UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC
DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia

Mercado virtual genérico basado en la tecnología de Sistemas Multi-Agente

Trabajo Fin de Máster

Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital (MIARFID)

Autor: María Gabriela Juárez Juárez

Dirigido por: Adriana Giret Boggino

Vicente Botti Navarro

Valencia 2018

Agradecimientos

En primer lugar, agradezco a Dios Todopoderoso por permitirme finalizar con éxito una de las etapas más importantes de mi vida académica.

A mi madre por su infinito amor, cariño y apoyo, a mi padre que sé que desde donde se encuentra siempre está conmigo.

A mis hermanos por darme siempre su amor y cariño incondicional.

A mis amigos por apoyarme y hacerme sonreír en los momentos difíciles.

A mi directora de TFM Adriana Giret por sus conocimientos, su tiempo, sus orientaciones, su paciencia y motivación, ya que han sido fundamentales para la finalización de este trabajo.

Al Grupo de Tecnología Informática- Inteligencia Artificial por la confianza depositada en mi por y por la oportunidad brindada.

Y a todas las personas que con su amor, cariño, compresión, conocimiento y apoyo colaboraron de alguna manera para ayudarme en la finalización de este trabajo.

Resumen

El presente trabajo se centra en la utilización de la tecnología de sistemas multiagentes, específicamente utilizando plataformas virtuales, mercados virtuales, protocolos y estrategias de negociación, para la creación de un mercado virtual genérico que funciona mediante, agentes inteligentes que representan a un usuario común y cuyo comportamiento está basado en las preferencias del usuario, el funcionamiento del agente depende de los protocolos y estrategias de negociación asignados al mismo por el usuario que representa.

Índice de Contenido

1. Introducción 1
1.1. Introducción 1
1.2. Motivación 2
1.3. Objetivos 3
1.3.1. Objetivo General 3
1.3.2. Objetivos Específicos 3
1.4. Estructura del trabajo 4
2. Estado del Arte 5
2.1. Mercados, Comercio Electrónico y Plataformas Virtuales. 5
2.1.1. Mercado 5
2.1.2. Comercio Electrónico (E-Commerce) 6
2.1.3. Mercados virtuales o Marketplaces 6
2.1.4. Plataformas de mercados virtuales 7
2.2. Agentes inteligentes, Sistemas Multi-agentes y Negociación 8
2.2.1. Agente Inteligente 8
2.2.2. Sistemas Multiagentes 9
2.2.3. Negociación 10
2.2.4. Protocolos de negociación 12
2.3. Comparación de plataformas de mercados virtuales existentes 13
2.4. Trabajos relacionados 14
3. Diseño metodológico 18
3.1. Materiales utilizados 18
3.1.1. Materiales Hardware 18
3.1.2. Materiales Software 18

3.2. Metodología de Desarrollo Software 19
3.3. Etapas del proyecto de investigación 19
3.3.1. Etapa I: Formulación del problema 19
3.3.2. Etapa II: Estudio del Estado del Arte 19
3.3.3. Etapa III: Integración de Tecnologías 20
3.3.4. Etapa IV: Prueba y Depuración 20
3.3.5. Etapa V: Conclusiones y Trabajos Futuros 20
4. Desarrollo 21
4.1. Elementos del mercado virtual 21
4.1.1. Plataforma basada en Agentes 22
4.1.2. Fichero properties 27
4.1.3. Tareas de inicialización 30
4.1.4. Agentes vendedores 31
4.1.5. Agentes Compradores 36
4.1.6. Módulo Gestor 49
5. Experimentación y Resultados 58
5.1. Experimentación 58
5.2. Resultados 58
6. Conclusiones y Trabajos Futuros 60
6.1. Conclusiones 60
6.2. Trabajos Futuros 61
7. Bibliografía 62

Índice de Ilustraciones

Ilustración 1. Etapas del proyecto de investigación. 19
Ilustración 2. Elementos principales del mercado virtual..... 22
Ilustración 3. Logotipo y esquema de funcionalidades de MAGENTIX2. 23
Ilustración 4. Intercambio de Mensajes a través del QPID Broker 24
Ilustración 5. Gráfica del CProcessor, protocolo REQUEST..... 26
Ilustración 6. Clases contenidas CAgents 26
Ilustración 7. Estructura Fichero properties. 27
Ilustración 8. Lectura del fichero properties y acceso a los parámetros..... 27
Ilustración 9. Datos iniciales 27
Ilustración 10. Ejemplo de Datos del Comprador. 28
Ilustración 11. Ejempo de Datos del Vendedor..... 29
Ilustración 12. Ejemplo de Datos del Producto. 29
Ilustración 13. Ejemplo de Datos del Protocolo. 30
Ilustración 14. Ejemplo de Datos de la Mesa. 30
Ilustración 15. Configuración y establecimiento del Logger. 30
Ilustración 16. Conexión Qpid Broker. 31
Ilustración 17. Variables de la clase Java de Vendedor. 31
Ilustración 18. Extracto de código para establecer el ID del Agente. 32
Ilustración 19. Extracto de código del execute. 32
Ilustración 20. Extracto de código que representa DoBegin. 32
Ilustración 21. Extracto de código de DoEvaluateProposals. 33
Ilustración 22. Extracto de código de doReceiveInform. 33
Ilustración 23. Extracto de código para iniciar la conversación. 34
Ilustración 24. Extracto de código del reporte gráfico-propuestas aceptadas. 35
Ilustración 25. Reporte Gráfico de las propuestas aceptadas. 35
Ilustración 26. Extracto de código establecimiento del ID Comprador. 36
Ilustración 27. Extracto de código doTask. 37
Ilustración 28. Extracto de código SendProposal 37
Ilustración 29. Pseudocódigo Temporal Ascendente 40

Ilustración 30. Pseudocódigo Temporal Descendente 41
Ilustración 31. Pseudocódigo TitforTatAscendente 43
Ilustración 32. Pseudocódigo TitforTatDescendente 45
Ilustración 33. Pseudocódigo RandomAscendente 46
Ilustración 34. Pseudocódigo RandomDescendente 47
Ilustración 35. CFactory Comprador 47
Ilustración 36.Extracto de código al añadir comprador. 47
Ilustración 37. Resultado propuestas enviadas a los compradores. 48
Ilustración 38. Resultado protocolos de negociación. 48
Ilustración 39.Resultado estrategias de negociación. 49
Ilustración 40. Diagrama de protocolo Contract-Net.[6] 50
Ilustración 41. Diagrama de protocolo iterated contrac net (MAS, 2005). 51
Ilustración 42. Diagrama de funcionamiento protocolo Contract-Net. 52
Ilustración 43. Formato básico de comunicación. 53
Ilustración 44. Con decisión o exclusivo. 53
Ilustración 45. Con decisión o inclusivo. 53
Ilustración 46. Mensajes Concurrentes. 53
Ilustración 47. Definición, métodos y transiciones de estados. 55
Ilustración 48. Grafo Vendedor(Seller). 56
Ilustración 49. Grafo Comprador(Buyer). 57
Ilustración 50. Porcentaje de aceptación. 59

Índice de Tablas

Tabla 1. Descripción de materiales hardware 18
Tabla 2. Descripción de materiales Software 18
Tabla 3. Mensajes intercambiados protocolo Contract-Net. 50
Tabla 4. Colección de Estados CAgents. 54
Tabla 5. Estrategias Aceptadas. 59

1. Introducción

En este capítulo se describirá brevemente el problema que se pretende resolver con el presente trabajo, a continuación, se relatará la motivación que ha dado lugar a la realización de este, posteriormente se definirán los objetivos que se pretenden alcanzar y la estructura del trabajo.

1.1. Introducción

Debido al desarrollo actual de la sociedad, la adquisición de un producto o un servicio cada vez se encuentra más automatizado, esto debido a que las personas o empresas quieren reducir cada vez más los costes y el tiempo invertidos para dicha adquisición.

Recientemente se han desarrollado numerosas formas de facilitar la obtención de dichos productos o servicios, que permiten que el proceso de obtención sea cada vez más sencillo, rápido y efectivo.

Una de las formas que más se ha popularizado para dicho fin es el uso de Internet a través de medios electrónicos , esto debido a que tanto para el comprador como para el vendedor presenta muchas ventajas , ya que al comprador le permite poder realizar una investigación mucho más rápida del precio que tiene el producto o servicio que desea adquirir , de igual manera el comprador tiene mucho más opciones de vendedor a la hora de realizar la compra, para encontrar el producto y el precio que más se adapten a sus necesidades , todo esto desde la comodidad de su hogar y en un tiempo muy breve. Al vendedor le permite ofrecer sus productos o servicios, a una gran cantidad de compradores interesados, minimizando los costes de publicidad de estos y facilitando el proceso de venta. El hacer uso del internet también permite reducir el número de intermediarios entre el comprador y el vendedor lo que resulta beneficioso para ambos, ya que permite reducir los costes de manera significativa.

Si bien al hacer uso del internet se reducen los intermediarios y el proceso de automatización es mucho más rápido, siempre se necesita tener usuarios presentes en los roles de comprador y vendedor , lo cual puede resultar contraproducente cuando el número de compradores o vendedores aumente de forma significativa , cuando exista una gran variedad de productos ofertados cuyas características sean diferentes y en operaciones de venta o compra que deben

ser realizadas de manera rápida o que requieran respuesta de manera inmediata por parte ya sea del comprador o del vendedor .

Con el fin de resolver este problema se propone la creación y utilización de un mercado virtual que permita agilizar la interacción y negociación entre compradores y vendedores, por medio de agentes inteligentes automatizados cuyo comportamiento va a depender de las necesidades del usuario que represente.

1.2. Motivación

En la actualidad todas las empresas quieren poder ofertar sus productos o servicios a la mayor cantidad de público objetivo posible. Debido a la crisis actual que afecta al país lo anterior se ha vuelto indispensable, por lo cual todas las empresas que quieren mantenerse en el mercado y no cerrar sus puertas, tienen la necesidad de innovar y de reducir costes en la medida de lo posible.

Una herramienta que permite que las empresas puedan innovar y reducir costes es el uso de Internet como medio de venta de un producto o servicio, debido a esto cada vez más empresas invierten en este campo, sin embargo, aunque las empresas realicen dicha inversión es necesario tener recursos humanos para interactuar con los clientes, debido a la situación actual del país la cantidad de recursos humanos que presenta una empresa se ha reducido, lo cual entorpece dicha interacción, ya que las empresas no tienen disponible personal las 24 horas del día, los 365 días del año para gestionar acuerdos con los clientes, lo que tiene como consecuencia perder ganancias y clientes.

De este problema surge la necesidad de automatizar el proceso, para que por medio del mercado virtual basado en agentes inteligentes, la interacción con los clientes sea más rápida y efectiva ,el comportamiento que tendrán los agentes será confiable debido a que se basara en los objetivos establecidos por la empresa , el uso de dicho mercado virtual permitirá que las empresas puedan destinar más recursos humanos a otras áreas cruciales de la misma ,todo esto sin sacrificar acuerdos con los clientes, lo cual beneficiara en gran medida a la empresa ya que minimizara costes y aumentara ganancias, lo cual permitirá su crecimiento y fortalecimiento en el mercado.

El mercado virtual que se propone se encuentra compuesto por agentes inteligentes, cuyas acciones se basan en los objetivos del usuario que representa, dichos objetivos permiten restringir los acuerdos que serán aceptados ,para de esta manera garantizar que la ganancia requerida sea alcanzada ,otra parte fundamental del mercado son los protocolos por medio de los cuales se delimitara el tipo de subasta , las interacciones de la misma , el tiempo de respuesta y otros datos necesarios para llevar a cabo dicha subasta, también se tienen mesas de negociación por medio de las cuales un usuario puede empezar un proceso de negociación para vender un producto en específico, al iniciar una negociación se lanza un proceso de ejecución del mercado que permite a los usuarios de la mesa interactuar, con el objetivo de llegar a un acuerdo sobre la compra-venta de un producto determinado, una vez que se llega a un acuerdo este se formaliza entre los usuarios participantes del mismo.

Con el fin de facilitar la interacción entre compradores y vendedores, la salida del mercado virtual será representada por medio de reportes gráficos fáciles de comprender, que permitan que los usuarios que utilicen el prototipo además de gestionar las operaciones de compra y venta de productos puedan apreciar los acuerdos alcanzados, las propuestas recibidas, los protocolos y estrategias de negociación utilizados.

1.3. Objetivos

Los objetos que presenta este trabajo se dividen en general y específicos.

1.3.1. Objetivo General

Diseño, creación e implementación de un ambiente de desarrollo que facilite la creación de mercados virtuales genéricos, utilizando las tecnologías de: sistemas multiagentes, protocolos de negociación y estrategias de negociación.

1.3.2. Objetivos Específicos

- Análisis del estado del arte:
  - Definición de los elementos y tecnologías utilizados.
  - Estudio y comparación de las diferentes plataformas de mercados virtuales existentes.
  - Análisis de trabajos relacionados.

- Estudio y utilización de la plataforma MAGENTIX2, para llevar a cabo la creación del entorno de desarrollo.
- Implementar un prototipo de mercado virtual que permita la correcta interacción entre vendedores y compradores.

1.4. Estructura del trabajo

Los capítulos que se desarrollaran en este trabajo son los siguientes:

- **Capítulo 2, "Estado del arte":** se llevará a cabo un análisis con los siguientes objetivos: conocer el estado actual de la creación y diseño de mercados virtuales utilizando la tecnología de sistemas MultiAgente, conocer los protocolos de negociación que deben tenerse en cuenta para la creación de estos y conocer los conceptos fundamentales para la compresión de dicha tecnología.
- **Capítulo 3, "Diseño metodológico":** contiene el diseño metodológico utilizado para crear el presente trabajo, así como las etapas que lo componen.
- **Capítulo 4, "Desarrollo":** se explicará detalladamente todos los elementos que componen el prototipo, así como las características y funcionamiento de dichos elementos.
- **Capítulo 5, "Experimentación y Resultados":** se realizará el proceso de experimentación para comprobar el correcto funcionamiento del prototipo y se analizarán los resultados de dicho proceso.
- **Capítulo 6, "Conclusiones y Trabajos Futuros":** este capítulo contiene las conclusiones alcanzadas al finalizar el presente trabajo y los trabajos que pueden ser desarrollados basándose en este trabajo.

Al finalizar este capítulo podemos concluir que existe suficiente motivación para crear un prototipo de mercado virtual genérico, ya que los mercados comerciales que existen actualmente requieren que los usuarios realicen las interacciones de forma presencial.

2. Estado del Arte

En este capítulo se realizará un análisis de la literatura científica revisada. Primero una breve definición de los términos que constituyen el estado del arte del presente trabajo. Además, se realizará una comparación de las plataformas de mercados virtuales existentes, por último, se llevará a cabo un análisis de los trabajos relacionados.

2.1. Mercados, Comercio Electrónico y Plataformas Virtuales.

En esta sección se detallarán los términos mercado, comercio electrónico y plataformas virtuales, ya que estos son utilizados a lo largo del desarrollo del presente trabajo y es fundamental su correcta compresión.

2.1.1. Mercado

El término mercado se puede definir como un conjunto de personas y organizaciones que participan de alguna forma en la compra y venta de bienes o servicios, o en la utilización de estos. En el mercado existen diversos agentes que se influyen entre sí, dando lugar a un proceso dinámico de relaciones entre ellos.

Existen numerosos tipos de mercados, pueden clasificarse principalmente con base en las características de los compradores y con base en la naturaleza de los productos. Siendo los principales los mercados de consumo de acuerdo con las características de los compradores, en estos mercados se realizan transacciones de bienes y servicios, que son adquiridos por las unidades finales de consumo; y los mercados industriales, en los que se realizan transacciones de bienes y servicios empleados en la obtención de diferentes productos que son objeto de una transacción posterior o que se adquieren para obtener un beneficio mediante su posterior reventa. (María, 2002)

2.1.2. Comercio Electrónico (E-Commerce)

Cuando de comercio electrónico se trata se hace referencia al tipo de comercio que es desarrollado por medios electrónicos. Lo que significa que es una nueva forma de realizar el comercio tradicional, haciendo uso de los medios que las Nuevas Tecnologías de la Información y las comunicaciones, las TIC, ponen a nuestro alcance.

El comercio electrónico, es en sí una actividad económica que se desarrolla por medios electrónicos, y en la cual se aplican todas las normas del comercio tradicional y, además, las normas propias del medio en cual se desarrolla.

