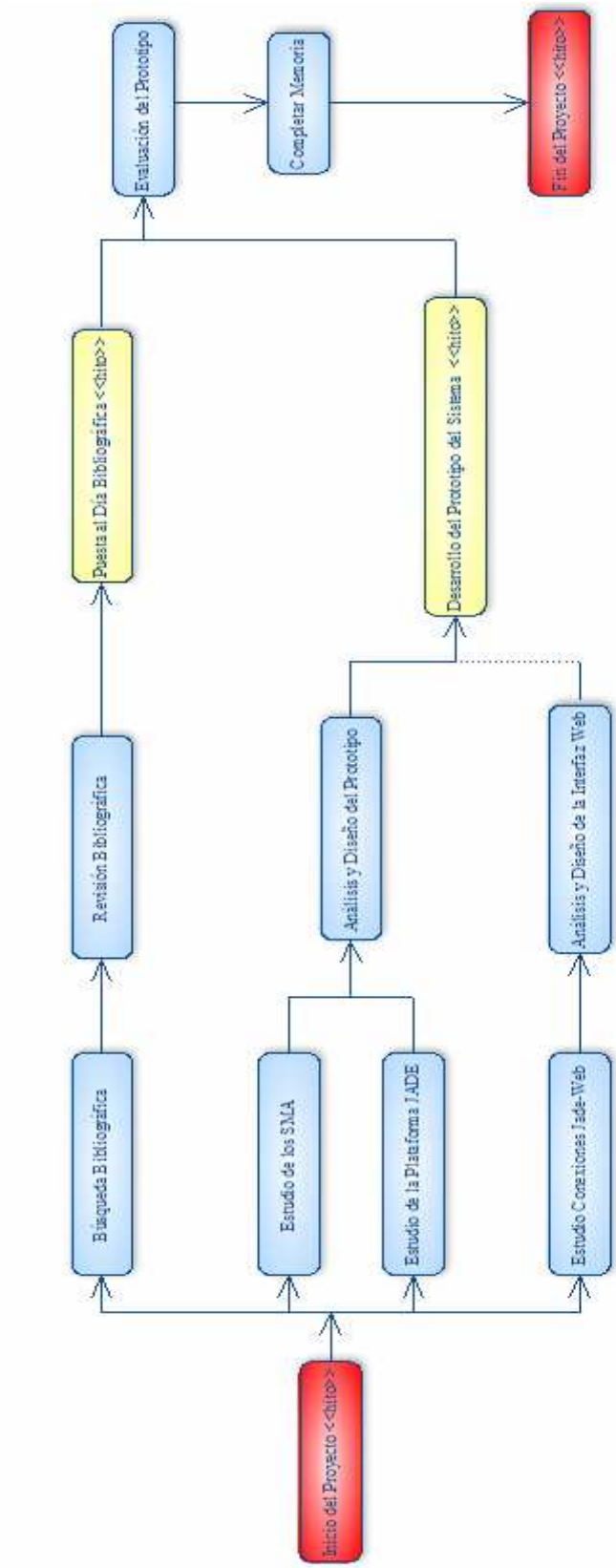


Figura 1.1. Diagrama de red con las etapas del Proyecto.

En la planificación general del proyecto la duración estimada para el *desarrollo del software* es de 18 semanas. Es necesario dividir a su vez este proceso en un conjunto de tareas que pueden observarse en la siguiente tabla:

Nombre de la Tarea	Duración (semanas)	Comienzo	Fin
A. Manual Preliminar del Usuario	2	01/05/06	19/05/06
B. Análisis del Sistema	2	22/05/06	09/06/06
C. Diseño del Mercado	3	12/06/06	10/07/06
D. Diseño de los Agentes	3	12/06/06	10/07/06
E. Diseño de la Interfaz	2	12/06/06	30/06/07
F. Implementación del Prototipo	3	11/07/06	08/08/06
G. Implementación de la Interfaz	3	11/07/06	08/08/06
Duración total	**18**	**01/05/06**	**08/08/06**

En el cuadro anterior se detallan las etapas que componen el desarrollo del software y se especifica la duración de las mismas y el período de realización en el que se llevan a cabo.

Para representar de forma gráfica la programación de las actividades del proyecto, es decir, su distribución conforme a un calendario, y visualizar el periodo de duración de cada actividad, sus fechas de inicio y fin así como el tiempo total requerido para cada trabajo se muestra a continuación un Diagrama de Gantt.

En el diagrama se representan en el eje horizontal una escala de tiempo expresados en semanas; en el eje vertical se encuentran las actividades que constituyen el trabajo a ejecutar.

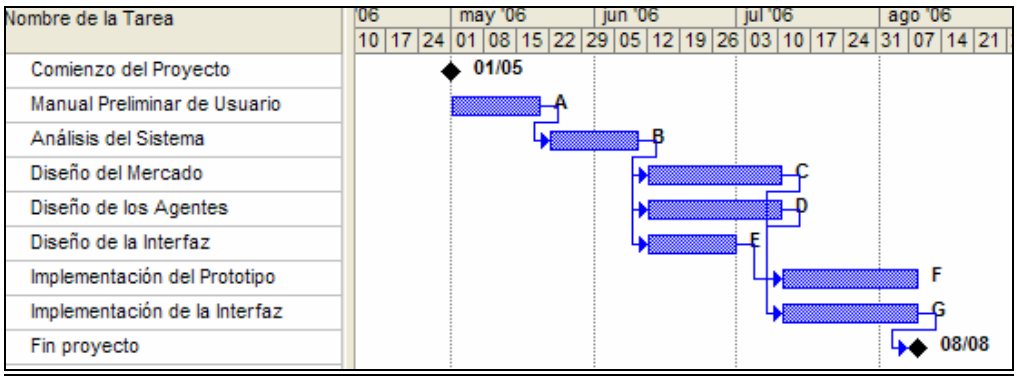


Figura 1.2. Diagrama de Gantt.

A continuación se muestra una representación, en forma de diagrama de red, que vincula las actividades y los eventos del proyecto entre sí para reflejar las interdependencias entre las mismas.

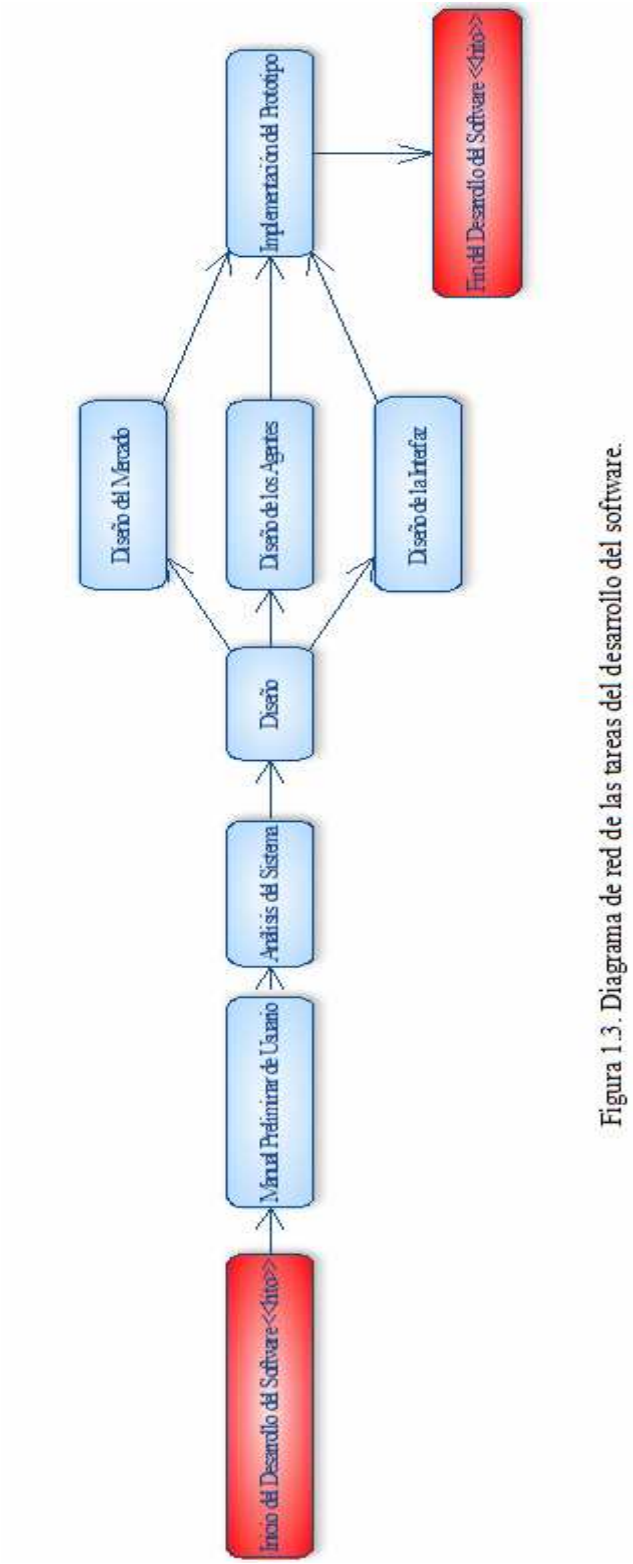


Figura 1.3. Diagrama de red de las tareas del desarrollo del software.

1.4. Estructura de la memoria

En este apartado se describe cómo se encuentra estructurada la memoria y cómo se ha organizado la información en función de los distintos capítulos que la forman siguiendo las fases de la Ingeniería del Software.

En el *capítulo 2* se realiza una introducción a los agentes y los Sistemas Multiagente así como su relación con el Comercio Electrónico. De forma similar, en el *capítulo 3* se realiza una introducción a la plataforma JADE usada para el desarrollo del software.

El *capítulo 4* está dedicado al Análisis y Diseño. Antes de comenzar es necesario realizar un Análisis y estudiar en qué ámbito se va a desarrollar el proyecto. Por ello ha sido necesario investigar el campo de los Agentes para comprender su funcionamiento. Una vez se ha comprendido el objeto del proyecto se realiza en profundidad el estudio sobre los requisitos y funciones que debe reunir el prototipo.

Establecidos previamente los requisitos en el análisis se realiza el Diseño y se establece el funcionamiento del sistema: cómo funcionará el Mercado y los agentes, cómo serán las conversaciones, cómo será la interfaz,…

Tras la fase de Diseño se realizará la Implementación del prototipo siguiendo el diseño establecido y se lleva a cabo la fase de Verificación y Validación en *el capítulo 5* donde se realizan las pruebas oportunas sobre el correcto funcionamiento del software.

En el *capítulo 6* se expondrá una valoración personal y las conclusiones finales del proyecto realizado, así como unas sugerencias sobre líneas de trabajo futuro. Además, la memoria posee unos anexos con el manual del administrador y el manual del usuario. Por último, se presenta en el anexo III los materiales incluidos en el CD que acompaña al proyecto, un glosario de términos y la bibliografía.

Capítulo 2

Los Agentes y el comercio electrónico

En este capítulo se introducen los conceptos principales sobre los Agentes y Sistemas Multiagentes así como su uso en el Comercio Electrónico que son necesarios conocer para entender el prototipo objeto de esta propuesta.

2.1. Introducción

Los agentes software [14] suelen presentarse como entidades computacionales con un comportamiento humanoide. Pueden funcionar en los equipos de los usuarios- PCs, PDAs, teléfonos móviles- y en los nodos de las redes; tienen autonomía y capacidad de decisión, razonan, aprenden, se comunican con otros agentes, pueden organizarse, y desplazarse de un nodo a otro. Solos o colaborando con otros agentes, utilizan sus capacidades para ayudar al usuario a resolver sus problemas de forma inteligente, servicial y pro-activa. Por ejemplo, en Internet los agentes pueden ayudar al usuario a buscar los productos más baratos, negociar los precios, aconsejar las mejores inversiones, y multitud de cosas más. De hecho este prototipo está enmarcado en el Comercio Electrónico donde los agentes realizarán las tareas de negociación en un proceso de compra-venta a través de una plataforma accesible desde Internet.

Los sistemas de información [14] basados en agentes software se han convertido en una tecnología, que cada vez más, se está afianzando como solución a muchos problemas complejos. Hablar de tecnología implica disponer de conceptos, teoría, métodos, procedimientos e

instrumentos para que los ingenieros puedan construir sistemas con agentes. En el *capítulo 3* se especifica en detalle el uso de esta tecnología a través de la plataforma JADE.

Al auge de esta tecnología ha ayudado de manera decisiva no sólo la disponibilidad de metodologías que asistan en el proceso de análisis y diseño sino también un número cada vez mayor de herramientas de programación y plataformas para la implementación de los Sistemas Multiagente una vez diseñados.

Por otra parte, el desarrollo de aplicaciones utilizando agentes [14] tiene numerosas ventajas:

- *Mejora la funcionalidad y la calidad*. Los sistemas basados en agentes son amigables y flexibles frente a los sistemas actuales caracterizados por ser poco intuitivos y de difícil uso por el gran público.

- *Menor coste.* Los agentes facilitan la reusabilidad. Requieren menos recursos y menor tiempo de desarrollo que otros sistemas convencionales con menores prestaciones.

- *Se reduce el mantenimiento facilitando la transformación y la evolución.* La funcionalidad puede cambiarse de forma rápida modificando el conocimiento de los agentes, sus estrategias o sus objetivos. Puede extenderse la funcionalidad incluyendo nuevos agentes, nuevo conocimiento y nuevos objetivos en cada agente.

- *Se integran adecuadamente con otras tecnologías.* Los agentes se integran, interactúan y son compatibles con otras aplicaciones desarrolladas mediante otras tecnologías (Web, bases de datos,…).

- *Simplifican la labor de los ingenieros.* Pueden utilizarse patrones de agentes que permiten al ingeniero concentrarse en definir el comportamiento del agente, evitando tediosas tareas de codificación.

2.2. Sistemas multiagente

Antes de comenzar es conveniente establecer la diferencia terminológica entre "*Sistema basado en agentes*" [10] y "*Sistema Multiagente*" (SMA) [10].

Un *Sistema basado en Agentes* es aquel que utiliza el concepto de agente como mecanismo de abstracción, pero aunque sea modelado en términos de agentes podría ser implementado sin ninguna estructura de software correspondiente a éstos.

Por otro lado, un *Sistema Multiagente* es aquel que se diseña e implementa pensando en que estará compuesto por varios agentes que interactúan entre sí, de forma que juntos permitan alcanzar la funcionalidad deseada. En este caso, es necesario realizar un mayor esfuerzo de abstracción, identificar mecanismos de aprendizaje, coordinación, negociación, etc. Los sistemas multiagente son adecuados para solucionar problemas para los que hay múltiples métodos de resolución y/o múltiples entidades capaces de trabajar conjuntamente para solucionarlos. Por ello, uno de los aspectos básicos en estos sistemas es la interacción entre los agentes que los forman y la definición de modelos concretos de cooperación, coordinación o negociación de los agentes.

Los Sistemas Multiagente tratan sobre la coordinación inteligente entre una colección de agentes autónomos, cómo pueden coordinar sus conocimientos, metas, propiedades y planes para tomar una decisión o resolver un problema caracterizan por una serie de propiedades:

- **Descripción de Competencias.** Consiste en la abstracción del problema para determinar las tareas, subtareas y relaciones para determinar la resolución del problema, cómo distribuirlo entre los diferentes agentes y las relaciones entre los mismos.

- **Modelos de Agentes Conocidos.** Se utiliza para representar la información relativa a cada agente (capacidades, comportamientos, creencias) para poder razonar y satisfacer sus necesidades de comunicación con otros agentes.

- **Comunicación.** A través de la comunicación tienen lugar todas las interacciones entre los agentes. Los canales de comunicación permiten que un agente tenga conocimiento sobre los otros agentes y la coordinación entre ellos; intercambiando información se realiza a través de un protocolo de comunicación.

- **Comportamiento del Sistema.** Se refiere al comportamiento global coherente de los agentes en un sistema, es decir, el grado de eficiencia, calidad y claridad de las soluciones obtenidas y el nivel de fallos no críticos del sistema.

- **Puntos de Interacción de un Agente.** Cada agente se encuentra en la necesidad de intercambiar información con otros agentes. Esto se realiza mediante las llamadas funciones de petición, de envío o de información. A todos los puntos presentes en la aplicación donde se produce una interacción con otros agentes se les denomina Puntos de Interacción (IP). Los IP son importantes porque son las localizaciones iniciales para las actividades cooperativas entre los agentes.

2.3. Agentes

En este apartado se va a realizar una introducción a los agentes, sus tipos y sus características para comprenderlos mejor ya que son un concepto fundamental en este proyecto.

2.3.1. Definición y características

El término agente es algo confuso existiendo múltiples definiciones de agentes. No existe una definición precisa aunque una de las más citadas [14] es la siguiente: "un *agente* es un sistema informático situado en un entorno y que es capaz de realizar acciones de forma autónoma para conseguir sus objetivos de diseño". Algunas de las propiedades [10] deseables en un agente son:

- **Autonomía:** capacidad de actuar sin intervención humana directa o de otros agentes.
- **Sociabilidad:** capacidad de interaccionar con otros agentes, utilizando como medio algún lenguaje de comunicación entre agentes.
- **Reactividad:** un agente está inmerso en un determinado entorno (hábitat), del que percibe estímulos y ante los que debe reaccionar en un tiempo establecido.
- **Iniciativa/Pro-Actividad:** un agente no sólo debe reaccionar a los cambios que se produzcan en su entorno, sino que tiene que poseer un carácter emprendedor y tomar la iniciativa para actuar guiado por los objetivos que debe de satisfacer.
- **Movilidad:** habilidad de trasladarse en una red de comunicación informática.
- **Veracidad:** no comunica información falsa intencionadamente.
- **Benevolencia:** no tiene objetos contradictorios.
- **Racionalidad:** tiene unos objetivos específicos y siempre intenta llevarlos a cabo.

2.3.2. Tipos de agentes

Se pueden establecer los tipos de agentes [14] en función de distintos criterios que se muestran a continuación.

- Tipos de agentes según sus *características individuales*:

    - **Agentes reactivos.** Realizan tareas sencillas. Su modelo computacional está basado en un *ciclo de recepción de eventos externos/reacción.* La reacción consiste en la ejecución de procedimientos o rutinas sencillas según el estado interno del agente. El comportamiento reactivo puede consistir en cambiar el estado interno, o en ejecutar funciones internas o funciones externas sobre el entorno. En todo caso, un agente reactivo no realiza procesos de razonamiento, ni tiene ningún mecanismo explícito de representación del conocimiento. El mecanismo de control se basa normalmente en autómatas de estados finitos extendidos. Pueden también incorporarse mecanismos de aprendizaje simbólico, sustituyendo el autómata de control por una red neuronal.

    - **Agentes cognitivos.** Realizan tareas complejas. Utilizan algún tipo de representación explícita (simbólica) del conocimiento. Para realizar las tareas necesitan llevar a cabo procesos de razonamiento y otros procesos cognitivos como la planificación y el aprendizaje. Las arquitecturas de los agentes cognitivos tienen como núcleo central algún tipo de procesador de conocimiento que integra la información procedente del entorno y controla el comportamiento interno y las acciones del agente de acuerdo con el conocimiento de éste. El modelo computacional se basa en un *ciclo percepción-asimilación-razonamiento-actuación.*

- Tipos de agentes según su *autonomía*:

La autonomía es una de las propiedades ideales de los agentes y como se comentó anteriormente, puede definirse como la capacidad de actuar sin intervención humana directa o de otros agentes. Atendiendo a este criterio podemos encontrarnos tres tipos de agentes:

    - **Agentes autónomos.** Es capaz de decidir qué acciones ejecutar o qué objetivos elegir sin un control exterior explícito.

    - **Agentes dependientes.** Necesitan de un control exterior explícito.

- **Agentes semi-autónomos.** Son un tipo de agentes que realizan algunas acciones de forma autónoma y otras de forma dependiente.

- Tipos de agentes según la *movilidad*:

La movilidad de un agente tiene que ver con su capacidad para desplazarse por los nodos de una red para realizar una determinada tarea. Con la movilidad se pretende conseguir que un agente se desplace de un nodo a otro conservando su estado interno: por ejemplo, un agente parte de un nodo con un conjunto de tareas a resolver y de datos iniciales sobre el problema, pero debe volver con la traza del trabajo realizado y con los resultados obtenidos. La movilidad está desarrollada en las distintas plataformas de agentes como JADE. Por lo tanto, la clasificación de agentes sería la siguiente:

- **Agentes Fijos.** Permanecen en el lugar donde fueron creados. Los agentes cognitivos suelen ser agentes fijos debido a su complejidad interna.

- **Agentes móviles.** Aquellos que poseen la capacidad de desplazarse por los nodos de una red para realizar una determinada tarea.

- Tipos de agentes según en el *entorno* en que funcionen:

Se puede considerar como entorno del agente toda la infraestructura computacional que le rodea, por ejemplo, el sistema operativo, los protocolos de comunicaciones, protocolos de comunicaciones,… El entorno proporciona los medios necesarios para que un agente desarrolle su ciclo de vida, es decir, pueda ser creado, funcione y desaparezca. De este modo, se pueden considerar dos tipos de agentes:

- los que requieren un **entorno especial**, es decir, una plataforma software específica para su funcionamiento, Un ejemplo son los agentes móviles donde cada plataforma proporciona los mecanismos para gestionar su ciclo de vida y para que se desplacen de un nodo a otro.

- los que se ejecutan en las **plataformas computacionales existentes.** Un ejemplo son los agentes que usan la plataforma FIPA [22]. Se denominan agentes FIPA porque usan los servicios de gestión de la plataforma (creación, registro, eliminación) para poder comunicarse con ellos.

- Tipos de agentes según el *modo de interacción*:

La interacción comprende la comunicación y el intercambio de información entre un agente y otras entidades. Se distinguen tres tipos de interacciones:

  - Interacción **agente-agente.** El agente sólo interactúa con otros agentes implicando el uso de estándares de comunicación como es ACL (Lenguaje de comunicación de agentes) y de protocolos de comunicación como RMI usado en JADE.

  - Interacción **agente-persona** (Agentes de interfaz). Interactúa con el usuario usando los medios adecuados para que las personas entiendan y se expresen con la mayor sencillez y naturalidad.

  - Interacción **agente- entorno.** Interactúa con el entorno del agente y comprende la comunicación con los distintos elementos de los cuales tiene que obtener información o recibirla de manera espontánea: sistema operativo, librerías, interfaces de usuario,…

- Tipos de agentes según el *modo de organización*:

Desde el punto de vista de la organización permite considerar a los agentes como entidades individuales que tienen un rol o función dentro de un conjunto o SMA con una finalidad común. Se pueden distinguir:

  - **Agentes individualistas.** No poseen la capacidad de cooperación y realizan tareas solos, sin requerir la colaboración de otros agentes.

  - **Agentes cooperantes.** Pueden realizar tareas solos o colaborando con otros agentes.

2.3.3. Arquitecturas de agente

Una arquitectura define los mecanismos que permiten interconectar los componentes tanto software como hardware, que hacen que un agente se comporte como tal. Las arquitecturas utilizadas para construir agentes especifican cómo se descomponen los agentes en un conjunto de módulos que interactúan entre sí para lograr la funcionalidad requerida.

Uno de los aspectos básicos que diferencia una arquitectura de otra es el método de descomposición del trabajo en tareas particulares y su planificación.

A continuación se presentan tres arquitecturas diferentes establecidas por Wooldridge que se diferencian por el modelo de razonamiento que utilizan: **Deliberativa**, **Reactiva** e **Híbrida.**

En un principio se pueden determinar que las dos grandes familias clásicas de agentes son la de los deliberativos y la de los reactivos. Un **agente deliberativo** [6] es aquel que "*contiene un modelo simbólico representado internamente de forma explícita y que toma decisiones mediante razonamiento lógico*". Por otro lado, un **agente reactivo** [6] es aquel en que "*no incluye un modelo simbólico del mundo ni usa razonamiento simbólico complejo de ningún tipo*".

2.3.3.1. Agentes deliberativos (Agentes BDI)

Los agentes deliberativos [14] usan un modelo del mundo explícitamente representado y funcionan siguiendo el paradigma de los sistemas clásicos basado en el ciclo percepción-planificación-acción.

El modelo de agente más representativo del tipo de los deliberativos es el BDI (**Belief-Desire-Intention**). Estos agentes se caracterizan por tener "ciertas actitudes mentales" como son:

- **Creencias:** corresponden al conocimiento que el agente posee sobre el resto del mundo.
- **Deseos:** cómo se ordenan los objetivos del agente.
- **Intenciones:** representa la estrategia de acción que el agente está siguiendo actualmente.

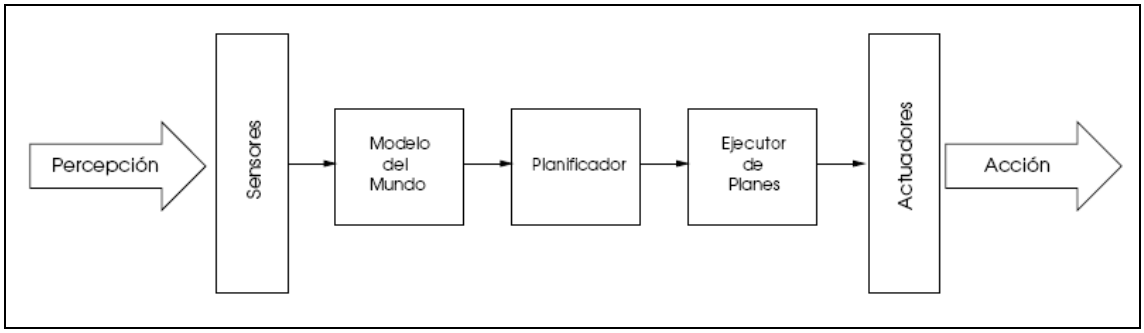


Figura 2.1. Diagrama estructural de la arquitectura de un agente deliberativo.

En la figura 2.1 el agente recibe un estímulo del exterior que percibe por los sensores. A partir de este momento realiza la planificación de sus acciones hasta producir como resultado la acción deseada.

2.3.3.2 Agentes reactivos

Los agentes reactivos [14], como se ha comentado anteriormente, no presentan representación explícita de conocimiento simbólico complejo. Se trata de ofrecer respuestas inmediatas a estímulos del entorno.

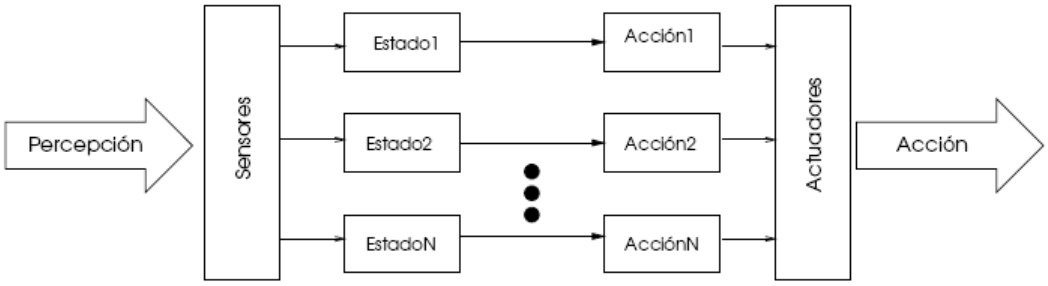


Figura 2.2. Diagrama estructural de la arquitectura de un agente reactivo.

Se trata de agentes que tienen una percepción concreta a partir de la cual se disparan las acciones pertinentes, dependiendo de las condiciones activadas en el proceso perceptivo. Este tipo de agentes puede implementarse usando una infraestructura de comunicación a través de mensajes ACL.

En la figura 2.2 puede observarse como percibe por los sensores los estímulos que le hacen realizar una serie de acciones y pasar por unos estados hasta producir como resultado la acción deseada.

2.3.3.3. Agentes híbridos

Existe un enfoque híbrido que combina aspectos de las arquitecturas deliberativa y reactiva. Un agente híbrido [14] típico dispone de componentes deliberativos que permiten realizar razonamientos complejos, realizar planes y tomar decisiones; combinado con componentes reactivos que permiten la reacción inmediata ante eventos para los cuales es necesario ese tipo de reacción.

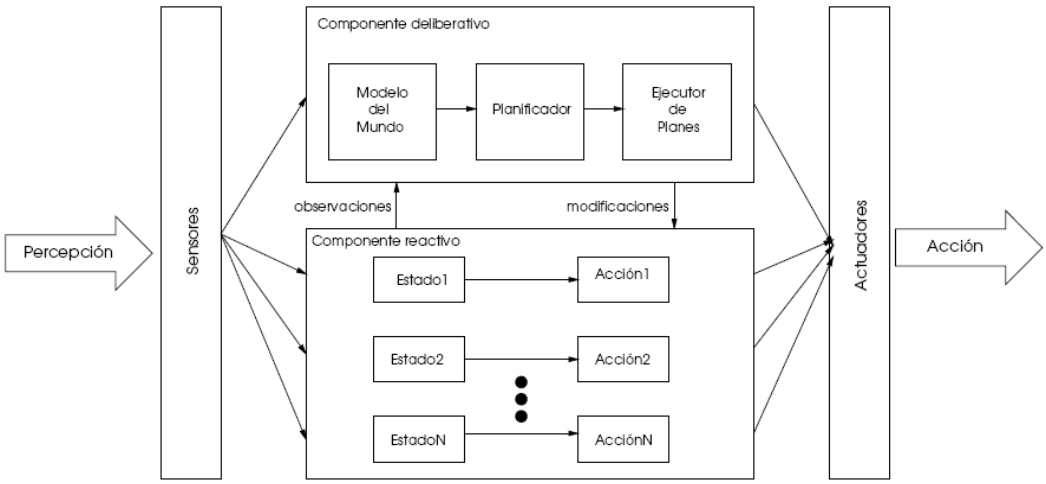


Figura 2.3. Diagrama estructural de la arquitectura de un agente híbrido.

En la figura se puede observar cómo la estructura del agente híbrido está compuesta por un componente deliberativo y un componente reactivo. Ambos componentes se intercambian información: observaciones y modificaciones.

2.4. La arquitectura FIPA

En esta tecnología de agentes dos de los problemas que hay que resolver para facilitar su aplicabilidad son la interoperabilidad (facilidad de conexión e integración de sistemas basados en dicha tecnología) y la apertura (posibilidad de extensión). Es por ello por lo que es importante disponer de estándares y en el tema de los agentes la organización que los establece es FIPA (*Foundation for Intelligen Physical Agent*)**.**

En las especificaciones FIPA [14] se definen diversas características que deben cumplir las plataformas de gestión de sistemas multiagentes. FIPA define sólo el comportamiento externo (interfaz), dejando las decisiones de diseño a cada equipo de desarrollo. Otro principio del estándar es conseguir un sistema totalmente abierto de tal forma que sistemas heterogéneos puedan interactuar a nivel de sociedades de agentes. FIPA establece el modelo lógico referente a la creación, destrucción, registro, localización y comunicación de agentes.

En el *capítulo 3* se detalla el funcionamiento del estándar FIPA en la plataforma de agentes JADE siendo esta plataforma de agentes el núcleo del modelo de referencia de FIPA.

2.5. Comunicación entre agentes

La solución de un problema de forma distribuida entre un grupo de agentes requiere de la *coordinación* [14] de la actividad de los mismos, y a veces, de la *cooperación* entre alguno de ellos. Para que esto sea posible es necesario establecer nodos y *mecanismos de comunicación* entre agentes.

Para solucionar el problema de la comunicación han surgido una serie de iniciativas que han tratado de especificar una serie de estándares que consiguen una interacción real y homogénea entre sistemas tan distintos como son los sistemas multiagente. En el *apartado 2.4.*se ha comentado la arquitectura FIPA y en el *capítulo 3* se detallará el lenguaje de comunicación de agentes (ACL) que sigue los estándares de FIPA.

Además de FIPA existen otras organizaciones que trabajan en crear estándares para la comunicación entre agentes, como puede ser el grupo de *Knowledge Sharing* del DARPA (*Defense Advanced Research Projects Agency*). Este grupo creó el lenguaje KQML (*Knowledge Query and Manipulation Language*) a partir del cual se basa FIPA ACL.

2.6. Coordinación en los sistemas multiagente

Los *Sistemas Multiagente* (SMA) [14] suelen concebirse a menudo como sistemas software compuestos de múltiples entidades computacionales encapsuladas, embebidos en un entorno e (inter-)actuando de forma autónoma e inteligente.

La *coordinación* puede entenderse como la gestión de dependencias entre agentes y es característica clave de los SMA y por otro lado, la capacidad de coordinarse con los demás es una propiedad clave de cualquier agente.

Una característica fundamental de cualquier mecanismo de coordinación es identificar la necesidad de coordinarse, y determinar las diferentes alternativas para hacerlo, es decir: *establecer el espacio de coordinación*.

El tipo de coordinación que llevan a cabo los agentes del prototipo basado en la *Negociación* que puede encontrarse en el *apartado 2.6.* En el *capítulo 4* cuando se plantea el diseño Web se especifica que será necesaria además la *coordinación mediante agentes intermediario* .

2.7. Negociación

Los mecanismos de negociación [14] modelan la coordinación entre agentes auto-interesados capaces de llegar a acuerdos vinculantes. Estos mecanismos permiten la convergencia hacia decisiones de coordinación conjuntas mediante una comunicación explícita.

Tres aspectos básicos de la negociación son:

- **Lenguaje:** trata de las primitivas de comunicación para negociar (proponer, refinar, confirmar, etc.), su semántica, el objeto de la negociación (plan, oferta de tarea, etc.) y los protocolos de negociación.

- **Decisión:** trata de qué aspectos tienen en cuenta los agentes para decidir en la negociación cómo maximizar una función de utilidad, preferencias, estrategias de negociación (ser cooperativos, competitivos, etc.).

- **Proceso:** estudio de modelos generales del proceso de negociación y de la conducta global de los participantes.

La *complejidad* de la negociación dependerá de los atributos involucrados, los tipos de valores y el número de agentes que negocian (uno a uno, uno a muchos, muchos a muchos).

El mecanismo usado en este proyecto es el **modelo de Mercado** [14]. Este modelo de coordinación equipara un SMA con un mercado virtual, con el objetivo de que dicho sistema tenga algunas de las propiedades deseables de los mercados reales. Existirán una serie de agentes vendedores que ofrezcan unos productos y unos agentes compradores que deseen conversar con los vendedores para realizar la compra de esos productos.

2.8. Comercio electrónico

En el proceso del comercio se encuentran enfrentados: empresa y consumidor, cada uno con unos intereses distintos. Por un lado, las metas de las empresas (vendedores) consisten básicamente en: conseguir la fidelización del cliente, aumentar el nivel de ventas y conocer al cliente. Por otro lado el consumidor (compradores) desea recibir un servicio personalizado, obtener recomendaciones y tener un acceso fácil a la información. Ambos, compradores y vendedores, necesitan automatizar el manejo de sus transacciones comerciales.

El Comercio Electrónico o **e-commerce** puede definirse como el intercambio comercial de valor (productos, servicios e información) en el que algunas o todas las fases se desarrollan mediante redes de comunicación (por ejemplo, Internet).

El **comercio electrónico** ha ido evolucionando con el tiempo pudiendo hacerse una distinción entre dos generaciones. La p*rimera generación:* simplemente el negocio electrónico consistía en la navegación por un catálogo en línea donde el usuario podía consultar los productos de los vendedores. La *segunda generación:* se amplía este concepto y se pretende analizar los comportamientos del usuario en la Web: identificación de las necesidades, búsqueda del producto, búsqueda de vendedor, negociación, compra y recepción, evaluación del producto y del servicio. Existen distintos tipos de mercados electrónicos:

- **B2B (Business to Business).** Se refiere a las transacciones económicas o las relaciones efectuadas entre empresas.

- **B2C (Business to Consumer).** Es el comercio entre empresas y particulares. El éxito en Internet está potenciado por el aseguramiento de los sistemas de pago.

- **C2C (Consumer to Consumer).** Se refiere a las transacciones privadas entre consumidores que pueden tener lugar mediante el intercambio de correos electrónicos o el uso de tecnologías **P2P** (Peer to Peer).

2.8.1. Roles de los agentes

Los agentes para el comercio electrónico pueden asumir distintos roles:

- **Agentes notificadores.** Son los encargados de notificar a sus usuarios la aparición y/o detección de productos acordes a sus preferencias o necesidades. Esta notificación puede realizarse por correo electrónico o sms. También se denominan servicios de alerta.

- **Agentes de recomendación.** Tienen como misión realizar recomendaciones a los usuarios de productos que podrían interesarles, basándose en su perfil y en el conocimiento del contexto de negocio.

- **Agentes de compra comparativa.** Su objetivo es encontrar al comerciante que ofrece las mejores condiciones de compra de un producto deseado por el usuario. Posee limitaciones como son los prejuicios del consumidor y que no sólo se deberían realizar la comparativa de los productos basándose en el precio.

- **Agentes de subasta.** Permiten a los usuarios realizar pujas en la red existiendo agentes vendedores y compradores para los cuales se deben establecer unos parámetros.

- **Agentes de negociación.** Este es el tipo de agentes con los que se trabajará en este proyecto. Pretenden trasladar al mercado electrónico los procesos de negociación que se producen normalmente a la hora de realizar una transacción comercial. Existen agentes compradores y vendedores que colaboran para llegar a algún acuerdo en las condiciones de adquisición de un producto.

2.8.2. Estrategias en los mercados electrónicos

El proceso de negociación entre agentes se ha de producir teniendo cada uno de los agentes unas estrategias definidas para interactuar en ese proceso de negociación en el cual ellos mismos son los que intentan vender el producto. Se dan de alta en el mercado, contactan con los agentes interesados y negocian para obtener el mejor acuerdo.

Se pueden clasificar las estrategias como [9]: '*Homogéneas*' donde los agentes usan una única táctica y las '*Flexibles*' donde se combinan varias tácticas a la vez.

Analizando el Mercado se puede distinguir entre mercado no controlado y mercado controlado.