Existen diferentes criterios para clasificar el comercio electrónico. El más utilizado se basa en los posibles sujetos que intervienen en el proceso, empresarios o comerciantes, consumidor o usuario y administración pública, es posible distinguir las siguientes relaciones: entre empresarios *Business to Business* (B2B), entre empresario y consumidor *Business to Consumers* (B2C), entre empresario y administración *Business to Administrations* (B2A) , y entre Consumidor y Consumidor *Costumer to Costumer*(C2C) (Pou, 2006).

Por otro lado, también se pueden clasificar los tipos de comercio electrónico según el tipo de venta de. Estos se dividen en: Tiendas Virtuales, utiliza el modelo de ventas *B2C*. E-Procurement o *buysite*, consiste en el aprovisionamiento o suministro de productos o servicios. Centro de Comercio Virtual o *emall*, posee un gran número de tiendas virtuales asociadas bajo características comunes tales como marca o sector del mercado. Mercado virtual del cual se hablará más adelante. Comunidades Virtuales, que permiten a distintos usuarios de un mismo sector compartir dudas creando plataformas para la gestión del conocimiento (e-Commerce: aplicación y desarrollo, 2010).

2.1.3. Mercados virtuales o Marketplaces

Los mercados virtuales, se pueden definir como plataformas creadas por terceros. Ofrecen una web por medio de la cual un gran número de comerciantes o vendedores pueden ofrecer sus productos. Estas plataformas les ofrecen flujos de tráfico (flujo de compradores) ya existente, además de contar con los elementos necesarios para presentar sus productos y establecer escaparates online individuales. También pueden poner en contacto a minoristas

que ofrecen sus productos en el comercio electrónico con proveedores de servicios de confianza para que les ayuden en todos los aspectos relacionados con sus actividades, tales como atención al cliente, logística y soluciones para el almacenamiento (KROKOU, 2015).

También podemos definir a un mercador virtual, como un mercado tradicional, el cual se encuentra formado por compradores y vendedores. La diferencia reside en que, gracias a la transmisión electrónica de los datos, el proceso se vuelve más ágil debido a que tanto compradores como vendedores, poseen más información, lo que permite disminuir costos y obtener mayores beneficios. El mercado virtual se encuentra representado por medio de una plataforma "on-line", en la que interaccionan las demandas y ofertas de compradores y vendedores. En este caso el mercado virtual actúa como intermediario entre las dos partes interesadas.

Pueden existir tres tipos de mercados virtuales, según la estrategia de interacción, que se dé entre el comprador y el vendedor: Agregador de catálogos, hipermercados con precios relativamente fijos. Sistema de precios, fluctuando según oferta y demanda, el cual presenta un comportamiento similar al de la bolsa de valores. Subasta, consiste en un vendedor y múltiples compradores. Los precios varían dependiendo del tipo de subasta del que se trate (e-Commerce: aplicación y desarrollo, 2010)

2.1.4. Plataformas de mercados virtuales

Una plataforma de mercado virtual es un sistema que sirve como base, para permitir el correcto funcionamiento y ejecución, de los módulos de hardware y software, que componen un mercado virtual, y que proporciona los elementos necesarios para realizar una transacción comercial exitosa entre vendedores y compradores.

En la actualidad existen numerosas plataformas virtuales, debido a los grandes beneficios que estas presentan para compradores y vendedores, ya que permiten las transacciones de manera más ágil y con menores recursos, lo que permite tener mayores ingresos y disminuir los costos.

Algunas de las famosas en la actualidad son: Amazon, Ebay, AliExpress(perteneciente al grupo AliBaba), Milanuncios, Mercado Libre, Google Shooping, entre otras.

Todas las anteriormente mencionadas tienen como objetivo principal realizar el mayor número de transacciones entre sus usuarios. Para poder llevar a cabo dicho objetivo estas plataformas utilizan diferentes estrategias de interacción entre vendedores y usuarios, con el fin de aumentar sus ganancias.

También existen propuestas de plataformas virtuales que presentan el mismo objetivo, pero funcionan de manera diferente, debido a que emplean tecnologías de inteligencia artificial, en estas plataformas es necesario contar con un conjunto de reglas, por medio de las cuales se pueda regir el comportamiento y propuestas de los agentes (que representan a los compradores y a los vendedores) que las componen para poder llegar a un arreglo que resulte beneficioso para ambos (Chavez, 1996), (Tsvetovatyy M. &., 1996), (Wurman, 1998), (Richter, 1998), (Oliveira, 2001), (Cuní, 2004). El conjunto de reglas que se emplea es lo que en inteligencia artificial es conocido como protocolo de negociación.

2.2. Agentes inteligentes, Sistemas Multi-agentes y Negociación

En la presente sección se definirá el significado de: agentes inteligentes, sistemas Multiagentes, negociación y protocolos de negociación, debido a que estos elementos forman parte de la tecnología de Sistemas MultiAgente utilizada para el desarrollo del prototipo.

A continuación, se realizará una comparación de las plataformas virtuales existentes tomando en cuenta elementos en común y las limitaciones que presentan, por último, se realizará una breve discusión de los trabajos relacionados.

2.2.1. Agente Inteligente

Un agente software, es una entidad software que funciona de manera continua y autónoma en un entorno particular. Cada agente software tiene funciones específicas y responde a eventos específicos, basado ya sea en reglas de conocimiento, las colaboraciones de otros agentes o las instrucciones de los usuarios (Brenner, 1998).

Según (MAS, 2005) las características destacables del comportamiento de un agente son:

- Funcionamiento continuo y autónomo.
- Comunicación con el entorno y con otros agentes.

- Robustez.
- Adaptabilidad como capacidad de realizar objetivos y tareas en distintos dominios de forma incremental y flexible.

2.2.2. Sistemas Multiagentes

Un sistema Multi-Agente es un conjunto de agentes de software que interactúan para resolver problemas, que están más allá de las capacidades individuales o el conocimiento de cada agente individual. Los agentes interactúan a través de una plataforma, la cual generalmente proporciona a los agentes un conjunto de servicios que dependen de las necesidades del sistema y se considera como una herramienta para los agentes (K Potiron, 2013).

De acuerdo con (Wooldridge, 1995) los agentes, tienen las siguientes propiedades:

- Autonomía: Un agente posee metas individuales, recursos y competencias; debido a esto opera sin intervención humana directa o de otro tipo, y presenta un grado de control sobre sus acciones y su estado interno. Una de las consecuencias más importantes de la autonomía del agente es la adaptabilidad de este, ya que un agente tiene el control sobre su propio estado y así puede regular su propio funcionamiento sin asistencia externo o supervisión.
- Sociabilidad. Un agente puede interactuar con otros agentes, y posiblemente humanos, a través de algún tipo de lenguaje de comunicación de agente. Por este medio, un agente es capaz de proporcionar y solicitar servicios.
- Reactividad. Un agente percibe y actúa, hasta cierto punto, en su entorno cercano; puede responder de manera oportuna a los cambios que ocurren a su alrededor.
- Proactividad. Aunque algunos agentes llamados agentes reactivos, actúan en respuesta a estímulos de su entorno, un agente puede exhibir comportamiento dirigido a objetivos tomando la iniciativa.

Los sistemas multiagentes pueden ser clasificados de un sin número de maneras, con el fin de ejemplificar su clasificación de manera breve se utilizarán los siguientes criterios:

1. Autonomía, partiendo del control de la autonomía del agente podemos tener las siguientes categorías (Florea, 2004):

- Autonomía del usuario: es cuando el agente depende del usuario para tomar una decisión, ejemplo de esto es un asistente personal, donde el usuario debe elegir el comportamiento y el plan del agente.
- Autonomía social: sucede cuando el agente debe tomar objetivos de otros agentes como propios dependiendo del contexto en el que se encuentre.
- Autonomía basada en normas: en esta categoría se utilizan leyes, o estructuras organizativas para restringir la autonomía de los agentes.
- Autonomía con respecto al medio ambiente: se refiere al hecho de que el medio ambiente solo puede influir en el comportamiento de un agente, no puede imponerlo.
- Autonomía propia: el agente posee control local de su comportamiento.

2. Interacción entre sus agentes, haciendo uso de la interacción de los agentes de un sistema, podemos obtener los siguientes modelos (Ngobye, 2010) :
    - NI- *Interacciones no directas:* sucede cuando los agentes realizan sus tareas o actividades sin tener interacción con los demás agentes del sistema.
    - SI- *Interacciones Simples:* en este modelo los agentes tienen interacciones de carácter simple.
    - CI- *Interacciones Complejos:* los agentes presentan interacciones complejas entre ellos y se caracterizan por él, “Si tú mueves, Yo muevo”, por ejemplo, la regla del Tit For Tat y el Prisoner's dilemma.

2.2.3. Negociación

Podemos definir negociación, como un conjunto de mecanismos que modelan la coordinación entre agentes auto-interesados capaces de llegar a acuerdos vinculantes. Estos mecanismos permiten la convergencia hacia decisiones de coordinación conjunta mediante una comunicación explicita. (H., 1996). El sistema PERSUADER de Sycara (Sycara, 1990) y el trabajo de Rosenschein y Zlotkin (Rosenschein, 1994) figuran en los primeros trabajos sobre negociación entre agentes auto-interesados.

En la actualidad existen la negociación manual y la negociación automática, la negociación manual, es realizada por un usuario común, esta negociación conlleva mucho tiempo y, por

lo tanto, es costosa, además de esto es considerada vergonzosa o frustrante para consumidores comunes. En cambio, la negociación automática es ejecutada por una agente, y no requiere que los usuarios estén ubicados en el espacio o el tiempo, la automatización presenta mejores resultados debido a que elimina las sensibilidades humanas y pueden conducir a resultados más satisfactorios (R. Das, 2001), (P. Maes, 1999), (Luo, 2003) .

De acuerdo con la cardinalidad y la naturaleza de la interacción, los modelos de negociación automatizados se pueden clasificar en tres categorías principales (Lomuscio, 2001):

1. Many-to-one o Many-to-many: en este modelo varios agentes negocian con un solo agente o con muchos otros agentes. Esta categoría se maneja predominantemente utilizando varios modelos basados en subastas (Goyal, 2012), (Sandholm, 1999), (Walsh, 1998).Estos modelos son ampliamente utilizados en el campo del comercio minorista en línea (por ejemplo, eBay, Amazon).
2. One-to-one: en este modelo un par de agentes interactúan directamente entre sí (M. Barbuceanu, 2000), (P. Faratin, 2002), (R. Kowalczyk, 2000), (N. Matos, 1998), (S. Paurobally, 2001).
3. Basados en argumentación: en este modelo los agentes emplean varios tipos de argumentos, como amenazas, recompensas y apelaciones, para persuadir a sus oponentes de que acepten un trato que antes no habrían tolerado o aceptado (Kraus, 2001), (S. Kraus, 1998), (S. Parsons, 1998).

Los componentes principales de un proceso de negociación son (Lomuscio, 2001):

- Objetos de Negociación: la gama de cuestiones sobre las cuales se debe llegar a un acuerdo. En un extremo, el objeto puede contener un único problema (como el precio), mientras que, por otro lado, puede cubrir cientos de problemas (relacionados con el precio, la calidad, los tiempos, las penalizaciones, los términos y condiciones, etc.).
- Protocolos de Negociación: es el conjunto de reglas que rigen la interacción. Esto engloba los tipos permitidos de participantes (por ejemplo, los negociadores y terceros relevantes), los estados de negociación (por ejemplo, aceptación de ofertas, negociación cerrada), los eventos que provocan que la negociación cambie (por ejemplo, no hay más ofertas, oferta aceptada) y las acciones válidas de los

participantes en estados particulares (por ejemplo, qué mensajes pueden enviarse por quién, a quién, en qué etapa).

- Estrategias de negociación: es el motor de toma de decisiones que los participantes emplean, para actuar de acuerdo con el protocolo de negociación para alcanzar sus objetivos. La sofisticación del modelo, así como la gama de decisiones que deben tomarse, están influenciadas por el protocolo vigente, por la naturaleza del objeto de negociación, y por el rango de operaciones que se pueden realizar en él protocolo.

2.2.4. Protocolos de negociación

Existen numerosos protocolos de negociación, entre los más utilizados tenemos (MAS, 2005):

- Mecanismos de mercado: este modelo equipara un SMA con un mercado virtual, con el objetivo de que dicho sistema tenga algunas de las propiedades deseables de los mercados reales. Se basa en la denominada *Teoría del Equilibrio General* (Varian, 1992), que proporciona un método distribuido para una asignación eficiente de bienes y recursos entre los agentes. En este modelo es posible implementar un mecanismo de subasta distribuido para asignar recursos(bienes) entre múltiples agentes (productores y consumidores) o simplemente realizar un proceso de subasta entre compradores y vendedores.
- Teoría del regateo: En un regateo los agentes pueden llegar a un acuerdo mutuamente beneficioso, pero tienen un conflicto de intereses acerca de qué acuerdo seleccionar (Aumann & Hart, 1994), (Friedman, 1991), (Thomson, 1997). Esta teoría se distingue entre dos aproximaciones: la no cooperativa o estratégica y la cooperativa o axiomática. En la aproximación estratégica la atención se centra en las acciones que cada jugador es capaz de adoptar, de forma que alcance algún tipo de equilibrio. Por el contrario, en la aproximación axiomática se estudia directamente el espacio de resultados posibles y no la forma en la que se llega a éstos, estableciendo las propiedades deseables de la solución de regateo.
- Formación de coaliciones: el modelo teórico subyacente de este mecanismo de cooperación es la teoría de juegos cooperativa, que estudia modelos matemáticos de

cooperación y de conflictos en situaciones multi-personales, en las que el objetivo de los participantes es maximizar su propia utilidad sabiendo que la misma actitud será adoptada por el resto de los participantes (Aumann & Hart, 1994), (Friedman, 1991).

- Votaciones: el objetivo de este mecanismo es derivar una regla de elección social que clasifique los resultados sociales posibles, basándose en las clasificaciones individuales de estos resultados (Sandholm, 1999).

2.3. Comparación de plataformas de mercados virtuales existentes

Como se mencionó anteriormente existen numerosas plataformas de mercados virtuales, por lo tanto, es necesario realizar una comparación entre ellas, en esta sección se realizará dicha comparación para poder delimitar los aspectos comunes y las limitaciones que presentan.

Para realizar la comparación se tendrán en cuenta los siguientes aspectos:

- La relación o relaciones que existen entre los sujetos que interactúan en la plataforma, siendo estas B2B, B2C y C2C.
- Estrategia de interacción que se da entre el comprador y el vendedor, Subastas y Agregador de catálogos.

Basándonos en el primer aspecto podemos agrupar las plataformas de mercados virtuales en dos grupos: Las que presentan una única relación y las que presentan dos o más, dentro del primer grupo podemos encontrar a: AliExpress (B2C), Alibaba (B2B), Mercado Libre (C2C), Rakuten (B2C), Google Shooping (B2C), Fnac (B2C) y Etsy (C2C). En el segundo grupo se encuentran: Amazon (B2B, B2C, C2C), Ebay (B2C, C2C) y Dixmania (B2C, C2C).

Haciendo uso del segundo aspecto, de igual manera podemos agrupar las plataformas en dos grupos: Las que utilizan una sola estrategia de interacción y las que utilizan dos tipos de estrategias diferentes. En el primer grupo se encuentran: Amazon, AliExpress, Alibaba, Rakuten, Dixmania, Google Shooping, Fnac y Etsy, empleando todos únicamente, agregador de catálogos. En el segundo grupo se encuentran Ebay y Mercado Libre debido a que, aparte de utilizar agregador de catálogos también emplean Subastas.

Sin importar que relaciones presenten o que estrategias de interacción utilizan, las plataformas mencionadas anteriormente presentan una limitación en común todas necesitan que tanto el vendedor, como el comprador, sean los responsables de manejar las transacciones que ocurran entre ellos. La estrategia de subasta presenta una limitación adicional, debido a que para llevar a cabo la subasta es necesario la presencia constante de los usuarios que figuran como compradores interesados en el producto, ya que dichos usuarios deben estar pendientes de la fluctuación del precio que presente el objeto que deseen comprar o el servicio que desean adquirir y posteriormente realizar ofertas sin ningún tipo de estrategia.

Basándonos en estas limitaciones podemos claramente identificar que las plataformas virtuales que se mencionaron presentan una diferencia con la propuesta que se realiza en el presente proyecto, debido a que en dicha propuesta como se ha mencionado anteriormente se pretende implementar una plataforma de mercado virtual, que funcione por medio de la tecnología de sistemas multiagentes, en la cual cada usuario tendrá un agente asignado que realizara las transacciones del usuario, basándose en un protocolo, una estrategia de negociación , en los objetivos y demás parámetros establecidos por el usuario. Por lo tanto, la presencia del usuario en el espacio tiempo no será necesaria. Además, la negociación utilizando agentes puede disminuir en gran manera el tiempo de las transacciones, debido a que los agentes pueden encontrar, enumerar y evaluar ofertas potenciales mucho más rápido que los usuarios comunes.

2.4. Trabajos relacionados

Al analizar la literatura científica investigada, se puede valorar que existen varios trabajos relacionados que hacen uso de la negociación automática para crear mercados virtuales basados en agentes inteligentes, pero cada uno desde enfoques diferentes. (Krulwich, 1996) propone una de las primeras aproximaciones realizada para la creación de un agente inteligente, en su obra presenta dos agentes, BargainFinder es un agente que realiza una comparación de precio entre varias tiendas de CD en línea y NewsFinder recupera artículos de noticias en línea, los compara y los transmite a través de ordenadores portátiles. Entre los primeros trabajos realizados sobre este tema también podemos encontrar a los autores (Chavez, 1996) y (Tsvetovatyy M. &., 1996), siendo el primero el creador de Kasbah, un mercado virtual donde los usuarios pueden crear agentes autónomos para comprar y vender

bienes en su nombre, los agentes de los usuarios negocian entre sí para realizar transacciones con el fin de obtener el mejor precio posible, este mercado utiliza la teoría del regateo como protocolo de negociación y como estrategia un algoritmo basado en procesos matemáticos simples. El segundo autor es el responsable de realizar la primera aproximación de MAGMA, en la cual se plantea una arquitectura extensible que proporciona todos los servicios indispensables para realizar actividades comerciales basadas en agentes. Al año siguiente los creadores de MAGMA (Tsvetovatyy M. G., 1997), realizaron una aproximación más completa, en la cual definen a MAGMA, como un mercado virtual basado en agentes inteligentes, que contiene numerosos elementos de un mercado real, como infraestructura, almacenamiento, comunicación, transferencia, almacenamiento de bienes y banca. Como protocolo utiliza la Subasta y la estrategia utilizada por los agentes de MAGMA para negociar es la de Vickrey.

Por otro lado, en la parte teórica existen algunos estudios que analizan distintos comportamientos que presentan los agentes inteligentes. Por ejemplo, en la obra del autor (Wu, 2001) , se plantean diferentes mecanismos de coordinación que pueden ser utilizados en las cadenas de suministros basadas en agentes y también mecanismos para subastas mediadas por agentes. Otro estudio relacionado lo podemos encontrar en (Guttman, 1999), en este estudio el autor examina los roles de los agentes que figuran como mediadores en un sistema de comercio electrónico, basándose en el modelo CBB (*Consumer Buying Behavior*) y la venta al por menor.

En la parte práctica, existen dos grupos: en el primero se propone la creación de arquitecturas de mercados virtuales basadas en agentes y en el segundo se plantea el desarrollo de agentes inteligentes automatizados que representen los intereses de un usuario común. Dentro del primer grupo podemos encontrar el reconocido Auction Bot (Wurman, 1998), que no es más que un servidor flexible, robusto y escalable, que soporta la negociación entre humanos y agentes software. Su protocolo de negociación es la Subasta, los agentes pueden participar en distintos tipos de subastas, (ejemplo: de Vickrey, alemana, inglesa, etc..) pujando por el usuario que representan. Dentro de este grupo también se encuentra la obra de (Richter, 1998), en la cual propone un entorno en el que es posible comprar y vender a través de subastas dobles implementadas en una bolsa regional de productos básicos, los agentes que

pertenecen a este ambiente tienen como estrategia el uso de algoritmos genéticos. Oliveria (Oliveira, 2001), plantea una arquitectura de mercado virtual, dicha arquitectura incluye un agente de mercado(coordinador) y agentes inteligentes que representan a socios comerciales y cuyos comportamientos y estrategias , se basan en los objetivos del socio que representan, el protocolo de negociación que utilizan incluye criterios múltiples y formalismos basados en restricciones, como estrategia emplean el algoritmo Q-learning. COMPRANET (Jaso, 2004) es un sistema creado para administrar procesos de compras(licitaciones) gubernamentales. A través de este sistema se realizan de forma electrónica todas las compras de todas las dependencias del Gobierno Federal mexicano. En este sistema se utiliza TA (Tecnología de Agentes) en dos versiones: agentes de uso interno para recopilar y buscar información anómala y agentes externos que realizan servicios para los compradores o vendedores sobre la información pública de COMPRANET (MAS, 2005). iBlunder es un sistema basado en agentes (compuesto por agentes y componentes software) cuyo propósito es facilitar la negociación de procesos complejos de compras (o ventas), transacciones que típicamente suponen la agregación de ofertas de distintos proveedores para cumplir un contrato de forma óptima (Giunchiglia, 2002). En iBlunder los agentes median con el comprador y con los proveedores y ejecutan las labores propias de un agente de licitación (MAS, 2005). Masfit es un sistema de subasta intermediada por agentes que permite a los compradores participar en varias subastas que suceden simultáneamente (Cuní, 2004). Masfit fue diseñado sobre TA, debido a esto en una misma subasta pueden participar compradores humanos y agentes software (MAS, 2005).

En el segundo grupo se encuentran los agentes: ITA e ISA. ITA (*Intelligent Trading Agency*), plantea un sistema de negociación basado en subastas, de cardinalidad uno-a-muchos, el proceso de negociación se lleva a cabo a través de varias negociaciones de uno-a-uno simultáneas. El razonamiento de los agentes de ITA se basa en varios aspectos: la teoría de utilidad mediante atributos múltiples y técnicas basadas en restricciones para la evaluación y generación de ofertas y contraofertas (Rahwan, 2002). ISA (*Intelligent Sales Agent*), propone un agente autónomo y fácil de utilizar, que funciona como un empleado virtual de una tienda electrónica. Este agente adopta dinámicamente diferentes estrategias de persuasión y negociación, que le permiten realizar argumentos y ofertas de acuerdo con las características propias de los compradores humanos con los que interacciona. Como protocolo de

negociación utiliza la teoría del regateo y como estrategia el algoritmo Q-learning (Huang, 2005).

En este capítulo, podemos concluir que existen numerosas plataformas de mercados virtuales, las comerciales que requieren interacción presencial de los usuarios y las que utilizan la tecnología MultiAgente, cuyo funcionamiento depende de agentes inteligentes.

3. Diseño metodológico

En este capítulo se enumerará y explicará los materiales utilizados para la creación e implementación del prototipo de mercado virtual, también se describirá la metodología de desarrollo software utilizada y las etapas desarrolladas en el proceso de investigación.

3.1. Materiales utilizados

Los materiales utilizados se pueden dividir en materiales Hardware y materiales Software.

3.1.1. Materiales Hardware

Los materiales hardware implementados son:

Material	Descripción	Características
Ordenador portátil.	HP Pavilion Notebook Componente físico en el cual el prototipo se desarrollará e implementará.	Windows 10 Home (64 bits) Intel Core i7-6500U 2,5 GHz 8 GB de Ram Disco duro de 1 TB NVIDIA GeForce 940M

Tabla 1. Descripción de materiales hardware

3.1.2. Materiales Software

Los materiales software utilizados fueron los siguientes:

Software	Descripción
EclipseJava Neon	Software utilizado para la creación del prototipo y para el proceso de experimentación.
Axure RP8	Software por medio del cual se modelaron las diferentes máquinas de estados y procesos que se realizan en el prototipo.
PSeInt	Software utilizado para crear archivos de pseudocódigo y diagramas de flujo.

Tabla 2. Descripción de materiales Software

3.2. Metodología de Desarrollo Software

El presente trabajo de investigación está basado en el modelo de desarrollo software en espiral.

El modelo espiral en el desarrollo del software es un modelo meta del ciclo de vida del software donde el esfuerzo del desarrollo es iterativo, tan pronto culmina un esfuerzo del desarrollo por ahí mismo comienza otro (FARIÑO, 2011). Este modelo permite llevar a cabo paso a paso el desarrollo del mercado virtual, comenzando con la formulación del problema que motiva el proyecto a desarrollar hasta culminar con el prototipo finalizado.

La serie de etapas utilizadas para la creación del prototipo se pueden observar en la siguiente imagen.

Ilustración 1. Etapas del proyecto de investigación.

3.3. Etapas del proyecto de investigación

En esta sección se describirán las etapas de desarrollo en las cuales se dividió el proceso de investigación del presente trabajo.

Con el fin de dar cumplimiento a cada uno de los objetivos planteados anteriormente, el presente trabajo se dividió en las siguientes etapas:

3.3.1. Etapa I: Formulación del problema

En esta etapa se realizó la delimitación de los aspectos a ser desarrollados en el trabajo y se plantearon los objetivos a cumplir.

3.3.2. Etapa II: Estudio del Estado del Arte

Esta etapa fue dedicada a la recolección y estudio de literatura relacionada a los temas Sistemas Multi-Agente y mercados virtuales, la búsqueda de información se realizó en diversos medios de información como libros, páginas web confiables, artículos de

revistas. Posteriormente se realizó un análisis de la literatura encontrada para conocer el estado del arte del problema planteado.

3.3.3. Etapa III: Integración de Tecnologías

3.3.3.1.1. Estudio y Reconocimiento de la plataforma basada en agentes que se utilizará.

Se realizó la lectura detallada y el análisis del manual de usuario correspondiente a la plataforma MAGENTIX2.

3.3.3.1.2. Instalación y configuración de la plataforma MAGENTIX2.

Se procedió a la instalación y posterior configuración de la plataforma, a continuación, se realizaron pruebas de los ejemplos que esta incluye con el fin de conocer mejor su funcionamiento.

3.3.3.2. Creación del mercado virtual dentro de la plataforma.

Se diseño y programo todo lo relacionado al mercado virtual, los agentes principales (Vendedor y Comprador) fueron desarrollados, también se crearon los ficheros de propiedades que son utilizados como archivos de configuración, por último, se diseñaron y programaron reportes gráficos para facilitar la compresión de los resultados.

3.3.4. Etapa IV: Prueba y Depuración

Una vez que el diseño y la programación estuvieron terminados, se realizaron las pruebas y posteriormente se corrigieron los errores encontrados, se realizó esto hasta que se consideró que el mercado virtual funcionaba de forma satisfactoria.

3.3.5. Etapa V: Conclusiones y Trabajos Futuros

Se obtendrán las conclusiones alcanzadas al finalizar el trabajo y los posibles trabajos futuros a desarrollar.

Al terminar este capítulo es posible concluir, que fue necesario seguir cada una de las etapas de desarrollo, para llevar a cabo con el éxito la creación del prototipo.

4. Desarrollo

En el presente capítulo se explicará detalladamente cada uno de los elementos principales del mercado virtual, de igual manera se introducirán las herramientas utilizadas para el desarrollo de un prototipo funcional.

4.1. Elementos del mercado virtual

En esta sección se definirán los elementos que componen el prototipo del mercado virtual y que son fundamentales para su correcto funcionamiento.

Para el desarrollo y funcionamiento del mercado virtual basado en la tecnología de Sistemas Multi-Agente, podemos identificar seis elementos principales:

1. **Plataforma basada en Agentes**

La plataforma basada en agentes es un elemento indispensable, ya que a través de esta y de las funcionales o servicios que presenta es posible la creación e implementación de un prototipo de mercado virtual funcional.

2. **Fichero properties o de propiedades**

Los diferentes ficheros properties, permiten almacenar parámetros de configuración, que son utilizados al iniciar la ejecución del mercado virtual.

3. **Tareas de inicialización**

Son tareas ejecutadas antes de iniciar los agentes que serán miembros del mercado, ya que son necesarias para el correcto funcionamiento de estos.

4. **Agentes Vendedores**

Son los agentes que actúan con el rol de "Vendedor" en el mercado, y los iniciadores de las conversaciones.

5. Agentes Compradores

Agentes que actúan con el rol de "Comprador" en el mercado, y son los participantes voluntarios de las conversaciones.

6. Módulo Gestor

Es el elemento principal del mercado virtual, ya que es el encargado de controlar los cambios de estado en los agentes, las conversaciones e interacciones que se desarrollan entre estos, crear nuevas conversaciones y llegar a acuerdos en las conversaciones creadas.

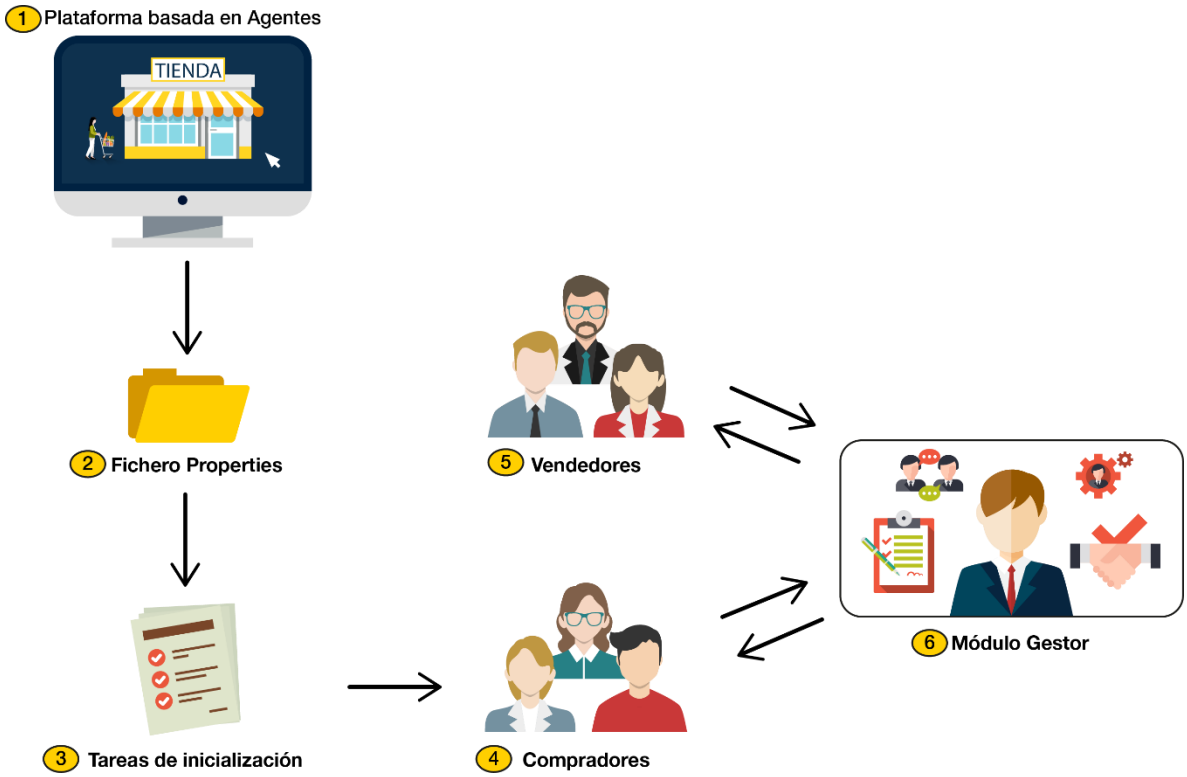

Ilustración 2. Elementos principales del mercado virtual

4.1.1. Plataforma basada en Agentes

En esta subsección se definirá el lenguaje de programación Java ya que fue utilizado en todos los elementos que pertenecen al mercado virtual, a continuación, se detallarán las características, el funcionamiento y los componentes de la plataforma virtual sobre la cual fue desarrollado el prototipo de mercado virtual genérico.

4.1.1.1. Lenguaje de Programación Java

Java es un lenguaje de programación simple, distribuido recurrente y orientado a objetos, cuyo principal objetivo es el desarrollo de software con el menor número de requisitos de implementación como sea posible. Esto permite facilitar en gran medida la portabilidad del software desarrollado, lo que presenta una gran ventaja. Presenta numerosas ventajas, contiene un sin número de clases preexistentes expresadas en librerías que permiten facilitar la labor del programador, permite el uso de hilos en la ejecución de procesos y su entorno, compilador e interprete fueron creados con el fin de proveer un ambiente de seguridad a los desarrolladores. De igual manera presenta algunas desventajas, su rendimiento es significativamente más lento y utiliza más espacio de memoria comparado con otros lenguajes de programación como C# y C++.

El prototipo de mercado virtual fue programado con este lenguaje, principalmente por la compatibilidad que este presenta con la plataforma basada en agentes donde fue desarrollado el mercado y por las facilidades que ofrece al momento de graficar los resultados obtenidos de las conversaciones entre los agentes.