- Un **Mercado Electrónico No Controlado** es aquel en el que cada participante inicializa un agente que actúa en el interés de su propietario y hace uso de estrategias definidas exclusivamente por él. No existe ningún ente que controle al mercado. Es difícil de implementar.

- En un **Mercado Electrónico Controlado** los participantes tienen que llegar a un acuerdo acerca de qué se compra y vende y cómo puede hacerse. Por tanto, el proceso de negociación requiere tres partes :

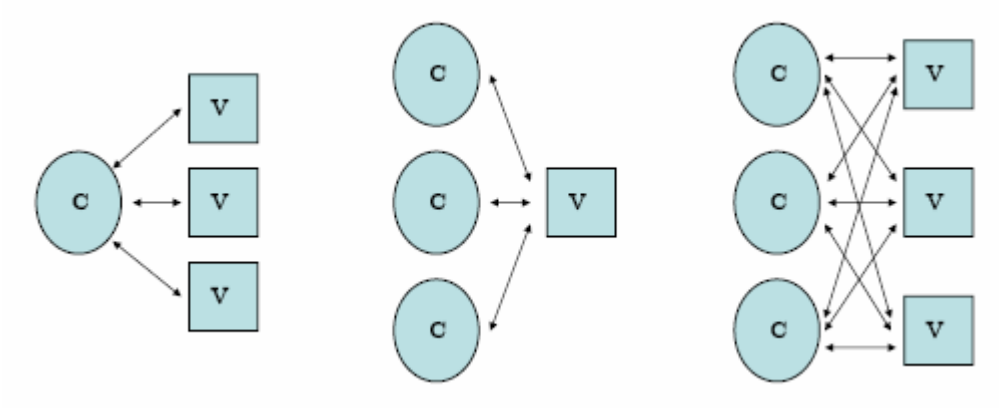
    - *Protocolos de Negociación.* Definen el conjunto de reglas que gobiernan la interacción de los agentes.

    - *Objetos de la Negociación.* Definen el rango de aspectos sobre los que debe alcanzarse un acuerdo.

    - *Modelo de Toma de Decisiones de los Agentes*. Describe el funcionamiento de la toma de decisiones de los agentes y el protocolo de negociación necesarios para conseguir los objetivos.

Una vez definidos estas tres partes, el descubrimiento de información, la comunicación, las ontologías y la monitorización se simplifican bastante en comparación con los mercados no controlados.

Se pueden presentar distintas topologías en cuanto a los tipos de mercado y negociaciones.



Modelo A Modelo B Modelo C

Figura 2.4. Modelos de negociación competitiva en un mercado electrónico.

En la figura anterior se presentan las constelaciones de compradores y vendedores que son posibles en los diferentes tipos de mercados y las comunicaciones que se producen entre los agentes compradores y los agentes vendedores. En el modelo A se produce una negociación de 1 comprador con 'n' vendedores; en el modelo B se produce una negociación de 1 vendedor con 'n' compradores y en el modelo C la negociación se producen entre 'n' compradores y 'm' vendedores.

Dado un tipo de mercado, se debe conocer el rango de aspectos sobre los que se quiere alcanzar un acuerdo. El caso más sencillo puede ser una negociación basada en el precio; mientras que en una negociación B2B la negociación puede cubrir varios aspectos distintos (precio, calidad, penalizaciones, condiciones, plazos,…). Se puede hacer una distinción entre las negociaciones: de *un solo atributo* y *multiatributo.*

Otra catalogación se puede hacer en función de las restricciones sobre el rango de valores de los diferentes aspectos sobre los que se negocia, estableciéndose: *restricciones extrictas* y *restricciones difusas*. En el marco de restricciones extrictas el problema se simplifica dado que todo se reduce a una simple comparación de preferencias entre las partes que negocian. Sin embargo, en una negociación real, los acuerdos se alcanzan mediante la realización de concesiones.

Según lo expuesto puede realizarse la siguiente clasificación de los mercados controlados:

criterio	valores posibles		
tipo de mercado	B2B	B2C	C2C
tipo de modelo	1:n (A)	m:1 (B)	n:m (C)
atributos de negociación	un atributo	multiatributo	
tipo de restricciones del consumidor	estrictas	difusas	
tipo de restricciones del vendedor	estrictas	difusas	

Figura 2.5. Estructura de los mercados controlados.

En un mercado real, cada participante se ve implicado normalmente en una negociación multilateral sobre múltiples atributos. Este proceso negociador es un proceso de concesiones mutuas que giran entorno a unas determinadas estrategias. Las concesiones se producirán según la flexibilidad de las restricciones establecidas en los parámetros. Este tipo de negociación correspondería por ejemplo al modelo C2C con una negociación multiatributo y restricciones difusas en ambos lados.

En función de los criterios expuestos existe una gran cantidad de mercados electrónicos que pueden diseñarse y hay muchos en funcionamiento. En mercados B2B y con el Modelo A puede destacarse MAGMA [*Tsvetovatyy et al., 1997*], MAGNET [*Collins et al.,1998*] Y MASCOT [*Sadeh et al., 1999*] de subastas electrónicas.

Las subastas electrónicas son el mercado electrónico con mayor éxito en Internet, tanto en el ámbito B2B, como B2C y C2C. Ejemplos de este tipo de mercado son 'Ebay' y 'Fishmarket'. De todas formas, no parece que la delegación en un agente de las pujas sobre el precio vaya a tener excesivo éxito o por lo menos la decisión del precio final, ya que al consumidor le resulta divertido intervenir en el proceso.

En el modelo C se pueden destacar 'Kasbah'[8] [*Chavez y Maes,199*6], 'Tete-aTete' [Guttman y Maes, 1998] y 'CASBA' [Vetter y Pitsch,1999]. Estos prototipos utilizan restricciones difusas en ambos lados, negocian un único atributo en el caso de Kasbah [8] y CASBA, y múltiples atributos en el caso de Tete-a-Tete (en este caso las ofertas se realizan manualmente).

2.8.3. Diseño de las estrategias

En el diseño de la estrategia hay que establecer la propuesta con la que se va a empezar, cuándo y cuánto hay que ceder en función del riesgo.

El riesgo puede definirse como la pérdida relativa máxima si el agente A cede en la ronda t. (Zeuthen)

$$\text{Riesgo}(A, t) = \frac{\text{Pérdida máxima de A si cede (y acepta la oferta de B)}}{\text{Pérdida máxima de A (si no cede y llega a un conflicto)}}$$

Lo ideal sería que el agente con el menor riesgo (el que realiza la menor pérdida relativa máxima) ceda en el proceso. Si el propio riesgo es igual o más pequeño que el del contrario, entonces hacer la oferta mínima suficiente; de lo contrario, no ceder y repetir la misma oferta realizada.

Los agentes pueden seguir diversas estrategias, las cuales serán opuestas para compradores y vendedores.

Para el diseño de las estrategias se ha seguido el modelo 'Kasbah' [8] a partir del cual surgió la idea de realizar este proyecto. La estrategia del *agente vendedor* consiste en vender el producto al precio deseado antes de la fecha acordada, es decir desea maximizar el precio de venta. Si con el paso del tiempo el vendedor no encuentra agentes compradores dispuestos a pagar el precio deseado, tendrá que ir disminuyendo el precio de venta conforme a la estrategia seleccionada y teniendo en cuenta cuál es el precio más bajo al que está dispuesto a vender el producto. Este proceso puede asociarse a una función, por ejemplo: función lineal, función cuadrática y función cúbica.

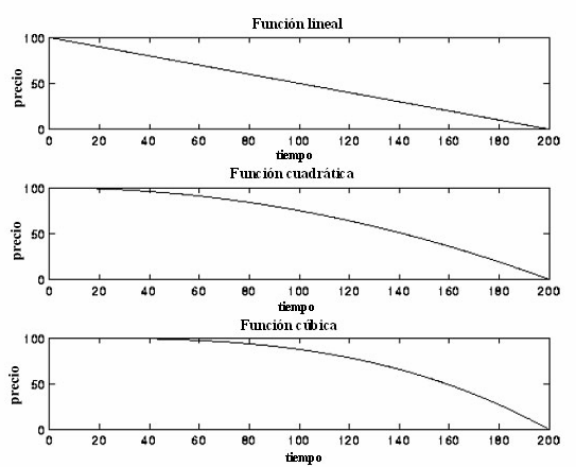


Figura 2.6. Funciones de 'decremento' del precio del producto.

En la figura 2.6 se muestra el funcionamiento de tres funciones de decremento del precio del producto de venta con respecto al paso del tiempo.

De forma análoga al agente vendedor, el *agente comprador* tendrá una estrategia para determinar el incremento del precio que está dispuesto a pagar por el producto. En este caso las estrategias serían similares a las del vendedor pero en sentido ascendente ya que se desea minimizar el precio final que ha de pagar por el producto.

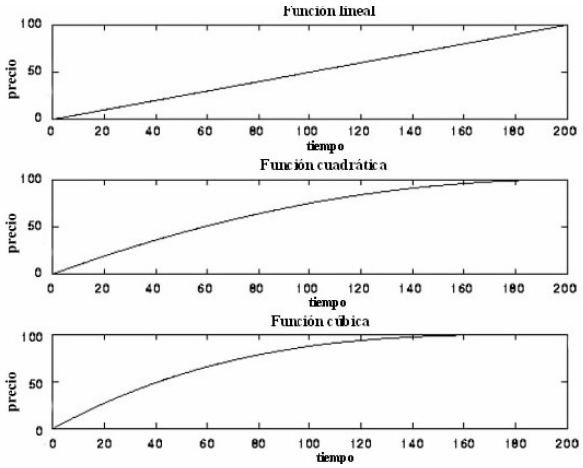


Figura 2.7. Funciones de 'incremento' del precio del producto.

Capítulo 3

La Plataforma JADE y los Agentes

Este capítulo está dedicado a explicar en detalle el funcionamiento de los agentes en la plataforma JADE. Es importante conocer cómo se comportan los agentes en la plataforma y qué posibilidades se ofrecen para el desarrollo del proyecto.

3.1. Definición y características

JADE (*Java Agent Development Framework*) es un software para el desarrollo de sistemas multiagente diseñado bajo los estándares FIPA. Proporciona una *plataforma de agentes distribuida* que puede ser ejecutada a lo largo de varios hosts pudiendo ser conectados vía RMI, es decir, solo una aplicación Java, y solo una máquina virtual de Java ejecutada en cada host. Los agentes son implementados como hebras de Java y viven dentro de Contenedores de Agentes que proporcionan el soporte para la ejecución de los mismos.

Posee una *Interfaz Gráfica* para manejar a los agentes y los contenedores de agentes desde un host remoto y *Herramientas de Depuración* para ayudar en el desarrollo de aplicaciones multiagentes basadas en JADE.

Permite la *movilidad de agentes* (en el *apartado 2.3.3* ya se hablaba de agentes móviles) entre las plataformas y el *registro y desregistro* automático de los agentes con el agente AMS.

Posee soportes para el transporte de *mensajes ACL* dentro de la misma plataforma de agentes, para aplicaciones definidas mediante *lenguajes* y *ontologías* y para la ejecución múltiple, paralela y concurrente de actividades de agentes mediante los comportamientos (*behaviours*).

Con respecto a los estándares FIPA (*organización de estándares de la sociedad de los computadores de IEEE que promueve las tecnologías basabas en agentes y la interoperabilidad de sus estándares con otras tecnologías*) proporciona:

- Una *plataforma de agentes* que incluye el ***AMS*** (*Agent Management System*), el ***DF*** (*Directory Facilitator*), y el ***ACC*** (*Agent Communication Channel*). Todos estos componentes son automáticamente activados cuando la plataforma se pone en funcionamiento.

- Una *librería FIPA de protocolos* de interacción listos para ser usados.

- *Servicio de Nombres FIPA*: cuando un agente comienza se le asigna un GUID (*Globally Unique Identifier*) desde la plataforma, con el cual se distinguirá de forma unívoca del resto de agentes.

3.2. Composición de la plataforma JADE

JADE está escrito en lenguaje Java y está formado por varios paquetes que proporcionan a los programadores las piezas de la funcionalidad y las interfaces abstractas para los usuarios.

Java fue el lenguaje de programación elegido porque posee unas características muy atractivas, entre ellas: la serialización de objetos, *Reflection API* y Método Remoto de Invocación (RMI).

JADE está compuesto por un **conjunto de paquetes**, entre ellos cabe destacar:

- **jade.core** que implementa el kernel del sistema. Incluye la clase **Agent** que debe ser extendida por las aplicaciones; además, una clase comportamiento (**Behaviour)** que está contenida en el subpaquete **jade.core.behaviours**. Los comportamientos implementan las tareas, o intenciones, de un agente las cuales son acciones lógicas que pueden estar formadas por varios caminos de ejecución y pueden ser ejecutadas concurrentemente.

- El **jade.lang.acl** es el subpaquete que proporciona el proceso de comunicación entre agentes (Agent Communication Language, ACL) de acuerdo con las especificaciones de los estándares FIPA.

- El paquete **jade.content** contiene un conjunto de clases que soporta la definición de las ontologías y el contenido de los lenguajes.

- El paquete **jade.domain** contiene todas las clases Java que representan el manejo de los agentes definidos en los estándares FIPA, en particular el "*AMS*" y el "*DF Agents*" que proveen el ciclo de vida a los agentes y los servicios de páginas blancas y amarillas.

- El paquete **jade.mtp** contiene una interfaz de java para el protocolo de transporte de mensajes (*Message Transport Protocol*).

- El paquete **jade.wrapper** contiene funcionalidades de alto nivel de JADE que permiten usar JADE como una librería donde las aplicaciones externas de Java puedan lanzar agentes JADE y contenedores de agentes.

3.3. Herramientas

Las herramientas de la plataforma se encuentran en el subpaquete **jade.tools** y son las siguientes:

- **Remote Management Agent (RMA).** Es el gestor de la plataforma.

- **Dummy Agent.** Permite interactuar con los agentes; usar la interfaz para mandar mensajes ACL a otros agentes.

- **Sniffer.** Es un agente que puede interceptar los mensajes ACL mientras se están enviando de un agente a otro y mostrar gráficamente este proceso en un diagrama de secuencia UML. Esto es útil para depurar los mensajes que se envían entre los agentes.

- **InstrospectorAgent.** Permite monitorizar y controlar el ciclo de vida de un agente.

- **SocketProxyAgent.** Es una agente simple activado como una puerta de comunicación bidireccional entre la plataforma JADE y el protocolo TCP/IP.

- **DF Gui.** Es una interfaz que entre otras opciones permite ver las descripciones de los agentes registrados.

3.4. La Plataforma Agent

La plataforma JADE está organizada en *contenedores*; por un lado, un contenedor principal (*main-container*) donde están localizados el AMS, DF y el registro RMI; y 'n' contenedores secundarios y conectados al principal.

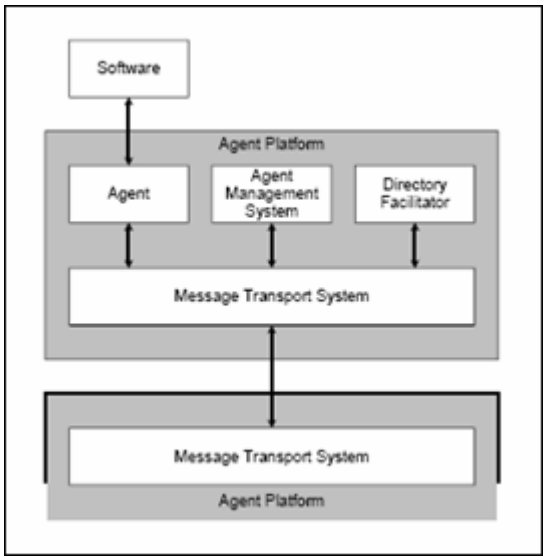


Figura 3.1. Referencia de arquitectura de una Plataforma de Agentes FIPA.

El Agente Gestor del Sistema (**Agent Management Sistema, AMS**) es el agente que realiza el proceso de supervisión para acceder y usar la Plataforma de Agentes. Solo un existirá un AMS en cada plataforma. Los servicios que ofrece son:

- Creación, destrucción y control del cambio de estado de los agentes.
- Supervisar los permisos para que nuevos agentes se registren.
- Control de la movilidad de los agentes.
- Gestión de recursos compartidos.
- Gestión del canal de comunicación.
- Servicio de nombres (ANS) o Páginas Blancas (Nombre-Dirección).

El **Directory Facilitator (DF)** es un agente que provee el servicio de páginas amarillas en la plataforma. Los agentes se registran indicando los servicios que ofrecen. Al darse de alta los agentes en la plataforma solicitan un servicio y se busca cuales son los agentes que lo ofrecen.

El Sistema de Transporte de Mensajes, también llamado **Canal de Comunicación de Agentes (ACC**), es el componente software que controla todos los intercambios de mensajes dentro de la plataforma, incluidos los mensajes de plataformas remotas. Realiza una comunicación asíncrona.

Cuando la plataforma JADE se lanza, AMS y DF son inmediatamente creados y el módulo del ACC se inicializa para permitir la comunicación mediante mensajes. La plataforma de agentes puede ser separada en varios host. Cada JVM es un simple contenedor de agentes que provee del entorno de ejecución de los agentes y permite que se ejecuten varios agentes de forma concurrente en el mismo host.

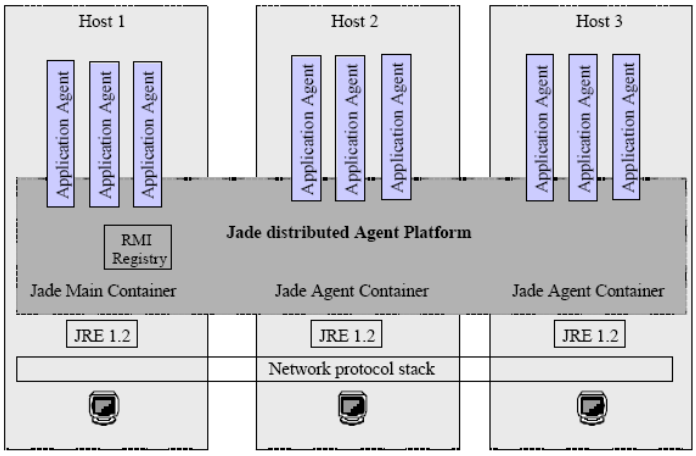


Figura 3.2. Plataforma JADE de agentes distribuida sobre varios contenedores.

El contenedor principal (***main-container***) es el contenedor de agentes donde el AMS y el DF están en funcionamiento y donde el Registro RMI, que es usado internamente por JADE, es creado. Los otros contenedores de agentes, además, conectan con el contenedor principal y proporcionan un completo entorno de ejecución para otro conjunto de agentes de JADE.

3.5. La clase Agent

Representa una clase base para la definición de los agentes. De esta manera, un agente de JADE es una simple instancia de una clase de Java que extiende de la clase base Agent. Esto implica la herencia de características básicas de interacción con la plataforma de agentes (registro, configuración, control remoto,…) y un conjunto básico de métodos que pueden ser llamados para implementar un comportamiento del agente (enviar/recibir mensajes, usar el estándar de protocolos de interacción,…).

El modelo computacional de un agente es multitarea, donde las tareas/comportamientos son ejecutadas concurrentemente. Cada funcionalidad/servicio proporcionado al agente debe ser implementado como uno o más comportamientos (behaviours).

3.6. Ciclo de vida del Agente

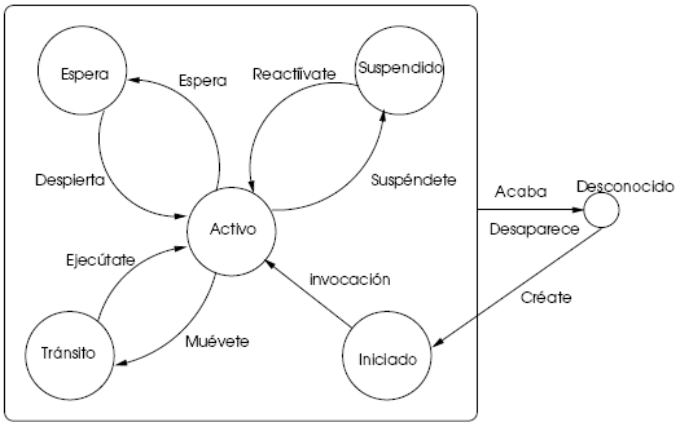


Figura 3.3. Ciclo de vida del Agente definido por FIPA.

Un agente de JADE puede encontrarse en varios estados, de acuerdo al Ciclo de Vida de los agentes establecido en las especificaciones FIPA; estos son representados como algunas constantes en la clase *Agent*. Los estados son:

- **Iniciado** (AP_INITIATED): el objeto Agente es construido, pero aún no ha sido registrado en el AMS, no tiene aún un nombre ni una dirección con lo cual no puede comunicarse con otros agentes.

- **Activo** (AP_ACTIVE): el objeto Agente es registrado con el AMS, obteniendo un nombre y una dirección pudiendo acceder a todas las características de JADE.

- **Suspendido** (AP_SUSPENDED): el objeto Agente está actualmente parado. Su hebra interna está suspendida y no está ejecutándose ningún comportamiento.

- **Espera** (AP_WAITING)*:* el objeto Agente está bloqueado mientras ocurra alguna acción determinada que lo despierte, como por ejemplo la recepción de un determinado mensaje.

- **Eliminado** (AP_DELETED)**:** el objeto Agente está muerto. Su hebra interna ha terminado su ejecución y el Agente no está más registrado en el AMS.

- **Tránsito** (AP_TRANSIT)*:* los agentes entran en este estado mientras están siendo enviados a una nueva localización. Son los agentes móviles.

- **Copiado** (AP_COPY): este estado es usado internamente por JADE para realizar una clonación del agente.

- **Migrado y Estable** (AP_GONE): este estado es usado internamente por JADE cuando un agente móvil ha migrado a una nueva localización y tiene un estado estable.

La clase Agent proporciona métodos públicos para realizar las transiciones entre los distintos estados; esos métodos toman sus nombres según las especificaciones de FIPA de manera que los métodos se componen de "do" seguido del nuevo estado que obtiene el agente. Ejemplo: el método *"DoWait()"* hace pasar al agente del estado AP_ACTIVE al estado AP_WAITING.

3.7. Ejecución del Agente

El marco de trabajo de JADE controla el nacimiento de nuevos agentes de acuerdo a los siguientes pasos:

- Se ejecuta el constructor del Agente.
- El agente recibe un identificador único (AID).
- El agente es registrado en el AMS.
- El agente adopta el estado AP_ACTIVE.
- Se ejecuta el método de inicialización "*setup()*".

De acuerdo a las especificaciones FIPA un agente es identificado por los siguientes atributos:

- Un **nombre único.** Por defecto JADE crea este nombre compuesto del nombre local más el símbolo "*@*" más el identificador de la plataforma del agente. Ejemplo: *Mercado@home-2007:1099/JADE*.

- Un **conjunto de direcciones de agentes**. Cada agente hereda las direcciones de transporte de la plataforma donde se haya creado.

- Un **conjunto de proposiciones**, como por ejemplo el servicio de páginas blancas donde el agente es registrado.

El método "*setup()"* es por lo tanto el punto donde comienza toda actividad del agente. Cuando el método "*setup()*" es ejecutado, el agente ya ha sido registrado en el AMS y su estado en la plataforma es AP_ACTIVE. En este método se debe realizar:

- De manera voluntaria el registro del agente en el AMS y establecer la descripción del agente y proveerle de servicios y si es necesario de uno o más dominios.

- De manera obligatoria añadir las tareas a la cola de tareas listas usando el método "*addBehaviour()*". Estos métodos comienzan a ejecutarse tan pronto como se haya ejecutado el método "*setup()*".

Para parar la ejecución del Agente se usa el método "*doDelete()*" pasando al estado P_DELETED.

3.8. Mensajes ACL

En el *apartado 2.5* sobre la comunicación entre agentes se hizo referencia a distintos estándares entre ellos el ACL. En este apartado se detalla este estándar ue ha sido el empleado en la comunicación entre los agentes del proyecto.

ACL [14] define la estructura que deben de tener los mensajes que se envíen los distintos agentes a través del AMS (*Agent Management System*). Un mensaje FIPA ACL contiene un conjunto de uno o más elementos de mensaje de los que no todos son obligatorios:

- *Performative:* indica el tipo de acto comunicativo del mensaje.
- *Sender:* el emisor del mensaje.
- *Receiver:* el receptor del mensaje.
- *Reply-to:* el receptor de la réplica al mensaje.
- *Content, language, enconding, ontology:* el contenido del mensaje, el lenguaje en el que va a ser representado, la codificación y la ontología a la que pertenecen los datos del mismo, respectivamente.
- *Protocol:* el protocolo de conversación subyacente entre los agentes implicados.
- *Conversation-id:* identificador de la conversación particular a la que pertenece el mensaje.
- *Reply-with:* identificador de mensaje que ha de llevar la réplica a este.
- *In-reply-to:* identifica la réplica con respecto al mensaje original.
- *Reply-by:* dato de tiempo o fecha, dentro de la cual el agente emisor desearía recibir la réplica.

Una agente que envíe un mensaje debe crear un nuevo objeto del tipo ACLMessage, rellenar sus atributos con los valores apropiados y finalmente llamar al método "*Agent.send()*" . El agente receptor deberá llamar al método "receive()" o al método "blockingReceive()", ambos implementados en la clase Agent.

El envío y recepción de mensajes puede producirse con independencia de las actividades del agente añadiendo los comportamientos de *ReceiverBehaviour* y *SenderBehaviour* a la cola de tareas del agente. Todos los atributos de un mensaje ACL son accesibles mediante determinados métodos (*set/get <atribute> ()*).

3.8.1. Tipos de Mensajes ACL

Los tipos de Mensajes [11] que pueden enviarse con ACL y su significado son los siguientes:

- **Accept-proposal.** Aceptación de una propuesta para realizar cierta acción, utilizando como propuesta a *propose* (aceptar una propuesta).
- **Agree.** Acuerdo para realizar cierta acción y es utilizado en respuesta a request (aceptar petición).
- **Cancel**. Cancelar una petición.
- **Call-for-proposal (CFP).** Pide candidaturas para realizar cierta acción, por lo que inicia procesos de negociación (pedir propuestas para realizar una acción).
- **Confirm.** Informa al receptor de que cierta proposición es cierta, siempre que sea cierto que el receptor tiene incertidumbre sobre ella.
- **Disconfirm.** Contrario de Confirm.
- **Failure.** Informa que se intentó cierta acción pero falló (explica por qué ha fallado la acción).
- **Inform.** Para dar información al receptor.
- **Inform-if**. Informa al receptor de su creencia o descreencia de cierta proposición.
- **Inform-ref**. Comunica al receptor una descripción referencial de un objeto (por ejemplo, un nombre).
- **Not-understood.** Informa al receptor que recibió cierta petición de acción pero que no la entendió.
- **Propagate.** El receptor recibe un mensaje y además debe propagarlo al resto de la lista de receptores.
- **Propose.** Hacer una propuesta para realizar cierta acción bajo ciertas precondiciones (hacer una propuesta).
- **Proxy.** El emisor quiere que el receptor seleccione agentes objetivo a partir de cierta descripción y les envíe cierto mensaje.
- **Query-if.** Pregunta a otro agente si cierta proposición es cierta o no (preguntar al receptor alguna cosa).
- **Quero-ref**. Pide al otro agente un objeto denotado por cierta expresión referencial.
- **Refuse.** Acción de rehusar realizar cierta acción (no aceptar petición).
- **Reject-proposal.** Acción de rechazar una propuesta para realizar alguna acción durante una negociación (rechazar una propuesta).
- **Request.** El emisor desea que el receptor realice cierta acción.
- **Request-when**. Se desea que el receptor realice cierta acción tan pronto cuando cierta proposición sea cierta.
- **Request- whenever.** Igual que request-when pero continuamente.

- **Subscribe.** Suscripción a un servicio del receptor, de manera tal que cuando ocurra algo que interese al emisor el receptor lo notificará.

3.8.2. La Clase MessageTemplate

El modelo de comportamiento de JADE permite que un agente ejecute varias tareas en paralelo. De todas formas cada agente debería poseer la capacidad de llevar varias conversaciones a la vez.

La clase *MessageTemplate()* permite construir patrones para poder trabajar con varios mensajes simultáneamente.

3.9. Los comportamientos del agente

Las tareas o servicios que un agente hace realmente se especifican en comportamientos. Un *comportamiento* hace referencia a una funcionalidad que incorpora el agente. Un agente debe poder llevar a cabo varias tareas concurrentemente. Para hacer que la gestión del agente sea eficiente, todos los agentes de JADE están compuestos por una simple hebra de ejecución y todas las tareas/comportamientos son diseñados y pueden ser implementados mediante objetos *Behaviour*.

Los agentes multi-hebra pueden ser implementados pero JADE no proporciona un soporte específico (salvo la sincronización de la cola de mensajes ACL).

Si se quiere implementar una tarea específica para un agente deben definirse una o más subclases de *Behaviour* , instanciándolas y añadiendo los comportamientos como objetos a la lista de tareas del agente.

La clase Agente posee dos métodos: *addBehaviour(Behaviour)* y *removeBehaviour(Behaviour)* que permiten manejar las tareas listas de la cola de un agente específico. Estos comportamientos y sub-comportamientos pueden añadirse donde sea necesario y no obligatoriamente en *Agent.setup()* .

Puede ser implementada una clase Agente y llevar a cabo una política de programación de las tareas mediante el método round-robin para los comportamientos de la cola, ejecutar una clase derivada de Behaviour hasta que el control sea liberado (esto ocurre cuando se ejecuta el método *action()*. Si el abandono de la tarea no ha sido completo, se programará para la siguiente ronda. Un comportamiento también puede ser bloqueado por la espera de mensajes. El agente planificador ejecuta el método *action()* de cada comportamiento presente en la cola de

comportamientos listos; cuando finaliza el método *action()* , se llama al método *done()* para chequear si la tarea del comportamiento ha sido completada. Si es así, el objeto de este comportamiento es eliminado de la cola.

Para evitar una espera activa para los mensajes, a cada comportamiento simple se le permite bloquear esta computación. El método *block()* coloca el comportamiento de una cola comportamientos bloqueados tan pronto como se retorna del método *action().* Por lo tanto, el bloqueo no es alcanzado inmediatamente tras la llamada a *block()* , sino justo después de retornar del método *action().* Todos los comportamientos bloqueados son reorganizados tan pronto como llega un nuevo mensaje, por lo que hay que tener cuidado si bloquea de nuevo un comportamiento si este no está esperando la llegada de mensajes. Un comportamiento puede bloquearse a sí mismo por un tiempo limitado mediante el método *block().*

Debido al tipo de modelo multitarea elegido de antemano para el comportamiento de los agentes, se debe evitar el uso de bucles infinitos para largas operaciones dentro del método *action().* Hay que tener en cuenta que algunos de los métodos *action()* de los agentes está en continuo funcionamiento, otros pueden llegar al final de la ejecución del método.

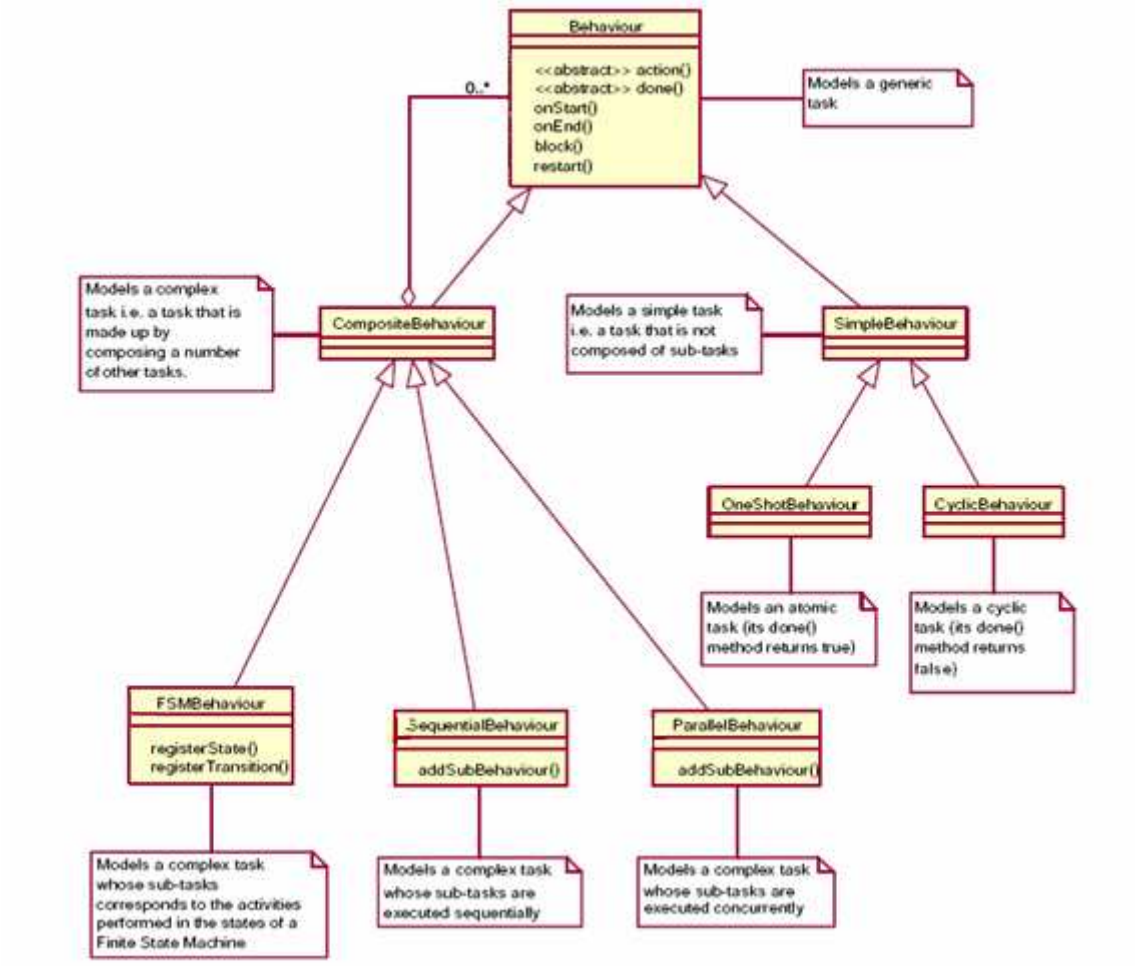


Figura 3.4. Modelo UML sobre la jerarquía de la Clase Behaviour.

Clase Behaviour

Esta clase abstracta proporciona una clase base para modelar las tareas de los agentes, y un conjunto de bases para la programación de los comportamientos como permitir las transiciones entre estados (por ejemplo, comenzar, bloquear y restaurar un comportamiento). Es decir, que el código que se ejecute en este tipo de comportamientos dependerá del estatus del agente, y eventualmente finalizarán su ejecución.

El método *block()* permite bloquear un objeto comportamiento hasta que ocurra algún evento (normalmente la llegada de un mensaje. Un comportamiento bloqueado puede volver a continuar su ejecución en uno de estos tres casos:

- Cuando recibe un mensaje ACL.
- Cuando alcanza el tiempo máximo que puede estar bloqueado el método *block()* expira.
- Cuando se realiza una llamada al método *restart().*

Class SimpleBehaviour

Esta clase abstracta modela los comportamientos simples y atómicos. En el método *reset()* no se hace nada por defecto, pero pueden sobrescribirse en las subclases definidas por el usuario.

Class OneShotBehaviour

Esta clase abstracta modela los comportamientos atómicos que deben ser ejecutados una sola vez de manera casi instantánea y que no pueden ser bloqueados.

Class CyclicBehaviour

Esta clase abstracta modela los comportamientos atómicos que se ejecutan para siempre. Este tipo de comportamientos nunca son sacados del conjunto de comportamientos del agente y su método *action()* siempre ejecuta el mimo código. Son comportamientos que nunca finalizan.

Class CompositeBehaviour

Esta clase abstracta modela los comportamientos que están compuestos por un conjunto de comportamientos. Todas las acciones a ejecutarse no son definidas en el propio comportamiento compuesto sino en los sub-comportamientos. En el método compuesto existe una política de planificación que establece cuáles de los sub-comportamientos debe ejecutarse cada ronda.

Class SequentialBehaviour

Esta clase es un *CompositeBehaviour* que ejecuta sus sub-comportamientos de forma secuencial y termina cuando todos los sub-comportamientos lo han hecho. Se usa cuando el agente tiene que realizar una tarea compleja que puede ser definida mediante una secuencia de pasos atómicos.

Class ParallelBehaviour

Esta clase en un *CompositeBehaviour* que ejecuta sus sub-comportamientos concurrentemente y termina cuando ocurre una determinada condición en uno de los sub-comportamientos. Finaliza cuando todos los sub-comportamientos están hechos, cuando alguno de los sub-comportamientos ha terminado o cuando un número de sub-comportamientos establecido de antemano por el usuario ha finalizado. Esta clase se usa cuando el agente tiene que realizar una tarea compleja que puede ser expresada como una colección de operaciones paralelas alternativas, con algún tipo de condición de finalización en las subtareas.

Class FSMBehaviour

Esta clase es un *CompositeBehaviour* que ejecuta sus sub-tareas de acuerdo a un estado finito de la máquina (*Finite State Machine*) definido por el usuario**.** Cada sub-tarea representa la activad que debe realizarse con un estado de la FSM y el usuario puede definir las transiciones entre los estados del FSM.

Class SenderBehaviour

Encapsula una unidad atómica la cual realiza la acción de enviar. Extiende de la clase *OneShotBehaviour* y se ejecuta una sola vez.

Class ReceiverBehaviour

Encapsula una operación atómica que realiza la acción de recibir. Esta acción termina cuando se ha recibido un mensaje. Si la cola de mensajes está vacía o no hay mensajes en el parámetro MessageTemplate, el método *action()* llama al método *block()* y retorna.