4.1.1.2. Magentix2

Magentix2, es una plataforma basada en agentes, creada e implementada por el Grupo de Tecnología Informática- Inteligencia Artificial de la Universidad Politécnica de Valencia. Esta plataforma puede ser utilizada en sistemas abiertos con tecnología multiagentes. El objetivo principal es permitir y facilitar la utilización de la tecnología de agentes en dominios reales: negocios, industria, logística, comercio electrónico, atención médica, etc.

Ilustración 3. Logotipo y esquema de funcionalidades de MAGENTIX2. [1]

[1] http://www.gti-ia.upv.es/sma/tools/magentix2/

Una de las principales características que presenta es que admite protocolos y conversaciones de interacción flexibles entre sus agentes, ha sido desarrollada en el lenguaje de programación Java con librerías independientes, lo cual permite garantizar su portabilidad ya que no tiene como requisito su instalación y uso en un sistema operativo específico, también incluye una API de argumentación que permite a los agentes participar en diálogos argumentativos utiliza THOMAS con el fin de permitir a los usuarios gestionar fácilmente aspectos de organización y servicio, presenta compatibilidad para crear interfaces basadas en el protocolo HTPP y presenta soporte a agentes que utilizan Jason (GTI, 2015).Magentix2 proporciona mecanismos de comunicación avanzados, tales como grupos de agentes, protocolos de interacción predefinidos y un mecanismo de seguridad para proporcionar autenticación, integridad, confidencialidad y control de acceso (Alberola, 2008).

Permite la creación de varios tipos de agentes con niveles de complejidad diferentes: agentes básicos y simples (BaseAgent, SimpleAgent), agentes conversacionales avanzados (CAgents), agentes BDI (BDI Agents) y agentes argumentativos (Argumentative Agents). De igual manera permite que estos agentes puedan ser agrupados en organizaciones virtuales.

Los siguientes párrafos provienen del trabajo de (GTI, 2015):

El uso de los protocolos de interacción es monitoreado por Magentix2 a través del ID (Qpid Broker) de cada agente, este ID es único por cada agente y debe ser definido antes de lanzarlo a ejecución, el ID funciona como una cola de intercambio de mensajes entre los agentes.

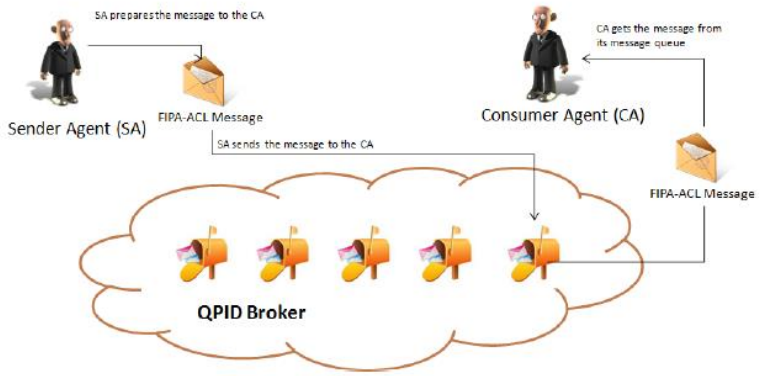

Ilustración 4. Intercambio de Mensajes a través del QPID Broker [2]

[2] http://www.gti-ia.upv.es/sma/tools/magentix2/

La comunicación de los agentes se puede realizar a través de los siguientes tres protocolos FIPA de interacción:

a) Request: permite a los agentes iniciadores, solicitar a otros agentes que ejecuten una determinada acción.
b) Query: permite a los agentes iniciadores, solicitar a otros agentes la consulta de una propuesta particular para saber si es verdadera o falsa.
c) Contract-Net: presenta el siguiente comportamiento, el agente que actúa como iniciador envía una propuesta a varios respondedores, luego evalúa sus respuestas y elige al agente que según sus criterios sea el mejor postor.

Los protocolos de interacción tienen dos roles definidos:

a) Iniciadores, estos agentes son ejecutados una vez por conversación, son los encargados de iniciar la conversación, solicitando respuesta con el fin de iniciar una nueva conversación, para requerir una acción, una consulta o una propuesta.
b) Respondedores, son ejecutados de manera cíclica, ya que regresan a su estado inicial después de alcanzar su estado final, estos agentes deciden aceptar la solicitud de respuesta por parte de los iniciadores para entablar una conversación.

Estos roles permiten facilitar la programación de los agentes ya que, gracias a estos, los programadores solo deben definir qué acciones se debe realizar en cada estado que se alcance en el protocolo y preparar los mensajes antes de enviarlos. Las acciones realizadas en cada estado son definidas por los manejadores en el rol de Iniciador y por los preparadores en el rol de Respondedor. Para utilizar los protocolos de interacción predefinidos existen plantillas, que pueden ser editadas por el programador, de acuerdo con sus necesidades.

El tipo de agente seleccionado para utilizarse en el mercado virtual es el CAgent (Agente conversacional) debido a que utilizan CFactories (Fábricas) y CProcessors(Procesadores):

a) CFactories: actúan como Protocolos de Interacción y están a cargo de la creación de nuevos CProcessors.
b) CProcessors: actúan como instancias de CFactories, y representan cada conversación que se da entre dos agentes. Cada CProcessor contiene una gráfica que representa la secuencia de acciones que una conversación desarrollada dentro de un protocolo

debe seguir, cada estado representa una acción específica y cada aro representa una posible transición entre estados (GTI, 2015).

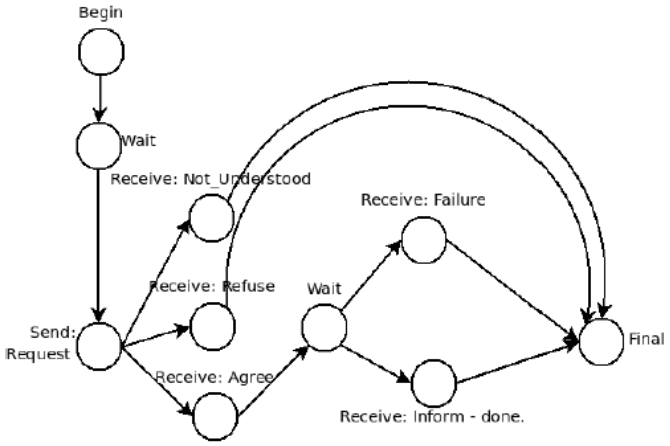

Ilustración 5. Gráfica del CProcessor, protocolo REQUEST.[3]

Otra de las razones para elegir a los CAgents es que poseen una definición por defecto de los estados necesarios para utilizar el protocolo de interacción Contract-Net.

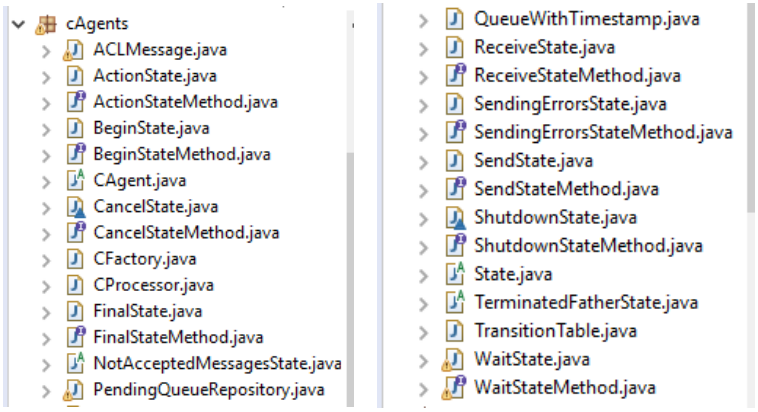

Ilustración 6. Clases contenidas CAgents

Cabe recalcar que en este proyecto se utilizará el protocolo de interacción Contract-Net debido a que presenta un comportamiento ideal para ser utilizado en Compra y Venta, en este caso el rol de Iniciador será representado por los agentes vendedores y el rol de participante será representado por los compradores, en la sección del módulo gestor se explicará con mayor profundidad las características y el funcionamiento que presenta este protocolo de interacción.

[3] http://www.gti-ia.upv.es/sma/tools/magentix2/

4.1.2. Fichero properties

Como se mencionó anteriormente los ficheros properties o de propiedades, contienen los parámetros de configuración necesarios para ejecutar el mercado virtual.

En cada línea se puede visualizar una pareja de clave y valor, el nombre de la clave es correspondiente al nombre del parámetro y el valor, es el contenido de dicho parámetro.

```
# Esto es un comentario
path=c:/un/path
username=Pedro Arenas
password=un secreto
```

Ilustración 7. Estructura Fichero properties. [4]

Para acceder al contenido del parámetro es necesario crear una instancia de la clase Properties, cargar el fichero en ese objeto por medio de la función load, y a continuación acceder al contenido de cada parámetro con la propiedad getProperty seguida del nombre de dicho parámetro.

```
Properties p = new Properties();
p.load(new FileReader("files/properties/config.properties"));

System.out.println("uno="+p.getProperty("uno"));
```

Ilustración 8. Lectura del fichero properties y acceso a los parámetros. [4]

4.1.2.1. Estructura fichero Properties

A continuación, se describirá la estructura definida y empleada en todos los ficheros properties.

a) Datos iniciales: como datos iniciales figuran el número total de agentes en el universo (Vendedores+Compradores), el número de vendedores o mesas y el número de compradores.

```
#Número de agentes en el universo
num_agentes=2

#Número de mesas del fichero
num_mesas=1

#Número de compradores
num_compradores=1
```

Ilustración 9. Datos iniciales

[4] http://chuwiki.chuidiang.org/index.php?title=Leer_y_modificar_fichero_de_propiedades_en_java

b) Datos de los compradores: los datos de los compradores son, el ID del agente que debe ser único e irrepetible, aspiración que es la cantidad máxima que el usuario del agente pretende gastar en las negociaciones efectuadas, utilidad de reserva la cantidad mínima a gastar en las negociaciones, el incremento que puede darse en las negociaciones, el tipo de agente en este caso buyer (Comprador), conversaciones habilitadas que corresponde al número de mesas del fichero, preferencias del agentes es decir los productos en los que esta interesado y estrategias de negociación que se utilizaran según el tipo de subasta, todos estos parámetros se repetiran en cada comprador variando sus valores y su cantidad en los parámetros preferencias y estrategias según el tipo de subasta .

```
#Datos de los compradores
#Comprador 1

#Id del agente
id_agente_b1=Buyer0
#Nivel de aspiracion
aspiracion_b1=2700
#Utilidad de reserva
utilidad_reserva_b1=1200
#Incremento
incremento_b1=3
#tipo_agente
tipo_buyer=buyer
#Conversaciones habilitadas
conversations_buyer=1
#Preferencias
preferencia1_b1=Verduras
preferencia2_b1=Ropa

#Estrategias
#Estrategias subasta americana
estrategia_americana1_b1=TitforTatAscendente
#Estrategias subasta inglesa1
estrategia_inglesa11_b1=RandomAscendenteInglesa
estrategia_inglesa12_b1=TemporalAscendenteInglesa
#Estrategias subasta inglesa2
estrategia_inglesa21_b1=TemporalDescendenteInglesa
estrategia_inglesa22_b1=TitforTatDescendenteInglesa
#Estrategias subasta sobrecerrado
estrategia_sobrecerrado1_b1=RandomAscendenteSobreCerrado
estrategia_sobrecerrado2_b1=AscendenteSobreCerrado
#Estrategias subasta libre
estrategia_libre1_b1=TitforTatAscendente
```

Ilustración 10. Ejemplo de Datos del Comprador.

c) Datos del vendedor: ID del Vendedor que al igual que en el caso del Comprador debe ser único e irrepetible, el precio base del producto que se desea vender, el timeout es decir el tiempo que esperará el vendedor para recibir respuesta por parte de los

compradores (variable estática) expresado en segundos, conversaciones habilitadas (variable estática) que corresponden al número de iteraciones máximas posibles, tipo de agente en este caso seller (Vendedor), ID de la mesas a la que corresponde el vendedor y el producto que se desea vender

```
#Datos de los Vendedores
#Vendedor 1

#Id del agente
id_agente_s1=Seller1

#Precio Base
precio_base_s1=1000

#Timeout variable estática
timeout_seller=10

#conversaciones habilitadas variable estática
conversations_seller=15

#tipo agente variable estática
tipo_seller=seller


#Producto a vender
producto_s1=Verduras
```

Ilustración 11. Ejemplo de Datos del Vendedor.

d) Datos del producto: ID del producto, nombre del producto, descripción del producto, cantidad del producto.

```
##Productos
#Producto 1

#Id del producto
id_producto_1=1

#nombre del producto
nombre_1=Verduras

#Descripcion
descripcion_1=Lotes Variado de Verduras

#Cantidad
cantidad_1=3
```

Ilustración 12. Ejemplo de Datos del Producto.

e) Datos del protocolo: ID del protocolo, tipo de subasta corresponde al nombre de subasta a ser empleado, estado en el que inicia el protocolo, iteraciones es decir el número máximo de iteraciones permitido según el tipo de subasta.

```
#Datos de los Protocolos
#Protocolo 1

#Id del protocolo
id_protocolo1=1

#Tipo Subasta
tipo_subasta_1=libre

#estado
estado_1=esperando

#iteracciones
iteracciones_1=6
```

Ilustración 13. Ejemplo de Datos del Protocolo.

f) Datos de la mesa: ID de la mesa, ID del vendedor dueño de la mesa, ID del protocolo a utilizar en la mesa.

```
#Datos de las Mesas
#Mesa 1

#Id de la mesa
id_mesa1=1

#Id del vendedor dueño de la mesa
Id_vendedor_mesa1=Seller1

#Id del protocolo que utilizara la mesa
Id_protocolo_mesa1=1
```

Ilustración 14. Ejemplo de Datos de la Mesa.

4.1.3. Tareas de inicialización

Como se mencionó anteriormente las tareas de inicialización son necesarias para el correcto funcionamiento del prototipo de mercado virtual, las tareas son las siguientes:

a) La plataforma Magentix2 utiliza log4j como una instalación de registro, ya que su velocidad y flexibilidad permite que las instrucciones de registro permanezcan en el código enviado mientras que le da al usuario la capacidad de habilitar el registro en tiempo de ejecución sin modificar ninguno de los binarios de la aplicación (GTI, 2015). Esto se realiza por medio de la configuración del fichero de loggin , se inicia el Logger dentro del método Main del prototipo.

```
// Inicializando Tareas
//Configure
DOMConfigurator.configure("configuration/loggin.xml");
//Logger
logger= Logger.getLogger(Main.class);
```

Ilustración 15. Configuración y establecimiento del Logger.

b) Conexión al Qpid Broker, se realiza la conexión al Qpid Broker para garantizar que pueda existir comunicación entre los agentes, esta conexión debe ser establecida antes de iniciar los agentes.

```
//Conexión de los agentes con  Qpid Broker
AgentsConnection.connect();
```

Ilustración 16. Conexión Qpid Broker.

4.1.4. Agentes vendedores

Los agentes vendedores (Seller) son los que cumplirán con el rol de Iniciador dentro del protocolo de interacción ContractNet y el rol de Vendedor dentro del mercado virtual. Su funcionamiento depende los siguientes elementos:

4.1.4.1. Variables de la clase Seller (Vendedor)

Como se señaló anteriormente el Vendedor posee los siguientes parámetros dentro del fichero properties: ID, precio base, timeout, conversaciones posibles, tipo de agente y producto a vender, dentro de la clase Java que representa al vendedor estos parámetros, tienen su variable correspondiente y métodos de Set y Get. La clase Java también utiliza otras variables imprescindibles para su correcto funcionamiento, son las siguientes: variable auxiliar que representan el tiempo en el estado BEGIN y que es utilizada para una de las estrategias de negociación del Comprador, variables utilizadas para representar el estado del protocolo, variables auxiliares que se utilizan como contadores y las variables necesarias para crear los reportes gráficos.

```
//Tiempo en el inicio
    static long ttemp;

//Precio Base
    final double precio=precioBase;

//Lista de Productos
static List<Producto> productos;

//Estados
static String espera="EN ESPERA DE COMENZAR";
static String ejecutandose="EN EJECUCIÓN";

//Iteracciones
private static List<Integer> iteracciones;
```

```
//Contador
int contador=0;
int it=0;

//Grafica num_propuestas
//ArrayList de propuestas aceptadas por los agentes
static ArrayList<ACLMessage> propuestas_aceptadas=new ArrayList<
//ArrayList de nombre de los agentes
static ArrayList<String> nombres_agentes=new ArrayList<String>()
//Variable auxiliar
String aux_nombre;
//Variable estática que obtiene el número de mesas
static int vendedores=Main.num_mesas;
//Número de veces que los vendedores llegan al estado final
static int finals=vendedores*2;
//Variable estática que cuenta el número de vendedores
static int contador_vendedores=0;
```

Ilustración 17. Variables de la clase Java de Vendedor.

4.1.4.2. Establecimiento del ID del agente Vendedor

```
//Id del agente Seller
public Seller(AgentID aid) throws Exception {
    super(aid);
}
```

Ilustración 18. Extracto de código para establecer el ID del Agente.

4.1.4.3. Execute del agente Vendedor

```
//Execute del agente Seller
protected void execution(CProcessor myProcessor, ACLMessage welcomeMessage) {
    //Mensaje que se recibe
    ACLMessage msg;


    //Se realiza la extension del FIPA Initiator
    class myFIPA_CONTRACTNET extends Initiator {
```

Ilustración 19. Extracto de código del execute.

La parte que corresponde al execute del vendedor, debe realizar una extensión a la clase que representa el rol del Iniciador en el protocolo de interacción de Contract-Net, y deben ser implementados los siguientes métodos:

a) doBegin, método que se utiliza cuando el agente alcanza el estado inicio (BEGIN) y que es utilizado para mostrar por consola el timeout, deadline y estado del agente. De igual manera se utiliza para obtener el tiempo en milisegundos al iniciar y para añadir los nombres de los agentes al ArrayList de nombres.

```
protected void doBegin(CProcessor myProcessor, ACLMessage msg) {
    myProcessor.getInternalData().put("InitialMessage", msg);
    ttemp=System.currentTimeMillis();
    it=iteracciones.get(contador);
    System.out.println("Timeout: "+ timeout + "\n");
    System.out.println("Deadline: "+ deadline+ "\n");
    setEstado(espera);
    System.out.println("ESTADO DEL PROTOCOLO: "+ estado + "\n");
    aux_nombre=getName();
    if (nombres_agentes.size()>1) {
        for(int j=0; j<nombres_agentes.size();j++)
        {
            if (nombres_agentes.get(j).equals(aux_nombre)) {
                break;

            }
            else{
                nombres_agentes.add(aux_nombre);
            }
        }

    }
    else{
        nombres_agentes.add(aux_nombre);
    }

}
```

Ilustración 20. Extracto de código que representa DoBegin.

b) doEvaluateProposals, es el método en el cual se evalúan las propuestas recibidas por parte de los Compradores, se elige la propuesta que será aceptada y se rechazan las demás. En este caso se elige la propuesta del comprador que ofrezca la cantidad más alta y se añade a las propuestas aceptadas.

```
@Override
protected void doEvaluateProposals(CProcessor myProcessor,
        ArrayList<ACLMessage> proposes,
        ArrayList<ACLMessage> acceptances,
        ArrayList<ACLMessage> rejections) {

    setEstado(ejecutandose);
    System.out.println("ESTADO DEL PROTOCOLO: "+ estado + "\n");

    double min = precio;
    double proposal;
    DecimalFormat format = new DecimalFormat("#.##");

    System.out.println("Propuestas recibidas por "+ getName());
    for(int i=0; i < proposes.size(); i++){
        proposal = Double.parseDouble(proposes.get(i).getContent());
        int  aux=i+1;
        System.out.println("Propuesta "+ aux  +" "+  format.format(proposal)+ " enviada por "+ proposes.get(i).getSender().name +" recibida por "+ getName());
    }
    //Revisa que las propuestas sean mayor que el precio base
    int index = -1;
    for(int i=0; i < proposes.size(); i++){
        if(Double.parseDouble(proposes.get(i).getContent()) > min){
            min = Double.parseDouble(proposes.get(i).getContent());
            index = i;
        }
    }
    System.out.println(getName()+": Quiero la mejor propuesta para mi producto");
    for(int i=0; i < proposes.size(); i++){
        if(i == index){ // accept the highest proposal
            proposal= Double.parseDouble(proposes.get(i).getContent());

            System.out.println(getName()+":Acepto la propuesta de "+proposes.get(i).getSender().name + " " +"Cantidad:" +format.format(proposal));
            ACLMessage accept = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
            accept.setContent("Acepto tu propuesta");
            accept.setReceiver(proposes.get(i).getSender());
            accept.setSender(getAid());
            accept.setProtocol("fipa-contract-net");
            acceptances.add(accept);
            propuestas_aceptadas.add(proposes.get(i));
        }
        else{ // reject the rest
            proposal= Double.parseDouble(proposes.get(i).getContent());
            System.out.println(getName()+": Rechazo la propuesta de  "+proposes.get(i).getSender().name +" " +"por:" +format.format(proposal));
            ACLMessage reject = new ACLMessage(ACLMessage.REJECT_PROPOSAL);
            reject.setContent("No  me gusta tu propuesta, la rechazo");
            reject.setReceiver(proposes.get(i).getSender());
            reject.setSender(getAid());
            reject.setProtocol("fipa-contract-net");
            rejections.add(reject);
        }
    }
```

Ilustración 21. Extracto de código de DoEvaluateProposals.

c) doReceiveInform, método en el cual se recibe un mensaje enviado por el Comprador seleccionado.

```
protected void doReceiveInform(CProcessor myProcessor,
        ACLMessage msg) {
    // método que recibe el resultado de la propuesta aceptada
    System.out.println(getName()+": He recibido el siguiente mensaje : "+msg.getContent());


}
```

Ilustración 22. Extracto de código de doReceiveInform.

4.1.4.4. Iniciar conversación

Para establecer una conversación con los agentes del rol Comprador es necesario seguir los siguientes pasos:

a) Establecer el mensaje que se enviara y los agentes que lo recibirán, en este caso los Compradores que pertenezcan al universo del agente.

b) Crear y configurar el CFactory del agente con los datos obtenidos del fichero properties.

c) Añadir el CFactory como Iniciador.

d) Crear la conversación síncrona.

e) Terminar el CProcesor una vez que el agente haya finalizado.

```
//Se añaden los datos que debe llevar el mensaje
msg = new ACLMessage(ACLMessage.CFP);
for (int i = 0; i <compradores; i++) {
msg.addReceiver(new AgentID("Buyer" + i));
}
msg.setContent("Cuanto quieres gastar en mi producto?");
msg.setSender(getAid());
msg.setProtocol("fipa-contract-net");

//Se crear el filtro

MessageFilter filter;
filter = new MessageFilter("performative =CFP");

//Se crea el CFactory y se configura

CFactory talk = new myFIPA_CONTRACTNET().newFactory("TALK", null, msg,
        conversaciones, myProcessor.getMyAgent(),conversaciones,deadline,timeout);

//Se añade el CFactory como Initiator
this.addFactoryAsInitiator(talk);

//Se crea la conversacion Sincrona
System.out.println("Le hare propuestas a los compradores");
myProcessor.createSyncConversation(msg);

//Se termina el CProcessor
myProcessor.ShutdownAgent();
```

Ilustración 23. Extracto de código para iniciar la conversación.

4.1.4.5. Reportes Gráficos

Con el fin de crear un ambiente más amigable para el usuario, se crearon reportes gráficos que permiten visualizar la salida de la ejecución de cada archivo de configuración (fichero properties), de forma que la compresión de esta resulte más fácil.

Para crear los reportes Gráficos se utilizaron las librerias JFreeChart y JCommon.

La librería JFreeChart ofrece la posibilidad de crear varios tipos de gráficas con procedimientos sencillos Presenta un diseño flexible y de fácil compresión, además permite la exportación de fichero en formatos JPEG y PNG.

JCommon,es una librería de clases utilizada en JFreeChart , ya que permite facilitar la configuración de dependencias y el uso de hilos.

Cabe recalcar que en este proyecto se utilizaron dos tipos de gráficas proporcionados por estas librerías, gráficas de barras en 3D y gráficas de pastel en 3D.

Para crear los reportes gráficos es necesario utilizar DataSets en los cuales se agreguen los datos a visualizar, crear una instancia del tipo de gráfico que se quiera utilizar y posteriormente añadir esa instancia a una ventana.

```
//Grafica num propuestas
public void SetGrafica_propuestas_aceptadas(ArrayList<ACLMessage> acceptados)
{
    DefaultCategoryDataset dataset= new DefaultCategoryDataset();
    Double propose;
    String nombre;
    String vend=String.valueOf(vendedores);
    for (int i = 0; i < acceptados.size(); i++) {
        propose=Double.valueOf(acceptados.get(i).getContent());
        nombre=String.valueOf(acceptados.get(i).getReceiver());
        dataset.addValue(propose, nombre,vend);

    }

    JFreeChart grafica=ChartFactory.createBarChart3D("Propuestas aceptadas", "Número de Mesas",
    ChartPanel contenedor= new ChartPanel(grafica);
    JFrame ventana=new JFrame(" Propuestas aceptadas");
    ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    ventana.add(contenedor);
    ventana.setSize(550,350);
    ventana.setVisible(true);
    ventana.setLocation(600, 15);
}
```

Ilustración 24. Extracto de código del reporte gráfico-propuestas aceptadas.

Como podemos visualizar en la imagen se muestran las propuestas aceptadas, se obtiene la estrategia utilizada en cada propuesta, el monto de esta y el nombre del vendedor que la acepto.

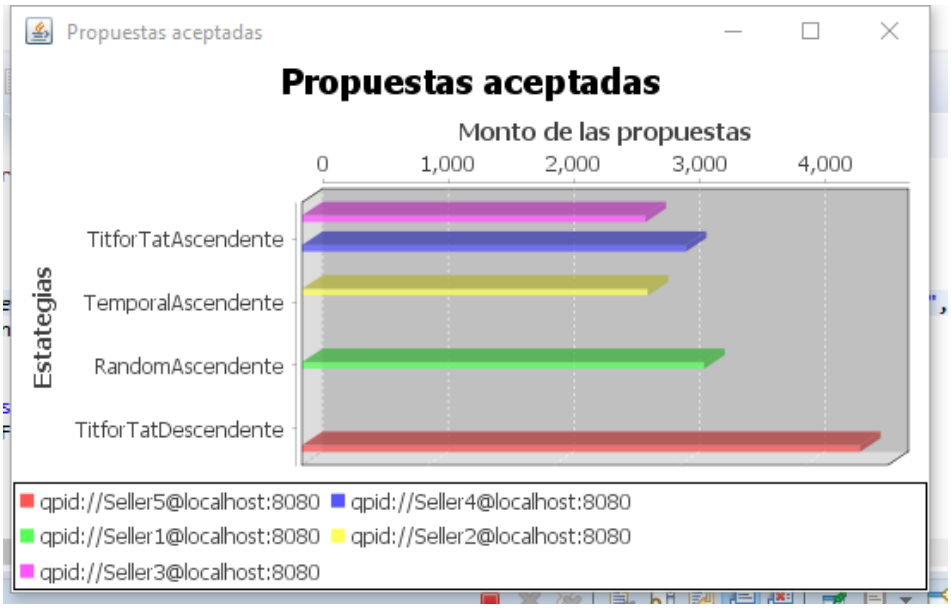

Ilustración 25. Reporte Gráfico de las propuestas aceptadas.

4.1.5. Agentes Compradores

Los agentes compradores (Buyer) son los que cumplirán con el rol de Participante dentro del protocolo de interacción ContractNet y el rol de Comprador dentro del mercado virtual. Su funcionamiento depende los siguientes elementos:

4.1.5.1. Variables de la clase Comprador (Buyer)

Como se mencionó anteriormente en el fichero properties el comprador posee los siguientes parámetros: ID, aspiración, utilidad de reserva, incremento, tipo de agente, conversaciones habilitadas, preferencias y estrategias de negociación. Estos parámetros tienen su variable correspondiente y métodos Set y Get en la clase que representa al comprador.

También existen otras variables necesarias para su funcionamiento, son las siguientes: nivel de aspiración corresponde a la propuesta que se enviará al vendedor, subasta representa el tipo de protocolo de negociación que utiliza el vendedor, timeout del agente vendedor este es utilizado en las estrategias de negociación, algunas variables de tipo String que representan las estrategias de negociación, contadores utilizados para saber la cantidad de veces que fueron empleados los protocolos de negociación y las estrategias.

4.1.5.2. Establecimiento del ID

Para establecer el ID del Comprador el procedimiento es el mismo que el utilizado en el Vendedor.

```
//Id del agente Buyer
public Buyer(AgentID aid) throws Exception {
    super(aid);
}
```

Ilustración 26. Extracto de código establecimiento del ID Comprador.

4.1.5.3. Execute del Agente Comprador (Buyer)

En el fragmento que corresponde al execute del comprador, se debe realizar una extensión a la clase que representa el rol del Participante en el protocolo de interacción de Contract-Net, y deben ser implementados los siguientes métodos:

a) doReceiveSolicit, es el método que recibe la solicitud por parte de los Vendedores. En este método se agrega el nombre de cada Vendedor a un ArrayList, se accede al número de iteraciones(propuestas) que los compradores pueden enviar a los

vendedores (valor establecido en el protocolo) y se decide si aceptar o no la solicitud de respuesta de cada vendedor, dependiendo de si el producto que vende pertenece o no a la lista de preferencias de cada comprador.

b) doTask, envía un mensaje al vendedor que ha aceptado una propuesta del comprador, este método solo es utilizado para enviar información.

```
protected String doTask(CProcessor myProcessor,
        ACLMessage solicitMessage) {
    // no realiza ninguna acción solo informa
    System.out.println(getName()+":Mi propuesta fue aceptada!");
    return "SEND_INFORM";
}
```

Ilustración 27. Extracto de código doTask.

c) doSendInform, método que informa el resultado de la ejecución de doTask.

d) doFinal, este método se ejecuta cuando el agente ha alcanzado el estado FINAL, y es el encargado de llamar a las funciones que crean los reportes gráficos.

e) doSendProposal, método encargado de enviar propuestas a los vendedores.

Es el método más importante y complejo dentro del agente Comprador, su funcionamiento se puede dividir en las siguientes etapas:

I. doSendProposal: es la declaración del método como tal, en esta se realiza una llamada a la función update, se añade el valor de la propuesta y todos los datos necesarios de la misma (agente que la envía, agente que la recibirá entre otros), se imprime la propuesta y se añade la misma a las propuestas enviadas.

```
protected void doSendProposal(CProcessor myProcessor,
        ACLMessage messageToSend)
{
    update();
    messageToSend.setSender(getAid());
    messageToSend.setReceiver(myProcessor.getLastReceivedMessage().getSender());
    messageToSend.setContent(String.valueOf(aspiracion));
    messageToSend.setPerformative(ACLMessage.PROPOSE);
    messageToSend.setProtocol("fipa-contract-net");
    double proposal= Double.parseDouble(messageToSend.getContent());
    System.out.println(getName()+":Propongo gastar "+ format.format(proposal)+" en tu producto");
    AddProposal(messageToSend);
}
```

Ilustración 28. Extracto de código SendProposal

II. Función update: en la función update se conoce que tipo de protocolo de negociación fue asignado a cada vendedor, y de acuerdo al funcionamiento del protocolo se elige la estrategia de negociación a utilizar.

Protocolos de negociación utilizados

Como se mencióno en los primeros capitulos del presente proyecto, el protocolo de negociación que se utilizara es la subasta.

Podemos definir subasta como, la venta al público de un servicio o bien , cuya adjudicación es hecha al mejor postor.

Los tipos de subastas utilizados fueron los siguientes:

1. Subasta de sobrecerrado al primer precio: Cada participante presenta su puja en un "sobrecerrado" dentro del plazo que dure la subasta. Una vez que el plazo ha caducado, se abren todos los sobres y al comprador que ofreció la mejor oferta se le adjudica el producto.
   Estrategias de negociación utilizadas:
   - TemporalAscendente.
   - TemporalDescendente.
   - RandomAscendente.
   - RandomDescendente.

2. Subasta ascendente Inglesa: en este tipo de subasta es posible conocer la cantidad ofrecida por los competidores, si se desea. Las ofertas parten de un precio mínimo, los participantes pujan de manera progesiva, ofreciendo cada vez una cantidad más alta , hasta que la subasta termine o hasta que alcancen su oferta máxima.
   Estrategias de negociación utilizadas:
   - TemporalAscendente.
   - TitforTatAscendente.
   - RandomAscendente.

3. Subasta descendente Inglesa: funciona de manera similar a la subasta anterior, el cambio radica en que la cantidad a ofrecer disminuye en lugar de aumentar.
   Estrategias de negociación utilizadas:
   - TemporalDescendente.

- TitforTatDescendente.
- RandomDescendente.

4. Subasta Americana: todos pagan el valor de la puja ,el que realiza la oferta mayor es al que se le adjudica el producto o servicio subastado, esta subasta es de tipo ascendente.