ClassWakerBehaviour

Esta clase abstracta implementa una tarea *OneShot* que se ejecuta una sola vez antes de sobrepasar el tiempo permitido.

3.10. Interacción de protocolos

FIPA especifica un conjunto de protocolos de interacción estándar. Para la conversación entre los agentes JAVA distingue dos agentes: "*Initiator*" (el agente que inicia la conversación) y "*Responder*" (el agente que responde al agente *Initiato*r).

JADE proporciona comportamientos para ambos roles en conversaciones siguiendo los protocolos de interacción de FIPA que se encuentran en el paquete **JADE.proto**.

Alguno de estos protocolos para controlar la comunicación entre los agentes son: FIPA-Request, FIPA-query, FIPA-propose, FIPA-Request-When,..; que permiten al *Initiator* verificar si el Efecto Racional del acto de comunicación ha sido alcanzado. Porque ellos comparten la misma estructura, JADE proporciona el par de clases *AchieveREInitiator/Responder* los cuales son implementaciones simples homogéneas de todos los tipos de protocolos de interacción.

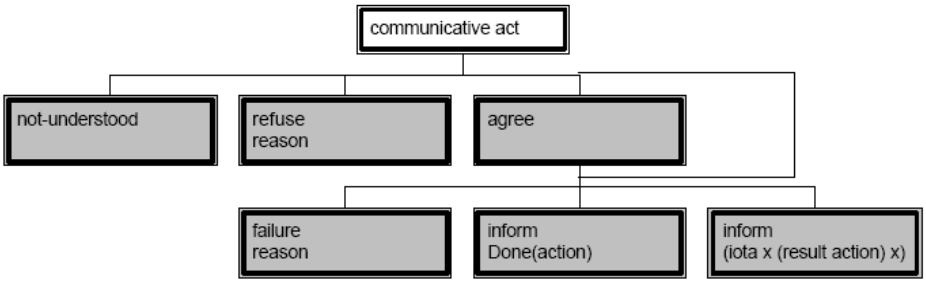


Figura 3.5. Estructura de los protocolos de interacción.

El agente emisor (*Initiator*) envía un mensaje; el Receptor que debe responder puede entonces responder enviando distintos tipos de mensajes: **not-understood, refuse** para alcanzar el Efecto Racional del acto de comunicación, o también un mensaje **agree** para comunicar la aceptación a la petición (posibilidad en el futuro) del acto de comunicación, como se muestra al principio del diagrama de la figura 3.6.

El agente *Responder* realiza la acción y finalmente, debe responder con un mensaje **inform** acerca del resultado de la acción (normalmente justo cuando se ha hecho la acción) o con un mensaje **failure** si algo salió mal en la ejecución del acción.

El Protocolo FIPA-Contract-Net

Este protocolo de interacción permite al "*Initiator*" enviar una llamada de "*Proposal*" (Propuesta) a un conjunto de agentes (Responder), evaluar esas proposiciones y entonces aceptar la "preferida"**.** La interacción de este protocolo está descrita al detalle en las especificaciones de FIPA.

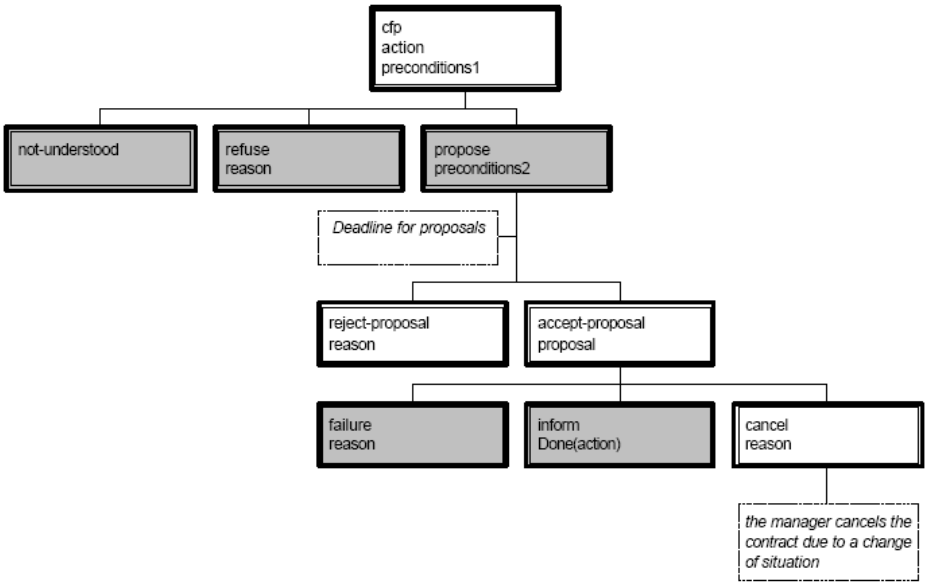


Figura 3.6. Protocolo de Interacción FIPA-Contract-Net.

El *Initiator* solicita ofertas de otros agentes enviando un mensaje CFP (CALL FOR PROPOSAL) que especifica la acción a realizar y si necesita unas condiciones sobre su ejecución. Los agentes que responden (*Responders*) pueden responder enviando un mensaje PROPOSE incluyendo las precondiciones que ellos precisan para realizar la acción, por ejemplo el precio o el tiempo.

Alternativamente, los *Responders* pueden enviar un mensaje REFUSE para comunicar su negativa a la realización de la proposición o un mensaje NO-UNDERSTOOD para comunicar problemas de comunicación. El *Initiator* puede entonces evaluar todas las ofertas recibidas y determinar cuáles aceptará y cuáles no; por ejemplo, si ha recibido un mensaje ACCEPT-PROPOSAL , ha completado su tarea y puede enviar un mensaje INFORM donde informe sobre el resultado de la acción o un mensaje FAILURE si hubo algún error.

Antes de que la acción haya sido realizada y se haya recibido el último mensaje, el *Initiator* puede decidir cancelar el protocolo enviando un mensaje CANCEL aunque no ha sido aún implementado en JADE.

3.11. Soporte para la movilidad de agentes

En el apartado 2.3.2. se han definido los agentes móviles como "*aquellos que poseen la capacidad de desplazarse por los nodos de una red para realizar una determinada tarea*". La plataforma JADE permite construir estos agentes móviles que pueden migrar o copiarse a sí mismos a través de múltiples nodos de una red. A continuación se definen los conceptos de *migración* y *clonación* para entender mejor estos procesos.

El concepto de **Migración** [7,14] es complejo, pues requiere la transferencia de código y datos del agente a la plataforma remota. Cuando un agente va a migrar se "*suspende*", se transfiere por la red y se "*despierta*" en el otro extremo prosiguiendo la ejecución de sus tareas en el mismo punto donde las había dejado.

En la **Clonación** [7, 14] se crea una copia del agente en el nodo remoto que se encarga de hacer el trabajo solicitado por su clon en la máquina origen. No existe ninguna transferencia de código y datos, sino solo un conjunto de órdenes que el clon debe realizar.

3.12. Ontologías

Toda la plataforma FIPA se basa en la comunicación entre agentes mediante los mensajes ACL (descritos en el apartado 3.8). El contenido de un mensaje ACL [14] consiste en una expresión de un determinado lenguaje con un conjunto de términos específicos de la ontología usada. Es decir, cuando dos agentes están conversando es necesario que ambos estén de acuerdo en el lenguaje y la ontología empleados para que puedan comprenderse.

Las ontologías [14] pueden definirse como un conjunto de símbolos o términos junto a su correspondiente interpretación o significado. Cuando dos agentes usan una misma ontología se aseguran de que sus interpretaciones de lo que se está diciendo coinciden.

Una ontología en JADE es una instancia de la clase **JADE.content.onto.Ontology** en la cual se definen los esquemas de la *estructura* de los predicados (expresiones que relacionan conceptos), *acciones* de los agentes y *conceptos* (entidades*)* relevantes al dominio del problema.

Capítulo 4

Análisis y Diseño del Sistema

El desarrollo del software se realizará siguiendo las etapas de la Ingeniería del Software, ocupándose este capítulo de las tres primeras:

- *Análisis del Sistema:* análisis de requisitos del producto software.
- *Especificación del sistema*: detallar el funcionamiento del software.
- *Diseño y arquitectura*: definir los casos de uso para cubrir las funciones que realizará el sistema.
- *Programación:* reducir el diseño a código.
- *Prueba:* comprobar que el software realice correctamente las tareas indicadas en la especificación.
- *Documentación y Mantenimiento*: realizar el manual de usuario y proponer mejoras del software.

4.1. Lenguajes, herramientas y tecnologías

Antes de comenzar el análisis y diseño del sistema es necesario especificar qué lenguajes, herramientas y tecnologías van a ser las empleadas para el desarrollo del mismo.

Plataforma de Agentes JADE

JADE es un marco software con licencia abierta LGPL para desarrollar agentes en los sistemas inteligentes multiagente de acuerdo con los estándares FIPA. Se puede considerar un middle-ware que proporciona una plataforma y un marco de ejecución para los agentes, simplificando el desarrollo mientras que asegura conformidad estándar de un sistema comprensivo de servicios y agentes del sistema.

Lenguaje Java

Es un lenguaje de programación orientado a objetos independiente de cualquier plataforma. Fue desarrollado por Sun Microsystems siendo un lenguaje que en lugar de ser compilado a lenguaje nativo es compilado en un bytecode que es ejecutado por una máquina virtual Java.

Lenguaje HTML

Es el acrónimo de *HyperText Markup Language* (lenguaje de marcas de hipertexto). Es un lenguaje sencillo que permite describir hipertexto, es decir, texto estructurado en el que se incluyen hiperenlaces que conducen a otros documentos, objetos multimedia u otros elementos de información relacionados.

Con el hipertexto lo que se consigue realmente es especificar tanto la estructura lógica del contenido como los diferentes aspectos que se desean dar a dicho contenido. Sin embargo, el uso de CSS nos permite separar ambos conceptos: estructura y aspecto visual.

CSS

Las hojas de estilo (*Cascading Style Sheets*) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML. La idea principal del uso de CSS es separar la estructura de un documento de su presentación, con lo cual HTML define la estructura del documento y las hojas de estilo su aspecto visual.

Se puede definir una *hoja de estilo* es el conjunto de reglas de estilo que definen los formatos de los elementos HTML de la página a la que afecta. Actualmente existen dos versiones: CSS1 y CSS2 desarrolladas por el World Wide Web Consortium (W3C). Además, el W3C proporciona un validador de código en su página oficial para comprobar si el código generado cumple los estándares.

4.2. Análisis

4.2.1. Análisis de requisitos

Este análisis consiste en establecer los servicios que el sistema debe proporcionar y las restricciones bajos las cuales debe operar. Se establece qué debe hacer el sistema y cómo. Se distinguen dos tipos de requisitos [21]: funcionales y no funcionales.

4.2.1.1. Requisitos funcionales

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. El sistema debe permitir:

- El sistema ha de permitir al usuario interactuar con la plataforma de una forma cómoda y simple.

- El usuario podrá: Dar de alta a varios agentes compradores y vendedores en el mercado especificando los parámetros necesarios a través de la interfaz.

- El usuario podrá comprobar el resultado del proceso de negociación y aceptar o no su conformidad final con el acuerdo.

- El sistema deberá permitir realizar pruebas pudiendo observar distintos procesos de negociación y conflictos para comprobar cómo se desenvuelven los agentes y evaluar la estrategia elegida.

4.2.1.2. Requisitos no funcionales

Describen las restricciones sobre los requisitos funcionales. Se pueden englobar en distintas categorías:

Requerimientos de interfaz:

- *Recursos humanos.* La interfaz deberá ser amigable y accesible para lo cual se someterá previamente a un análisis de usabilidad. La aplicación debe ofrecer facilidad de operación, siendo intuitiva contando con una sencilla pero a la vez interfaz.
- *Software.* La aplicación trabajará internamente con la plataforma JADE proporcionando una interfaz donde el usuario pueda observar el proceso.
- *Hardware.* La aplicación necesitará de un equipo que pueda soportar los recursos que son necesarios.
- *Sistema Operativo.* El sistema debe funcionar con cualquier sistema operativo que posea una máquina virtual de Java y la plataforma JADE.

Requerimientos de capacidad:

El sistema necesita que las acciones se realicen rápidamente y la respuesta al usuario mientras está interactuando con la interfaz sea inmediata.

Requerimientos de interacción con el usuario y documentación:

El sistema no implica que los usuarios necesiten una formación específica sino los conocimientos básicos sobre mercado electrónico. Se realizará una documentación completa y simple del sistema, un manual de usuario del sistema así como de la plataforma JADE y la documentación en javadoc del código.

Requerimientos del proceso de desarrollo del proyecto:

La duración del mismo será de unos 4 meses y los costes mínimos ya que se hará uso de plataformas de libre distribución.

Requerimientos de Verificación:

Comprobar que el flujo de información entre la interfaz-usuario y en el interior del sistema se produce correctamente.

Requerimientos de Validación:

Comprobar que la aplicación contiene las opciones necesarias para satisfacer las necesidades expuestas por el usuario.

Requerimientos de pruebas de aceptación:

Realizar comprobaciones acerca del funcionamiento de la aplicación para comprobar si se realiza correctamente. Se deberán realizar pruebas tanto de caja blanca como de caja negra, y crear agentes con valores tanto concretos como incorrectos para verificar el correcto comportamiento de la aplicación en cada caso.

4.2.2.Análisis del sistema: casos de uso

Para comprender mejor el funcionamiento del sistema e identificar las características que debe cumplir el prototipo a diseñar se han especificado a qué tipo de usuarios está destinado y se han modelado una serie de *casos de uso.*

Primero es necesario determinar cuáles son los actores que interactúan con el mismo y las acciones que pueden realizar. En este caso nos vamos a encontrar con dos tipos de usuarios: el usuario *administrador* (es el usuario gestor del prototipo) y el usuario *comerciante* (es el usuario que desea realizar acciones de comercio a través del uso de la plataforma)

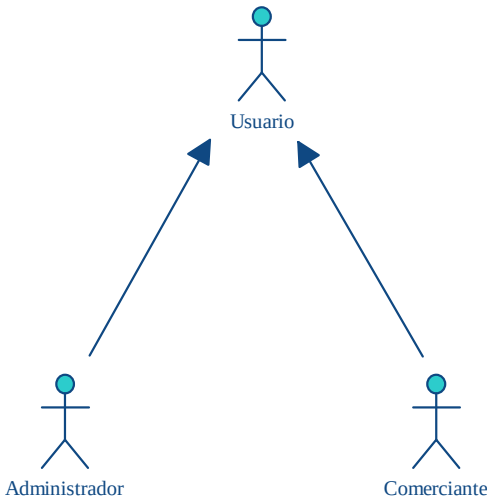


Figura 4.1. Perfiles de Usuario del Sistema.

Como usuario comerciante puede entenderse todo aquel usuario que realice una acción de comercio, ya sea comprar o vender por lo que el usuario comerciante puede dividirse a su vez en *usuario comprador* y *usuario vendedor*.

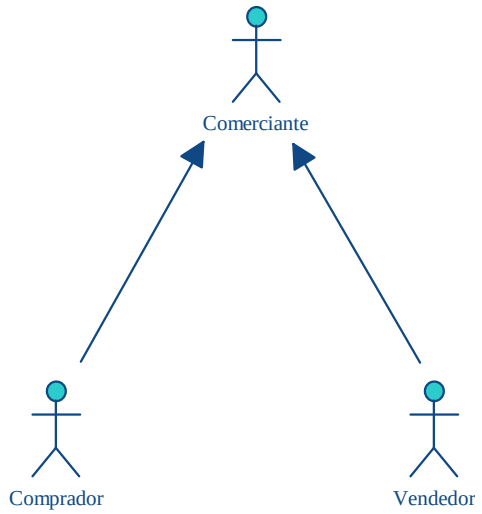


Figura 4.2. Tipos de usuario comerciante.

A continuación se muestra el *diagrama frontera* que muestra las relaciones entre los actores y el sistema. Las funcionalidades principales que pueden realizar los actores son "*Crear Mercado*", "*CrearAgenteComprador*" y "*CrearAgenteVendedor*", detallados como casos de uso y acompañados de su diagrama de secuencia.

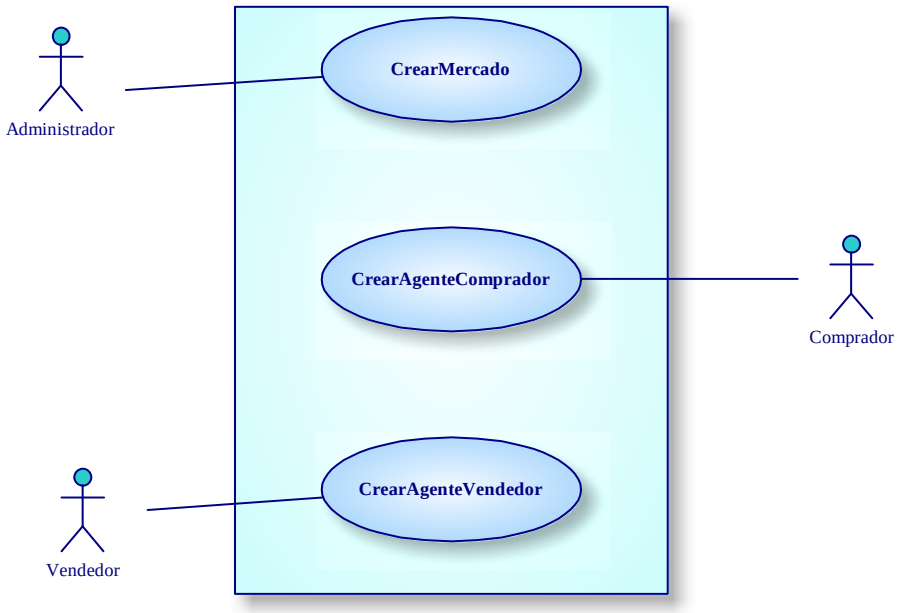


Figura 4.3. Diagrama frontera del Sistema.

Es necesario expresar mediante casos de uso una de las excepciones principales del sistema: *ErrorParámetroAgenteInválido* que es el que se produce cuando el usuario al cumplimentar los datos necesarios para la creación del agente introduce un valor incorrecto en alguno de los campos. Este error es controlado tanto por la plataforma JADE, si no se introduce el número de parámetros correcto, como por el prototipo que controla el tipo y valor de los parámetros introducidos.

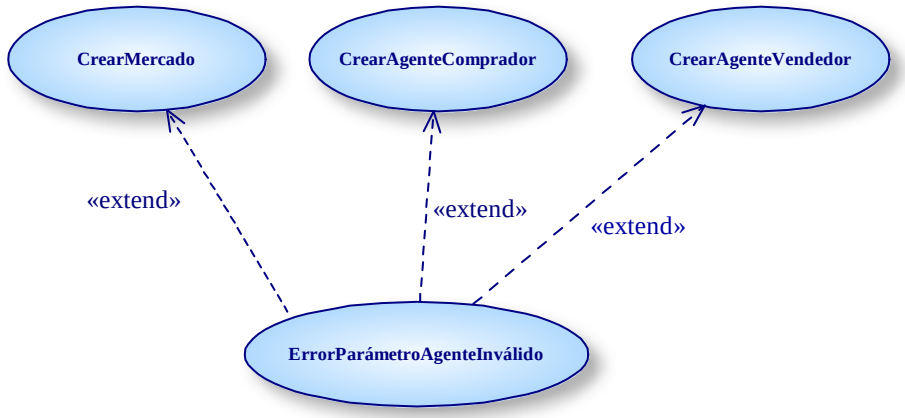


Figura 4.4. Excepción ErrorParámetroAgenteInválido.

Caso de uso: CrearMercado

Actores: Usuario Administrador	
Descripción: El usuario introduce los datos necesarios para crear un Mercado encargado de ser el nexo de unión entre los agentes compradores y vendedores que se creen a posteriori.	
Precondiciones: La plataforma se encuentra en funcionamiento.	
Postcondiciones: Se crea el agente Mercado.	
Flujo de eventos:	
Acciones del usuario:	**Acciones del sistema:**
1. Elige la opción de crear el mercado.	
	2. Muestra el formulario para la creación del agente mercado.
3. Cumplimenta el formulario con los datos necesarios para su creación.	
	4. Crea el agente mercado.
	5. El mercado solicita el registro de su servicio en las páginas amarillas con nombre del servicio "*JADE- mercado-libros*" y con tipo "*mercado-libros*". A través de este servicio puede ser identificado y localizado por el resto de agentes.
	6. Se crea la interfaz del agente Mercado.
7. Visualiza la interfaz del mercado a través del cual podrá consultar los procesos de negociación.	
Excepciones: *ErrorParámetroAgenteInválido.* Algún parámetro necesario para la creación del Agente Mercado posee un valor incorrecto. Se muestra un mensaje propio de la plataforma JADE.	

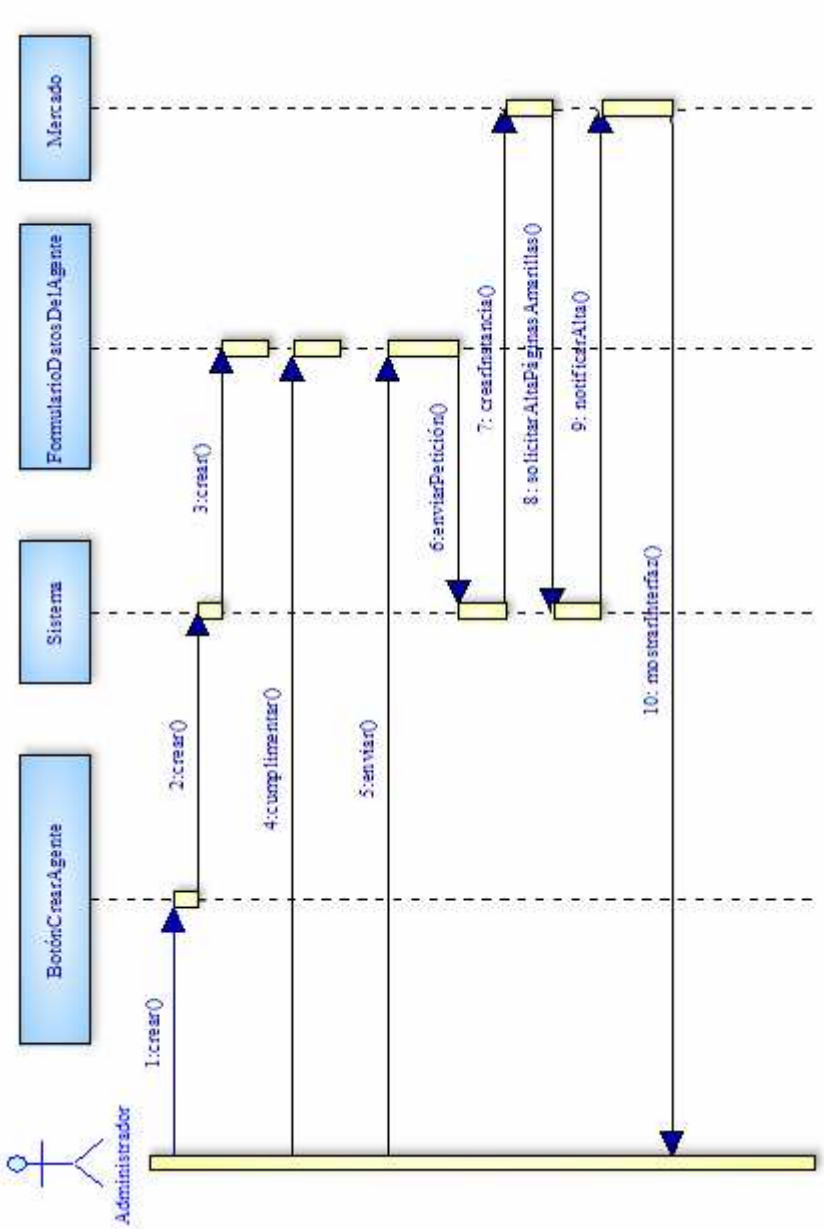


Figura 4.5. Diagrama de secuencia del caso de uso "CrearMercado".

Caso de uso: CrearAgenteComprador

Actores: Usuario Comprador	
Descripción: El usuario introduce los datos necesarios para crear un agente comprador encargado de realizar la compra de un producto en el Mercado.	
Precondiciones: La plataforma en funcionamiento y el agente Mercado está creado.	
Postcondiciones: Se crea un nuevo agente comprador en el Mercado.	
Flujo de eventos:	
Acciones del usuario:	**Acciones del sistema:**
1. Elige la opción de crear un agente comprador.	
	2. Muestra el formulario para la creación del agente.
3. Cumplimenta el formulario con los datos necesarios para su creación.	
	4. Crea el agente comprador.
	5. El agente comprador solicita a las páginas amarillas el identificador de agente que proporciona el servicio "*mercado-libros*" y si existe, el DF se lo proporciona.
	6. El agente comprador solicita darse de alta en el Mercado.
	7. El Mercado recibe la petición del agente comprador.
	8.El Mercado notifica a todos los agentes vendedores dados de alta la existencia de un nuevo comprador y el producto que demanda. Crea una lista de los agentes vendedores que poseen el producto solicitado por el comprador.
	9. El Mercado notifica el alta al agente comprador y le envía esa lista de vendedores.
	10. El comprador recibe el alta y la lista de vendedores.
	11. Se crea la interfaz del agente comprador.
12. Visualiza la interfaz del agente comprador con lo que se verifica que se ha creado correctamente.	
Excepciones: *ErrorParámetroAgenteInválido.* Algún parámetro necesario para la creación del Agente Comprador posee un valor incorrecto.	

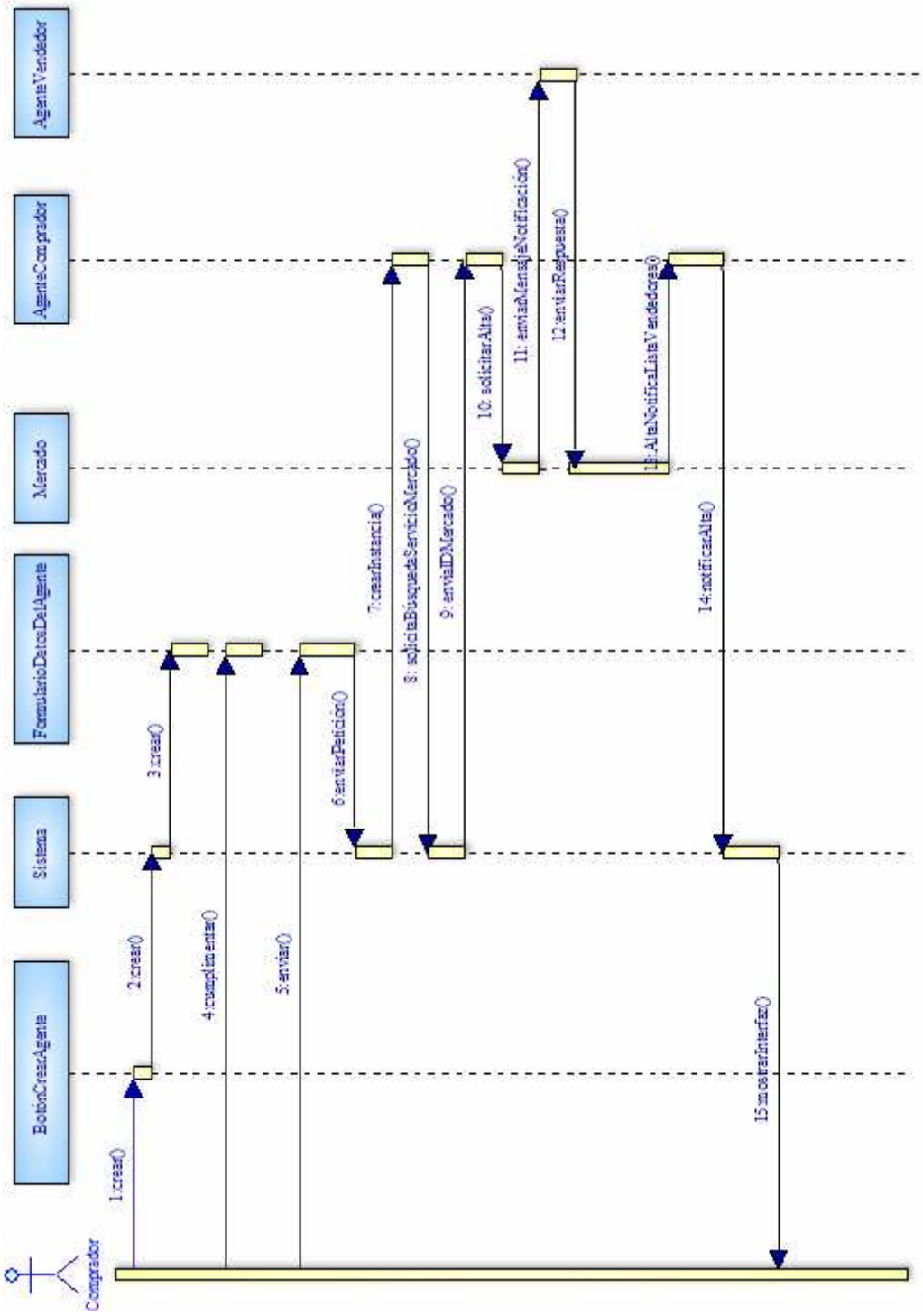


Figura 4.6. Diagrama de secuencia del caso de uso "CrearComprador".

Caso de uso: CrearAgenteVendedor

Actores: Usuario Vendedor
Descripción: El usuario introduce los datos necesarios para crear un agente vendedor encargado de realizar la venta de un catálogo de productos en el Mercado.
Precondiciones: La plataforma en funcionamiento y el agente Mercado está creado.
Postcondiciones: Se crea un nuevo agente vendedor en el Mercado.
Flujo de eventos: **Acciones del usuario:** **Acciones del sistema:** 1. Elige la opción de crear un agente vendedor. 2. Muestra el formulario para la creación del agente. 3. Cumplimenta el formulario con los datos necesarios para su creación. 4. Crea el agente vendedor. 5. El agente comprador solicita a las páginas amarillas el identificador del agente que proporciona el servicio "*mercado-libros*" y si existe, el DF se lo proporciona. 5. El agente vendedor solicita darse de alta en el Mercado. 6. El Mercado recibe la petición del comprador. 7. El Mercado envía al vendedor un mensaje para confirmar su alta con la lista de todos los agentes compradores y el libro que desean comprar. 8.Se envía un mensaje a la interfaz informando del resultado del proceso de creación del agente vendedor. 8. Visualiza la interfaz del agente vendedor comprobando su correcta creación.
Excepciones: *ErrorParámetroAgenteInválido.* Algún parámetro necesario para la creación del Agente Vendedor posee un valor incorrecto.

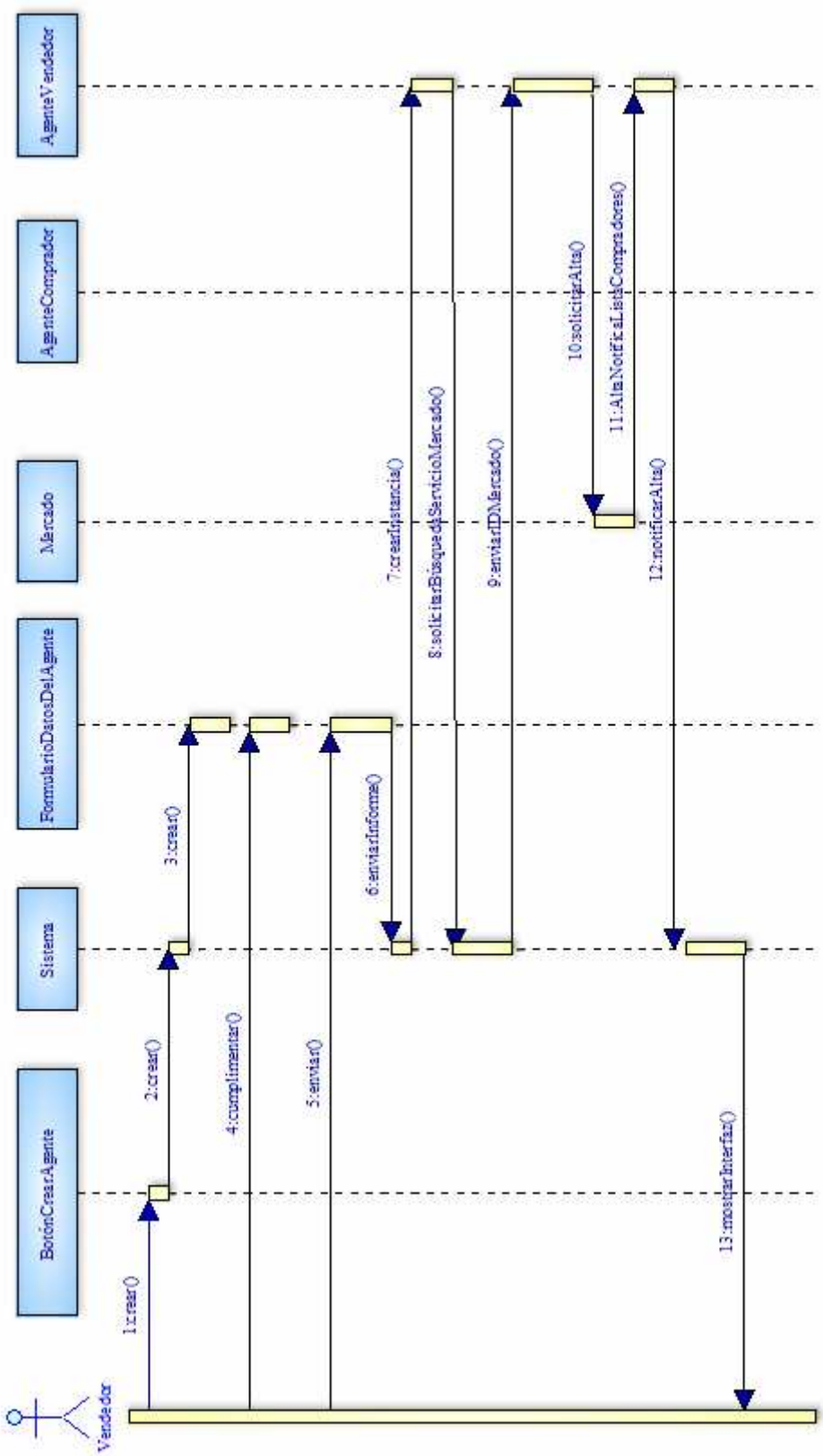


Figura 4.7. Diagrama de secuencia del caso de uso "CrearVendedor".

4.2. Diseño del sistema

En este apartado se especifica el diseño del sistema. Éste comprende el diseño de la estructura de los agentes, las estrategias de negociación, los tipos de mensajes que intercambian en cada caso, las conversaciones y los comportamientos que realizan.

4.2.1. Diseño de la estructura de los agentes

En el funcionamiento normal de la plataforma JADE los agentes que se van creando se dan de alta en el servicio DF de páginas amarillas indicando el servicio que proporcionan. Los agentes que se están ejecutando disponen de un servicio de *páginas blancas* para buscar a otros agentes pudiendo buscar también en las *páginas amarillas* por el servicio que proporcionan.

Sin embargo, en este proyecto el planteamiento es algo más complejo y se plantea siguiendo la estructura de la figura 4.8.

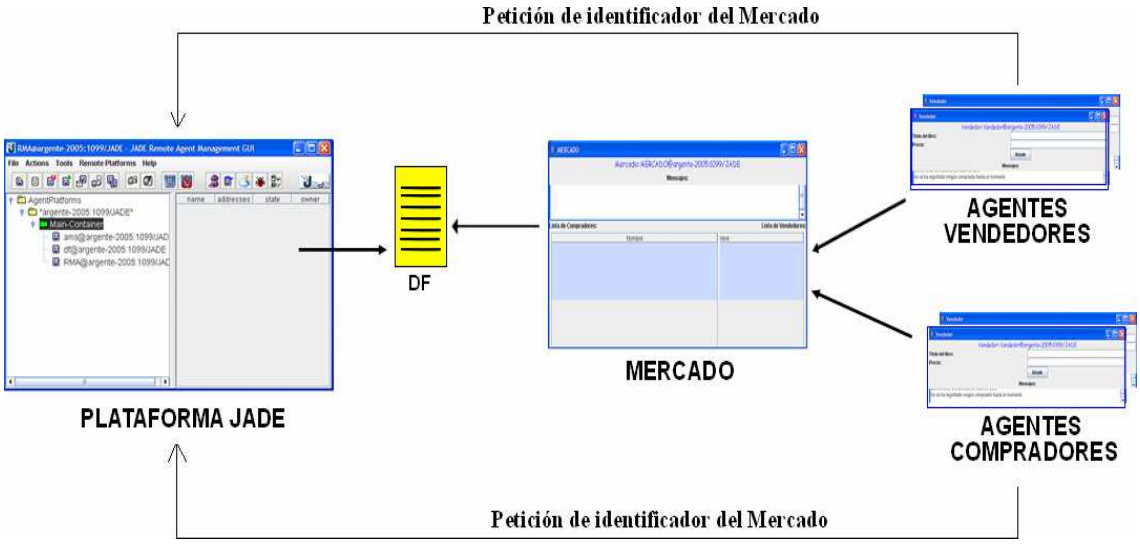


Figura 4.8. Estructura de los Agentes del prototipo.

En este caso, se va a contar con un agente intermediario entre los agentes compradores y vendedores que será el Mercado. Este agente al crearse registra en las páginas amarillas el servicio que proporciona "*mercado-libros*" para que pueda ser localizado por otros agentes.

El resto de agentes al crearse en lugar de solicitar al DF su alta en las páginas amarillas solicitan el identificador de aquel agente (el Mercado) que ofrezca el servicio "*mercado-libros*". Las páginas amarillas proporcionan el identificador del Mercado pudiendo ya los agentes entrar en contacto directamente con él. De esta manera, los agentes compradores y vendedores solicitan su alta en el Mercado manteniéndose en el mismo un listado para los compradores y otro para los vendedores.

Esta estructura provoca el efecto deseado, es decir, que exista un Mercado controlador de las altas, las bajas y los procesos de negociación entre los agentes. Sin la existencia del Mercado los agentes compradores y vendedores no podrían crearse ni comunicarse para realizar procesos de negociación.

Además, el Mercado permite controlar las acciones realizadas por todos los agentes y muestra información acerca de los procesos de negociación.

4.2.2. Diseño de la estrategia

En este prototipo la estrategia se establecerá en función del precio y del tiempo transcurrido, variando el primero respecto al segundo:

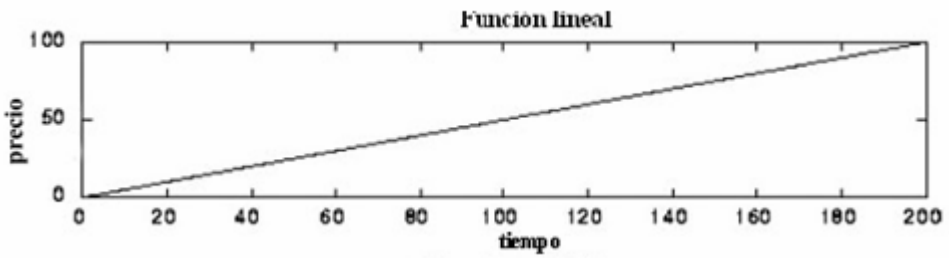


Figura 4.9. Estrategia con función lineal.

Un ejemplo de estrategia lineal que podría seguir el *agente comprador* sería "*Mejorar mi última oferta/contraoferta en un 1%"* siempre teniendo la restricción de un tope máximo que estaría dispuesto a pagar por el producto y cuyo valor sólo conoce él. En el caso del *agente vendedor* por ejemplo, *"Disminuir el precio de venta en un 2% en cada oferta/contraoferta"* teniendo a su vez un precio mínimo no conocido por el resto de agentes. De esta manera, cada uno de los agentes enfrentados en la negociación van cediendo hasta que lleguen a un acuerdo o a un momento en el cual no les interese seguir negociando.

En este prototipo el modelo de negociación que se va a llevar a cabo, según la estructura de mercados expuesta en la figura 2.4 seguirá el Modelo C con unas variaciones. En este

modelo C se produce una negociación de “m” a “n”; es decir, que un comprador negocia con varios vendedores al mismo tiempo, y que a su vez estos vendedores están negociando con varios compradores.

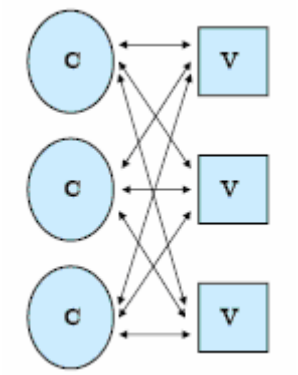


Figura 4.10. Modelo C de negociación en un mercado electrónico.

Según la estructura de los mercados controlados que aparece en la *figura 2.5* puede realizarse el siguiente esquema resumen del Mercado Electrónico a diseñar:

Tipo de Mercado	*C2C*
Tipo de Modelo	*Modelo C (m:n)*
Atributos de negociación	*Un atributo*
Tipos de restricciones del comprador	*Difusas*
Tipos de restricciones del vendedor	*Difusas*

El atributo elegido para la negociación va a ser “*el precio*” estableciendo, tanto los agentes comprados como los agentes vendedores, unos parámetros sobre los que variar el mismo y realizar la negociación.

Cuando el usuario crea un agente deberá especificar una serie de parámetros, para que en función de ellos puedan llevar a cabo una correcta negociación en función de los intereses del propio individuo. Alguno de los parámetros serán conocidos por todo aquel agente que intervenga en el proceso; pero otros estarán ocultos y formarán parte de la estrategia de los agentes.

En este caso, los parámetros serán los siguientes:

Agente Comprador:

- *Precio deseado.* Es el precio que desea pagar por el producto y con el cual iniciará la negociación.

- *Incremento.* Será el tanto por ciento en que irá aumentando su oferta en caso de que no obtenga de primeras el precio deseado para la compra. Este valor a priori no lo conocerán los otros agentes, aunque como se mantendrá fijo a lo largo del proceso cuando realice por primera vez un incremento de su oferta este valor ya será conocido.

- *Precio máximo aceptable que está dispuesto a pagar.* Este valor no será conocido por el resto de los agentes ya que es el elemento principal de la estrategia del comprador.

Agente Vendedor:

- *Precio deseado.* Es el precio por el cual desea vender el producto y será el precio de salida y el que figure en el catálogo del vendedor.

- *Decremento.* De manera similar al agente comprador, es el tanto por ciento en que irá disminuyendo el precio deseado por el producto.

- *Precio bajo más aceptable.* Es el precio más bajo al cual podría vender el producto y vendrá expresado por un tanto por ciento sobre el valor del producto. Es decir, cuando se crea un agente vendedor se establece que tanto por ciento de descuento realizará como máximo a los libros de su catálogo. Calculando el valor correspondiente sobre cada libro se obtiene el precio más bajo aceptable de cada uno. Este precio no es conocido por el resto de los agentes puesto que es uno de los principales elementos de la negociación del agente vendedor.

En un primer momento el agente comprador iniciará conversaciones con todos los agentes vendedores que se encuentren en la lista que el mercado le ha proporcionado. De todas las ofertas iniciales que recibe elegirá, en función del precio, aquella que, a pesar de no ser la idónea al comienzo para formalizar el trato, es la más ventajosa para el comprador y a partir de la cual inicia una negociación.

Puede darse el caso de que un comprador al darse de alta reciba una lista vacía de agentes vendedores. Esto es debido a que todavía no se ha registrado ningún vendedor que posea en su catálogo el artículo deseado por el comprador. El comprador al cabo de un minuto solicitará al Mercado una actualización de la lista de vendedores y así sucesivamente hasta que encuentre alguno/s con los que realizar la negociación.

Según los valores establecidos, los agentes entrarán en contacto y realizarán el proceso de negociación para llegar a un acuerdo. En el caso de que el precio deseado de compra sea el mismo que el precio deseado de venta o que incluso el precio deseado de venta sea inferior al precio deseado de compra, el acuerdo es inmediato.

En caso contrario, los agentes irán realizando sucesivas ofertas (los compradores incrementando el valor del precio de compra deseado y los vendedores decrementando el valor del precio de venta deseado) hasta llegar a un acuerdo satisfactorio para ambos, o bien no llegando a un acuerdo, finalizando en ambos casos el proceso de negociación.

Los incrementos y decrementos de los precios vendrán determinados por unos topes establecidos como "*precio máximo aceptable*" y "*precio bajo más aceptable*".

En la estrategia elegida el proceso finalizará en algunos de estos casos:

❖ Casos en los que se llega a un **ACUERDO**:

- El agente comprador solicita el precio del producto al vendedor; si este precio del catálogo es mejor que el precio deseado por el comprador, éste acepta directamente el trato ya que estaba dispuesto a pagar más.

- Tras un proceso de negociación el agente comprador está de acuerdo con el precio ofrecido por el vendedor y acepta el trato.

❖ Casos en los que **NO** se llega a un **ACUERDO**:

Tras un proceso de negociación en el cual se producen ofertas y contraofertas:

- El agente comprador no puede aumentar más el precio de su oferta ya que sobrepasa el precio máximo que está dispuesto a pagar.

- De forma análoga, otra posibilidad es que el agente vendedor no pueda disminuir más el precio de su oferta, ya que si lo hace estaría por debajo del precio más bajo por el está dispuesto a vender el producto.

- Es posible que cuando el agente comprador esté de acuerdo en el trato, al enviarle el mensaje de notificación al vendedor, éste al estar negociando con otros compradores a la vez, haya realizado la venta del producto a otro agente.

- Los agentes se dan de baja inesperadamente del Mercado con lo cual se rompen los contactos que habían realizado entre ellos.

- A pesar de que el Mercado haya incluido al agente vendedor en la lista de posibles vendedores para el agente, es posible que al solicitar el precio inicial del producto ya no posea existencias en el catálogo con lo cual no iniciarían ningún proceso de negociación.

4.2.3. Diseño de las conversaciones entre los agentes

En este apartado se van a describir las conversaciones que mantienen los agentes del sistema en los proceso de alta, baja y negociación. Van a estar representadas mediante diagramas de secuencia creados por la herramienta *Sniffer (*herramienta que permite visualizar los mensajes intercambiados entre los agentes*)* de la Plataforma JADE los cuales muestran los mensajes que se envían los agentes, su orden y el tipo de mensaje.

4.2.3.1. Tipos de mensajes

Es necesario determinar qué tipo de mensajes se intercambian en cada momento y qué tipo de información se transmite con cada uno. En la siguiente figura puede observarse además el sentido de los mensajes que pueden ser en una sola dirección o bidireccionales.

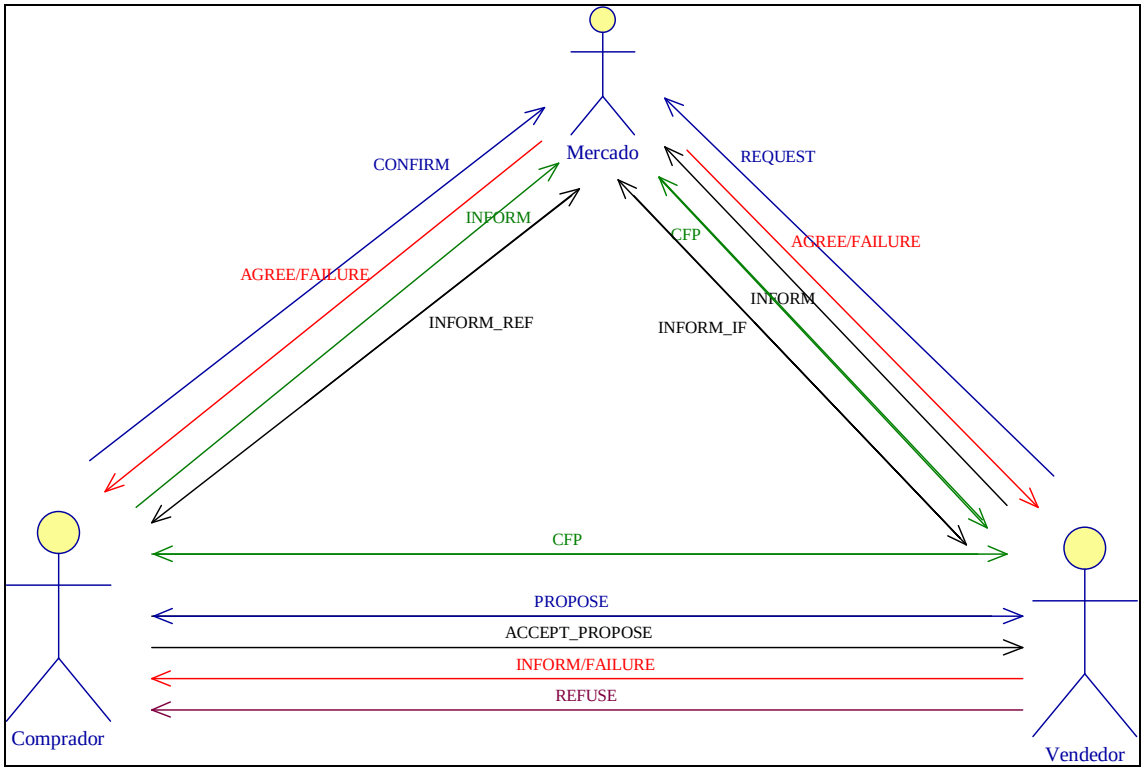


Figura 4.11. Tipos de mensajes que intercambian los agentes.

A continuación se describe el uso que se ha otorgado a cada uno de los tipos de mensajes empleados:

- **Confirm.** El comprador solicita su alta en el Mercado.
- **Agree/Failure.** El Mercado notifica a los agentes si ha realizado correctamente su inserción en el Mercado: aceptado (Agree) y rechazado (Failure).
- **Inform.** Los agentes notifican al Mercado su baja del mismo.
- **Request.** El agente vendedor solicita su alta en el Mercado.
- **CFP.** Este tipo de mensaje, como se comentó con anterioridad, pide candidaturas para realizar cierta acción, por lo que inicia procesos de negociación. Se ha utilizado en dos casos:
    - Entre el Mercado y los agentes vendedores. A través del mensaje CFP el Mercado busca entre los vendedores los candidatos que conformarán la lista de

vendedores posibles para un agente comprador. Esta acción se realiza al darse de alta un comprador y cuando el comprador solicita la actualización de su lista de vendedores.

- Entre comprador y vendedor. En este caso, es el agente comprador el que solicita el producto y su precio a los vendedores.

- **Propose.** Este tipo de mensajes se los intercambiarán en el proceso de negociación los compradores y los vendedores para realizar las ofertas y contraofertas.

- **Accept_propose.** El comprador acepta el acuerdo propuesto por el vendedor.

- **Refuse.** El vendedor notifica al comprador que no desea continuar con el proceso. Esto puede producirse porque llegue un momento en la negociación en que no le interese o que ya no posea el artículo.

- **Inform_if.** El vendedor solicita al Mercado la actualización de la lista de compradores. El mercado responde con un mensaje de este tipo a la petición.

- **Inform_ref.** El comprador solicita al Mercado la actualización de la lista de vendedores. El mercado responde con un mensaje de este tipo a la petición.

4.2.3.2. Comportamientos de los agentes

Mercado

Comportamiento	Tipo
AltaCompradores	CyclicBehaviour
AltaVendedores	CyclicBehaviour
BajaAgentes	CyclicBehaviour
NotificaCompradores	CyclicBehaviour
NotificaVendedores	CyclicBehaviour

AltaCompradores

Es un comportamiento cíclico que está constantemente ejecutándose para recibir las peticiones de los agentes compradores que desean darse de alta en el Mercado. El proceso se realiza en una serie de pasos:

Paso 0. Se encuentra bloqueado mientras no recibe ninguna petición de alta de los compradores. Cuando la recibe la procesa y se va al paso 1 para seguir con el proceso de alta.

Paso 1. Envía un mensaje CFP (cuyo contenido es el libro que el comprador desea) a todos los vendedores para formar la lista de vendedores potenciales que se enviará al agente comprador. Se prepara para recibir las respuestas de los vendedores.

Paso 2. Se encuentra en estado de bloqueo mientras no recibe ningún mensaje. Recibe las respuestas de los vendedores en la cual indican con un valor lógico si disponen del libro en su catálogo o no. Añade a la lista a los vendedores cuya respuesta es afirmativa. Este proceso en el cual pasa de bloqueado (mientras espera mensaje) a activo (cuando recibe un mensaje y lo procesa) se repite hasta que ha recibido la respuesta por parte de todos los vendedores y pasa al siguiente paso.

Paso 3. Añade al Mercado el nuevo agente y si esta acción se realiza correctamente le responde con un mensaje AGREE cuyo contenido es la lista de potenciales vendedores que acaba de formar; en caso contrario, se le envía un mensaje FAILURE para indicarle que su alta en el Mercado no ha sido posible. De este paso se vuelve al primero para seguir repitiéndose todo el proceso en el alta del resto de los compradores.

AltaVendedores

De manera similar al comportamiento anterior puede decirse que es un comportamiento cíclico que está constantemente ejecutándose para recibir las peticiones de los agentes vendedores que desean darse de alta en el Mercado. En este caso el proceso no requiere seguir una serie de pasos establecidos, aunque sí un pequeño proceso:

El Mercado se prepara para recibir las peticiones de los agentes vendedores que llegarán en forma de mensaje tipo REQUEST (petición). Mientras no se reciban mensajes este comportamiento se encontrará bloqueado activándose cada vez que se recibe un REQUEST. Es entonces cuando procesa el contenido del mensaje y envía al emisor un mensaje AGREE con la lista de los agentes compradores dados de alta en el Mercado, si el proceso de alta se ha realizado correctamente con la lista de todos los compradores dados de alta, o un mensaje FAILURE para indicarle que el proceso ha fallado.

BajaAgentes

Al dar de alta a los agentes los procesos eran distintos mientras que para procesar las bajas se realiza en el mismo comportamiento ya que el proceso es el mismo. Es un comportamiento cíclico que está continuamente ejecutándose.

El Mercado se prepara mediante un MessageTemplate para recibir los mensajes de tipo INFORM designados en esta estrategia para la baja de los agentes. Al igual que en comportamientos anteriores, mientras no recibe ningún mensaje el comportamiento permanece bloqueado. Al recibir el mensaje, examina su contenido para determinar si es un agente comprador (el contenido será "C") o vendedor (el contenido será "V"). En ambos casos, se localiza el agente con su AID en la lista correspondiente y se elimina.

NotificaCompradores

Es un comportamiento cíclico encargado de recibir y satisfacer las peticiones de los agentes vendedores que desean actualizar su lista de compradores añadiendo aquellos nuevos que realizaron el alta posteriormente al vendedor.

Este comportamiento está continuamente ejecutándose. Está preparado para recibir mensajes del tipo INFORM_IF cuya contenido sea "V+"; es decir, mensajes de petición de la nueva lista de agentes compradores por parte de agentes vendedores.

Mientras no recibe ningún mensaje el proceso permanece bloqueado. En caso contrario comprueba si el emisor es un agente vendedor le responde con un mensaje también de tipo INFORM_IF cuyo contenido es la lista actualizada de los agentes compradores dados de alta en el Mercado.

NotificaVendedores

Es similar al comportamiento de "NotificaCompradores". Este comportamiento recibe las peticiones de los agentes compradores que desean actualizar su lista de vendedores. El proceso es más complejo que para los vendedores ya que realiza acciones similares al alta de los agentes compradores cuando se procesa qué vendedores poseen en su catálogo el producto deseado por el comprador.

Se ejecuta en una serie de pasos:

Paso 0. Recibe los mensajes INFORM_REF de los compradores cuyo contenido posee la siguiente estructura: "C+título del libro". Mientras no recibe mensajes el

comportamiento se encuentra bloqueado. Una vez recibido y procesado el mensaje, si se ha determinado convenientemente que pertenece a un agente comprador se continúa con el paso 1.

Paso 1. Manda a todos los vendedores dados de alta en el Mercado un mensaje CFP solicitando que compruebe en su catálogo si dispone del artículo que desea el comprador.

Paso 2. Recibe las respuestas de los vendedores y con aquellos que respondan afirmativamente se crea una lista de vendedores potenciales para el comprador. Una vez recibidas todas las respuestas continúa con el paso 3.

Paso 3. Crea una respuesta de tipo INFORM_REF cuyo contenido es la lista actualizada de vendedores potenciales y el destinatario es el agente comprador que inició la conversación. Continúa en el paso 0 para seguir recibiendo más peticiones.

AgenteComprador

Comportamiento	Tipo
RecibeAltaComprador	CyclicBehaviour
RealizaPetición	Behaviour.Genérico
TickerBehaviour	Con time-out

RecibeAltaComprador

Es un comportamiento cíclico mediante el cual el agente comprador recibe el alta del Mercado.

Se prepara para recibir un AGREE o un FAILURE. Este comportamiento permanece bloqueado mientras no recibe mensajes de este tipo. Cuando recibe el mensaje lo procesa: en caso de ser del tipo AGREE crea su lista interna de vendedores con el valor que le ha pasado el Mercado; si es del tipo FAILURE el mercado no recibe ningún vendedor, no se ha dado de alta en el Mercado y por lo tanto no puede realizar negociaciones dándose de baja de la plataforma.

RealizaPeticion

Es un comportamiento de tipo genérico que hereda directamente de la clase Behaviour y que realiza su cometido siguiendo unos pasos:

Paso 0. Crea un mensaje CFP cuyo contenido está formado por el título de libro que desea, el precio de salida que está dispuesto a pagar, el precio que el vendedor ofrece (en este caso está inicializado a cero ya que el comprador no tiene constancia de este valor) y un contador de peticiones (inicializado a cero) para que el vendedor pueda controlar en qué momento se está realizando la petición. Una vez establecido el contenido lo envía a los vendedores que se encuentran en la lista que el Mercado le proporcionó. Se prepara para recibir la respuesta de los vendedores y continúa con el paso siguiente.

Paso 1. Recibe las respuestas de los vendedores y las procesa. Mientras no haya recibido la totalidad de las respuestas esperadas el comportamiento se bloqueará. Si el tipo de respuesta enviado es PROPOSE indica que el vendedor posee en su catálogo el libro solicitado por el comprador. El formato del mensaje será el mismo que el que envió el comprador estando en este caso el valor del precio del vendedor relleno con el valor correcto del precio del producto que figura en su catálogo. En esta estrategia se ha establecido que el comprador al ir procesando las respuestas elija al mejor vendedor en función del atributo "*precio*" eligiendo de esta manera a aquel cuyo precio de salida sea el menor.

Una vez ha recibido todas las respuestas esperadas comprueba si el precio deseado por el comprador es mayor o igual que el mejor precio del mejor vendedor. En tal caso es la mejor oferta que el comprador podía encontrar y decide realizar la compra directamente pasándose al paso 4 directamente. Si no es la oferta ideal necesitará iniciar un proceso de negociación en el cual obtener un precio más ventajoso; va al paso 2.

Paso 2. El vendedor ya le ha notificado al comprador cuál es su precio de salida y es hora de que el comprador realice una oferta mejor. Para ello modifica su precio aumentándolo en el tanto por ciento que indicó al darse de alta. Este tanto por ciento se calcula sobre el precio actual del comprador y se suma al mismo. El comprador obtiene tras los cálculos un nuevo precio que antes de ser enviado al vendedor ha de ser revisado.

Si el precio nuevo del comprador sea mayor o igual que el precio máximo que está dispuesto a pagar por el producto, el propio comprador finaliza el proceso de negociación ya que no le interesa el trato que pueda obtenerse.

En caso contrario, que la subida que ha realizado sobre su precio sea aceptable envía al vendedor elegido como mejor vendedor un mensaje PROPOSE donde ha actualizado el valor que ofrece por el producto.

Antes de continuar al paso 3 se prepara para recibir la respuesta del vendedor sobre la oferta que ha realizado.

Paso 3. Mientras no recibe la respuesta del vendedor se encuentra bloqueado. Al recibir un mensaje PROPOSE con la nueva oferta del vendedor, en la cual ha modificado su precio de venta, pasa a estar activo.

Si el nuevo precio del vendedor es menor o igual que el que en ese momento ofrece el agente comprador, éste decide aceptar el trato saltando al paso 4.

Si todavía el precio no satisface al comprador continúa con el proceso de negociación volviendo al paso 2. De este modo, se irán produciendo una sucesión de mensajes PROPOSE por ambas partes hasta que alguna de ellas decida acabar con la negociación o que el agente comprador esté dispuesto a aceptar las condiciones de venta.

Paso 4. Si se ha llegado a este paso quiere decir que el agente ha decidido aceptar las condiciones del vendedor y le manda un mensaje ACCEPT_PROPOSAL para confirmárselo. En el contenido del mensaje posee la misma estructura que durante el proceso de negociación.

A pesar de que el comprador haya decidido aceptar, aún debe esperar una respuesta por parte del vendedor para formalizar la venta y dar por finalizada la negociación. Sigue por el paso 5.

Paso 5. Se bloquea en este paso mientras no recibe una respuesta del vendedor. Si la respuesta es un mensaje de tipo INFORM el vendedor está de acuerdo y se realiza la venta. Si el tipo de mensaje recibido es REFUSE el trato no puede realizarse finalizando sin un acuerdo la negociación.

Este tipo de comportamiento genérico cuenta con una función *done()* que controla si la ejecución del comportamiento ha finalizado o no.

TickerBehaviour

Es un tipo especial de comportamiento cuya ejecución es diferida. Este comportamiento cíclico se invoca y guarda hasta que se ha cumplido un determinado time-out (un minuto en este caso).

Cada vez que se consuma el tiempo establecido este comportamiento crea un mensaje de tipo INFORM_REF con contenido "C+título del libro a comprar" y lo envía al Mercado para solicitarle que desea una actualización de la lista de vendedores.

Espera la respuesta del Mercado y permanece bloqueado mientras. Recibe la respuesta del Mercado en forma de mensaje de tipo INFORM_REF. Obtiene el contenido del mensaje que es una lista de vendedores y actualiza su lista interna con los nuevos vendedores.

AgenteVendedor

Comportamiento	Tipo
RecibeAlta	CyclicBehaviour
PeticionesComprador	CyclicBehaviour
PeticionesOfertas	CyclicBehaviour
Negocia	CyclicBehaviour
RealizaVenta	CyclicBehaviour
ModificarCatalogo	OneShotBehaviour
TickerBehaviour	Con time-out

RecibeAlta

Previamente, en el método setup() el agente envió un mensaje al Mercado para solicitar su alta. En este comportamiento espera la respuesta por parte del Mercado. Mientras no la recibe está bloqueado. Si recibe un mensaje AGREE sabrá que está dado de alta correctamente y además recibe la lista de los compradores proporcionada por el Mercado. Si no se ha realizado el alta de forma satisfactoria el mensaje recibido será FAILURE.

PeticionesComprador

Es un comportamiento cíclico que está continuamente ejecutándose para responder a las peticiones de los compradores acerca de si el vendedor posee un determinado artículo en su catálogo.

Se prepara para recibir los CFP de los compradores. Mientras no recibe mensajes el comportamiento estará ejecutando el método "*block()*". Al recibir el mensaje, obtiene del contenido del mismo el título de libro que desea el vendedor y realiza una búsqueda en su catálogo. Tanto si la búsqueda ha tenido éxito como si no, responderá al comprador indicándoselo con un valor lógico (1= disponible, 0= no disponible).

PeticionesOfertas

En este caso recibe las solicitudes de los compradores que le están preguntando por el valor de salida del producto que desean. Es un comportamiento cíclico que está bloqueado mientras no reciba mensajes del tipo que espera.

Se prepara para recibir los mensajes CFP de los compradores, obtiene el título de libro del contenido del mensaje e internamente busca el precio de salida del mismo. En caso de que ya no haya existencias responderá con un mensaje REFUSE indicando que el libro no está disponible y finalizando así la negociación.

Si ha obtenido correctamente el precio del artículo construye y envía la respuesta al comprador insertando en el contenido este valor. Así, le indica que el libro está disponible.

Negocia

En comportamientos anteriores el vendedor ha entrado en contacto con compradores que han iniciado la negociación solicitándole el precio de los productos de su catálogo.

En este comportamiento cíclico se va a suceder el envío y recepción de ofertas y contraofertas mediante mensajes PROPOSE. Como en tantos otros comportamientos permanece bloqueado mientras no recibe mensajes.

Cuando recibe el PROPOSE analiza su contenido y se centra en el precio de su última oferta. Como el comprador no ha aceptado todavía es el momento en el que el vendedor debe mejorar su oferta. Para ello realiza un descuento de un determinado tanto por ciento que se estipuló al crear el agente vendedor. El importe a descontar se irá calculando y descontando

sobre el precio por el que actualmente se está negociando. El vendedor manda un mensaje PROPOSE al comprador con el nuevo precio.

Si el nuevo precio calculado se encuentra por debajo del precio mínimo por el que está dispuesto a venderlo, la negociación finaliza enviando el vendedor un mensaje REFUSE. También la negociación no seguirá su curso si durante el proceso el vendedor detecta que el artículo lo ha vendido a otro comprador. En este último caso enviará también un mensaje de tipo REFUSE.

RealizaVenta

Tras haber realizado las negociaciones necesarias con el comprador, el vendedor en este comportamiento está esperando el mensaje de aceptación del trato por parte del comprador.

Tras recibirlo deja de estar bloqueado y analiza los precios con los cuales se llegó al acuerdo. En caso de que el precio del comprador sea mayor o igual que el precio del vendedor, el vendedor confirma la venta mediante un INFORM si el libro está disponible; si el libro no

está disponible o no está de acuerdo con los precios acordados el tipo de mensaje que enviará será de tipo FAILURE.

ModificarCatálogo

Una de las acciones que puede realizar el vendedor es la de dar de alta nuevos productos en catálogo. Esta acción se realiza a través de un comportamiento OneShot de manera que se ejecute una sola vez y de manera casi instantánea.

TickerBehaviour

Funciona de manera similar al comportamiento del Agente Comprador y sirve para que cada cierto tiempo el agente vendedor solicite al Mercado una actualización de la lista de agentes compradores. En este caso el mensaje que envía el vendedor es de tipo INFORM_IF y su contenido es “V+” (para indicar que es un vendedor).

Espera la respuesta del Mercado y permanece bloqueado mientras. Recibe la respuesta del Mercado en forma de mensaje de tipo INFORM_IF y actualiza su lista con la información proporcionada en el mensaje.

4.2.3.3 Descripción de las conversaciones

altaComprador

Cuando el usuario introduce los datos para la creación de un agente comprador se crea una instancia del agente comprador en el sistema siendo el siguiente paso su inserción en el Mercado. Es el propio agente comprador el que inicia esta conversación siendo el receptor el Mercado.

Para iniciar la conversación el agente comprador, en su función "*setup()*" , envía un mensaje al mercado de tipo CONFIRM indicándole qué tipo de agente es y el producto que desea comprar.

Cuando el Mercado recibe el mensaje prepara la respuesta que contendrá además una lista de los vendedores dados de alta y que posean el artículo solicitado por el comprador en su catálogo. Para confeccionar esta lista el Mercado establece contacto con los vendedores iniciando otra conversación mandando un mensaje de tipo CFP y espera que todos los vendedores le respondan indicando si contienen ese producto en su catálogo o no. Una vez que está formada la lista el Mercado envía un mensaje AGREE al agente Comprador con la lista de vendedores.

En caso de que no se realice correctamente la acción se enviará al Comprador un mensaje de tipo FAILURE.

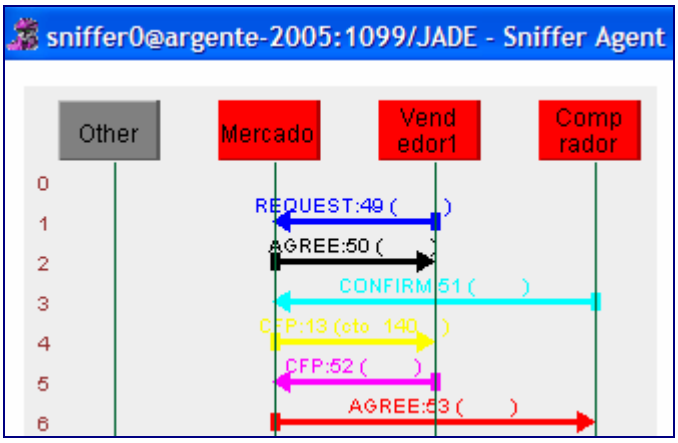


Figura 4.12. Conversación donde el comprador solicita su alta en el Mercado.

altaVendedor

Cuando el usuario introduce los datos para la creación de un agente vendedor se crea una instancia del agente vendedor en el sistema siendo el siguiente paso su inserción en el Mercado. Es el propio agente vendedor el que inicia esta conversación siendo el receptor el Mercado.

De forma similar al alta del comprador, inicia la conversación el agente vendedor, en su función "*setup()*" , enviando un mensaje al mercado REQUEST indicándole qué tipo de agente es. El Mercado responde con un mensaje AGREE confirmándole el alta o con un mensaje FAILURE en caso de que no haya podido realizarse la acción.

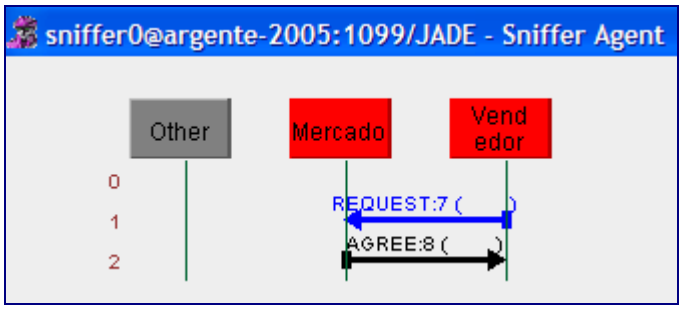


Figura 4.13. Conversación donde el Vendedor solicita su alta en el Mercado.

bajaAgentes

Esta acción es iniciada por los agentes compradores o vendedores cuando desean darse de alta en el Mercado porque hayan completado su trabajo. Se envía un mensaje de tipo INFORM al Mercado el cual por el contenido del mensaje determina qué tipo de agente es y le da de baja.

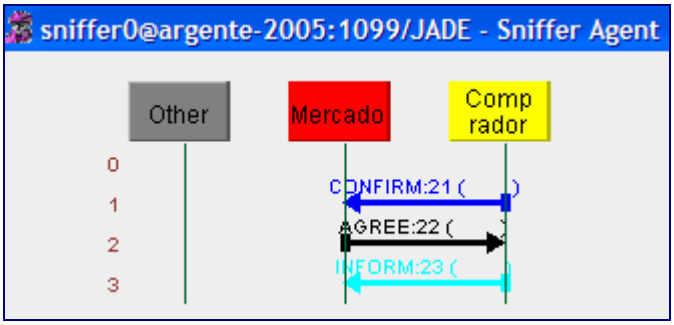


Figura 4.14. Conversación en la que el Comprador solicita su baja del Mercado.

Proceso de negociación.

Básicamente los pasos que se producen en el proceso de negociación son los siguientes:

1. El comprador indica qué producto desea comprar y a qué precio de salida; además también especifica en qué tanto por ciento irá aumentando su oferta y el precio máximo que está dispuesto a pagar por el artículo. Este paso se realiza cuando el agente se da de alta en el Mercado y es el propio Mercado el que establece contacto con los agentes vendedores para que realice el proceso de negociación. (Consultar la conversación de "altaComprador").

2. El vendedor responde si dispone del producto. Este paso se realiza en el mismo proceso de "altaComprador()" cuando el Mercado envía un mensaje CFP a todos los vendedores los cuales responden informando si disponen o no del producto. El Mercado crea así una lista de los potenciales agentes vendedores y se lo hace saber al agente comprador. De esta manera, el comprador ya dispone de los agentes con los que puede entrar en contacto para iniciar el proceso de negociación.

3. El agente comprador solicita el precio de salida a la venta que poseen los vendedores. Una vez que el comprador posee la lista de vendedores potenciales establece contacto con los mismos preguntando el precio de partida del producto. El agente vendedor responderá con un mensaje informativo acerca de dicho precio y a partir de este momento comenzará el proceso de negociación a través del cual y durante un tiempo determinado negociarán intentando llegar al mejor acuerdo.

4. El agente comprador elige al mejor vendedor. En este momento comienzan a sucederse las ofertas mediante mensajes PROPOSE: el agente comprador va aumentando el precio que está dispuesto a pagar y el agente vendedor disminuyendo el precio de venta.

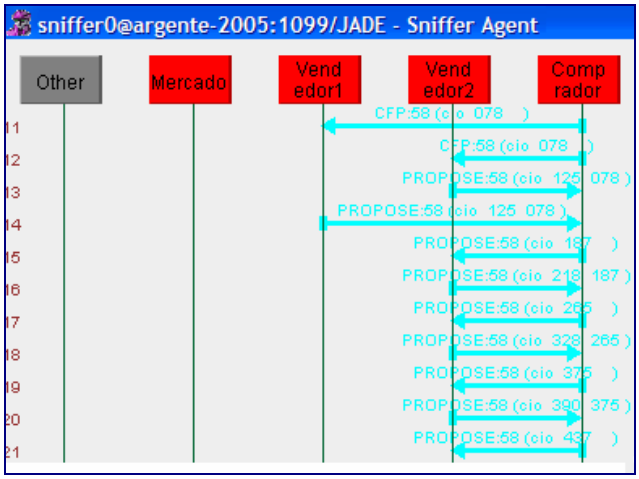


Figura 4.15. Conversación entre comprador y mejor vendedor.

5. Si tras el proceso de negociación el agente Comprador está dispuesto a comprar manda un mensaje ACCEPT_PROPOSAL al vendedor para notificarlo. A su vez el vendedor le envía un mensaje INFORM para cerrar el trato y en caso de no aceptarlo responderá con un mensaje FAILURE.

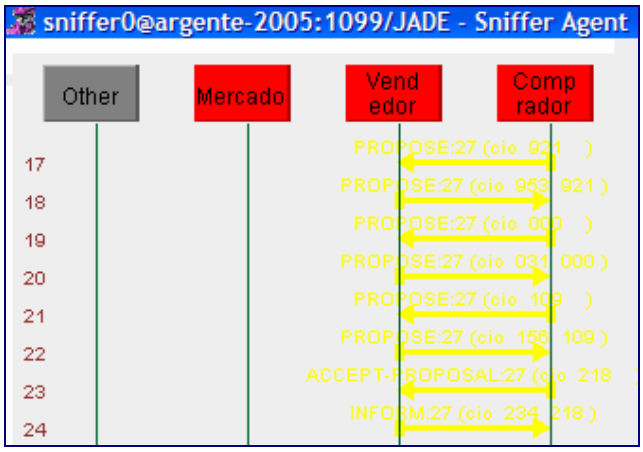


Figura 4.16. Acuerdo entre agentes.

El acuerdo puede que se produzca de inmediato sin necesidad de negociación si el precio del catálogo del artículo del mejor vendedor es menor que el precio de salida del comprador.

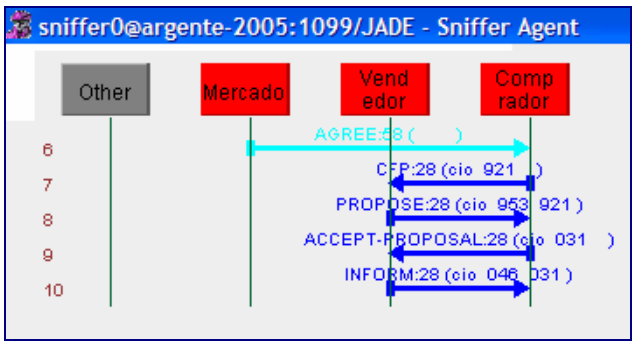


Figura 4.17. El comprador acepta el trato a la primera.