   Estrategias de negociación utilizadas:

   - TemporalAscendente
   - TitforTatAscendente
   - RandomAscendente

5. Subasta Libre: subasta en la cual los potenciales compradores pueden ofrecer la cantidad que deseen, sin ningún otro tipo de restricción.

   Estrategias de negociación utilizadas:

   - TemporalAscendente.
   - TemporalDescendente.
   - TitforTatAscendente.
   - TitforTatDescendente.
   - RandomAscendente.
   - RandomDescendente.

Estrategias de negociación

1. Temporal ascendente: el objetivo de esta estrategia es aumentar la propuesta que se enviará a medida que avanzan las rondas de negociación. Se realiza la estrategia de negociación temporal utilizando los valores de utilidad base, instante de tiempo, aspiración y el factor beta para variar el nivel de aspiración.

Al obtener el valor de la estrategia temporal, se calcula el valor a aumentar utilizando el incremento establecido en el fichero properties y esta cantidad se le suma a la cantidad inicial(estrategia temporal).

```
Algoritmo TemporalAscendente
    //Temporal  ascendente-- BOULWARE
    Imprimir "Método Temporal Ascendente"
    //Obtiene el tiempo en el inicio
    time=Seller.GetDead();
    //Resta el tiempo actual inicio para saber de que segundo se trata
    temp=System.currentTimeMillis()-time;
    //Se imprime el milisegundo en el que nos encontramos
    Imprimir "Tiempo en milisegundos :"+temp ;
    //Se realiza la estrategia de negociación temporal
    tempa=1 - (1 - this.utilidadr)*Math.pow( (temp),1.0/this.beta);
    //Se imprime los datos del agentes
    Imprimir "Precio base: "+precio;
    Imprimir "Aspiracion: "+as;
    Imprimir "Utilidad neta: "+utilidadr ;
    Imprimir "Propuesta inicial: "+tempa;
    //Basandose en el incremento establecido por el usuario
    //se calcula el valor que en el que se incrementara la propuesta inicial
    porcentaje=tempa*incremento/100;//Se realiza el calculo
    Imprimir "Incremento: "+incremento;
    Imprimir "Porcentaje: "+porcentaje;
    //Se añade el valor calculado a la propuesta inicial
    tempe=tempa+porcentaje;
    Imprimir "Resultado: "+tempe;
    //Se comprueba que la propuesta sea mayor que el precio base
    SI (tempe<precio) Entonces
        //Si el precio es mayor que la utilidad,la propuesta sera la aspiración
        SI(precio>utilidadr) Entonces
            aspiracion=as;
        FINSI
        //Si el precio es menor que la utilidad,la propuesta sera la utilidad
        SI(precio<utilidadr)Entonces
            aspiracion=utilidadr;
        FINSI
    FINSI
    //Se comprueba que la propuesta no sea menor que la utilidad
    Si (tempe<utilidadr) Entonces
        aspiracion=utilidadr;
        FINSI
    //Se comprueba la propuesta no se mayor que la aspiración
    Si(tempe>as)
    //Si es mayor que la aspiración , la aspiracion pasa a ser la propuesta
        aspiracion=as;
    SiNo
    //Sino el valor continua sin cambios
    aspiracion=tempe;
FINSI
FinAlgoritmo
```

Ilustración 29. Pseudocódigo Temporal Ascendente

2. Temporal Descendente: esta estrategia presenta un comportamiento similar a la anterior, lo que varía es el cálculo del tiempo ya que este se realiza restando el tiempo de duración de la subasta con el tiempo en el momento de la negociación, igual que en la estrategia anterior se utilizan los datos del agente y se va decrementando la cantidad a proponer de acuerdo con el incremento establecido en el fichero properties.

```
Algoritmo  TemporalDescendente
    //Temporal descendente
    t<-true
    Imprimir "Método Temporal descendente";
    //Se obtiene el instante de tiempo en el inicio
    time=Seller.GetDead();
    //Se suma el tiempo en el inicio más el timeout establecido por el usuario
    aux=time+timeout;
    //Se resta el tiempo actual para saber en que segundo se encuentra
    temp=aux-System.currentTimeMillis();
    Imprimir"Tiempo en milisegundos :"+temp;
    //Se hace el cálculo de la propuesta inicial
    tempa=1 - (1 - this.utilidadr)*Math.pow( (temp),1.0/this.beta);
    Imprimir "Precio base: "+precio;
    Imprimir "Aspiracion: "+as;
    Imprimir "Utilidad neta: "+utilidadr;
    Imprimir"Propuesta inicial: "+tempa;
    //Se comprueba que la aspiración no sea menor que la propuesta inicial
    Si(as<=tempa)Entonces
         tempa=as; Fin si
    Si (t==true) Entonces
    //Se comprueba si existen propuestas anteriores
    Si(proposess.size()>0) Entonces
    //Se establece el porcentaje a decrementar usando la propuesta inicial
    porcentaje=(tempa*incremento)/100;
    Imprimir "Porcentaje: "+porcentaje;
    //Se decrementa el valor del porcentaje
    tempb=tempa-porcentaje;
    Imprimir"Decremento: "+incremento;
    Imprimir"Resultado: "+tempb;
    //Se comprueba que la cantidad a proponer no sea menor que el precio base
    Si (tempb<precio) Entonces
        SI (precio<utilidadr) Entonces
            aspiracion=utilidadr;
        Sino
        aspiracion=as;
        Fin Si
    //Se comprueba que la cantidad a proponer no sea menor que la utilidad base
    Si(tempb<utilidadr) Entonces
    //Si lo es la propuesta pasa a ser la utilidad
        aspiracion=utilidadr; FinSi
    //Comprobar que la propuesta no sea mayor que la aspiración
    Si (tempb>as)
     aspiracion=as;
    Sino
    aspiracion=tempb;FinSi
     Fin Si
    SiNo //Sino se han realizado propuestas
    Si(tempa<precio) Entonces

        Si(precio<utilidadr) Entonces
        aspiracion=utilidadr;
        Sino
        aspiracion=as;FinSi
    //Se comprueba que la cantidad a proponer no sea menor que la utilidad base
    Si(tempa<utilidadr) Entonces
    //Si lo es la propuesta pasa a ser la utilidad
        aspiracion=utilidadr; FinSi
    Si(tempa>as) Entonces
    tempa=as;
    Sino
    aspiracion=tempa;FinSi
    Fin Si
Fin Si
```

Ilustración 30. Pseudocódigo Temporal Descendente

3. TitforTat Ascendente: consiste en conceder lo que creemos que el otro ha concedido. Al iniciar se comprueba si ya existen propuestas realizadas, sino se realiza la estrategia temporal normal. Si existen propuestas, se obtiene el contenido de estas, se localiza la propuesta mayor y se comprueba que dicha propuesta sea mayor que la utilidad base del agente, si es así se calcula la cantidad que se aumentará de acuerdo con el incremento establecido en el fichero properties y se le suma esta cantidad a la propuesta mayor, siempre que esta no sobrepase la aspiración del agente si la sobrepasa, la propuesta es igual a la aspiración.

```
Algoritmo  TitforTatAscendente
    //Tit for tat Ascendente
    Imprimir "Método Tit for Tat Ascendente";
    t<-false;
    time=Seller.GetDead();
    temp=System.currentTimeMillis()-time;
    //Se comprueba que existan propuestas realizadas anteriormente
    Si (proposess.size()>0) Entonces
        Imprimir "Propuestas hechas en tif fot tar ascendente "+proposess.size();
        index <- -1;
        //Se recorren las propuestas realizadas anteriormente
        Para i<-0 Hasta i < proposess.size() Con Paso 1 Hacer
        sel1=String.valueOf(proposess.get(i).getReceiver());
        Imprimir"Seller: "+vendedor;
        Imprimir "Seller que recibira la propuesta: " + proposess.get(i).getReceiver();
        //Si la propuesta fue enviada por ese comprador
        Si(sel.equals(sel1))Entonces
            t=true;
            //Se obtiene el contenido de las propuestas
            tempa = Double.parseDouble(proposess.get(i).getContent());
            //Propuestas hechas anteriormente
            Imprimir  "Propuestas tempa: "+tempa;
            //Se comprueba que la propuesta realizada sea mayor que la utilidad base
        Si(tempa>utilidadr)Entonces
            //Se calcula el resultado de restar la propuesta - menos la utilidad base
            tempb=tempa-utilidadr;
            //Se escoje la propuesta mayor
            Si(tempb >tempd)Entonces
            //Se almacena el index, la propuesta y se establece la resta como la mayor
            index=i;
            tempc=Double.parseDouble(proposess.get(index).getContent());
            tempd=tempb;
            FinSi
        FinSi
        finSi
        FinPara
        Imprimir "Porcentaje: " +porcentaje;
        Imprimir "Precio base: "+precio;
        Imprimir "Aspiracion: "+as;
        Imprimir "Utilidad neta: "+utilidadr;
        //Se le agrega a la cantidad base
        tempe=tempc+porcentaje;
        //Se comprueba que la propuesta sea mayor que el precio base
        Si (tempe<precio)Entonces
            Si(tempe<utilidadr) Entonces
            aspiracion=utilidadr;FinSi
            SI (tempe>utilidadr && tempe<as) entonces
             aspiracion=as; FinSi
        FinSi
        Si(tempe<utilidadr) Entonces
         aspiracion=utilidadr; FinSi
        //Si la propuestas es mayor que la aspiración se iguala a esta
        Si(tempe>as) Entonces
            aspiracion=as;
        Sino
        aspiracion=tempe;
        FinSi
        FinSi
```

```
    //Si es false
    Sino
    time=Seller.GetDead();
    temp=System.currentTimeMillis()-time;
    tempe = 1 - (1 - this.utilidadr)*Math.pow( (temp),1.0/this.beta);
    Si(tempe<precio) Entonces
        Si(tempe<utilidadr) Entonces
        aspiracion=utilidadr;FinSi
        Si(tempe>utilidadr && tempe<as) Entonces
         aspiracion=as;
        FinSi
    FinSi
    Si (tempe<utilidadr) Entonces
     aspiracion=utilidadr;
    FinSi
    //Si la propuesta es mayor que la aspiración se iguala a esta
    Si (tempe>as) Entonces
     aspiracion=as;
    Sino
     aspiracion=tempe;
    FinSi
FinSi
//Sino se realiza la estrategia temporal básica
Si(proposess.size()<0) Entonces
Imprimir "Precio base: "+precio;
Imprimir "Aspiracion: "+as;
Imprimir "Utilidad neta: "+utilidadr;
time=Seller.GetDead();
temp=System.currentTimeMillis()-time;
tempe = 1 - (1 - this.utilidadr)*Math.pow( (temp),1.0/this.beta);
Si(tempe<precio) Entonces
    Si(precio>utilidadr) Entonces
    tempe=as;FinSi
    Si (tempe<utilidadr && tempe<as) Entonces
    tempe=as;
    Fin Si
FinSi
Si (tempe>as) entonces
 aspiracion=as; FinSi
Si(tempe<utilidadr) Entonces
aspiracion=utilidadr;
Sino
aspiracion=tempe;
FinSi
 Fin Si
 FinAlgoritmo
```

Ilustración 31. Pseudocódigo TitforTatAscendente

4. TitforTatDescendente: Presenta un comportamiento análogo a la estrategia anterior, variando que en lugar de buscar la propuesta mayor se busca la propuesta menor, para calcular el decremento de esta con el incremento establecido en el fichero properties.

```
Algoritmo TitforTatDescendente
    //Tit for tat Descendente
    Imprimir "Método Tit for Tat Descendente";
    t<-false;
    tempd=5000;
    time=Seller.GetDead();
    aux=time+60000;
    temp=aux-System.currentTimeMillis();
    //Se obtiene el instante de tiempo actual de time out a -->1 decrementandolo
    Imprimir "Tiempo en milisegundos :"+temp;
    //se calcula la propuesta inicial
    tempa=1 - (1 - this.utilidadr)*Math.pow( (temp),1.0/this.beta);
    Imprimir "Propuesta inicial: "+tempa;
    tempc=0;
    //se comprueba si se han realizado propuestas anteriores
    Si (proposess.size()>0) Entonces
    Imprimir "Propuestas hechas en update descendente: "+proposess.size();
    Imprimir "Aspiracion: "+as;
    index = -1;
    //Se recorren las propuestas realizadas anteriormente
    Para i<-0 Hasta i < proposess.size() Con Paso 1 Hacer
    sel1=String.valueOf(proposess.get(i).getReceiver());
    Imprimir "Seller: "+sel;
    Imprimir "Seller que recibira la propuesta: " + sel1 + "\n";
    Si (sel.equals(sel1)) entonces
    t=true;
    //Se obtiene el contenido de la propuesta
    tempb=Double.parseDouble(proposess.get(i).getContent());
    //Si el contenido es menor que el nivel de aspiración
    Si (tempb<as) entonces
    //Propuestas
    Imprimir "Temporal b: "+tempb;
    //Restar la aspiració menos la propuesta
    //Se almacena la menor
    tempe=as-temp;
        Si (tempe<=tempd) entonces
        //Se almancena el index, la resta de la operación y se establece el valor men
        index=i;
        tempd=as-tempb;
        tempc=Double.parseDouble(proposess.get(i).getContent());
         Fin Si
    Fin Si
    Fin Si
   Fin Para
   Si(t==true) Entonces
   //Si la propuesta es mayor que 0 y la resta es mayor que 0
   Si (tempc>0 && tempd>0) Entonces
   //Propuesta inicial menor
   Imprimir "Propuesta inicial menor: "+ tempc;
   //Se cálcula el valor a decrementar
   porcentaje=(tempc*incremento)/100;
   Imprimir "Decremento: "+incremento;
   Imprimir "Porcentaje: "+ porcentaje;
   Imprimir "Utilidad: "+utilidadr;
   Imprimir "Aspiracion: "+as;
   Imprimir "Precio base: "+precio;
   //Se realiza el decremento
   tempe=tempc-porcentaje;
   //Se comprueba que la propuesta sea mayor que el precio base
   Si (tempe<precio) Entonces
       Si(tempe<utilidadr) Entonces
           aspiracion=utilidadr;FinSi
       Si(tempe>utilidadr && tempe<as) Entonces
           aspiracion=as;FinSi

   FinSi
```

```
//Comprobar que no se proponga una cantidad menor a la utilidad de reserva
Si (tempe<utilidadr) Entonces
    aspiracion=utilidadr;
Sino
    aspiracion=tempe;
FinSi
//Se comprueba que la propuesta no sea mayor que la utilidad
Si (tempe>as) Entonces
    aspiracion=as;FinSi
Fin Si
Sino
time=Seller.GetDead();
temp=System.currentTimeMillis()-time;
aspiracion = 1 - (1 - this.utilidadr)*Math.pow( (temp),1.0/beta);
FinSi
Sino
//Sino se realiza la estrategia temporal descendente con decremento
porcentaje=(tempa*incremento)/100;
Imprimir "Porcentaje: "+porcentaje;
tempb=tempa-porcentaje;
Imprimir "Decremento: "+incremento;
Imprimir "Resultado: "+tempb;
Si(tempb<precio) Entonces
    Si(tempb<utilidadr)Entonces
        aspiracion=utilidadr;FinSi
    Si(tempb>utilidadr && tempe<as)Entonces
        aspiracion=as;FinSi
FinSi
Si(tempb<utilidadr) Entonces
    aspiracion=utilidadr;FinSi
//Si la propuestas es mayor que la aspiración se iguala a esta
Si(tempb>as) Entonces
 aspiracion=as;
Sino
 aspiracion=tempb;
FinSi
FinSi
//sino se han realizado propuestas anteriores se realiza temporal descendente sin decremento
Si(proposess.size()<0) Entonces
Si (tempa<precio)  Entonces
    Si(precio>utilidadr)Entonces
    tempa=as;
    FinSi
```