4.2.4. Diseño Arquitectura Modelo-Vista-Controlador (MVC)

El prototipo está implementado usando la plataforma JADE y el lenguaje Java siguiendo la arquitectura Modelo-Vista-Controlador. Es un patrón de arquitectura que separa los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.

- **Modelo:** es la representación del dominio específico de la información sobre la cual funciona la aplicación.

- **Vista:** presenta el modelo en un formato adecuado para interactuar, normalmente la interfaz de usuario.

- **Controlador:** responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

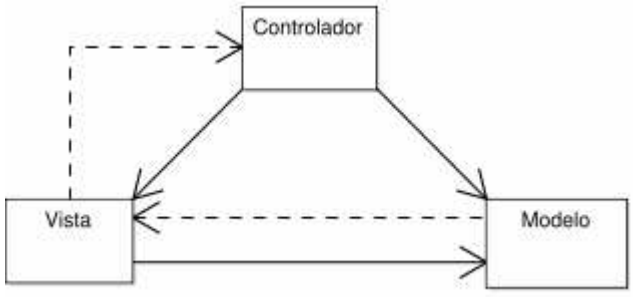


Figura 4.18. Estructura Modelo-Vista-Controlador.

4.2.5. Diagramas Conceptuales (UML)

Mediante los diagramas UML se permite modelar un sistema de software y ver los aspectos que se deben considerar en el análisis y diseño del mismo. Se puede observar cómo está estructurada la arquitectura Modelo-Vista-Controlador, los paquetes que forman el proyecto, las clases y las relaciones entre todos ellos.

Diagramas UML de los paquetes del proyecto

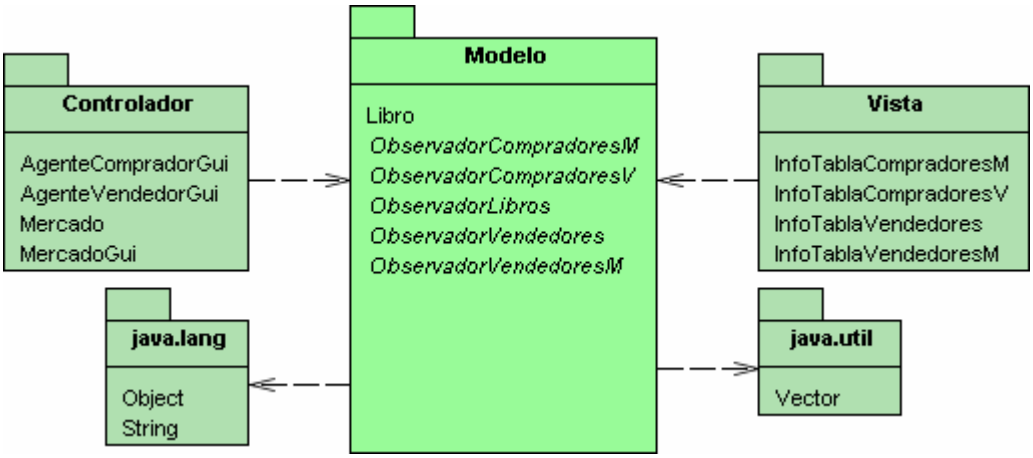


Figura 4.19. Diagrama UML del paquete Modelo.

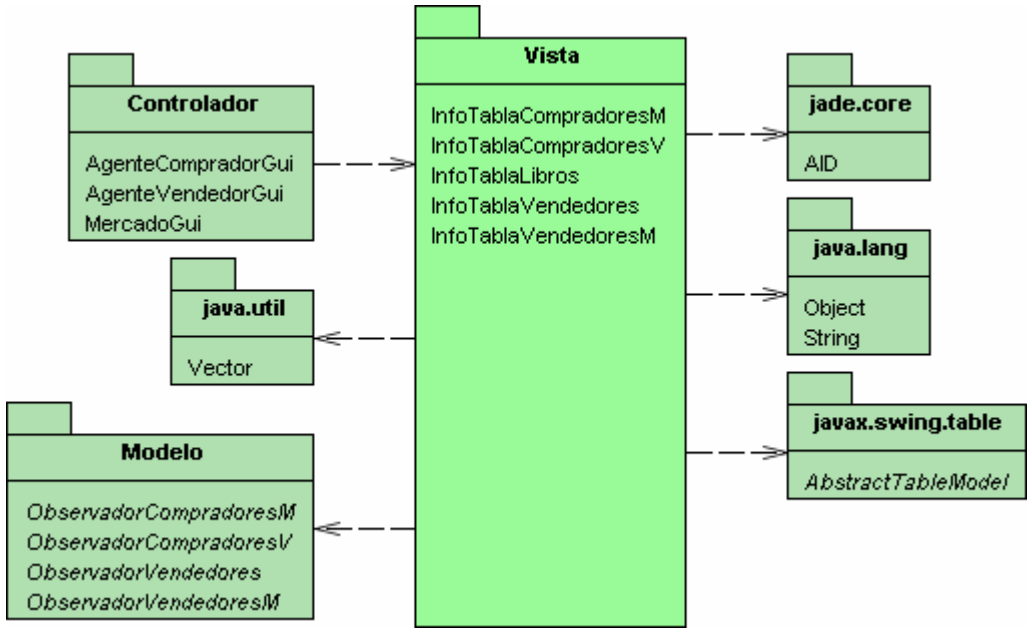


Figura 4.20. Diagrama UML del paquete Vista.

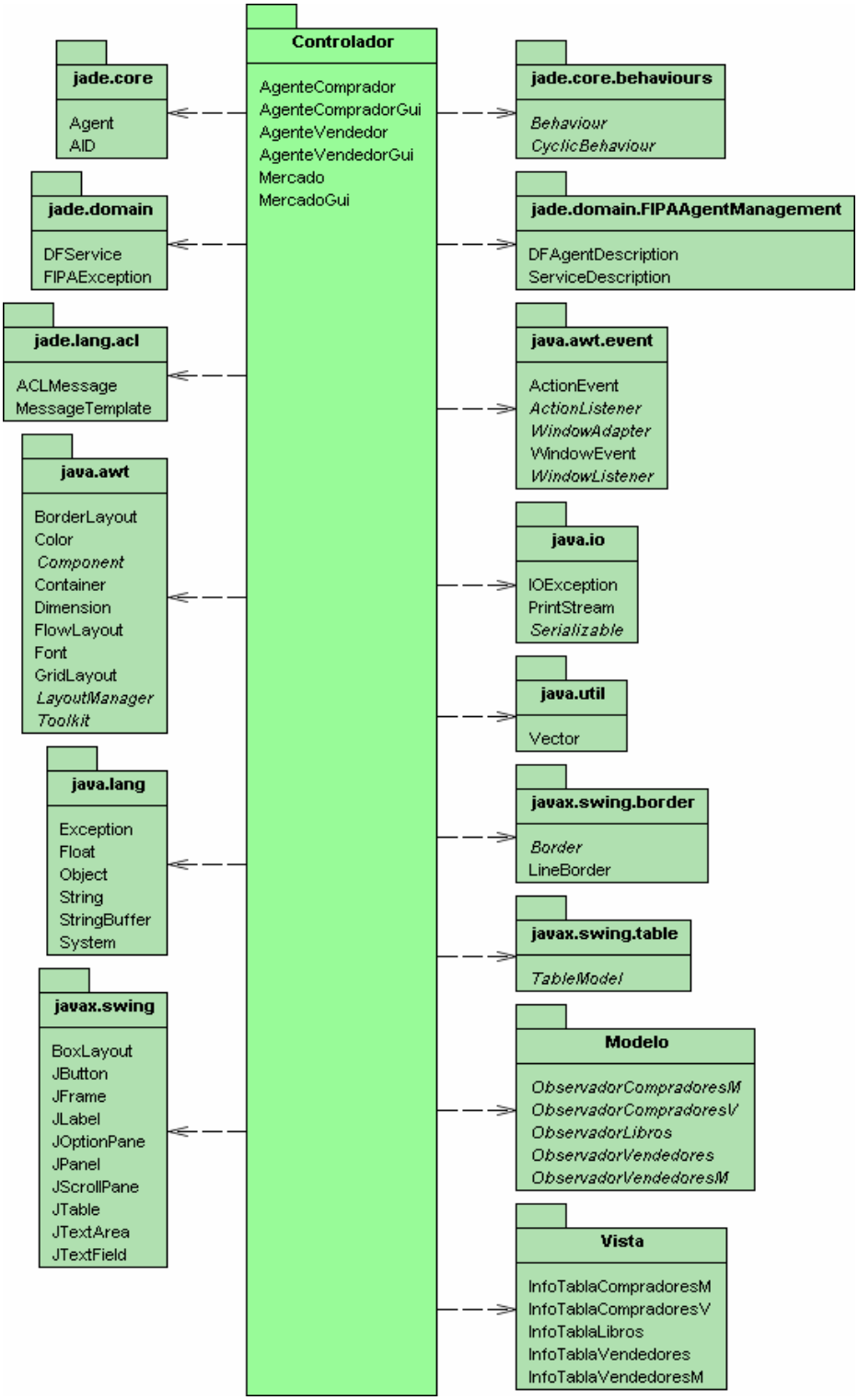


Figura 4.21. Diagrama UML del paquete Controlador.

Diagramas UML de las clases de los agentes

Agente Mercado

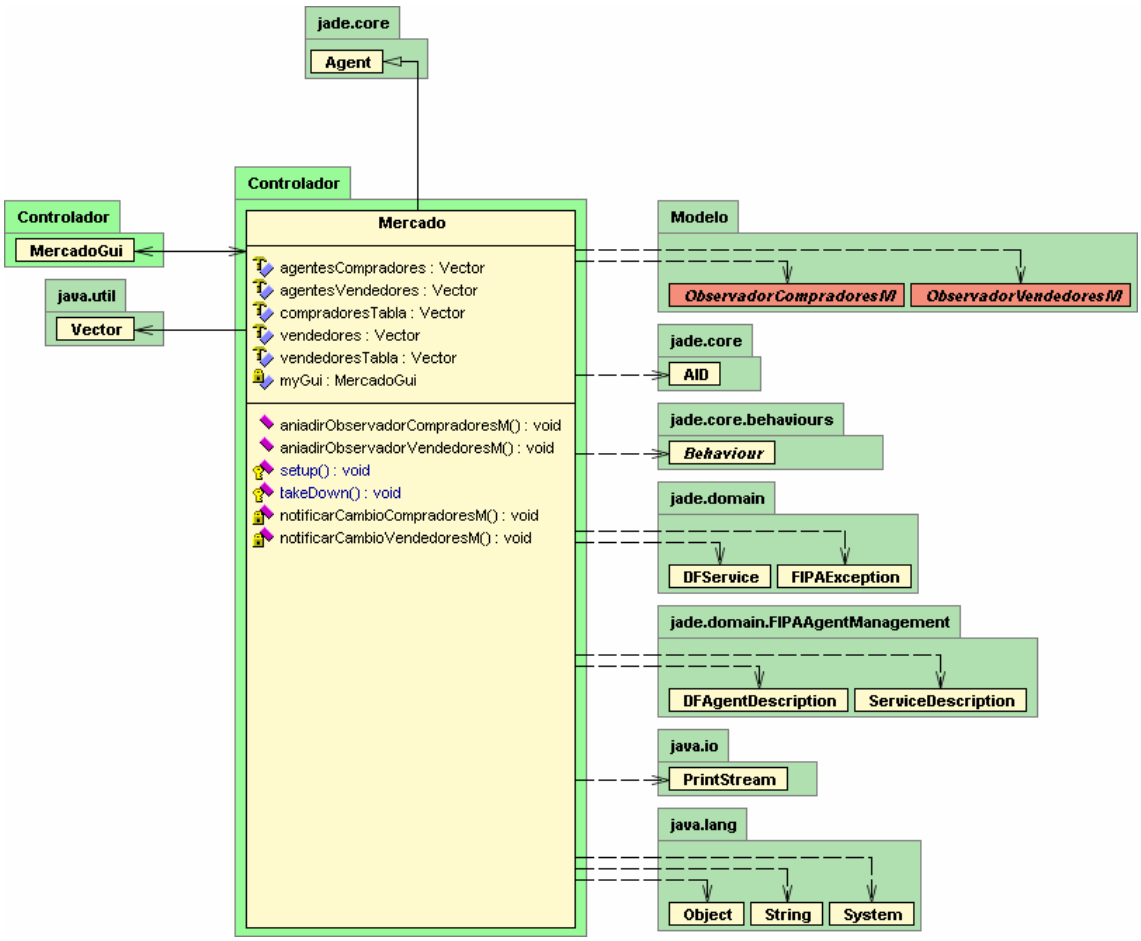


Figura 4.22. Diagrama UML de la clase Mercado.

Agente Comprador

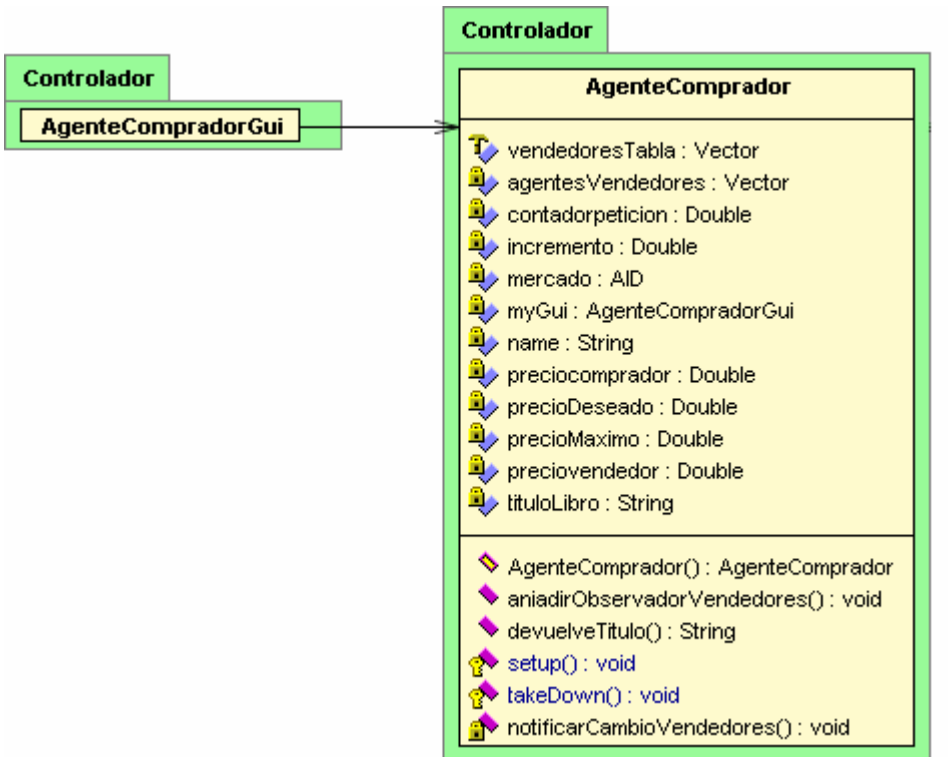


Figura 4.23. Diagrama UML de la clase AgenteComprador.

Agente Vendedor

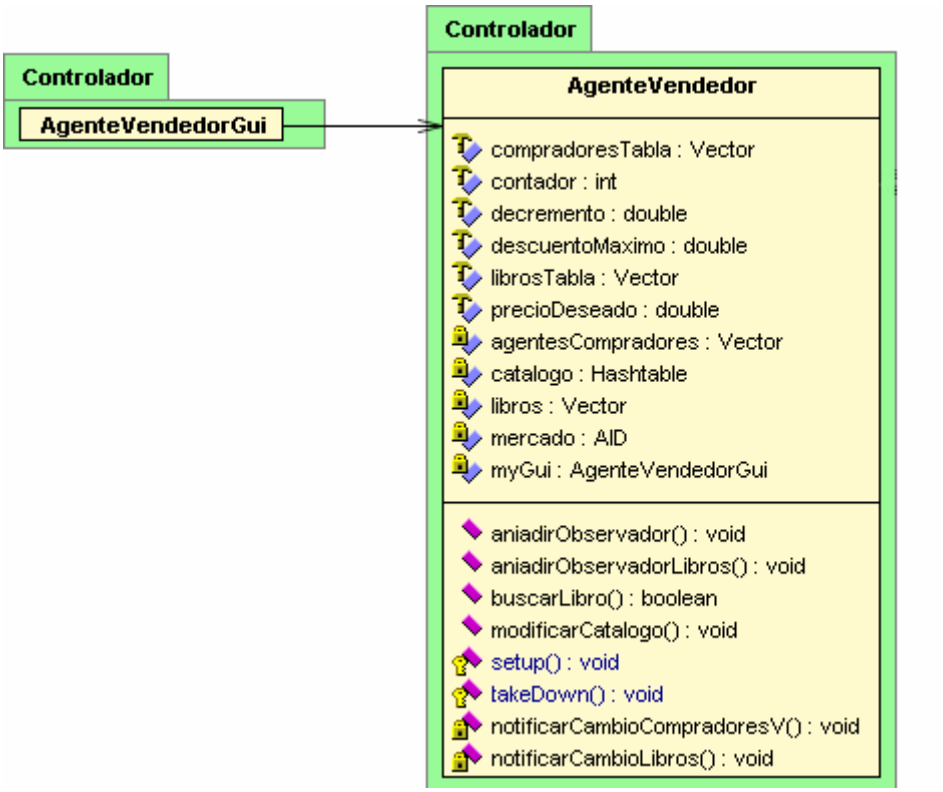


Figura 4.24. Diagrama UML de la clase AgenteVendedor.

Diagramas UML de las interfaces de los agentes

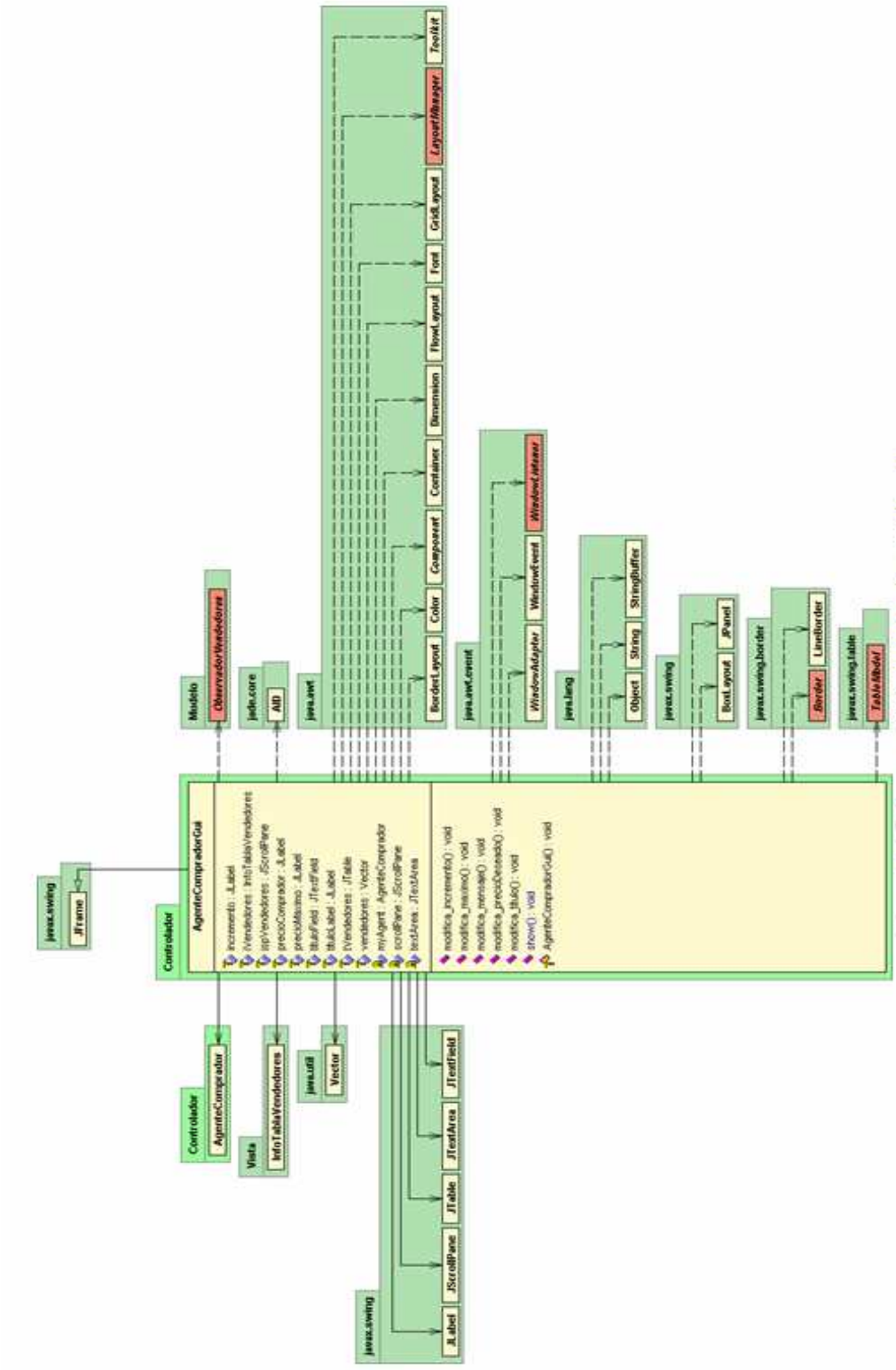


Figura 4.25. Diagrama UML de la interfaz del Mercado.

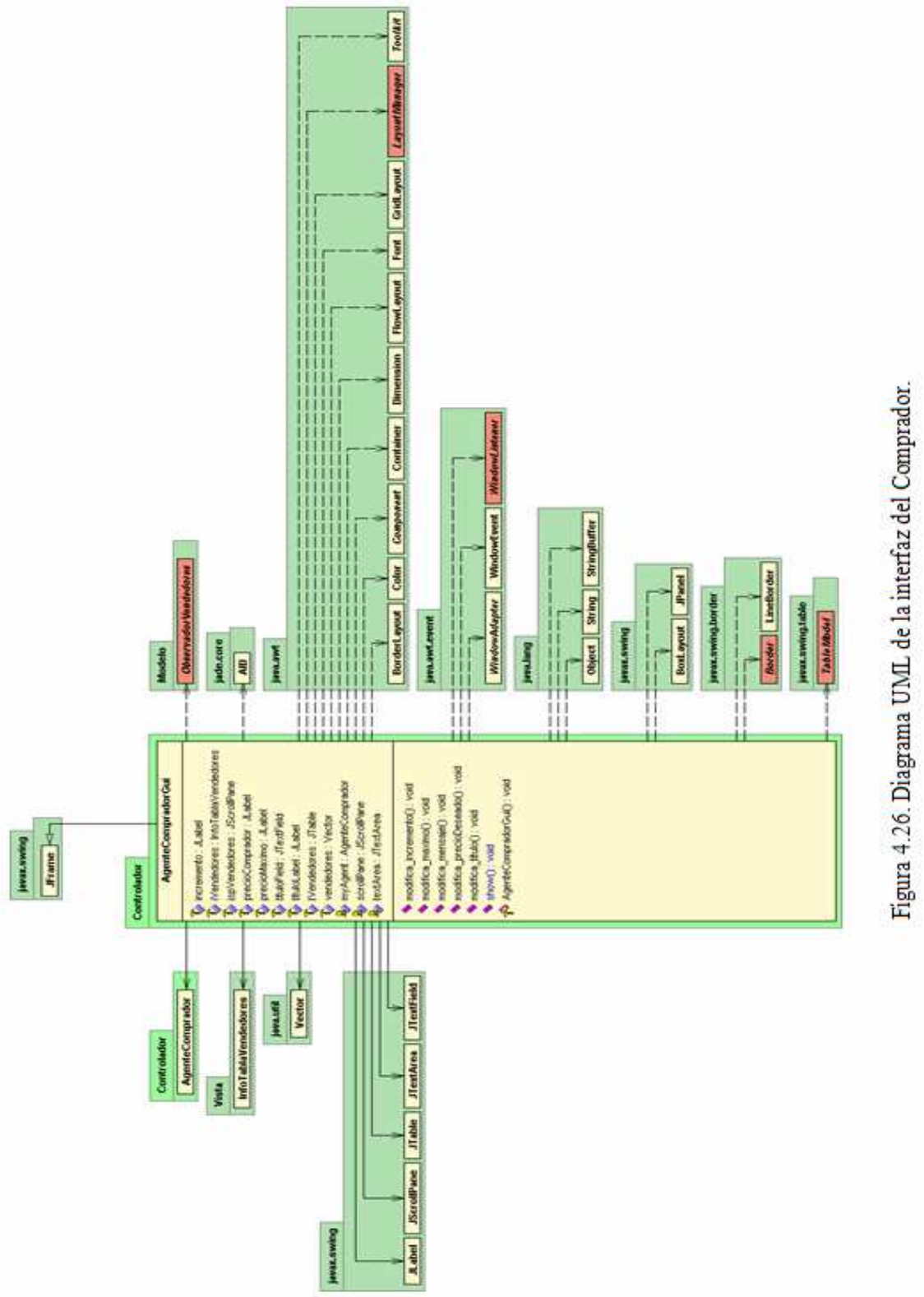


Figura 4.26. Diagrama UML de la interfaz del Comprador.

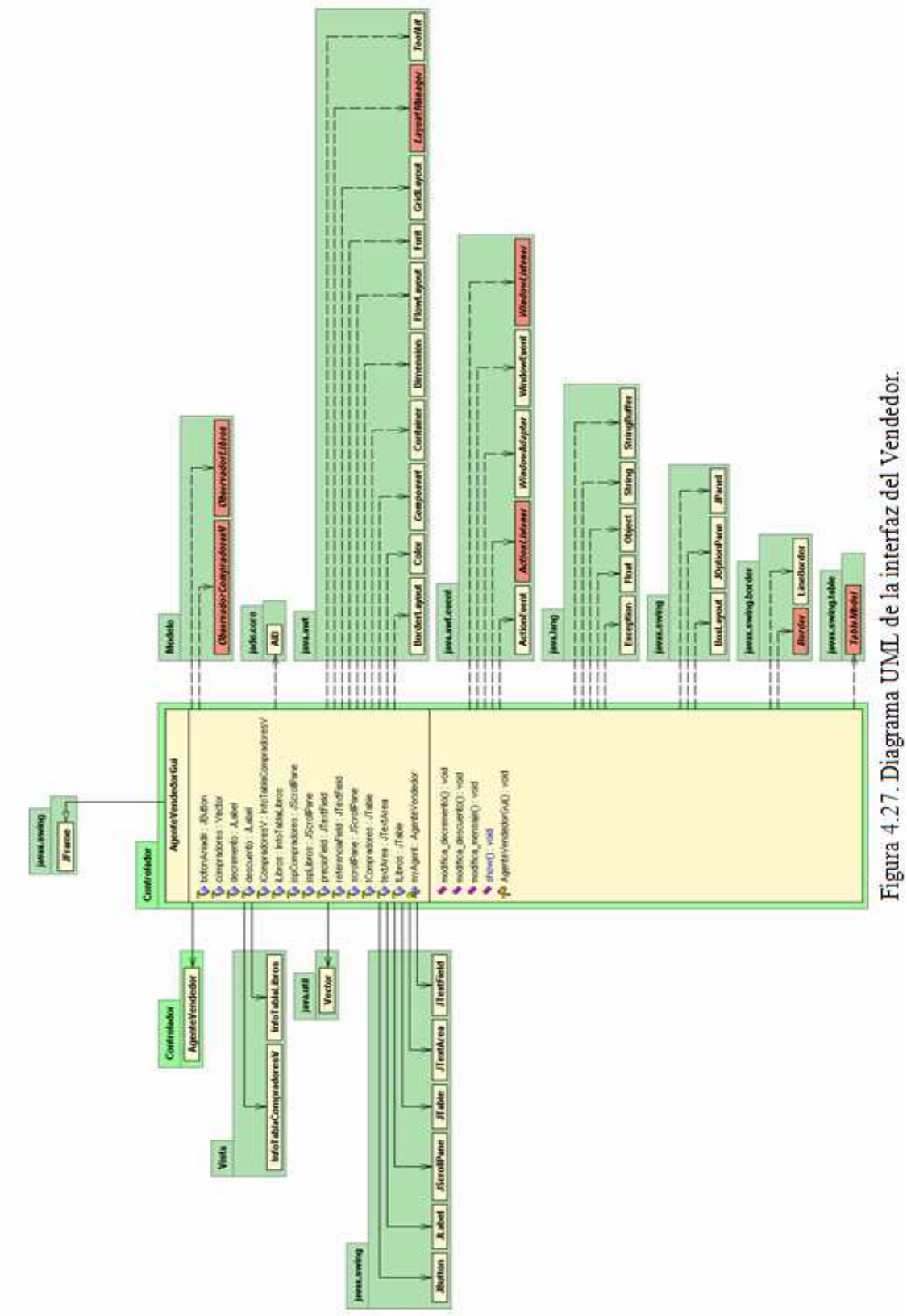


Figura 4.27. Diagrama UML de la interfaz del Vendedor.

Ejemplo de interacción del Modelo-Vista-Controlador

El resto de clases poseen diagramas similares ya que son los correspondientes a las tablas de información que se muestran en las interfaces de los agentes y sus observadores correspondientes. Por lo tanto, sólo se va a mostrar un ejemplo a partir del cual se explique además el funcionamiento de la arquitectura MVC.

La clase "*InfoTablaVendedoresM*" gestiona la información correspondiente a la tabla de los agentes vendedores dados de alta en el Mercado. Esta tabla es mostrada en la interfaz del Mercado y debe actualizarse cada vez que se da de alta o de baja un agente en el Mercado. Esta clase pertenece por lo tanto a la Vista y posee un método para cambiar el contenido de la tabla.

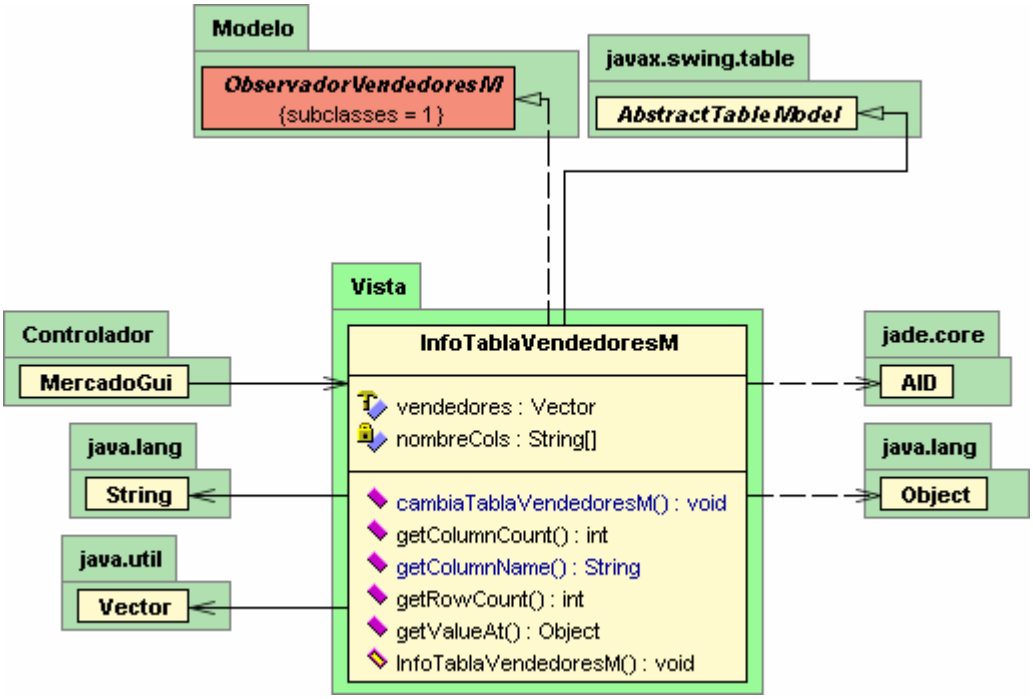


Figura 4.28. Diagrama UML de la clase InfoTablaVendedoresM.

Para controlar si se está modificando la lista de los agentes vendedores existe una clase "*ObservadorVendedoresM*" perteneciente al Modelo cuyo diagrama UML se muestra a continuación. Controla los cambios producidos en la lista de los vendedores y en cuanto se produce invoca al método de *cambiaTablaVendedores* de la clase explicada sobre estas líneas.

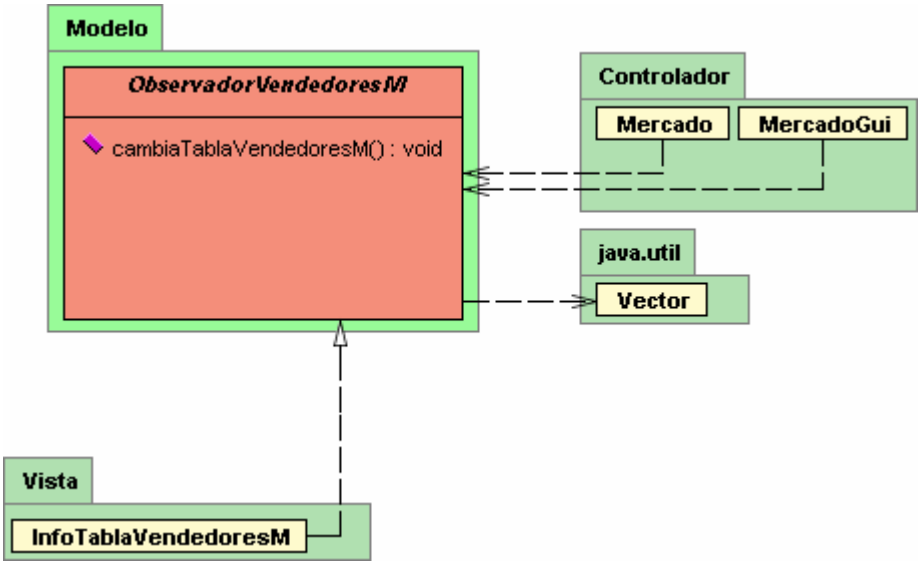


Figura 4.29. Diagrama UML de la clase ObservadorVendedoresM.

En el paquete Controlador se encuentra las clases asociadas al Mercado y la interfaz del Mercado.

De este modo, el funcionamiento del Modelo-Vista-Controlador con la tabla de los vendedores dados de alta en el Mercado sería el siguiente:

- El usuario tiene acceso a la interfaz Web del Mercado y controla los agentes vendedores a través de la tabla que muestra.

- Cuando el usuario crea un nuevo agente vendedor la lista que el Mercado posee aumenta en uno, lo cual se gestiona en la clase Mercado del Controlador.

- El Controlador accede al Modelo para actualizar los datos de la lista.

- El Modelo a su vez manda la modificación a la Vista que muestra al nuevo vendedor en la tabla.

4.3. Diseño de la interfaz

En este apartado se diseñará la interfaz del prototipo atendiendo a una serie de principios de usabilidad. Además, dado que el prototipo está orientado al trabajo a través de la Web, se planteará para un trabajo futuro una posible opción del diseño de la interfaz Web.

4.3.1. Principios de usabilidad

Puede definirse la *usabilidad* [20] como la medida en que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado.

Alguno de los principios de usabilidad que se espera cumpla la interfaz son [20]:

1. **Facilidad de aprendizaje:** el usuario debe poder aprender rápidamente su funcionamiento.

   - *Predecible y Sintetizable*: el usuario debe ser capaz de determinar el resultado de su acción y captar el estado resultante.
   - *Familiar:* el diseño ha de resultar familiar a los usados anteriormente por el usuario.
   - *Consistente:* los mecanismos a utilizar deben ser usados del mismo modo.

2. **Flexibilidad:** debe haber varias formas en que el sistema y el usuario intercambian información.

3. **Robustez:** es la capacidad de tolerar fallos durante la interacción.

   - *Navegable:* el usuario debe poder navegar por el prototipo y conocer en cada momento dónde se encuentra.
   - *Tiempo de Respuesta:* es el tiempo que necesita el sistema para reflejar los cambios de estado a su usuario.
   - *Adecuación de las tareas:* la interfaz debe permitir al usuario realizar todas las acciones disponibles.

4.3.2. Interfaz del prototipo

4.3.2.1. Análisis Previo

Este prototipo está orientado a dos tipos de usuarios:

- El *Administrador*: encargado de la gestión del prototipo y de la creación del agente mercado.
- El *Comerciante* (comprador/vendedor): que crea agentes compradores y/o vendedores.

La información que aparecerá en las distintas pantallas está destinada a conocer cuáles son los parámetros de los agentes y los mensajes que intercambian para comprender el proceso de negociación.

4.3.2.2. Diseño de pantallas

A continuación, se van a mostrar los diseños realizados para las pantallas del prototipo. Nos vamos a encontrar con tres tipos de pantalla correspondientes al Mercado, el Agente Comprador y el Agente Vendedor.

Todas tienen el mismo estilo para mantener la uniformidad pero la pantalla del Mercado tiene un color distinto a las del resto para poder diferenciarla sin dificultad del resto de pantallas.

Todos los aspectos referentes al estilo visual vienen incluidos en la guía de estilos de la interfaz del prototipo.

Pantalla del Mercado

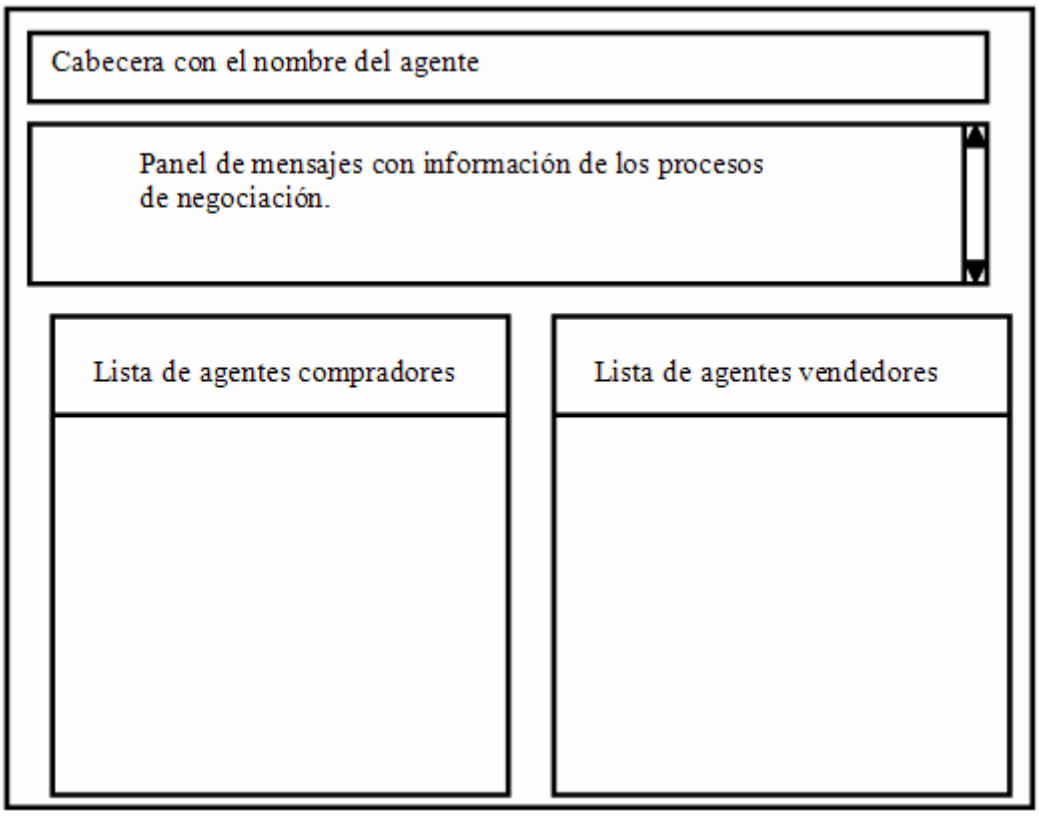


Figura 4.30. Diseño de la pantalla del agente Mercado.

La interfaz está compuesta por una zona superior donde aparece el nombre del agente y una zona destinada para mostrar mensajes de información sobre su funcionamiento. La zona inferior muestra dos tablas con los nombres de los agentes compradores y vendedores dados de alta en el Mercado.

Pantalla del Agente Comprador

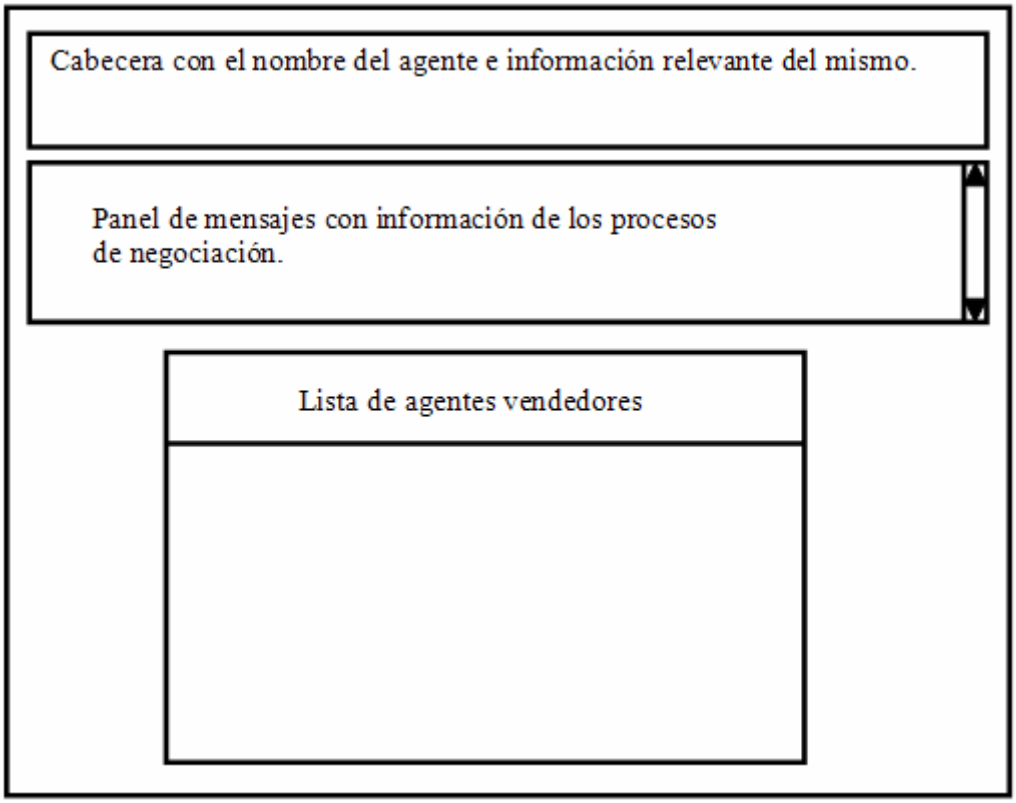


Figura 4.31. Diseño de la pantalla del Agente Comprador.

La interfaz del comprador muestra su nombre y los datos del proceso de negociación: el título del libro, el precio deseado, el tanto por ciento de incremento y el precio máximo.

Seguidamente hay un panel donde aparecen los mensajes donde se informa acerca de las acciones de los procesos de negociación del agente comprador. Por último, aparece una tabla donde se almacena la lista de vendedores potenciales que poseen en su catálogo el artículo que el comprador desea.

Pantalla del Agente Vendedor

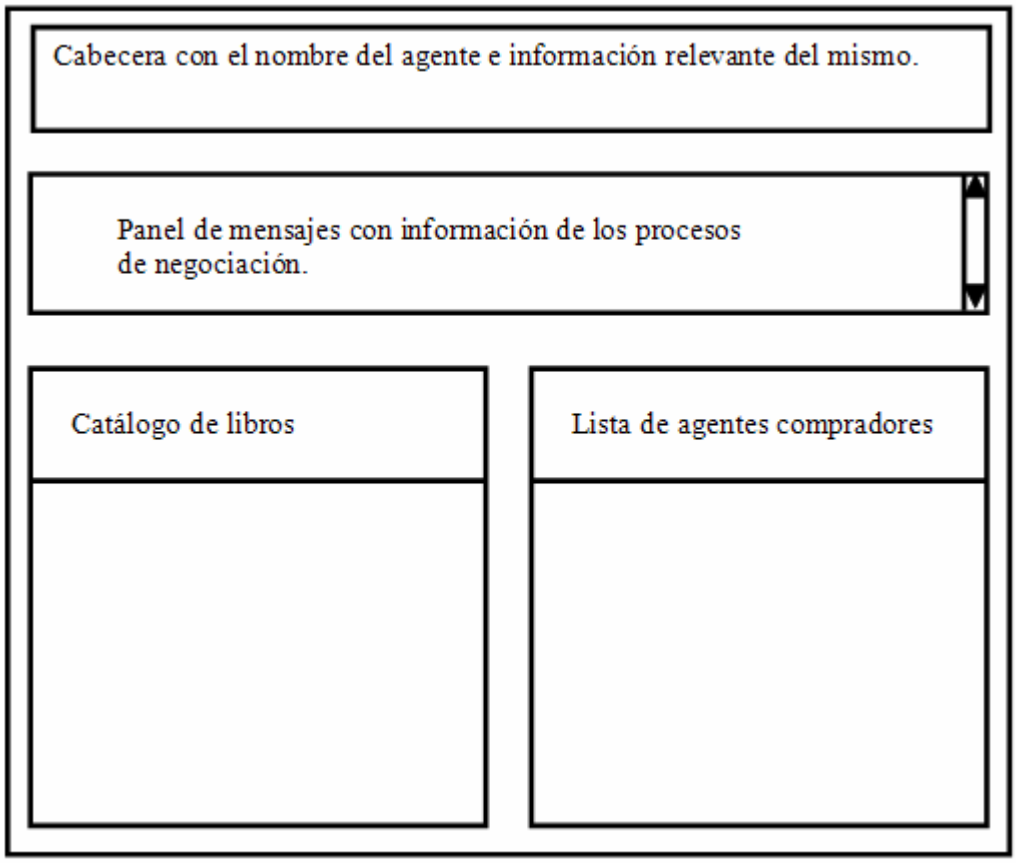


Figura 4.32. Diseño de la pantalla del agente vendedor.

La interfaz está compuesta por una zona superior donde aparece el nombre del agente y la información relacionada con la estrategia del agente, que son los datos que se piden al crearlo.

Para realizar la gestión del catálogo y poder insertar los artículos dispone de un formulario donde se puede especificar el título y el precio del libro. Una zona de mensajes nos muestra las acciones que va realizando el vendedor. Por último, en la zona inferior se muestran dos tablas: una para el catálogo de libros y otra para los compradores conocidos por el vendedor.

4.3.2.3. Guía de estilos

En esta guía de estilos de la interfaz del prototipo se determina la apariencia de las pantallas diseñadas especificando los colores, los tipos de letra, etc. Estas son algunas de las reglas de estilo:

- La apariencia de las tres pantallas será similar facilitando la familiaridad al usuario.
- El Look&Feel (apariencia que se proporciona a los diferentes componentes de una interfaz: botones, cajas de texto, listas,…) usado en todas las pantallas es la que ofrece por defecto Java.
- Los colores y fuentes usados para la pantalla del **agente mercado** son:
    - *Color1:* RVA (rojo, verde, azul): (255,255, 205) que corresponde a un tono beige.
    - *Color2:* RVA (163,255, 255) que corresponde a un tono granate.
    - *Fuente1:* "Comic Sans MS" con tamaño 18 y color2.
    - *Fuente2:* "Comic Sans MS" con tamaño 14 y color2.
    - *Fuente3:* "Comic Sans MS" con tamaño 12 y color2.
- En la interfaz del Mercado el *color1* (granate) se utiliza para todos los textos que aparecen y el *color2* (beige) como fondo de las tablas que muestran las listas.
- En la interfaz del Mercado el empleo de los tipos de fuentes son:
    - La *fuente1:* para el texto de la cabecera donde es necesario resaltar el nombre del agente Mercado.
    - La *fuente2:* para los títulos de las distintas zonas de la pantalla.
    - La *fuente3:* para los textos de las tablas y de la zona de mensajes.
- Los colores y fuentes usados para las pantallas del agente **comprador** y **vendedor** son:
    - *Color1:* RVA (204,219, 253) que corresponde a un tono azul claro.
    - *Color2:* BLUE que corresponde a un tono azul oscuro.
    - *Fuente1*: "Comic Sans MS" con tamaño 18 y color2.
    - *Fuente2:* "Comic Sans MS" con tamaño 14 y color2.
    - *Fuente3:* "Comic Sans MS" con tamaño 12 y color2.
- En la interfaz del Comprador el *color2* (azul oscuro) se utiliza para todos los textos que aparecen y el *color2* (azul claro) como fondo de las tablas que muestran las listas.
- En la interfaz del comprador y el vendedor el empleo de los tipos de fuentes son:
    - La *fuente1:* para el texto de la cabecera donde es necesario resaltar el nombre y el tipo de agente.

- La *fuente2:* para los títulos de las distintas zonas de la pantalla.
- La *fuente3:* para los textos de las tablas y de la zona de mensajes.

4.3.2.4. Mensajes de error

Una vez se conoce cuál va a ser el funcionamiento del sistema y se han diseñado las pantallas de la interfaz es necesario definir los mensajes de error. Estos mensajes son muy importantes para la usabilidad del prototipo ya que si no se especifican correctamente puede provocar que el usuario se encuentre confuso delante de la aplicación.

Algunas directrices para el diseño de los mensajes de error son:

- No recriminar al usuario lo que ha hecho mal sino indicar cómo hacerlo bien.
- Establecer una apariencia adecuada para todos los mensajes.
- Mostrar un mensaje claro que indique al usuario en qué se ha equivocado.

Creación de Agentes

- Si no se introducen todos los parámetros la propia plataforma muestra un mensaje de error en inglés indicando que no se ha podido crear el agente: "*RMA Error! FAILURE received during CreateAgent. Do you want to view the ACL Message?*". Los problemas de este mensaje es que es muy alarmante y aparece en inglés.
- Si los parámetros introducidos no poseen los valores adecuados se indicará que son incorrectos: "*Error: Valor Inválido*".

Dar de alta artículos en el catálogo del Vendedor

- Si se intenta dar de alta un libro que ya existe en el catálogo se muestra: "*Error: el libro se encuentra ya en el catálogo*."
- Si los parámetros introducidos son incorrectos: "*Error: Valor inválido*".

4.3.3. Planteamiento de la interfaz Web

4.3.3.1. Análisis previo

El sitio Web del cual se va a plantear su diseño estará orientado a ser la interfaz del prototipo desarrollado en este proyecto. Será la presentación en la Web de un Mercado Virtual de libros donde los usuarios podrán dar de alta agentes compradores y vendedores para comprar y vender estos artículos.

La aplicación Web va orientada a todos aquellos usuarios habituados al uso de Internet que desean realizar la compra-venta de libros a través de la red sin tener que realizar en persona el proceso de negociación.

Los servicios que se ofrecen serían la creación de agentes compradores y vendedores para que vivan en el Mercado y se comuniquen con otros agentes para realizar su cometido con el mayor grado de satisfacción para el usuario.

4.3.3.2. Diseño

Es importante el correcto diseño de la interfaz y ya que en este caso será un sitio Web se ha realizado un pequeño análisis de usabilidad. De esta manera se pretende que sea una interfaz fácil de usar para lo que el diseño Web estará centrado en el usuario.

Con respecto al *Modelado de Usuario*, es necesario analizar la audiencia del sitio Web que estará formada por usuarios habituales de la Web interesados en el Comercio Electrónico; es decir, en realizar la compra-venta de productos a través de la red.

Un posible *Diseño Conceptual* podría ser el siguiente*:* una estructura del prototipo del sitio Web que constase de un apartado para dar de alta a los agentes y otro para ayuda al usuario. En este caso solo podrían crearse agentes compradores y vendedores.

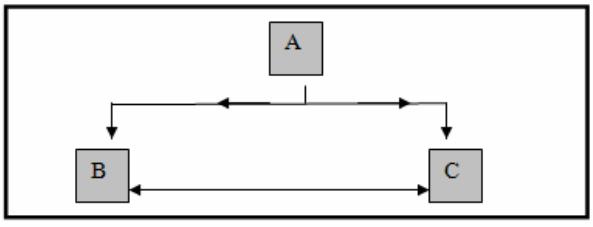


Figura 4.33. Diagrama de entidad-relación del sitio Web.

Existirían tres bloques como mínimo: A (inicio), B (creación de agentes) y C (ayuda). Desde el inicio de la página se pueden encontrar dos bloques: bloque "B" con información para el usuario sobre el proceso de comercio que se realiza y el bloque "C" donde se puede llevar a cabo dicho proceso.

A continuación se muestra la estructura general de la página mostrando su división en capas:

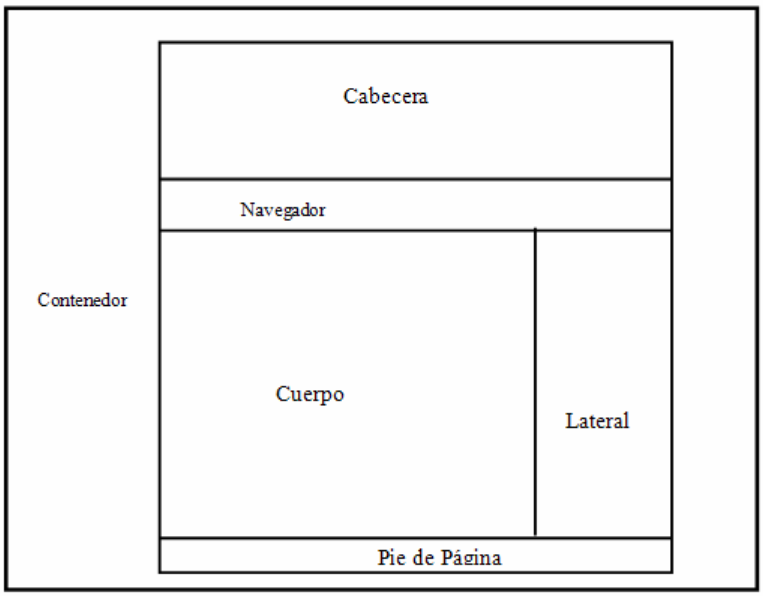


Figura 4.34. Estructura del sitio Web.

La página estaría dividida en varias capas que se encuentran dentro de una capa contenedor: la capa superior para la cabecera, el navegador, el lateral derecho y el cuerpo que será la parte principal de la página y donde irá apareciendo la distinta información.

4.3.3.3. Guía de estilos

Los colores usados para la página son básicamente: granate, beige y gris oscuro para el texto.

La cabecera será una imagen insertada que indicará el título de la Web. Debajo de la misma se encuentra el menú de navegación que constará de tres enlaces: Inicio, Creación de Agentes y Ayuda.

En la parte central de la página, el cuerpo, será donde se inserten los contenidos. En la capa "lateral" se ofrecerá otro tipo de información acerca del funcionamiento del Mercado.

Por último, en la capa inferior "pie" aparecerá información acerca de la autoría de la Web y de enlaces importantes como el del consorcio W3C para indicar que esta página ha sido validada correctamente con respecto a los estándares del w3c.

A continuación se muestra una posible implementación del diseño propuesto. El diseño estético de la interfaz debe estar ligado con la utilidad. Por ello, y siguiendo los principios de usabilidad comentados anteriormente se presentan las distintas pantallas que poseería la interfaz Web, pero sólo aquellas relacionadas con la creación de los agentes compradores y vendedores.

Figura 4.35. Pantalla de Creación del Agente Comprador.

Figura 4.36. Pantalla de Creación del Agente Vendedor.

Como se puede observar todas las páginas del sitio Web mantienen la misma estructura de manera que el usuario tenga constancia en cada momento del nivel de navegación donde se encuentra.

4.4. Diseño de la comunicación entre la interfaz Web y JADE

Para en un futuro integrar el mercado desarrollado en JADE con la interfaz Web puede realizarse simulando los servicios del prototipo como servicios Web. En este apartado se plantearán los conocimientos básicos de estos servicios Web.

4.4.1. Servicios Web

Esta solución requiere un previo de estudio de los servicios Web, su funcionamiento y las tecnologías usadas que a continuación se describen.

Una posible definición de servicios Web [16] sería hablar de ellos como un *conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web.* Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

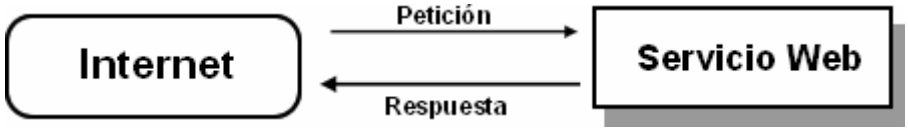


Figura 4.37. Internet y los Servicios Web.

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

En el proceso intervienen una serie de tecnologías estándar que hacen posible esta circulación de información y vienen recogidas en este esquema:

Protocolo de Transporte	HTTP/HTTPS
Codificación de datos y mensajes	SOAP
Descripción del Servicio	WSDL
Búsqueda y Localización de Servicios	UDDI

El grupo de trabajo *W3C Services Architecture Working Group* identifica los componentes funcionales y define las relaciones entre ellos para lograr las propiedades deseadas de una arquitectura global.

SOAP

Son las siglas de *Simple Object Access Protocol* . Este procolo [17] pretende realizar RPC o Remote Procedure Calls, es decir, permite realizar bien en cliente o servidor peticiones mediante http a un servidor Web. Los mensajes deben tener un formato determinado empleando XML para encapsular los parámetros de petición.

XML

Es el lenguaje sobre el que se asientan los servicios Web y sus siglas corresponden a "*Extensible Markup Language*". XML [17] es un conjunto de reglas para definir etiquetas semánticas que organiza un documento en diferentes partes. Además, es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados.

WSDL

El *Lenguaje de Descripción de Servicios Web* [17] es un dialecto basado en XML sobre el esquema que describe un Servicio Web. Un documento WSDL proporciona la información necesaria al cliente para interaccionar con el Servicio Web, establece qué puede hacer el servicio, dónde reside y cómo invocarlo.

UDDI

Corresponde con las siglas del catálogo de negocios de Internet denominado "*Universal Description Discovery and Integration*". Es uno de los estándares básicos de los servicios Web cuyo objetivo es ser accedido por los mensajes SOAP y dar paso a documentos WSDL, en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios Web del catálogo de registros.

El registro de un negocio en UDDI tiene tres partes:

- *Páginas blancas:* dirección, contacto y otros identificadores conocidos.
- *Páginas amarillas:* categorización industrial basada en taxonomías.
- *Páginas verdes*: información técnica sobre los servicios que aportan las propias empresas.

4.4.2. JADE Web Services Integration Gateway (WSIG)

Recientemente se ha desarrollado WSIG [18] que proporciona soporte para los agentes JADE en Servicios Web y para el cual se necesitan nociones de las tecnologías explicadas en el apartado anterior. La versión en funcionamiento en las 1.0 siendo un add-ons (añadido) de la plataforma JADE pudiendo usarse sólo a partir de la versión 3.5 de JADE.

El objetivo de WSIG [18] es exponer servicios proporcionados por los agentes y publicarlos en el JADE DF como un servicio Web con muy pocos esfuerzos adicionales,

proporcionando a los desarrolladores la suficiente flexibilidad para conocer los requerimientos específicos que deben poseer.

Requerimientos

- Java JRE v5.0 o superiores
- JADE v 3.5 o superiores.
- Contenedor de servlets como Jakarta Tomcat.

Arquitectura

WSIG soporta el estándar de los servicios Web, conteniendo WSDL para servicio de descripciones, soporte de mensajes SOAP y un repositorio UDDI para publicar servicios Web usando tModels. Una aplicación Web está compuesta por dos elementos principales [18]:

WSIG Servlet. Es el front-end hacia el mundo de Internet y es responsable de:
• Servir las peticiones HTTP/SOAP. • Extraer el mensaje SOAP. • Preparar la acción del agente y pasársela al agente WSIG. • Convertir el resultado de la acción en un mensaje SOAP. • Preparar la respuesta HTTP/SOAP para que sea enviada al cliente.
WSIG Agent. Es la puerta entre la Web y el mundo de los agentes y es responsable de:
• Recuperar las acciones de los agentes recibidas desde el servlet para los agentes que actualmente pueden servirles y conseguir las respuestas para ellos. • Suscribir al DF de JADE para recibir notificaciones sobre los registros y las bajas. • Crear la WSDL correspondiente para cada agente registrado con el DF y publicar el servicio en un UDDI si es necesario.

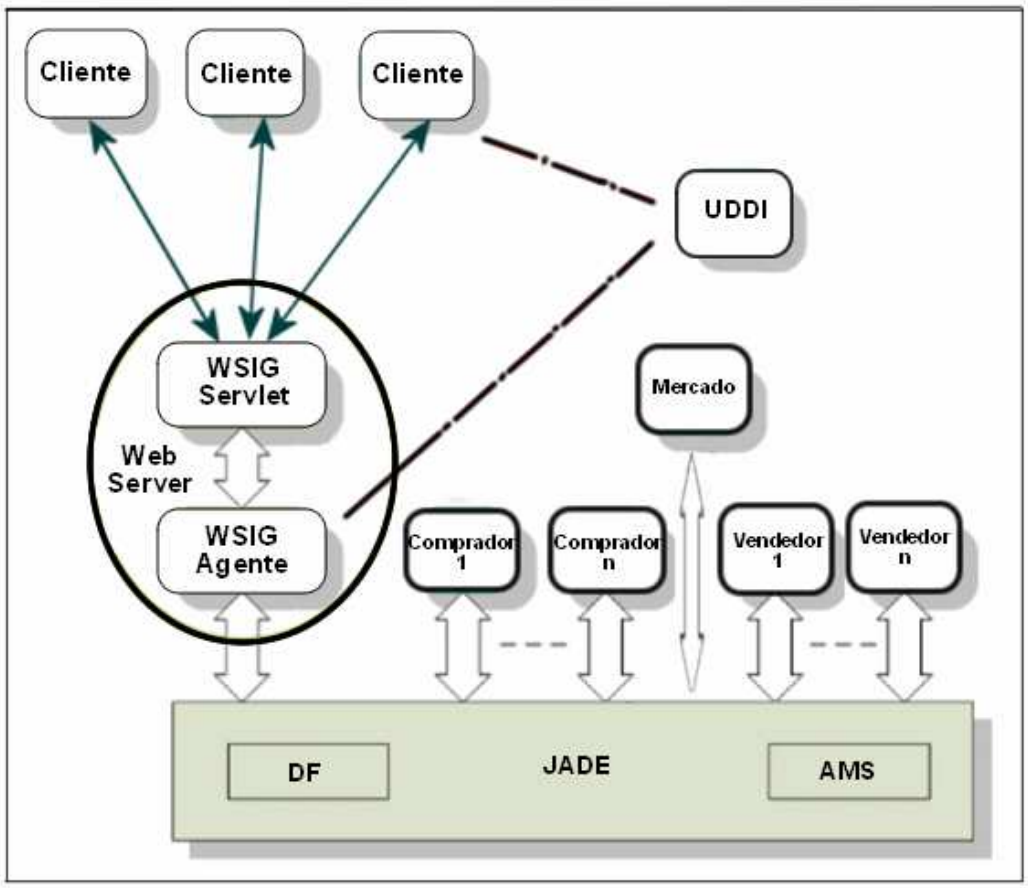


Figura 4.38. Arquitectura WSIG.

Los dos procesos principales están continuamente activos en la aplicación Web y son el proceso responsable de interceptar los registros y bajas del DF y el proceso responsable de servir las peticiones del servidor Web para lanzar las correspondientes acciones de los agentes. Lo importante ahora es establecer cuáles son los servicios Web que se ofrecerían y a qué corresponderían en la plataforma JADE.

En esta arquitectura se necesitará de un *agente intermediario* entre la interfaz Web y la plataforma JADE se encargará de tomar los datos de la página Web para enviarlos a los agentes de la plataforma y viceversa. Un *agente intermediario* [14] es un agente que ayuda a otros en la localización y conexión con proveedores de Servicios, en este caso, servicios Web que estarán asociados a las tareas de los agentes de la plataforma.

Los *Servicios Web* podrían ser:

- **Crear Comprador.** Poseerá una interfaz Web a través de la cual el usuario rellenará los datos necesarios para la creación del agente.

- **Crear Vendedor**. Contará con una interfaz Web similar a la del agente comprador.

- **Crear Mercado**. Este servicio no poseería de ninguna interfaz de creación ya que deberá encontrarse en funcionamiento junto con la plataforma.

- **Comprobar la evolución del Mercado**. En este caso la interfaz mostraría información relacionada con las actividades realizadas en el Mercado.

Capítulo 5

Resultados y análisis de la experimentación

En este capítulo se describirán las pruebas realizadas para la comprobación del correcto funcionamiento del prototipo: pruebas de caja blanca y de caja negra. También se analizará el resultado del impacto de la interfaz sobre el usuario. Por último, se explicarán los resultados obtenidos de la experimentación de los procesos de negociación.

5.1. Evaluación y pruebas

Durante el desarrollo de un software es necesario realizar una serie de pruebas para comprobar si el sistema que se está construyendo funciona correctamente. Para ello se realizan la "*Verificación y Validación*" que es el nombre genérico dado a los procesos de chequeo que aseguran que el software satisface sus especificaciones y cubre las necesidades del usuario.

La diferencia entre verificación y validación la presenta **Boehm** para determinar a qué responde cada una de ellas:

- La **Verificación** es: ¿Estamos construyendo bien el producto?
- La **Validación** es: ¿Estamos construyendo el producto adecuado?

Ambos conceptos persiguen: detectar y corregir defectos tan pronto como sea posible en el ciclo de vida del software y mejorar la calidad y fiabilidad del software, entre otros objetivos.

En los siguientes apartados se realizarán una serie de pruebas del software diseñadas de manera que tengan la mayor probabilidad de encontrar el mayor número de errores con la mínima cantidad de esfuerzo y tiempo.

5.1.1. Pruebas de caja blanca

Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba. De esta manera se intenta controlar que se ejecuten todos los caminos, que se ejerciten todas las decisiones lógicas, se ejecuten todos los bucles en sus límites,…

Las pruebas que podrían realizarse serían la "*Prueba del Camino Básico*" y la "*Prueba de los Bucles*" pero debido a que la ejecución del Sistema depende de los comportamientos de los agentes y no de un camino principal a seguir a partir de lo cual se ha determinado no realizar este tipo de pruebas.

Por otro parte, se puede resaltar que a lo largo del proceso de desarrollo del prototipo se ha controlado la ejecución correcta de las distintas tareas.

5.1.2. Pruebas de caja negra

Este tipo de pruebas se centra en los requisitos funcionales del software e intenta encontrar errores de funciones incorrectas, de interfaz, de estructura de datos, de rendimiento y de inicialización y terminación.

Existen distintos métodos y el usado en este caso ha sido el "*Método de la Partición Equivalente*" con el cual se trata de obtener un pequeño subconjunto de todas las posibles entradas con la mayor probabilidad de encontrar todos los errores.

Las pruebas realizadas sobre la aplicación se han ido realizando a medida que se añadían funcionalidades complicándose con el tiempo. Esto ha sido necesario para poder ir detectando los errores lo antes posible ya que hay que evitar dejar todas las pruebas para cuando el software ya está completamente programado.

Para realizar estos casos de prueba hay que determinar los valores de entrada y las clases de equivalencia válidas e inválidas y a partir de estas crear casos de prueba que engloben la mayor cantidad de clases de equivalencia a la vez.

Creación de un Agente Comprador

Condiciones de Entrada	Clases de equivalencia válidas	Clases de equivalencia inválidas
Nombre del Agente	(1) String no vacío	(2) Otro formato
Título del Libro	(3) String no vacío	(4) Otro Formato
Precio Deseado	(5) precio deseado>0	(6) Valores negativos
Precio Máximo	(7) precio máximo>0	(8) Valores Negativos
Incremento	(9) 0> incremento<=100	(10)Valores Negativos (11) incremento> 100

Caso de prueba 1 (engloba las clases de equivalencia: 1, 3, 5, 7, 9)	Nombre del agente: Comprador Título del libro: Marina Precio Deseado: 12 Precio Máximo: 15 Incremento: 2
Caso de prueba 2: (engloba las clases de equivalencia: 2)	Nombre del agente: Título del libro: El Ocho Precio Deseado: 12 Precio Máximo: 15 Incremento: 2
Caso de prueba 3: (engloba las clases de equivalencia: 4)	Nombre del agente: Comprador Título del libro: Precio Deseado: 12 Precio Máximo: 15 Incremento: 2
Caso de prueba 4: (engloba las clases de equivalencia: 6)	Nombre del agente: Comprador Título del libro: La Catedral del Mar Precio Deseado: -5 Precio Máximo: 15 Incremento: 2
Caso de prueba 5: (engloba las clases de equivalencia:8)	Nombre del agente: Comprador Título del libro: La Sombra del Viento Precio Deseado: 12 Precio Máximo: -1 Incremento: 2
Caso de prueba 6: (engloba las clases de equivalencia:10)	Nombre del agente: Comprador Título del libro: Los renglones torcidos de Dios Precio Deseado: 12 Precio Máximo: 15 Incremento: -2

Caso de prueba 7: (engloba las clases de equivalencia:11)	Nombre del agente: Comprador Título del libro: El Club Dumas Precio Deseado: 12 Precio Máximo: 15 Incremento: 120

Creación de un Agente Vendedor

Condiciones de Entrada	Clases de equivalencia válidas	Clases de equivalencia inválidas
Nombre del Agente	(1) Formato: String no vacío	(2) Otro formato
Descuento Máximo	(3) 0> incremento<=100	(4)Valores Negativos
Decremento	(5) 0> incremento<=100	(6) Valores Negativos (7) incremento >100

Caso de prueba 1 (engloba las clases de equivalencia: 1, 3, 5)
Nombre del agente: Vendedor Descuento Máximo: 15 Decremento: 2
Caso de prueba 2 (engloba las clases de equivalencia: 2)
Nombre del agente: Descuento Máximo: 15 Decremento: 2
Caso de prueba 3 (engloba las clases de equivalencia: 4)
Nombre del agente: Vendedor Descuento Máximo: -2 Decremento: 2
Caso de prueba 4 (engloba las clases de equivalencia: 6)
Nombre del agente: Vendedor Descuento Máximo: 15 Decremento: -2
Caso de prueba 5 (engloba las clases de equivalencia: 7)
Nombre del agente: Vendedor Descuento Máximo: 15 Decremento: 110

Resultados de las pruebas

Tras realizar estas pruebas se comprueba que las clases inválidas han sido controladas de manera que cuando se produzca alguna no se pueda realizar la creación del agente comprador ni la creación del agente vendedor y se notifique el error cometido.

5.2. Estudio de usabilidad

En este apartado se realiza un estudio sobre la interfaz que el usuario debe utilizar para interactuar con el prototipo. Este estudio se realizará a usuarios administradores y usuarios comerciantes (compradores/vendedores).

El problema que se plantea en el usuario **administrador** es que a este nivel el usuario debe aprender a lanzar la plataforma y a través de ella aprender a crear los agentes. Por lo tanto, se ha realizado sobre usuarios con conocimientos avanzados de informática capaces de ser administradores.

Número de usuarios a reclutar	1
Requisitos de los reclutados	El personal reclutado ha de estar familiarizado la informática.
Lugar de realización de la prueba	La prueba se realiza frente a un ordenador personal con las plataformas necesarias instaladas.
Preparación de los usuarios	Se informa a los reclutados del funcionamiento de la aplicación informándoles del funcionamiento de la plataforma JADE y los agentes.
Observador	La autora de este proyecto para poder advertir las reacciones de los usuarios.

Datos del usuario
Nombre: María **Ocupación:** Informática **Descripción:** Persona relacionada con el mundo de la informática usándola diariamente tanto en su vida personal como profesional.
Test de usabilidad
¿Sabe realizar la instalación del software requerido? Haciendo uso del manual de usuario es capaz de realizar las instalaciones oportunas. **¿Identifica el funcionamiento de la plataforma correctamente?** No tiene ningún problema. **¿Identifica la funcionalidad del Mercado?** A partir de la interfaz del Mercado y del trabajo que realizan los agentes detecta que el Mercado es el nexo de unión entre ellos. **¿Sabe realizar las acciones principales?** Tras consultar el manual de usuario no tiene ningún problema en crear el Mercado y comprobar su correcta creación. **¿Sabe consultar el resultado de las negociaciones?** Tanto en las pantallas de los agentes como del Mercado consulta las acciones del proceso de negociación comentando el resultado de los agentes de prueba que ha creado.

Es el momento de realizar un estudio de usabilidad sobre aquellas funcionalidades del prototipo que realizarán los **usuarios comerciantes** (compradores y vendedores). En la tabla siguiente se muestran los parámetros del estudio.

Número de usuarios a reclutar	2
Requisitos de los reclutados	El personal reclutado ha de estar familiarizado con Internet y a ser posible con el Comercio Electrónico.
Lugar de realización de la prueba	La prueba se realiza frente a un ordenador personal con las plataformas necesarias instaladas.
Preparación de los usuarios	Se informa a los reclutados del funcionamiento de la aplicación pero sin entrar en detalles del funcionamiento interno de la plataforma.
Observador	La autora de este proyecto para poder advertir las reacciones de los usuarios.

Los resultados de las pruebas determinan que los usuarios comprenden el funcionamiento de la plataforma sin dificultad encontrando la mayor dificultad en determinar qué estrategia llevarán a cabo.

Datos del usuario
Nombre: José Ángel **Ocupación:** Empresario **Descripción:** Persona acostumbrada al uso del ordenador y de Internet en su vida cotidiana como unas herramientas más de su trabajo. Además, ha comprado con anterioridad productos usando el negocio electrónico y está interesado en las nuevas tecnologías.
Test de usabilidad
¿Identifica el sitio correctamente? Sí, a partir de la información inicial que se le ha proporcionado sobre el prototipo no tiene ningún problema en advertir de qué trata. **¿Cuáles son las principales funcionalidades que puede realizar?** Determina sin dificultad cuáles son las principales funcionalidades: comprar y vender a través de los agentes. **¿Sabe crear un agente comprador? ¿Y un agente vendedor?** Tras consultar en el manual qué parámetros son necesarios no tiene ningún problema en crear un agente comprador. Tras conocer cómo se crea un agente comprador le resulta más fácil crear un agente vendedor. **¿Sabe consultar el resultado de las negociaciones?** Sí, advierte que puede controlar el resultado de las negociaciones en el Mercado y en el agente que ha creado. **¿Sabe qué función tiene el Mercado?** Como un típico mercado que sirve de encuentro entre los agentes compradores y vendedores. **¿Consulta el manual de usuario?** Sí, lo consulta para saber qué valores deben poseen los parámetros de los formularios.

Datos del usuario
Nombre: Miguel **Ocupación:** Empresario **Descripción:** Persona acostumbrada al uso del ordenador y de Internet en su vida cotidiana para el ocio y como comunicación en su trabajo.
Test de usabilidad
¿Identifica el sitio correctamente? No presenta ningún problema. **¿Cuáles son las principales funcionalidades que puede realizar?** Comprar y vender a través de la plataforma presentada. **¿Sabe crear un agente comprador? ¿Y un agente vendedor?** Primero decide crear un agente comprador para comprobar el funcionamiento y sólo pregunta qué parámetros son los que debe introducir y por qué. Crea varios agentes vendedores con diferente oferta de libros para comprobar su funcionamiento. **¿Sabe consultar el resultado de las negociaciones?** Observa cómo en el panel de sus agentes creados se le muestra el resultado de los procesos. **¿Sabe qué función tiene el Mercado?** El Mercado permite que los agentes se relacionen entre ellos. **¿Consulta el manual de usuario?** Hace uso del manual para consultar cómo crear los agentes y qué parámetros necesita especificar para la estrategia que llevará el agente a cabo.