Ilustración 32. Pseudocódigo TitforTatDescendente

5. RandomAscendente: en esta estrategia se resta la aspiración de la utilidad neta, del valor que es obtenido en esta resta se produce un Random (entre 1 y este valor), a la cantidad obtenida en el Random, se le calcula el porcentaje a aumentar utilizando el incremento del agente a este resultado lo llamaremos porcentaje, el porcentaje a aumentar es sumado a la utilidad base.

```
Algoritmo RandomAscendente
    Imprimir "Método Random Ascendente";
    // Se resta la aspiracion - la utilidad base
    tempa=as-utilidadr;
    ran <- Random();
    //Se obtiene un valor entre 1 y el resultado de la resta anterior
    result<-1 + (ran * (tempa - 1));
    Imprimir  "Resultado random: "+result;
    //Se calcula el valor a aumentar del resultado random
    porcentaje=result*incremento/100;
    Imprimir "Precio base: "+precio;
    Imprimir "Aspiracion: "+as;
    Imprimir "Utilidad neta: "+utilidadr ;
    Imprimir"Incremento: "+incremento;
    Imprimir "Porcentaje: "+porcentaje;
    //Se le suma el valor a la utilidad neta
    tempb=utilidadr+porcentaje;
    //Se comprueba que la propuesta sea mayor que el precio base
    Si(tempb<precio) Entonces
    Si(precio<utilidadr) Entonces
        aspiracion=utilidadr;FinSi

    Si(precio>utilidadr) Entonces
        aspiracion=as;FinSi
    FinSi
    //Se comprueba que la propuesta sea mayor que la utilidad
    Si(tempb<utilidadr) entonces
        aspiracion=utilidadr; FinSi
    //Se comprueba que la propuesta no sea mayor que la aspiracion
    Si(tempb>as) entonces
     aspiracion=as;
    Sino
    aspiracion=tempb;
    FinSi
    Imprimir "Propuesta ascendente: "+tempb;

FinAlgoritmo
```

Ilustración 33. Pseudocódigo RandomAscendente

6. RandomDescendente: presenta algunos cambios respecto a la estrategia anterior, se realizan comprobaciones del precio para restar la cantidad adecuada y en lugar de sumar el porcentaje obtenido a la utilidad base, este se le resta a la aspiración.

```
Algoritmo RandomDescendente
     t<-false;
    Imprimir "Método Random Descendente";
    //Se comprueba el valor del precio
    Si (precio>=utilidadr) Entonces
    tempa=as-precio;
    t=true;FinSi
    Si(precio<utilidadr) Entonces
    tempa=as-utilidadr;
    t=true;FinSi
    Si (precio>as)Entonces
    aspiracion=as;FinSi
    Si(t==true) Entonces
    rand <-Random();
    //Se calcula un random entre 1---> valor obtenido
    result <- 1 + (rand * (tempa - 1));
    Imprimir "Resultado random: "+result;
    //Ha dicho valor se le calcula el decremento establecido por el usario
    porcentaje=(result*incremento)/100;
    Imprimir "Precio base: "+precio;
    Imprimir "Aspiracion: "+as;
    Imprimir "Utilidad neta: "+utilidadr ;
    Imprimir"Decremento: "+incremento;
    Imprimir "Porcentaje: "+porcentaje;
    //Al nivel de aspiración se le resta el valor obtenido anteriormente
    tempb=as-porcentaje;
    //Se comprueba que la propuesta sea mayor que el precio base
    Si(tempb<precio) Entonces
        Si(precio<utilidadr) Entonces
            aspiracion=utilidadr; FinSi
        Si(precio>utilidadr)
            aspiracion=as;FinSi
    FinSi
    Si (tempb<utilidadr) Entonces
        aspiracion=utilidadr;Fin si
    //Se comprueba que la propuesta no sea mayor que la aspiración
    Si (tempb>as) Entonces
     aspiracion=as;
    Sino
        aspiracion=tempb; FinSi
    Imprimir "Propuesta descendente: "+tempb;
    FinSi
FinAlgoritmo
```

Ilustración 34. Pseudocódigo RandomDescendente

4.1.5.4. Creación y configuración de CFactory del Comprador

```
//Se crea el CFactory y se configura
CFactory talk = new myFIPA_CONTRACTNET().newFactory("TALK", null,
        null,conversaciones, myProcessor.getMyAgent(), 0);
```

Ilustración 35. CFactory Comprador

4.1.5.5. Añadir al comprador como participante

```
//Se añade el CFactory como participante
this.addFactoryAsParticipant(talk);
```

Ilustración 36.Extracto de código al añadir comprador.

4.1.5.6. Reportes Gráficos

Al igual que en el vendedor en el comprador se producen reportes gráficos, son los siguientes:

a) Número de propuestas enviadas: en este reporte se contabiliza el número de propuestas enviadas a los vendedores por todos los compradores del universo, se utilizan barras 3D.

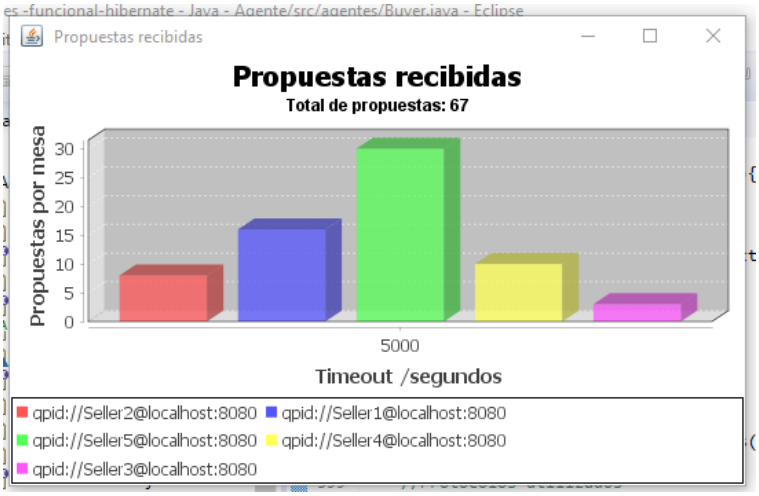

Ilustración 37. Resultado propuestas enviadas a los compradores.

b) Protocolos de negociación utilizados: se calcula el porcentaje en el cada protocolo de negociación fue utilizado.

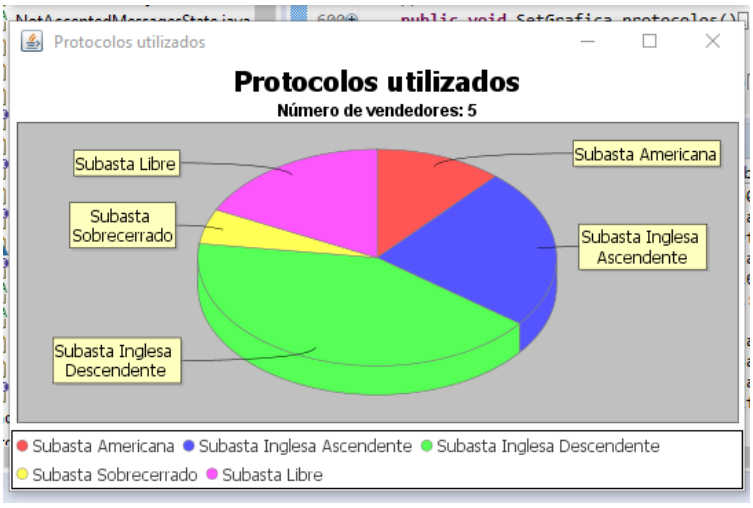

Ilustración 38. Resultado protocolos de negociación.

c) Estrategias de negociación utilizadas: se calcula el porcentaje de uso de las estrategias.

Ilustración 39.Resultado estrategias de negociación.

4.1.6. Módulo Gestor

Como se señaló anteriormente el módulo gestor es fundamental para el correcto funcionamiento del prototipo de mercado, ya que de este dependen, los cambios de estado de los agentes, el control de las conversaciones, acuerdos que se dan entre estos y la creación de nuevas conversaciones.

Este módulo se encuentra constituido por dos elementos principales:

4.1.6.1. Protocolo de comunicación Contract-Net

4.1.6.1.1. Contract-Net

Un agente inicia el protocolo enviando un mensaje *cfp*, al cual los participantes tendrán que notificar su disposición (para iniciar una conversación) antes de un *deadline*. Posteriormente, el agente inicial elige la mejor propuesta entre los participantes que han respondido afirmativamente a su solicitud (MAS, 2005) .

4.1.6.1.1.1. Mensajes Intercambiados

Los mensajes intercambiados entre los agentes del protocolo son:

Mensaje	**Descripción**
CFP (Call-ForProposal)	Pide candidaturas para realizar cierta acción. Inicia procesos de negociación.

Refuse	Cuando los respondedores rechazan participar en la conversación.
Not-Understood	Cuando la comunicación presento fallas.
Propose	Cuando un respondedor realiza una propuesta al iniciador.
Reject-Proposal	Es el caso cuando el iniciador recibe una propuesta y la rechaza.
Accept-Proposal	Cuando el iniciador recibe una propuesta y la acepta.
Failure	El respondedor envía este mensaje cuando su propuesta fue aceptada, pero ocurre un fallo.
Inform-Done	El respondedor envía este mensaje cuando su propuesta fue aceptada y la acción (método doTask)se realizó con éxito.
Inform-Results	El respondedor envía este mensaje cuando su propuesta fue aceptada y necesita informar solo los resultados de la acción realizada.

Tabla 3. Mensajes intercambiados protocolo Contract-Net.[5]

El funcionamiento del protocolo Contract-Net se puede visualizar en el siguiente diagrama:

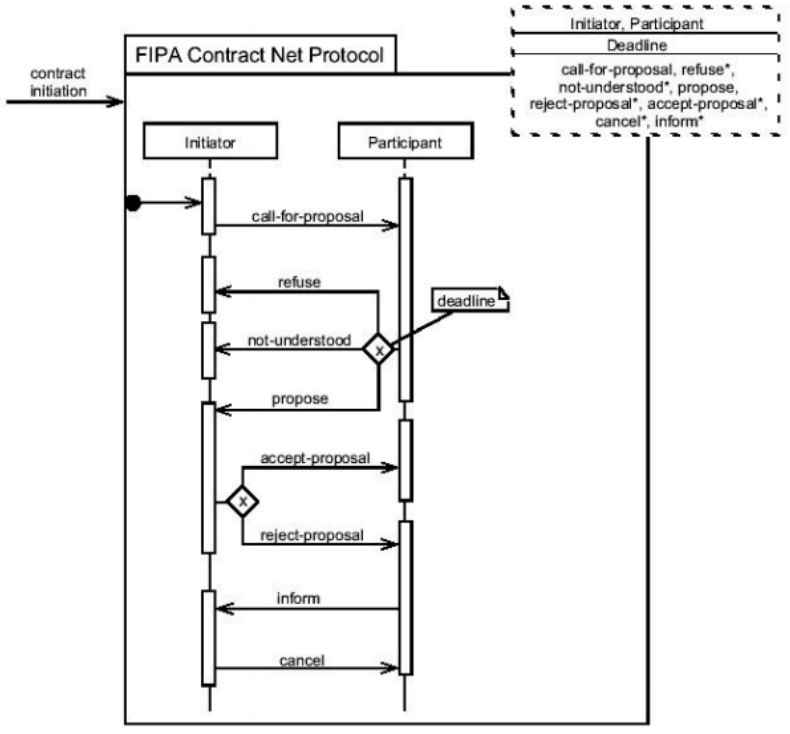

Ilustración 40. Diagrama de protocolo Contract-Net.[6]

[5] http://www.gti-ia.upv.es/sma/tools/magentix2/

4.1.6.1.2. Iterated Contract-Net

Amplía el protocolo descrito en la definición anterior, pues permite varias rondas de contrataciones. Al igual que en el Contrac-Net simple, un agente inicia el protocolo enviando un mensaje *cfp*, al cual los participantes tendrán que notificar su disposición antes de un plazo límite. Sin embargo, aparte de rechazar o aceptar ofertas también puede optar por volver a anunciar la tarea. Con tal fin, primeramente, rechaza la oferta anterior y seguidamente envía un nuevo *cfp*.

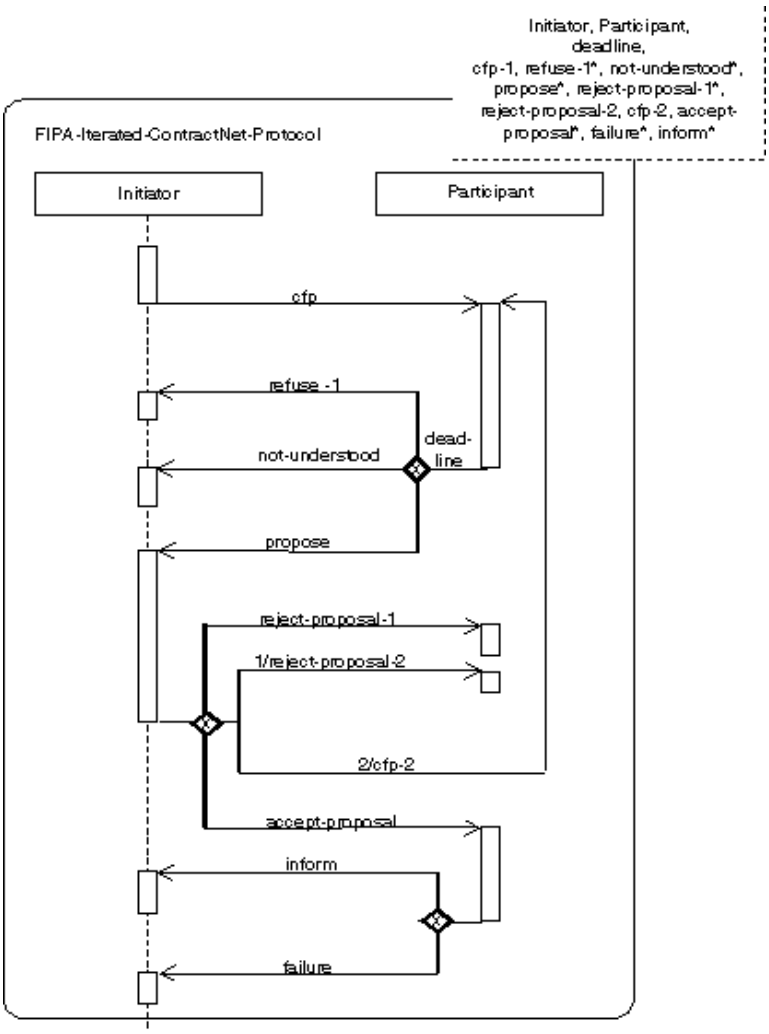

Ilustración 41. Diagrama de protocolo iterated contrac net (MAS, 2005).

En Magentix2 el protocolo Iterated Contract-Net no se encuentra definido, sin embargo, el protocolo Contract-Net fue modificado de tal manera que el prototipo permite que los respondedores puedan enviar varias propuestas al iniciador, sin embargo, el iniciador solo envía un mensaje *cfp* para solicitar a los respondedores que inicien una conversación con él.

4.1.6.1.3. Funcionamiento Contract-Net

El comportamiento del protocolo Contract-Net que utiliza el prototipo se puede visualizar en la siguiente ilustración:

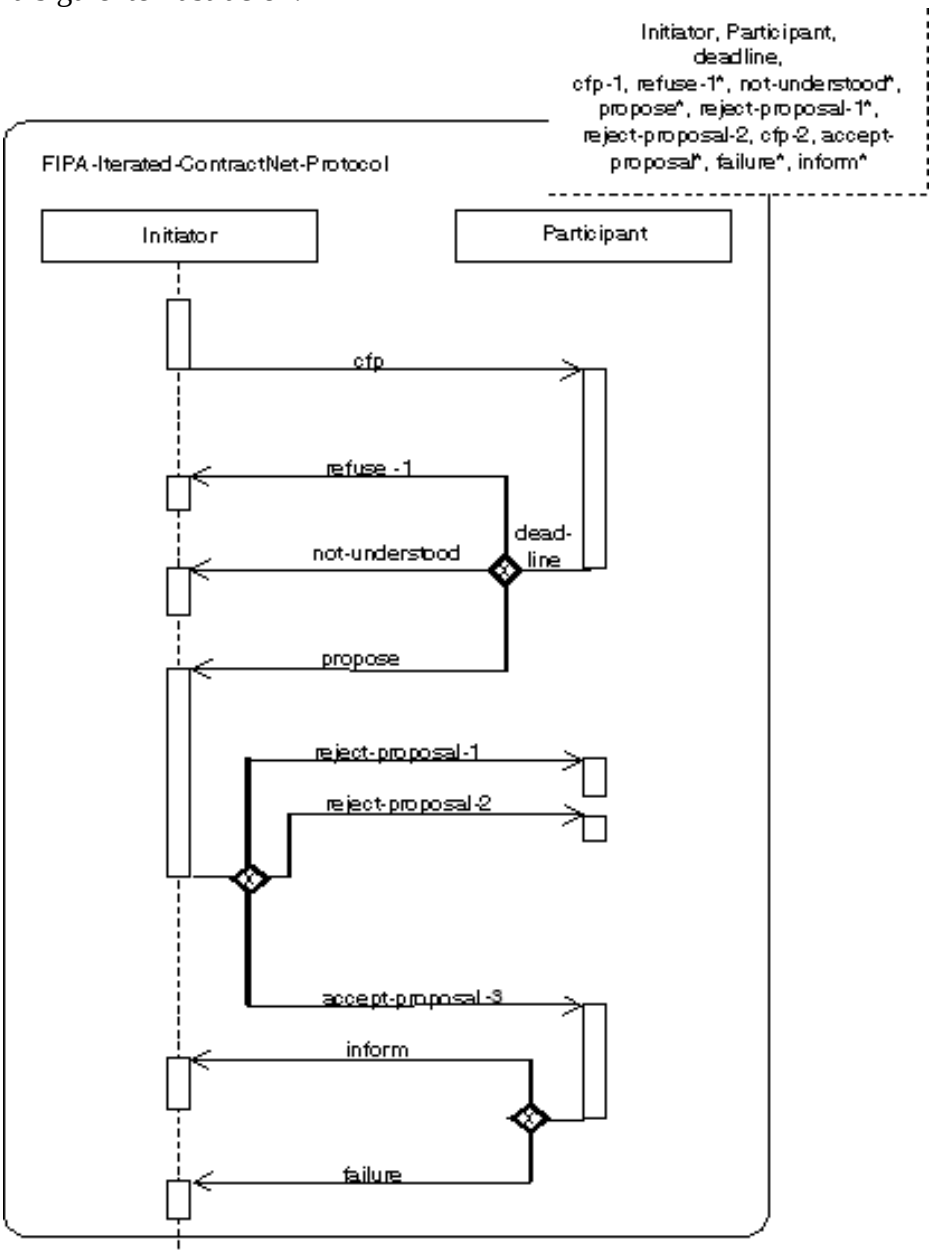

Ilustración 42. Diagrama de funcionamiento protocolo Contract-Net.