5.3. Experimentación

5.3.1. Estructura de la experimentación

Para realizar la comprobación del funcionamiento del prototipo es necesario someterlo a una serie de pruebas para comprobar los resultados obtenidos.

Debido al funcionamiento de la plataforma JADE puede determinarse que funciona correctamente si los agentes son capaces de contactar entre ellos, intercambiar mensajes, negociar e incluso llegar a acuerdos. El prototipo realiza estas actividades con éxito mostrándose a continuación una serie de ejemplos que lo ilustran.

Los resultados de la experimentación vienen determinados por el tipo de estrategia diseñada y los parámetros que se establezcan. En este prototipo se pretende que ambas partes negociadoras lleguen a un acuerdo que satisfaga a ambas partes estableciendo una serie de parámetros:

Agente Comprador:

- Libro solicitado
- Precio Deseado
- Incremento
- Precio Máximo

Agente Vendedor:

- Decremento
- Descuento Máximo

Para ilustrar este apartado de una manera sencilla se realizará a través de una serie de ejemplos en los cuales se podrá observar el correcto funcionamiento del proceso de negociación.

5.3.2. Resultados

De manera general puede resaltarse el éxito de la aplicación ya que cumple con su objetivo: ser una plataforma donde se comuniquen compradores y vendedores para obtener acuerdos de negociación ventajosos para ambos. Se ilustra este apartado con una batería de ejemplos a partir de los cuales se pueden observar los pasos del proceso de negociación y su resultado.

Ejemplo 1: En este primer ejemplo el agente comprador desea un determinado artículo. Tras establecer contacto con los agentes vendedores a través del Mercado elige al mejor vendedor, que teniendo el producto en su catálogo, ofrece un mejor precio de salida. A partir de este momento se suceden las ofertas y contraofertas hasta obtener un acuerdo que beneficie a ambas partes. Los parámetros de negociación son los siguientes:

Agente Comprador:	**Agente Vendedor:**
Libro Solicitado: "Tuareg"	*Catálogo:* disponible libro.
Precio Deseado: 20 €	*Decremento:* 2%
Incremento: 1 %	*Descuento Máximo:* 20 €
Precio Máximo: 21 €	

Figura 5.1. Negociación con acuerdo.

Ejemplo 2: El agente comprador desea un producto. Establece contacto a través del Mercado con los agentes vendedores y elige aquel que ofrezca un mejor precio de salida. Es con este vendedor con quien iniciará el proceso de negociación pero en este caso no se llega a un acuerdo debido a que llega un momento en la negociación en la que el precio que debe ofrecer el comprador es superior al precio máximo que está dispuesto a pagar.

Agente Comprador:	**Agente Vendedor:**
Libro Solicitado: "Tuareg"	*Catálogo:* disponible libro.
Precio Deseado: 16 €	*Decremento:* 2%
Incremento: 2 %	*Descuento Máximo:* 20 €
Precio Máximo: 18 €	

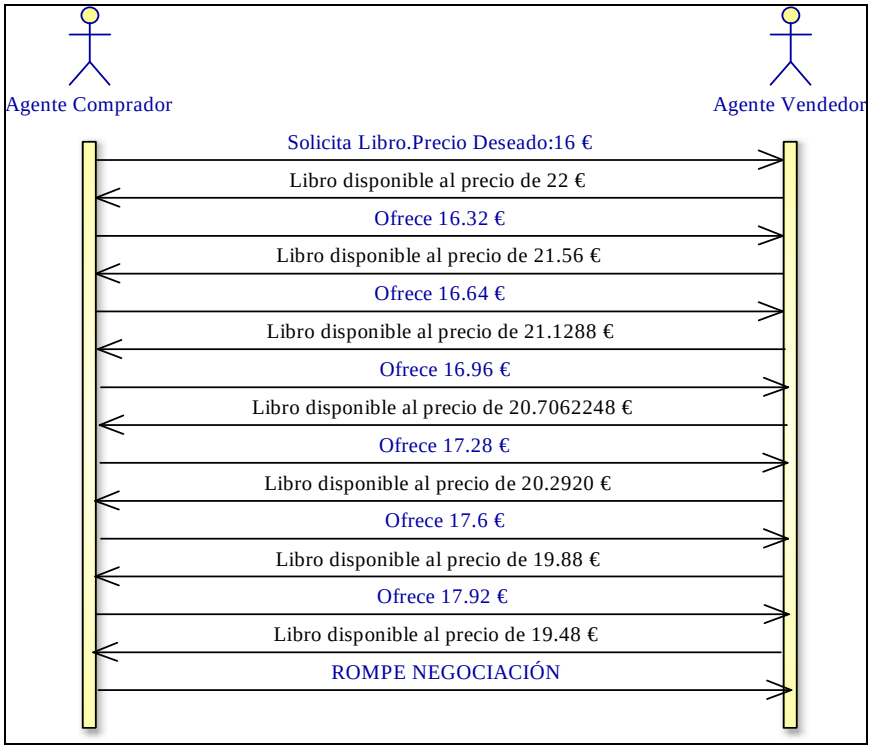


Figura 5.2. Negociación sin acuerdo.

Capítulo 6

Conclusiones finales

6.1. Valoración personal

Tras completar el desarrollo del proyecto y alcanzar los objetivos establecidos al comienzo, creo que es necesario resaltar que es de vital importancia a la hora de realizar un proyecto de estas características seguir las fases establecidas por la Ingeniería del Software.

En cuanto al *análisis y diseño del prototipo* considero que son los aspectos más importantes del proceso y que no pensaba que tuviera los conocimientos suficientes para realizarlos. Sin embargo, en el desarrollo de los mismos he descubierto que sí poseo estos conocimientos, adquiridos sobre todo en estos dos últimos años de carrera, y que son fundamentales. El establecer qué debe hacer el sistema y cómo, así como los errores posibles, las entradas y salidas de datos, son aspectos que son necesarios especificar antes de comenzar a programar. De este modo, me ha resultado mucho más sencilla la fase de implementación ya que tenía perfectamente claro qué era lo que debía hacer.

Sobre todo me gustaría resaltar en el *diseño de la interfaz* lo importante que es realizar un análisis y diseño correcto teniendo en cuenta la *usabilidad*. Al principio pueden parecer conceptos tan obvios que se comete el error de pasarlos por alto pero que son imprescindibles para diseñar una interfaz sencilla y útil en la que el usuario se encuentre lo más a gusto posible,

lo cual es muy importante ya que no hay que olvidar que estamos desarrollando un sistema para los usuarios no para nosotros mismos.

La *implementación del prototipo* me ha permitido aplicar los conocimientos de programación adquiridos a lo largo de la carrera en lenguajes tan usados como Java, HTML y CSS. Aplicando técnicas como el Modelo-Vista-Controlador que permiten tener el código mejor estructurado facilitando su desarrollo y posterior mantenimiento. Además, el uso de la plataforma y el modo en que se ejecutan los comportamientos, de forma cíclica y no de una forma tan lineal, me ha permitido comprobar que con unos buenos "cimientos" en programación puede hacerse frente a cualquier lenguaje o forma de programación.

La plataforma JADE ha presentado pequeños problemas debido a que al ser un proyecto de investigación existe una escasa documentación del mismo, y en algunos casos las documentaciones oficiales están obsoletas e incompletas lo que dificulta el entendimiento de diversos aspectos.

Para finalizar, he de comentar que ha sido una gran satisfacción personal poder desarrollar este proyecto por mí misma, aplicando los conocimientos que en estos cinco años he adquirido en esta escuela.

6.2. Líneas de trabajo futuras

Este prototipo presenta una estructura sencilla de comunicación y de negociación donde se puede observar el trabajo de los agentes a través de una sencilla interfaz. Una vez que se han visto los principios clave del funcionamiento de la plataforma y las posibilidades que ofrecen los agentes pueden realizarse una serie de ampliaciones:

- Diseñar una interfaz Web que posea restricciones de seguridad creando por ejemplo cuentas de usuario en las que un usuario dado de alta pueda ir comprobando cómo están funcionando sus agentes. Además, se podría proporcionar más información al usuario pudiendo a través de una sesión gestionar sus agentes, los catálogos, modificar su estrategia,…

- Aumentar la comunicación entre el usuario y la plataforma de manera que no sólo se reciba un mensaje en el que se muestre el resultado del proceso. Con la interfaz Web podría realizarse bien a través de los datos de la sesión y mediante el envío de un mensaje de correo con un informe completo del proceso.

- Establecer estrategias más complejas en las que por ejemplo:

    - El agente vendedor pueda establecer descuentos distintos para los productos.

    - El agente vendedor establezca un número máximo de ofertas/contraofertas tras las cuales si no se ha obtenido acuerdo finalice la negociación.

    - El agente vendedor no se limite sólo a esperar que los agentes compradores les pregunten por sus productos sino que realice ofertas de productos a los compradores.

    - El Mercado realice más labores para comunicar a los agentes entre ellos.

Anexo I

Manual del Administrador

Este manual está dirigido al usuario administrador encargado de la gestión del prototipo. Se especifican los requerimientos mínimos y los procesos de instalación de las herramientas y de la plataforma JADE. Se detallan las funcionalidades que puede realizar el administrador y que lo distinguen de resto, como es la creación, baja y gestión del agente Mercado.

I.I. Requerimientos mínimos e instalación

Para la utilización del prototipo desarrollado es necesario:

- *Una Máquina Virtual de Java.* Para instalarla no hay más que descargar la última versión desde la página de Sun: http://www.java.com .

- *La Plataforma JADE*. Cuyo software puede obtenerse en la página del grupo de investigación que creó este software: http://JADE.tilab.com/. El proceso de instalación se explica en el siguiente apartado.

I.I.I. Instalación del JRE de Java

Para realizar la instalación es necesario poseer conexión a Internet, acceder a la página http://www.java.com y seleccionar la descarga del JRE (entorno de ejecución de Java).

Al pinchar en el enlace aparece una nueva página Web donde se nos comunica que el sitio Web requiere el control de ActiveX : 'Java(TM) SE Runtime Environment 6 Update 2' de 'Sun Microsystems,Inc.'. Se indica que se haga clic sobre este mensaje para instalar apareciendo un pequeño menú despegable donde se elige la opción "Instalar Control ActiveX".

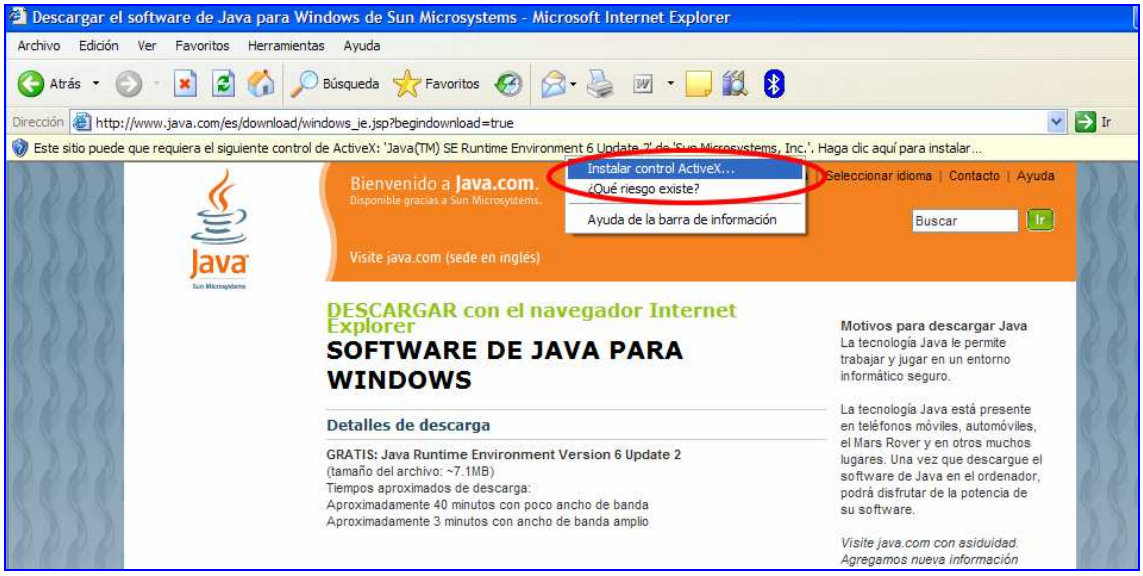


Figura I.1. Pantalla inicial de instalación del JRE.

Durante unos instantes detecta la configuración del JRE actual y determina cuál es el archivo a instalar. En este caso detecta que debe instalar una actualización de la máquina virtual. A continuación, se mostrará un mensaje donde se pregunta al usuario si desea instalar el producto. Hacemos clic en “Instalar”.

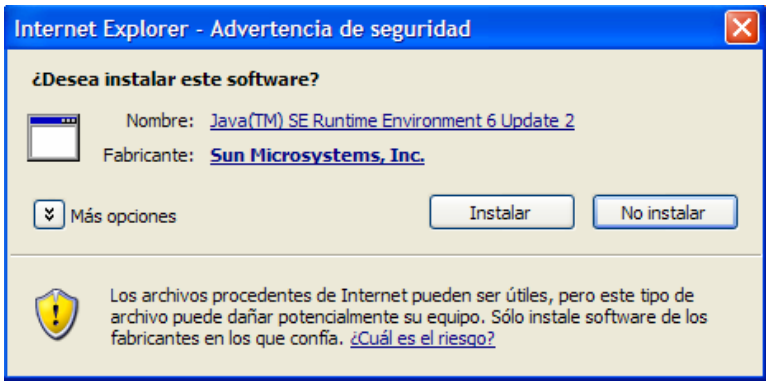


Figura I.2. Advertencia de Seguridad de la instalación del JRE.

Nos aparece la siguiente pantalla donde hay que aceptar el acuerdo de licencia y continuar con el proceso pulsando en “Aceptar>”.

Figura I.3. Pantalla del acuerdo de licencia del JRE.

En la siguiente pantalla Sun Microsystems ofrece la posibilidad de instalar programas gratuitos. Seguimos con el proceso pulsando el botón de “Siguiente>”:

Figura I.4. Pantalla de selección de otras aplicaciones.

En el siguiente paso se indica que se está realizando la instalación del producto de Java, mostrándose una barra de estado donde se indica la evolución del proceso de instalación.

Figura I.5. Pantalla del proceso de instalación del JRE.

Una vez finalizado el proceso de instalación se muestra una pantalla que informa que la instalación se ha realizado correctamente. Para terminar pulsar en el botón “Finalizar”.

Figura I.6. Pantalla de finalización de la instalación del JRE.

Se nos mostrará en la barra de "*Inicio*" el icono correspondiente a la máquina virtual, una pantalla que indica que están realizando actualizaciones y un bocadillo sobre el icono de la "taza de café" de Java:

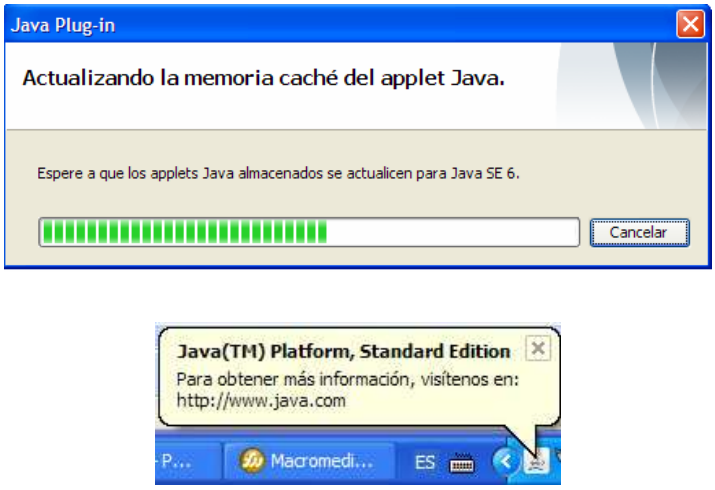


Figura I.7. Imágenes de la correcta instalación del JRE.

I.I.II. Instalación de la plataforma JADE

El software necesario está compuesto del archivo *JADE-all-3[1].4* que contiene la plataforma JADE y una correcta versión del *Java Run Time Environment*, a partir de la 1.4. A continuación, se detalla la instalación de la plataforma en los sistemas operativos Windows y Linux.

Instalación en Windows

El proceso de instalación de JADE es muy simple y consiste en la descompresión en C: del archivo "*JADE-all-3[1].4*" apareciendo cuatro ficheros comprimidos a su vez que contienen:

- *JADE-bin-3.4:* el código ya compilado y listo para ser interpretado.
- *JADE-doc-3.4*: la documentación javadoc, el manual del administrador, el del programador y un tutorial.
- *JADE-examples-3.4:* ejemplos de uso de la plataforma.
- *JADE-src-3.4:* el código fuente sin compilar.

Figura I.8. Archivos de JADE.

Al realizar la descompresión de los ficheros anteriores se crea una carpeta "*C:/JADE*" con lo cual finaliza el proceso de instalación. Su estructura de carpetas es la siguiente:

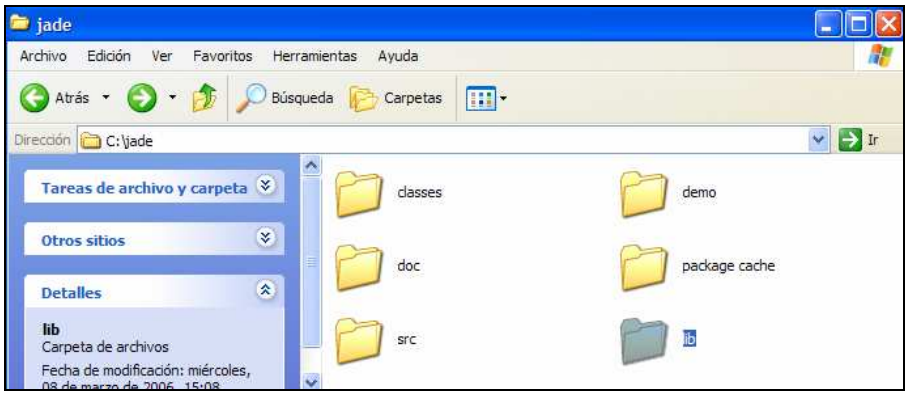


Figura I.9. Contenido de la carpeta "JADE".

Instalación en GNU/Linux

Los pasos para realizara la instalación de JADE en GNU/Linux son similares a los de la instalación en Windows.

1. Descargar JADE-all-3.4.1.zip de http://jade.tilab.com/ y descomprimir el archivo de JADE y los archivos comprimidos que contiene:

```
mkdir JADE
mv JADE-all-3.4.zip JADE/
cd JADE
unzip JADE-all-3.4.zip
unzip JADE-bin-3.4.zip
unzip JADE-doc-3.4.zip
unzip JADE-examples-3.4.zip
unzip JADE-src-3.4.zip
```

2. Crear los directorios /usb/lib/JADE y /usb/share; colocar en el primero las librerías y el resto de los contenidos en el segundo directorio.

```
sudo mkdir /usr/lib/JADE
sudo mv JADE/lib/* /usr/lib/JADE
sudo mkdir /usr/share/JADE
sudo mv JADE /usr/share/JADE
```

3. JADE ya está preparado para arrancar. Para facilitar su funcionamiento pueden crearse unas **variables de entorno** para JADE, para no tener que estar constantemente incluyendo todos los .jar en los classpath de java.
4. Para arrancarlo usar la siguiente sentencia:

```
java -cp $JADE_CP JADE.Boot –gui
```

I.II. Interfaz gráfica de JADE

En este apartado se especifica cómo lanzar JADE desde la distribución binaria y su funcionamiento desde la interfaz gráfica.

JADE, como ya se ha comentado, se distribuye de forma comprimida, en el árbol de directorios que se genera en JADE el root es "*JADE*" teniendo un subdirectorio "*lib*" el cual contiene algunos archivos *.jar* que tienen que tienen que ser añadidos a la variable CLASSPATH. Ejemplo:

SET
CLASSPATH=%CLASSPATH%;.;c./JADE/lib/Base64.jar;c:/JADE/lib/iiop.jar;c:/JADE/lib/JADE.jar

Una vez se ha establecido la CLASSPATH, el siguiente comando es usado para lanzar el contenedor principal de la plataforma:

Java JADE.Boot [options] [AgentSpecifier list]

El contenedor principal está compuesto por el agente DF, el agente AMS y el registro RMI (usado para la comunicación intra-plataforma). Para lanzar una agente contenedor:

Java JADE.Boot –container [options] [AgentSpecifier list]

Las *opciones* posibles son:

- **-container:** para especificar el contenedor principal.
- **-host:** para especificar donde el RMI Register debe ser creado, para el principal contenedor o para otros contenedores.
- **-port:** número del puerto RMI.
- **-name*:*** nombre de la plataforma.
- **-gui:** lanza la interfaz de JADE.
- **-mtp:** lista de protocolos de transporte de mensajes a activar.
- **-nomtp*:*** establecer por defecto el contenedor principal.
- **-aclcodec:** codificación ACL de los mensajes.
- **-nomobility:** para no permitir la migración o clonación de agentes.
- **-version*:*** versión de JADE.
- **-help:** ayuda de JADE.
- **-conf:** para elegir un fichero donde se encuentren los parámetros de configuración de JADE.

Los agentes pueden ser lanzados desde la línea de comandos mediante un string dividido en tres partes: *Nombreagente:nombre de la clase de java (lista de argumentos)*

En este proyecto se ha establecido primero la CLASSPATH teniendo en cuenta que es necesario añadir tanto los archivos .jar propios de JADE como el fichero .jar del proyecto. El paso siguiente es lanzar la interfaz de JADE y a través de ella ya pueden crearse el Mercado y los agentes.

```
Set CLASSPATH=c:/JADE/lib/JADE.jar;C:/JADE/proyecto/dist/proyecto.jar
Java JADE.Boot -gui
```

JADE posee varias herramientas para el desarrollo de las tareas multiagente. Cada herramienta está empaquetada en el propio agente, teniendo las mismas reglas, las mismas capacidades de comunicación y el mismo ciclo de vida de una aplicación general de un agente.

RMA (Remote Monitoring Agent)

Es el gestor que permite controlar el ciclo de vida de la plataforma y de los agentes registrados en ella. La arquitectura distribuida de JADE permite también control remoto, donde la GUI es usada para controlar la ejecución de los agentes y sus ciclos de vida desde un host remoto. Permite: iniciar, suspender, reiniciar agentes; matar agentes o contenedores; mandar mensajes; clonar agentes y añadir o quitar plataformas remotas.

Un RMA es un objeto de Java, instancia de la clase *JADE.Tools.rma.rma* y puede ser lanzado desde la línea de comandos como un agente normal: *java JADE.Boot –gui.*

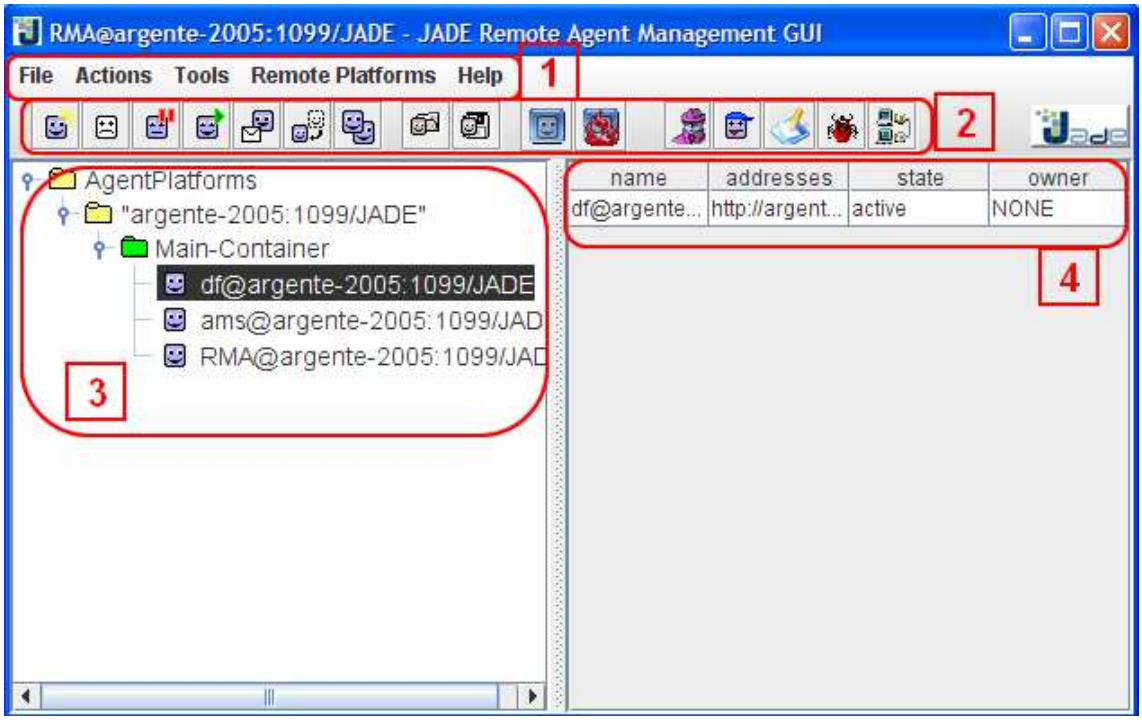


Figura I.10. Ventana principal de la interfaz de JADE.

Como puede observarse en la figura anterior la Interfaz de JADE está formada por cuatro partes numeradas en la imagen:

1. **Menú de Opciones** en la parte superior.

2. **Accesos rápidos** a las opciones más usadas de los menús situados debajo de los menús.

3. **Panel lateral izquierdo** donde aparecen los contenedores y los agentes.

4. **Panel lateral derecho** donde aparece información acerca del agente seleccionado: su nombre, su dirección, el estado en el que se encuentra y su dueño.

A continuación se detallan las opciones del menú superior.

Menú File

- **Close RMA Agent**: cierra la interfaz, aunque la plataforma de agentes continuará funcionando, así como todos aquellos agentes que estuvieran activos.

- **Exit this container**: Destruye el container donde el agente RMA se aloja, matando a este agente y todos aquellos que le acompañaran dentro del container. Si dicho container es el MainContainer será toda la plataforma la que se desactivará.

- **Shut down Agent Platform**: cierra totalmente la plataforma, eliminando todos los agentes y containers.

Menú "Actions"

Este menú contiene los ítems para invocar todas las acciones necesarias para los agentes o contenedores:

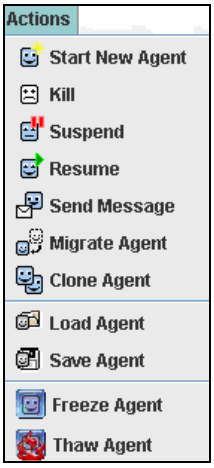


Figura I.11. Menú" Actions" de JADE.

- **Start New Agent.** Crea un nuevo agente pidiendo el nombre del agente y la clase de Java a instanciar pudiendo especificar el contenedor. Ejemplo:
- **Kill Selected Ítems.** Para "matar" a los agentes seleccionados. Si el MainContainer estuviera seleccionado, la plataforma entera se cerraría.
- **Suspend Select Agents.** Esta acción suspende a los agentes seleccionados.
- **Resume Selected Agents.** Vuelve a poner en estado de activo los agentes seleccionados que estuvieran suspendidos.
- **Send Message:** este comando permite enviar un mensaje ACL a un agente.
- **Send Custom Message to Selected Agents.** Esta acción permite enviar un mensaje ACL a un agente.
- **Migrate Agent.** Esta acción permite la migración de un agente, aunque no todos los agentes pueden ser migrados debido a su propia configuración.
- **Clone Agent.** Permite la clonación de un agente seleccionado.

Menú de Herramientas

Menú con herramientas para los programadores.

Figura I.12. Menú "Tools" de JADE.

Sniffer agent

Es una agente que muestra las interacciones que se producen. Permite controlar la comunicación entre los agentes y observar el envío de mensajes y su contenido.

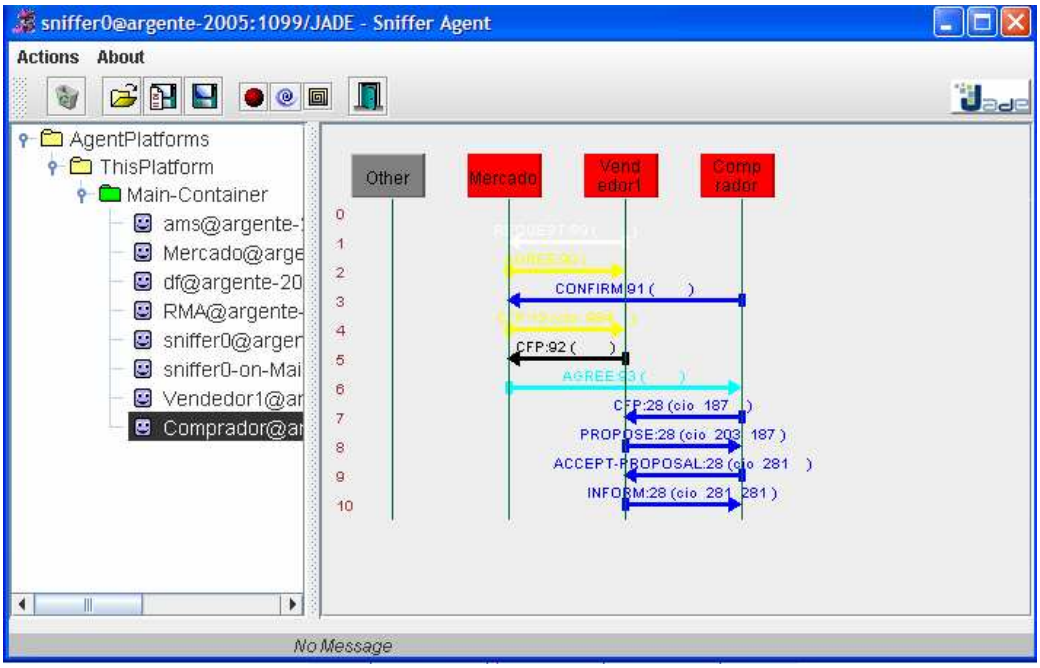


Figura I.13. Sniffer agent.

.

A la izquierda se muestra el árbol de plataformas, contenedores y agentes de la plataforma y en la parte derecha se puede observar el intercambio de mensajes entre los agentes que seleccionemos.

Los mensajes enviados y recibidos aparecen en orden cronológico comenzando desde la parte superior. Para analizar los distintos campos de un mensaje no hay más que hacer doble-click sobre el mensaje que se desee consultar.

Dummy agent

Esta herramienta permite la depuración del sistema multi-agente. Es un agente que envía mensajes para lo cual tiene asociada una interfaz gráfica que permite indicar los distintos campos del mensaje a enviar, así como visualizar los mensajes enviados y recibidos.

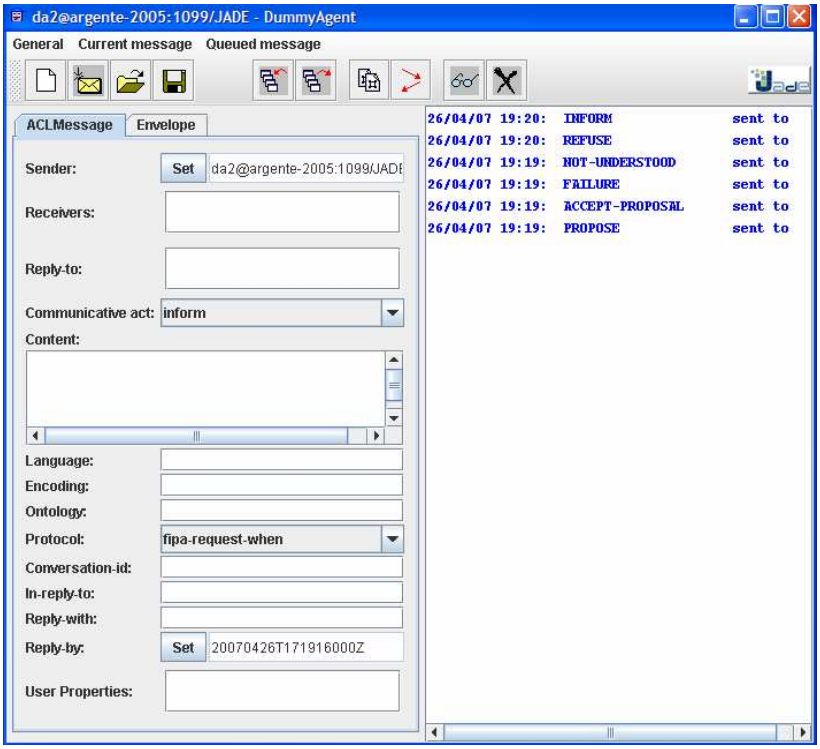


Figura I.14. Agente Dummy.

Como puede observarse en la figura siguiente la parte principal de la ventana está dividida en dos. En la izquierda disponemos de una serie de campos donde podemos introducir el valor de los campos del mensaje a enviar (emisor, receptores, contenido,…).

DF GUI

Es un interfaz del Directory *Facilitator*. Permite ver los agentes registrados con DF y controlar las operaciones básicas sobre los DF: ver, registrar, desregistrar, modificar y buscar.

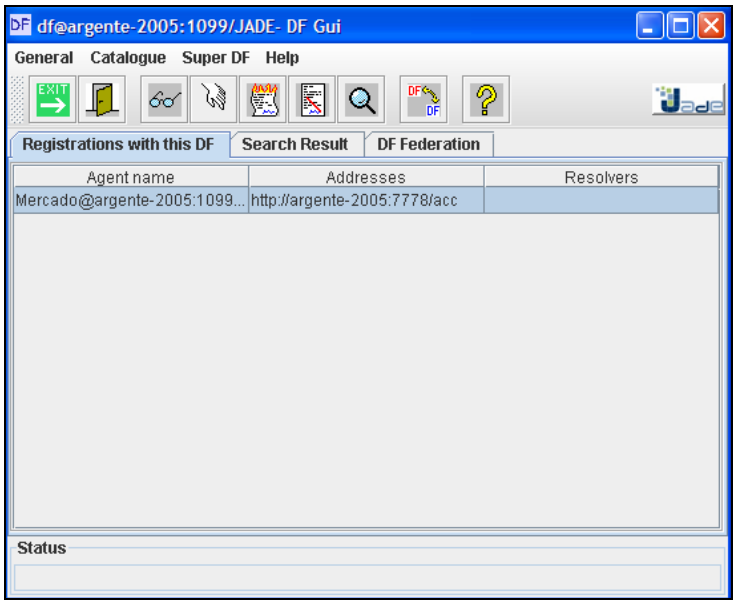


Figura I.15. DF Gui.

El Directory Facilitator a través de su interfaz nos permite visualizar los servicios que se han dado de alta en las páginas amarillas y qué agentes los proporcionan (Registrations with this DF) y buscar un servicio en concreto (Search Result), entre otras opciones.

Introspector agent

Permite monitorizar y controlar el ciclo de vida de una gente en funcionamiento y sus mensajes enviados y recibidos.

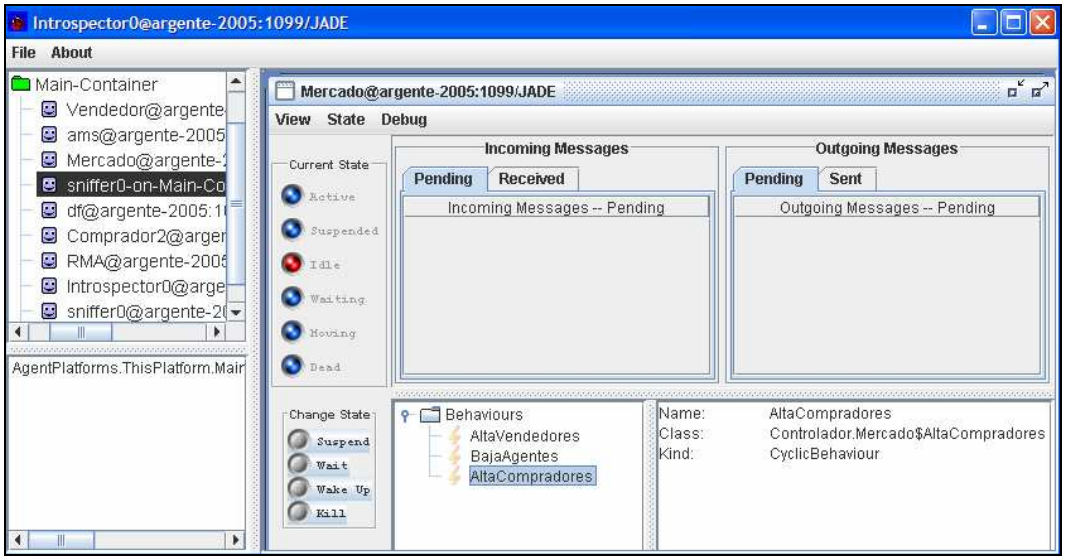


Figura I.16. Agente “Introspector” de JADE.

En su interfaz se puede observar todo lo relacionado con un agente, en este caso se ha seleccionado al agente Mercado. Puede comprobarse en qué estado se encuentra en este momento (Current State); los mensajes que ha recibido (Incoming Messages); los mensajes que ha enviado (Outgoing Messages); los comportamientos que posee y algunas características de ellos.

Remote Platforms Menú

Este menú permite controlar plataformas remotas que cumplan con las especificaciones FIPA; no teniendo por qué ser de JADE.

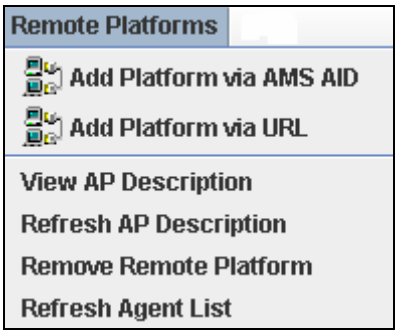


Figura I.17. Menú "Remote Platforms" de JADE.

- **Add Remote Platform Via AMS AID** y **Add Remote Platform Via URL**, para añadir plataformas remotas mediante AMS y URL.
- **View APDescription.** Para visualizar la descripción AP de una plataforma.
- **Refresh APDescription.** Esta acción pregunta al AMS remoto por la Descripción AP y refresca la antigua.
- **Remove Remote Platform.** Esta acción permite eliminar del GUI la plataforma remota seleccionada.
- **Refresh Agent List**. Esta acción permite una búsqueda con el AMS de la plataforma remota y la lista completa de agentes de la plataforma remota.

I.III. Funcionalidades del administrador

Las funcionalidades que posee el usuario administrador son el lanzamiento de la plataforma y la creación y control del agente Mercado.

Para lanzar la plataforma el usuario puede hacerlo desde el símbolo del sistema situándose dentro de la carpeta jade:

```
C:\jade\Proyecto\dist>set classpath=c:/jade/lib/jade.jar;C:/jade/proyecto/dist/p
royecto.jar

C:\jade\Proyecto\dist>java jade.Boot -gui
```

Figura I.18. Lanzamiento de la plataforma JADE.

Aparece la interfaz de JADE a través de la cual el usuario puede realizar diferentes acciones.

Creación del Mercado

Es el primer agente que debe lanzarse sin excepción ya que sin él no pueden comunicarse el resto de agentes. La creación del agente puede realizarse de varias maneras:

1. Seleccionando del menú *Actions* la opción ”Start New Agent”.

2. Seleccionando el acceso rápido: 
3. Con el botón derecho del ratón haciendo clic sobre el Main-Container nos aparece un menú en el cual se elige la opción “Start New Agent”.

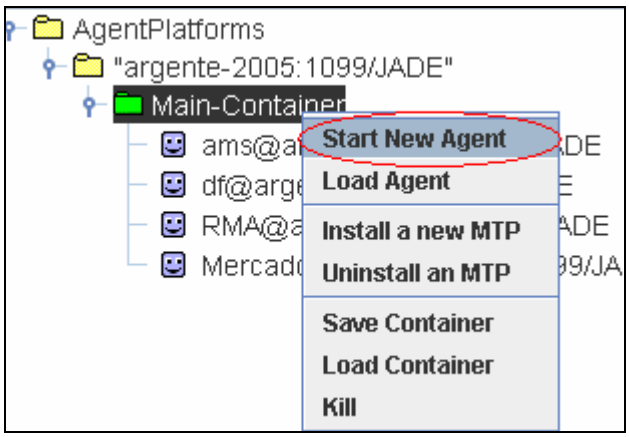


Figura I.19. Opción “Start New Agent”.

Aparece una ventana “Insert Start Parameters” donde es necesario introducir el nombre del agente, la clase a la que está asociada en el proyecto y en este caso nada más ya que el Mercado no necesita ningún parámetro de creación. En este caso se indica la clase y el paquete donde está contenido en el proyecto.

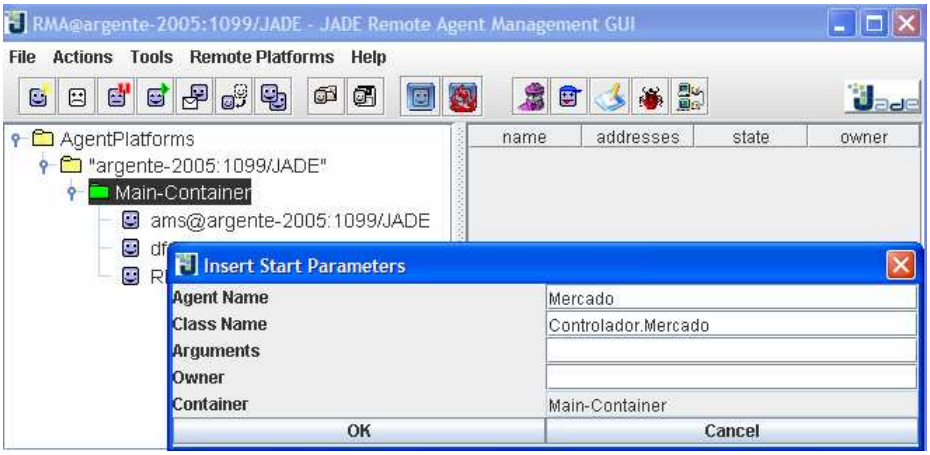


Figura I.20. Creación del Mercado en la plataforma JADE.

Una vez introducidos correctamente los valores se pulsa el botón “OK” y se crea el agente en el contenedor apareciendo su interfaz, cuyo contenido fue descrito en el apartado de Diseño.

Figura I.21. Interfaz JADE del Agente Mercado.

Dar de baja al agente Mercado

El administrador es el que puede dar de baja al agente Mercado no pudiendo el resto de los usuarios crear agentes compradores y vendedores, ya que estos necesitan del agente Mercado creado para poder comunicarse entre ellos.

Para dar de baja al agente puede seleccionarse la opción “Kill” del menú “Actions” o simplemente cerrando su interfaz mediante el aspa que aparece en la esquina superior derecha.

Consultar el resultado del proceso de negociación

El administrador mediante la *herramienta sniffer* puede consultar todo el proceso y los mensajes enviados durante el mismo.

La herramienta Sniffer permite, como se comentó anteriormente, observar los mensajes enviados en el proceso de negociación entre los agentes que han intervenido. En este caso no es tan obvio determinar si se ha llegado a un acuerdo o no. En las interfaces de los agentes el resultado se muestra de una forma explícita con un mensaje de texto, en el sniffer es necesario conocer el funcionamiento interno del prototipo para saber qué tipo de mensaje especifica que se ha llegado a un acuerdo.

A continuación se muestra un ejemplo en el cual se puede observar el intercambio de mensajes entre los agentes.

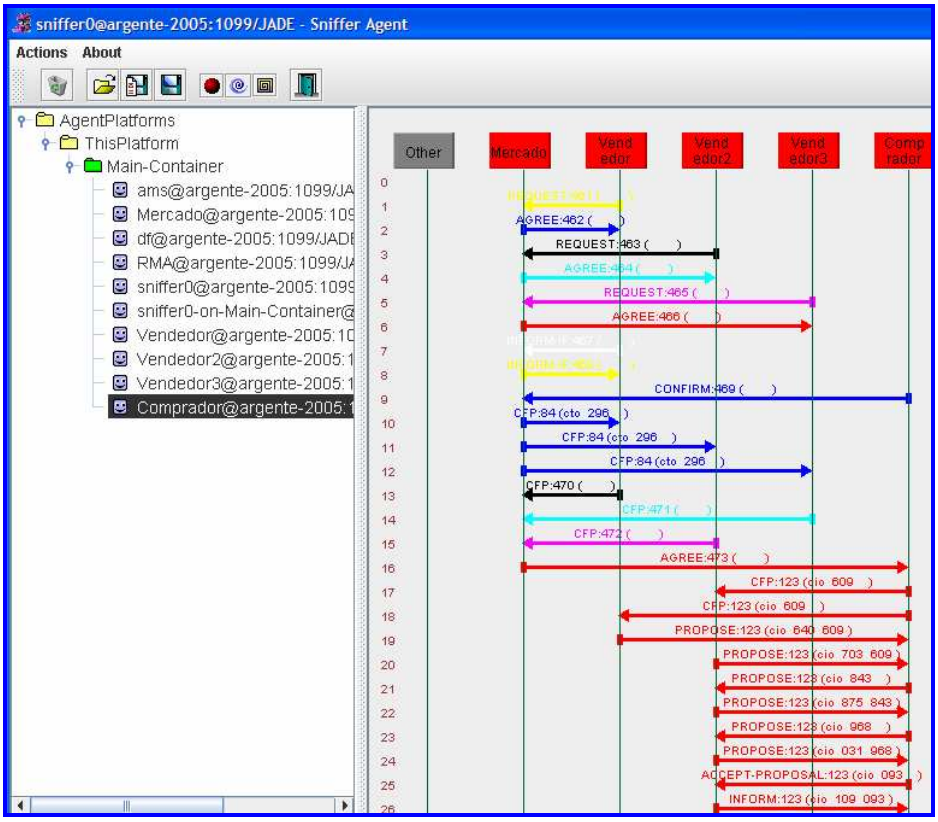


Figura I.22. Control del proceso de negociación a través del sniffer.

Entre el comprador y el vendedor2 existe un intercambio de mensajes continuo de tipo PROPOSE lo cual indica una negociación. Cuando finalizan estos mensajes el agente comprador envía un mensaje de tipo ACCEPT_PROPOSAL, que consultando en el capítulo de Análisis y Diseño, se comprueba que determina que se ha llegado a un acuerdo. Seguidamente el vendedor2 envía un mensaje al comprador de tipo INFORM confirmando la venta.

Anexo II

Manual de usuario

Este manual está destinado a los usuarios compradores y vendedores que desean hacer uso de la plataforma de agentes para comprar y/o vender productos. Se detalla cómo el usuario debe realizar las acciones que le están permitidas.

Creación de un Agente Vendedor

La creación del agente vendedor puede iniciarse de varias formas:

1. Seleccionando del menú *Actions* la opción ”Start New Agent”.
2. Seleccionando el acceso rápido: 
3. Con el botón derecho del ratón haciendo clic sobre el Main-Container nos aparece un menú en el cual se elige la opción “Start New Agent”.

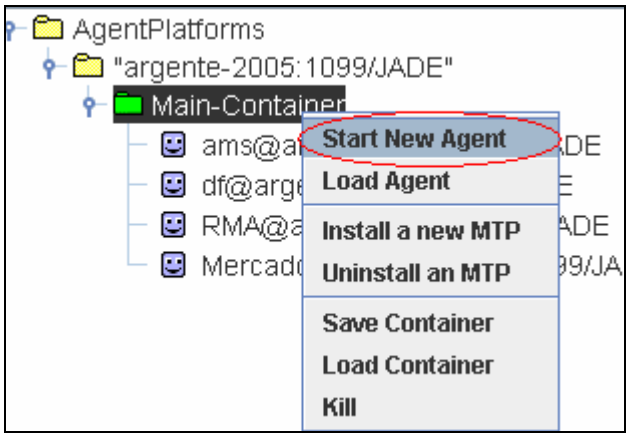


Figura II.1. Opción “Start New Agent”.

Aparece una ventana “Insert Start Parameters” donde es necesario introducir el nombre del agente, la clase a la que está asociada en el proyecto y como argumentos: el valor del tanto por ciento de decremento que irá realizando en cada oferta y el tanto por ciento máximo de descuento que hará sobre los productos de su catálogo. Se han de indicar directamente con el número sin especificar el símbolo “%”.

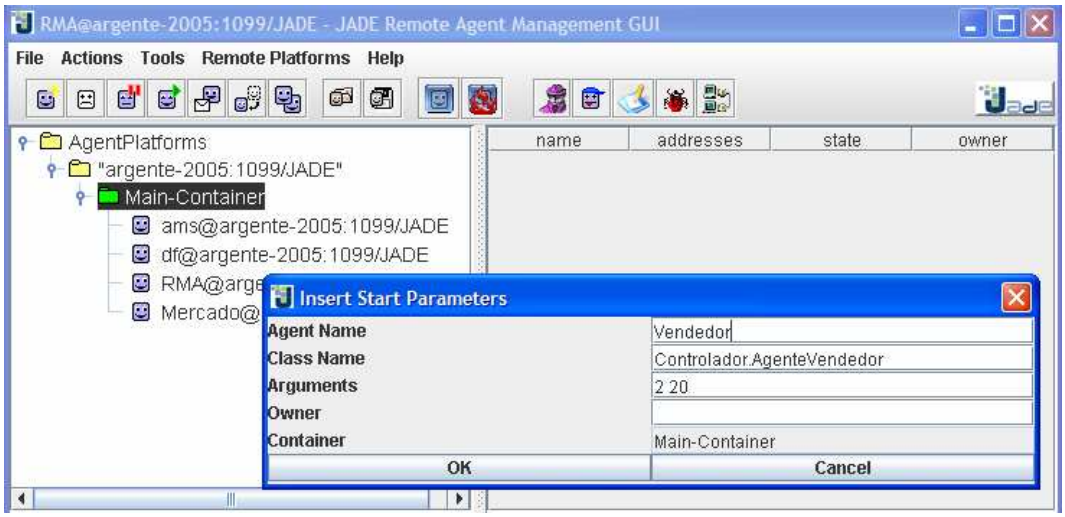


Figura II.2. Creación de un Agente Vendedor en la plataforma JADE.

Si no se han introducido todos los argumentos la plataforma muestra un mensaje de error indicando que no es posible crear el agente y que si desea ver el mensaje ACL que se ha creado en consecuencia. Consultando el contenido de este mensaje se puede detectar el error que se ha cometido.

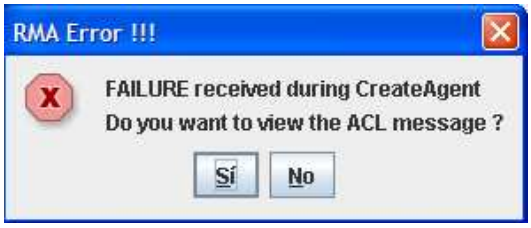


Figura II.3. Mensaje de error en la creación de agentes.

Si se han introducido correctamente los datos, tras pulsar el botón “OK” se crea el agente vendedor y aparece su interfaz.

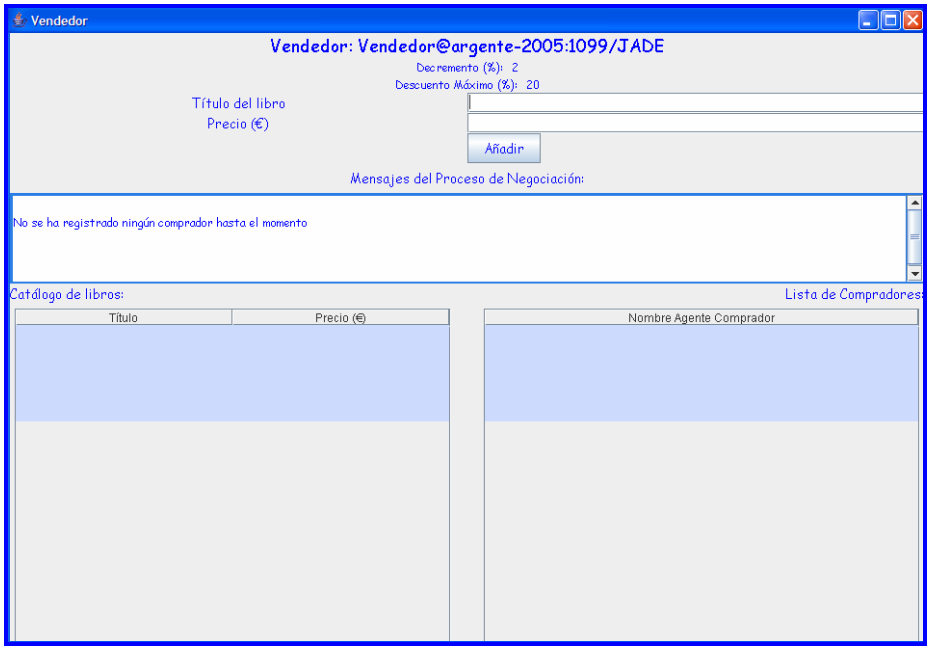


Figura II.4. Interfaz JADE del agente vendedor.

Se puede comprobar en la interfaz del Mercado si se ha dado de alta correctamente el vendedor. Para ello se observa el panel de Mensajes del Mercado y se comprueba la lista de los vendedores.

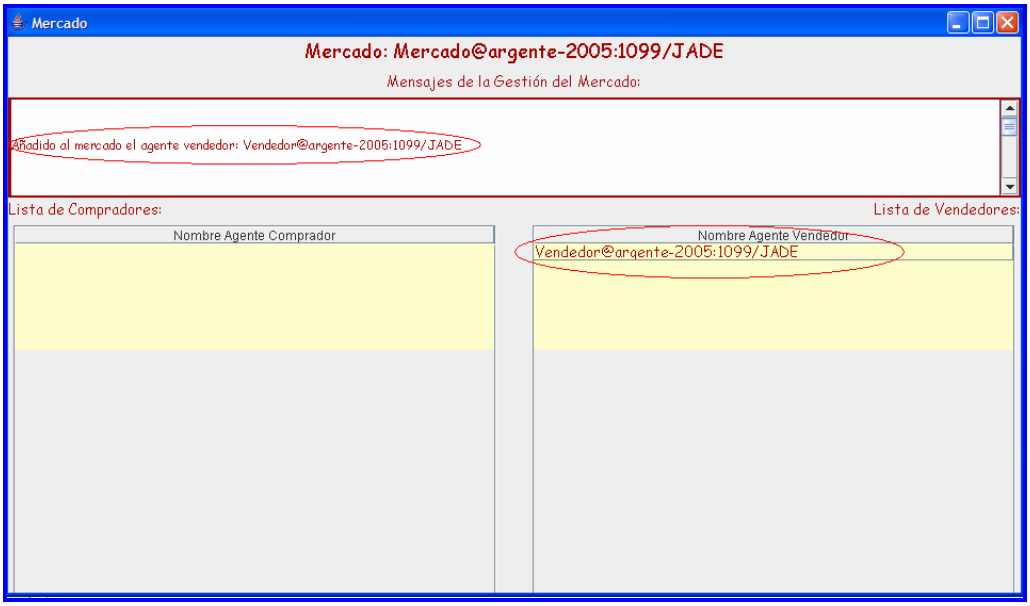


Figura II.5. El Vendedor se ha dado de alta en el Mercado.

Añadir libros al catálogo del vendedor

Para realizar esta acción se rellenarán los dos campos del formulario superior de la interfaz del vendedor y posteriormente se pulsará el botón "Añadir". Si el proceso se realiza correctamente el libro se añade al catálogo.

Figura II.6. Inserción de libros en el catálogo.

Si no se han especificado correctamente los parámetros o el libro ya está en el catálogo se muestra un mensaje de error para advertir al usuario.

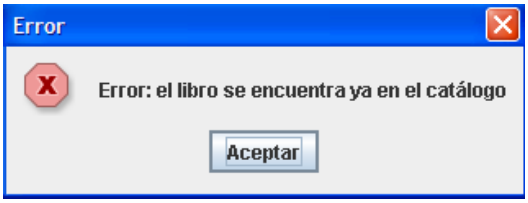


Figura II.7. Mensajes de error de la modificación del catálogo.

La tabla de libros irá mostrando las nuevas incorporaciones:

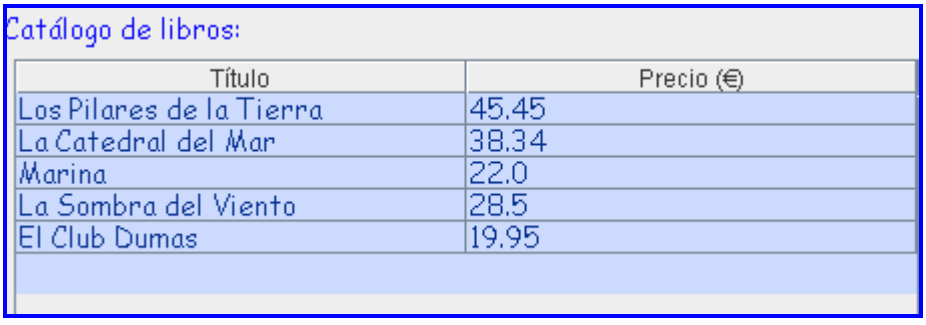


Figura II.8. Catálogo de Libros de la Interfaz del Vendedor.

Creación de un Agente Comprador

Para la creación del agente comprador se sigue los mismos pasos que para la creación del agente vendedor. Este agente tiene como argumentos de entrada: el título del libro que desea comprar, el precio deseado que le gustaría pagar, el tanto por ciento en que irá incrementando su oferta y el precio máximo que está dispuesto a pagar por el producto. En la *figura II.9* se muestra un ejemplo.

Figura II.9. Creación de un Agente Vendedor en la plataforma JADE.

Si el nombre del agente no coincide con ninguno de los agentes dados de alta hasta el momento y los parámetros introducidos son correctos se crea el agente y aparece su interfaz. En caso contrario se mostrarán mensajes de error similares a los del agente vendedor.

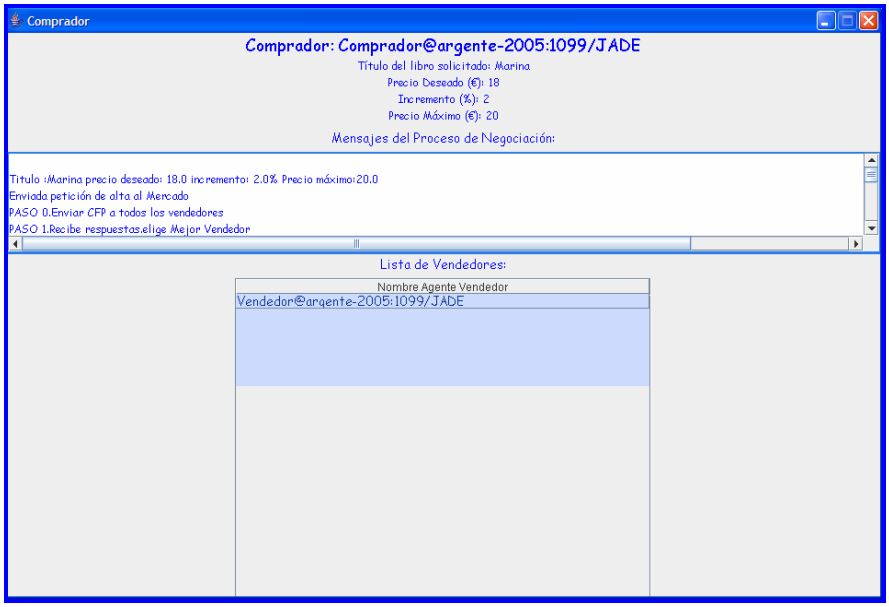


Figura II.10. Interfaz JADE del agente comprador.

Se puede comprobar en la interfaz del Mercado si se ha dado de alta correctamente el comprador. Para ello se observa el panel de Mensajes del Mercado y se comprueba la lista de los compradores.

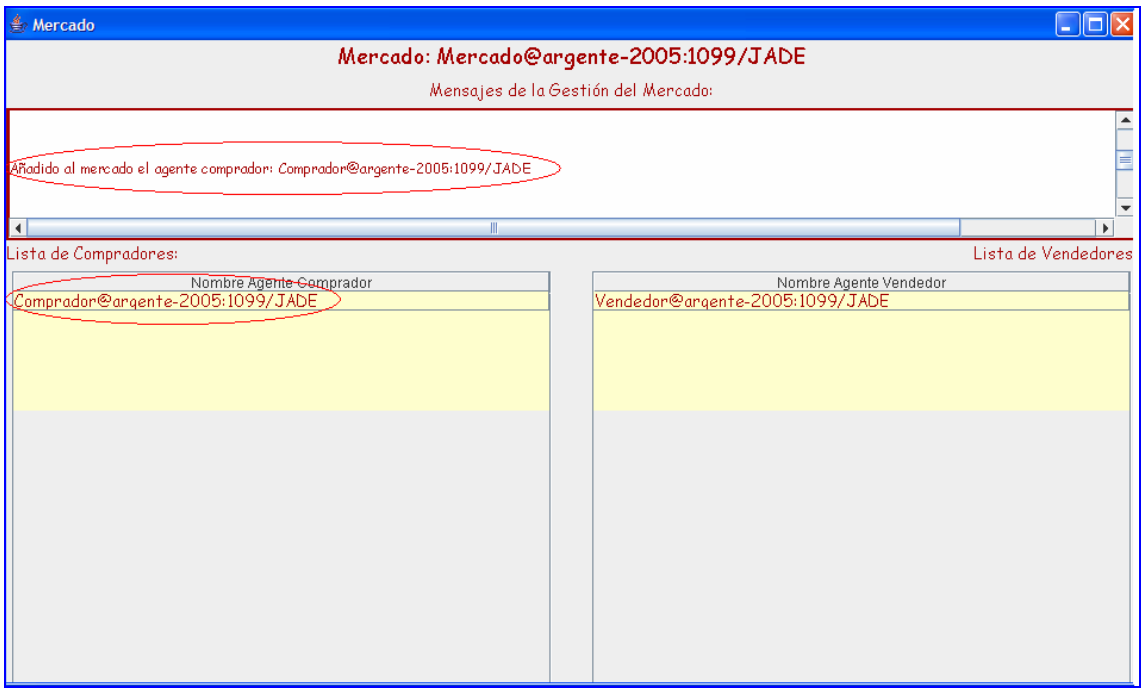


Figura II.11. El comprador se ha dado de alta en el Mercado.

Dar de baja a los agentes

El proceso de dar de baja a los agentes es el mismo en todos los casos y consiste en cerrar la ventana del agente desde el aspa o seleccionar la opción ”Kill” del menú “Actions”. En el caso de ser un agente comprador o vendedor al eliminarlo de la plataforma se da de baja en el Mercado con lo cual pierden las relaciones que habían establecido con los agentes.

- **Agente vendedor.** Cuando se cierra la ventana del vendedor con el aspa el agente es eliminado de la plataforma y desaparece de la lista del Mercado. A continuación, se muestra un ejemplo de cómo se indica en la interfaz del Mercado que el agente vendedor se ha dado de baja y cómo se modifica la lista.

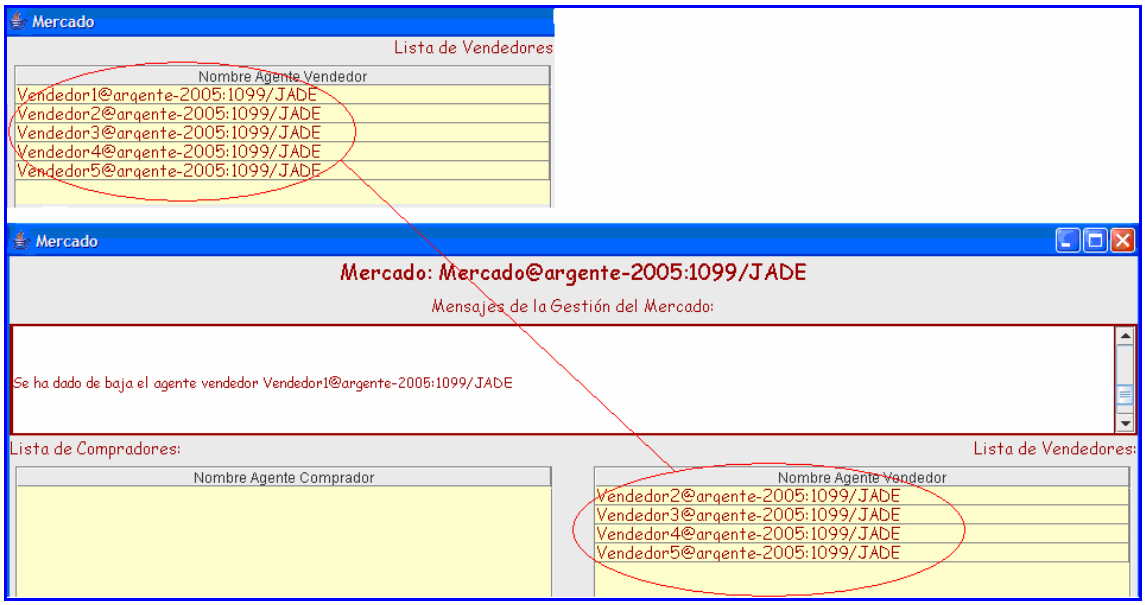


Figura II.12. Agente vendedor dado de baja.

- **Agente comprador.** Cuando se cierra la ventana del comprador con el aspa el agente es eliminado de la plataforma y desaparece de la lista del Mercado. de los compradores. El funcionamiento es similar al de los agentes vendedores.

Consultar el resultado de los procesos de negociación

Una vez lanzados los agentes con los parámetros correctamente definidos el usuario visualiza cómo durante unos instantes en la plataforma de producen los procesos de negociación oportunos. La consulta del resultado podrá realizarse en:

- En la *interfaz del agente vendedor*.

En la zona de "*Mensajes del Proceso de Negociación*" se puede consultar el resultado del proceso de negociación. Se mostrará un mensaje similar al de la figura en el que se indique el producto que se ha vendido, a qué precio y a qué agente.

Figura II.13. Control proceso de negociación en la interfaz del vendedor.

- En la *interfaz del agente comprador*.

También puede consultarse el resultado en la propia interfaz del agente que desea adquirir el producto en la zona dedicada a los mensajes del proceso de negociación. El mensaje mostrado indicará si se ha comprado el artículo deseado, a qué precio y a qué vendedor en concreto. Un ejemplo se muestra en la figura siguiente:

Figura II.14. Control proceso de negociación en la interfaz del comprador.

- En la *interfaz del Mercado.*

El mercado mostrará en el panel de Mensajes aquellos procesos de negociación en los que se haya alcanzado un acuerdo indicando entre qué agentes se ha producido, con qué artículo y a que precio.

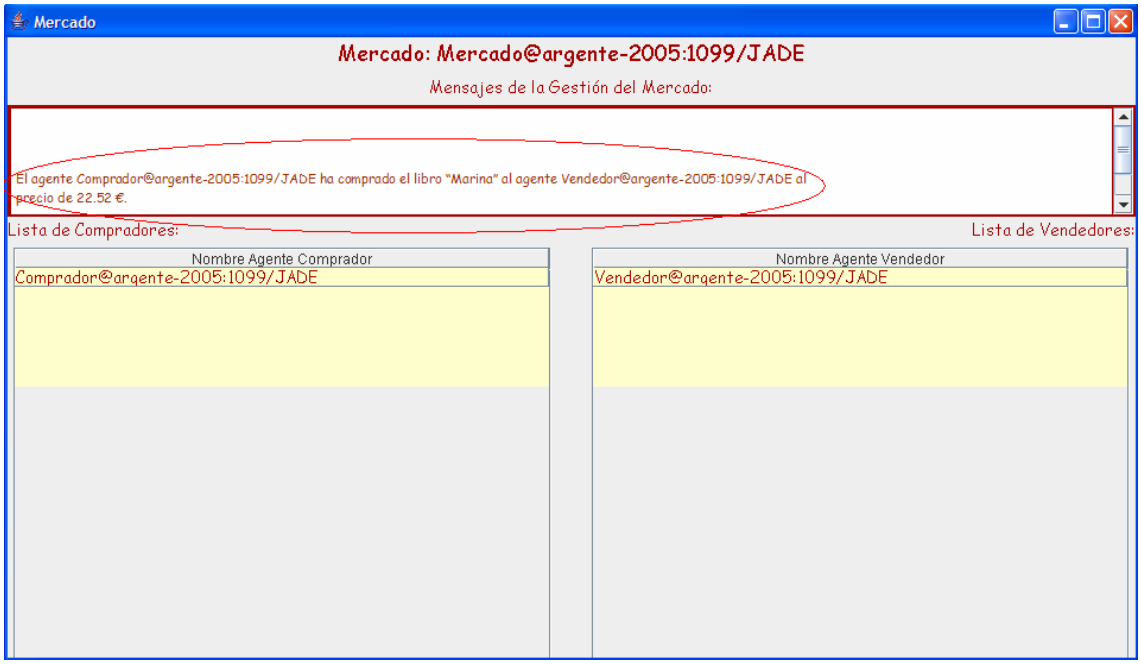


Figura II.15. Control proceso de negociación en la interfaz del Mercado.

Anexo III

Contenidos digitales del CD-ROM

- Archivos de instalación de la máquina virtual de Java.
- Archivos de instalación de la plataforma JADE.
- Código del prototipo.
- Documentación Javadoc del código.
- Memoria del Proyecto Fin de Carrera en formato PDF.

Glosario de términos

ACC (*Agent Communication Channel*). Es el componente software que controla todos los intercambios de mensajes dentro de la plataforma, incluidos los mensajes de plataformas remotas. Realiza una comunicación asíncrona.

ACL *(Agent Communication Language).* Lenguaje de comunicación de agentes. Es una especificación del la plataforma JADE para intercambiar mensajes entre agentes del sistema.

AMS (*Agent Management System).* es el agente que realiza el proceso de supervisión para acceder y usar la Plataforma de Agentes.

ANS (*Agent Name Service*). Servicio de nombres de agentes proporcionado por el AMS de JADE.

Agente. Sistema informático, situado en algún entorno (por ejemplo una plataforma), dentro del cual actúa de forma autónoma y flexible para así cumplir sus objetivos.

E-Commerce (*Comercio electrónico*). Intercambio comercial de valor (productos, servicios e información) en el que algunas o todas las fases se desarrollan mediante Internet.

DF (*Directory Facilitator).* Es una agente que provee el servicio de páginas amarillas en la plataforma.

FIPA (*Foundation for Intelligent Physical Agents*). Es una organización de estándares de la sociedad de los computadores de IEEE que promueve las tecnologías basabas en agentes y la interoperabilidad de sus estándares con otras tecnologías.

GUID (*Globally Unique Identifier*). Identificador único asignado a un agente cuando comienza su ejecución en JADE. Es proporcionado por el servicio de nombres de FIPA.

HTTP (*HyperText Transfer Protocol*). Protocolo de transferencia de hipertexto. Es un protocolo cliente-servidor por medio del cual se intercambian documentos HTML y sus componentes entre los servidores y los clientes que usualmente son navegadores Web.

JADE (*Java Agent Development Framework*). Es un software para el desarrollo de sistemas multiagente siguiendo los estándares FIPA.

JDK (*Java Development Kit*). Kit de desarrollo de Java, conjunto de herramientas para hacer programas basados en Java. Incluye una máquina virtual como entorno de ejecución de Java.

JRE (*Java Runtime Environment***).** Entorno de Ejecución de Java.

JVM (*Java Virtual Machine*). Máquina virtual de Java; es el nombre dado al intérprete de programas Java.

Licencia LGPL (*Lesser General Public Licence*) .Es un tipo de licencia GNU que pretende garantizar la libertad de compartir y cambiar el software libre. Los usuarios reciben unos derechos y obligaciones como, por ejemplo, libertad para distribuir copias de software libre.

RMA (*Remote Management Agent*). Es el gestor de la plataforma JADE.

SMA (*Sistema Multiagente***).** Es un sistema software compuesto por múltiples agentes que se comunican entre sí y en algunas ocasiones se mueven entre diferentes plataformas- agentes móviles.

Ontología. Simplificación de la lógica de algún dominio de conocimiento.

UML (*Unified Modelling Language*). Lenguaje Unificado del Modelo. Es una notación gráfica que permite modelar un sistema de software y ver los aspectos que se deben considerar en el análisis y diseño del mismo.

WSDL (*Web Service Definition Language*). Es un formato de XML creado por el W3C para describir servicios Web. Describe los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Se usa a menudo en combinación con SOAP y XML Schema, de manera que un programa cliente que se conecta a un servicio Web puede leer el WSDL para determinar qué funciones están disponibles en el servidor.

W3C (*World Wide Web Consortium***).** Es una organización sin ánimo de lucro dedicada a desarrollar normalizadamente las tecnologías relacionadas con Internet, en especial con la publicación de contenidos a través de la World Wide Web.

Bibliografía

Agentes, Sistemas Multiagente y JADE

[1] Fabio Belllifemine, Giovanni Caire, Tiziana Trucco, Giovanni Rimaza. *JADE Administrator's Guide, 2*002

[2] Fabio Belllifemine, Giovanni Caire, Tiziana Trucco, Giovanni Rimaza. JADE Programmer's Guide, 2002.

[3] Giovanni Caire. *JADE Tutorial. Application- Defined Content Languages and Ontologies*, 2002.

[4] Antonio F. Gómez Skarmeta y Juan A.Botía Blaya. *Tecnologías y Plataformas de Agentes*, 2002.

[5] Carlos Ángel Iglesias Fernández. *Definición de una Metodología para el Desarrollo de Sistemas Multiagente*, 1998.

[6] M. Wooldridge and N.R. Jennings. *Intelligent agents: Theory and practice. The Knowledge Engineering Review*, 1995.

[7] Juan Manuel Corchado (USAL). *Modelos y Arquitecturas de Agente*.

[8] Anthony Chavez y Pattie Maes. *Kasbah: An agent Marketplace for Buying and Selling Goods.*

[9] Carles Sierra, *Evolución de agentes inteligentes*, 2001.

[10] José M. Molina López, Jesús García Herrero y Ana Mª Bernardos Barbolla, *Agentes y Sistemas Multiagente*, 2004

[11] Universidad Tecnológica Metropolitana. Facultad de Informática. Escuela de Informática, *Informe Final de Agentes de Comercio*.

[12] Miguel Ángel López Carmona. *Estrategias de negociación automática basadas en restricciones difusas sobre Sistemas Multiagente*, 2006.

[13] Juan A. Botía Blaya. *Tutorial básico de JADE*. Escuela de Primavera de Agentes, Universidad de Sevilla, 2005.

[14] Ana Mas. *Agentes Software y Sistemas Multi-Agente: Conceptos, arquitecturas y aplicaciones.* 2005

Conexión Plataforma- Web

[15] Daniel Le Berre, Olivier Fourdrinoy. *Using JADE with Java Server Pages* .

[16] *XML, Servicios Web y Web Semántica*. Departamento de Informática, Universidad de Oviedo.

[17] W3C: http://www.w3c.com

[18] *JADE Web Services Integration Gateway (WSIG) Guide*, JADE Board, Junio 2007.

Otros

[19] Lina García Cabrera. *Apuntes de Sistemas Hipermedia,* curso 2005/2006.

[20] Francisco Conde de Asís. *Apuntes de Interacción-Persona-Ordenador,* curso 2005/2006.

[21] Pressman. *Ingeniería del Software*. McGraw Hill.

Estándar Fipa

[22] http://www.fipa.org

Licencia LGPL

[23] http://www.gnu.org

Java

[24] http://www.java.com