4.1.6.1.4. Representación de interacciones entre los agentes

Utilizando diagramas de secuencias las interacciones que ocurren entre los agentes pueden ser representadas de las siguientes maneras:

a) Formato básico de comunicación

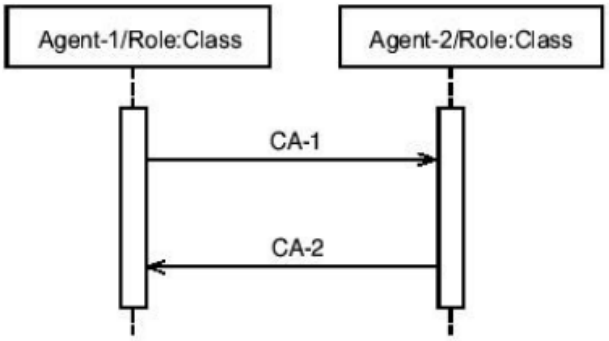

Ilustración 43. Formato básico de comunicación.[6]

b) Hilos de interacción

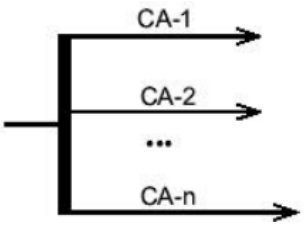

Ilustración 46. Mensajes Concurrentes.

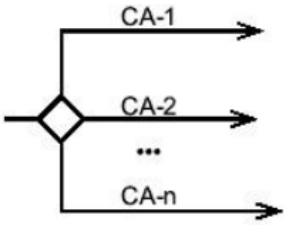

Ilustración 45. Con decisión o inclusivo.

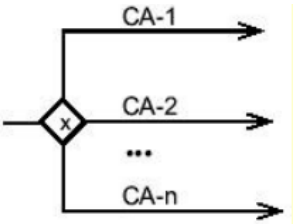

Ilustración 44. Con decisión o exclusivo.

[6] https://www2.infor.uva.es/~cllamas/MAS/ProInterAg.pdf

4.1.6.2. CFactories , CProcessors

Como se definió anteriormente un CFactory representa el protocolo de interacción utilizado, el CProcessor al ser una instancia de CFactory representa cada conversación entre los agentes, el CProcessor contiene una gráfica por cada rol del protocolo, esta gráfica representa los estados por los que el agente que represente determinado rol debe transitar para un correcto funcionamiento del protocolo, su funcionamiento es el de una máquina de estados.

Los CFactories pueden ser utilizados por agentes que utilicen el rol de Iniciador o de Participante, los CFactories Iniciadores empiezan una conversación basada en la lógica del agente, los Participantes comienzan la ejecución de un CProcessor cuando reciben un mensaje con los parámetros adecuados según su filtro o preferencias.

Durante la conversación el CProcessor ejecuta las acciones correspondientes al estado actual y realiza una transición entre estados cuando el agente recibe o envía un mensaje relacionado a una conversación especifica (GTI, 2015).

4.1.6.2.1. Colección de Estados

Existe una colección de estados predefinidos para los CAgents, estos son:

Estado	**Descripción**
BEGIN	Inicio de una conversación.
FINAL	Final de una conversación.
SEND	Envío de un mensaje.
WAIT	La conversación espera hasta un mensaje le sea asignado.
RECEIVE	Un agente recibe un mensaje, debe ser estar definido después de un estado WAIT.

Tabla 4. Colección de Estados CAgents.

4.1.6.2.2. Creación CProcessor

Para crear un CProcessor se debe seguir el siguiente procedimiento:

1. Definir los estados y las transiciones que componen el grafo asociado al CProcessor.

2. Asignar a todos los estados definidos exceptuando el estado WAIT, un método en el cual se encuentren especificadas todas las acciones que deben realizarse en ese estado, estos métodos deben retornar el nombre del estado al que deben transitar.

Los métodos del estado SEND, además de retornar el nombre del estado siguiente deben asignar los valores correspondientes a la variable que le enviaran al agente Iniciador.

```
BeginState BEGIN = (BeginState) talk.cProcessorTemplate().getState("
   BEGIN");
public String run(CProcessor myProcessor, ACLMessage msg) {
   // In this example there is nothing more to do than continue
   // to the next state which will send the message.
   return "PROPOSE";
};
}
BEGIN.setMethod(new BEGIN_Method());

SendState PROPOSE = new SendState("PROPOSE");
class PROPOSE_Method implements SendStateMethod {
   public String run(CProcessor myProcessor, ACLMessage
      messageToSend) {
      messageToSend.setContent("Would you like to come with me to
         the cinema?");
      messageToSend.setReceiver(new AgentID("Sally"));
      messageToSend.setSender(myProcessor.getMyAgent().getAid());
      return "WAIT";
   }
}
PROPOSE.setMethod(new PROPOSE_Method());
talk.cProcessorTemplate().registerState(PROPOSE);
talk.cProcessorTemplate().addTransition(BEGIN, PROPOSE);

talk.cProcessorTemplate().registerState(new WaitState("WAIT", 1000))
   ;
talk.cProcessorTemplate().addTransition(PROPOSE, WAIT);
```

Ilustración 47. Definición, métodos y transiciones de estados.

4.1.6.3. Grafos del Processor

A pesar de que se utilizaron las plantilla predefinidas por Magentix2 para los roles de Iniciador y Participante, estas plantillas recibieron varias modificaciones, la principal se realizó en la plantilla para el rol Participante debido a que solo era posible enviar una propuesta al Iniciador, se modificó de tal manera que se enviaran propuestas dependiendo del número iteraciones del Iniciador definido en el fichero properties.

4.1.6.3.1. Grafo Vendedor (Seller)

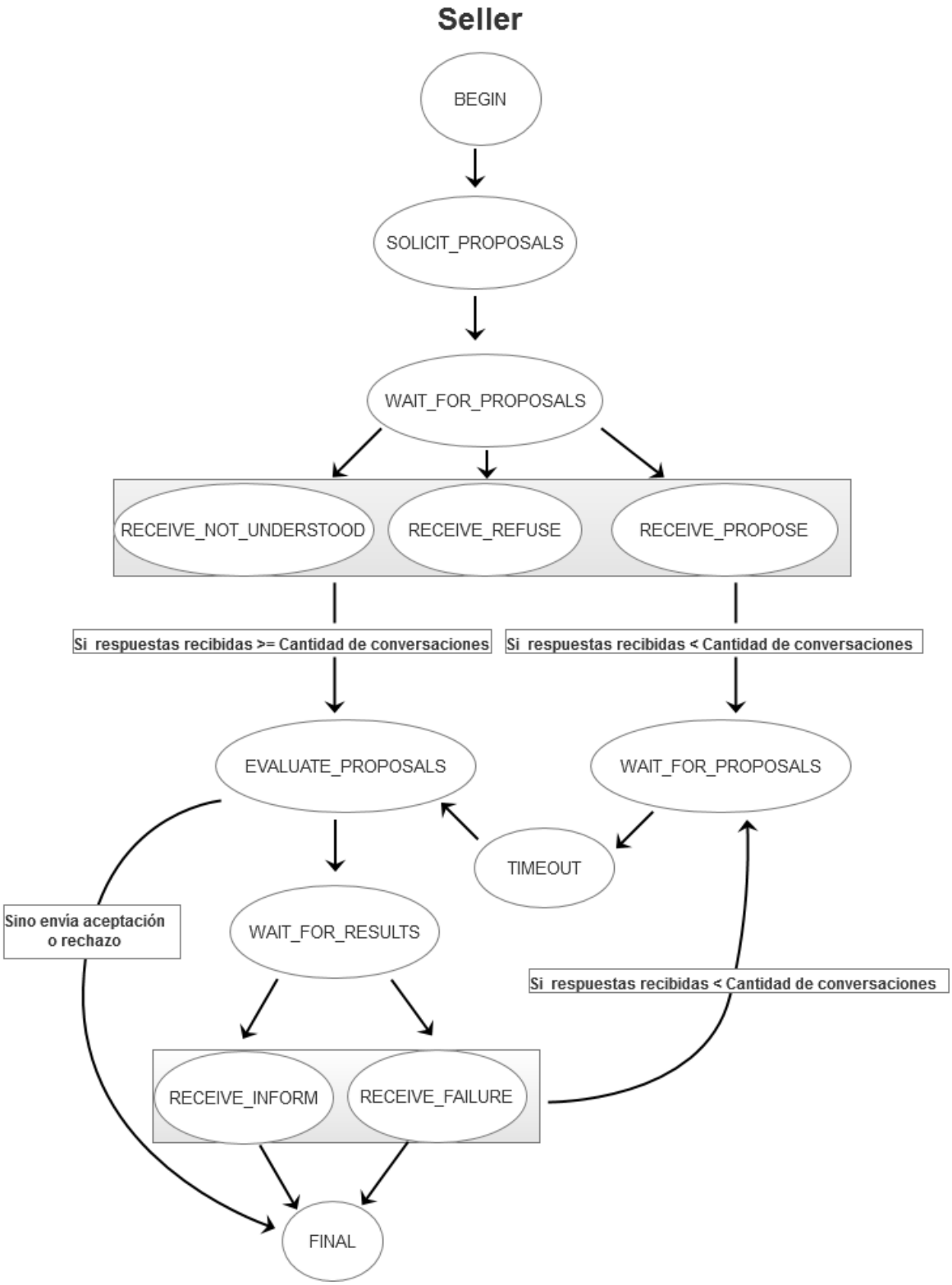

Ilustración 48. Grafo Vendedor(Seller).

4.1.6.3.2. Grafo Comprador(Buyer)

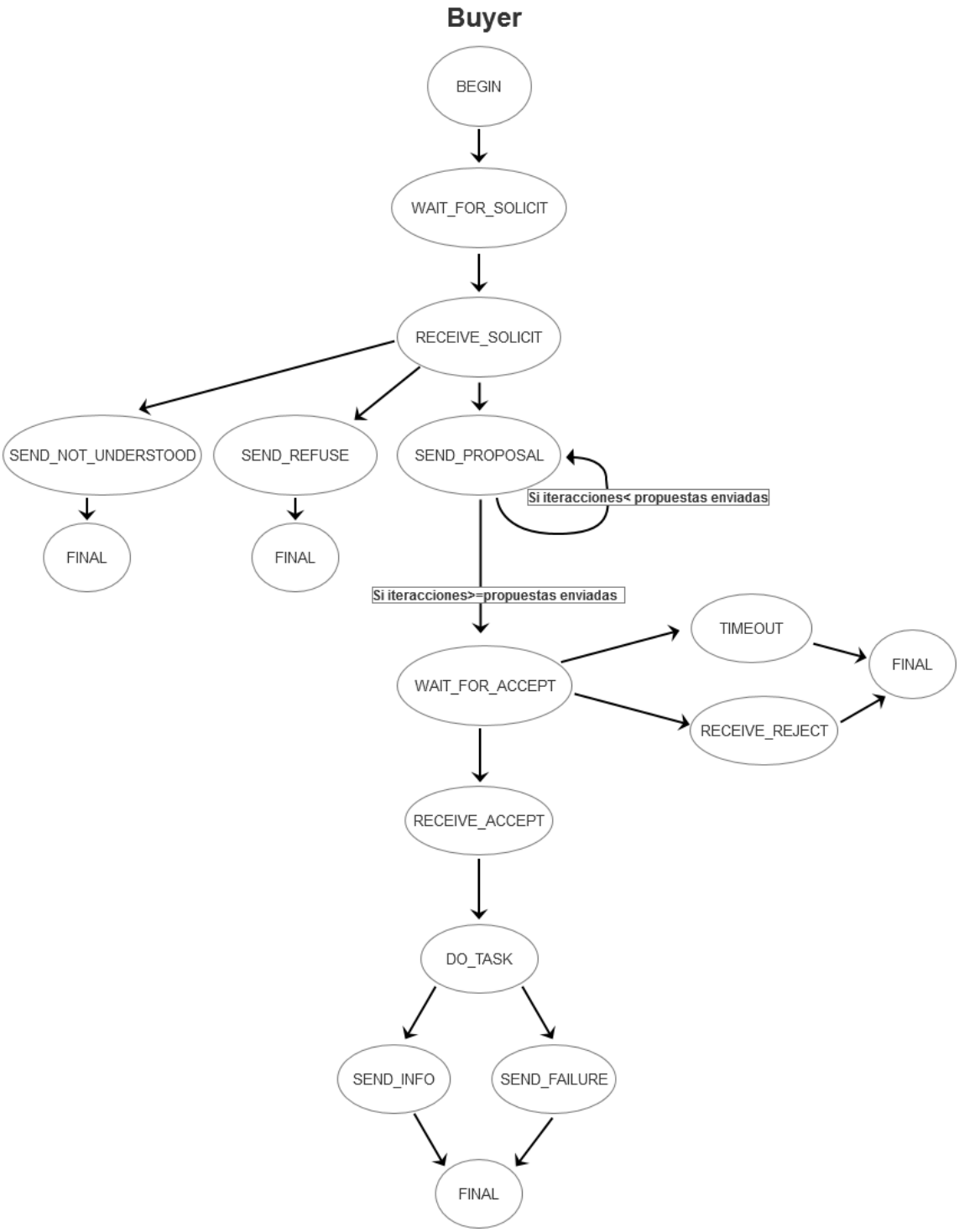

Ilustración 49. Grafo Comprador(Buyer).

Al finalizar este capítulo, se concluye que cada uno de los elementos que conforman el prototipo son esenciales para su correcto funcionamiento.

5. Experimentación y Resultados

En este capítulo se describen los experimentos que se llevaron a cabo para comprobar el correcto funcionamiento del prototipo y los resultados obtenidos al realizados estos experimentos.

5.1. Experimentación

Para llevar a cabo la experimentación se crearon diez plantillas de configuración (ficheros properties) en los cuales se establecieron la cantidad de agentes, vendedores y compradores entre los números uno y diez para de esta manera tener 2^{10} (1024) posibles configuraciones, en estas plantillas es necesario modificar: la utilidad base, la aspiración, el incremento, las preferencias y estrategias de negociación de los compradores ; el precio base y el timeout de los vendedores ;el tipo de subasta y las iteraciones máximas de los protocolos.

Después de que las plantillas fueron creadas, se realizaron veinte pruebas utilizando veinte archivos de configuración, cada uno de estos con valores diferentes para sus parámetros, esto con dos propósitos:

1. Observar el funcionamiento del prototipo para comprobar que sea el correcto.
2. Conocer el grado de aceptación que presenta cada una de las estrategias.

5.2. Resultados

De igual manera fue posible conocer el grado aceptación de cada una de las estrategias de negociación utilizadas, en la siguiente tabla podemos observar la cantidad de veces que fue aceptada cada estrategia de negociación:

Estrategia	Cantidad de veces aceptada
TitforTatAscendente	44
TitforTatDescendente	5
RandomAscendente	13
RandomDescendente	17
TemporalAscendente	17

TemporalDescendente	12
Total, de Iteracciones	108

Tabla 5. Estrategias Aceptadas.

En la siguiente gráfica podemos visualizar el porcentaje de aceptación de cada una de las estrategias:

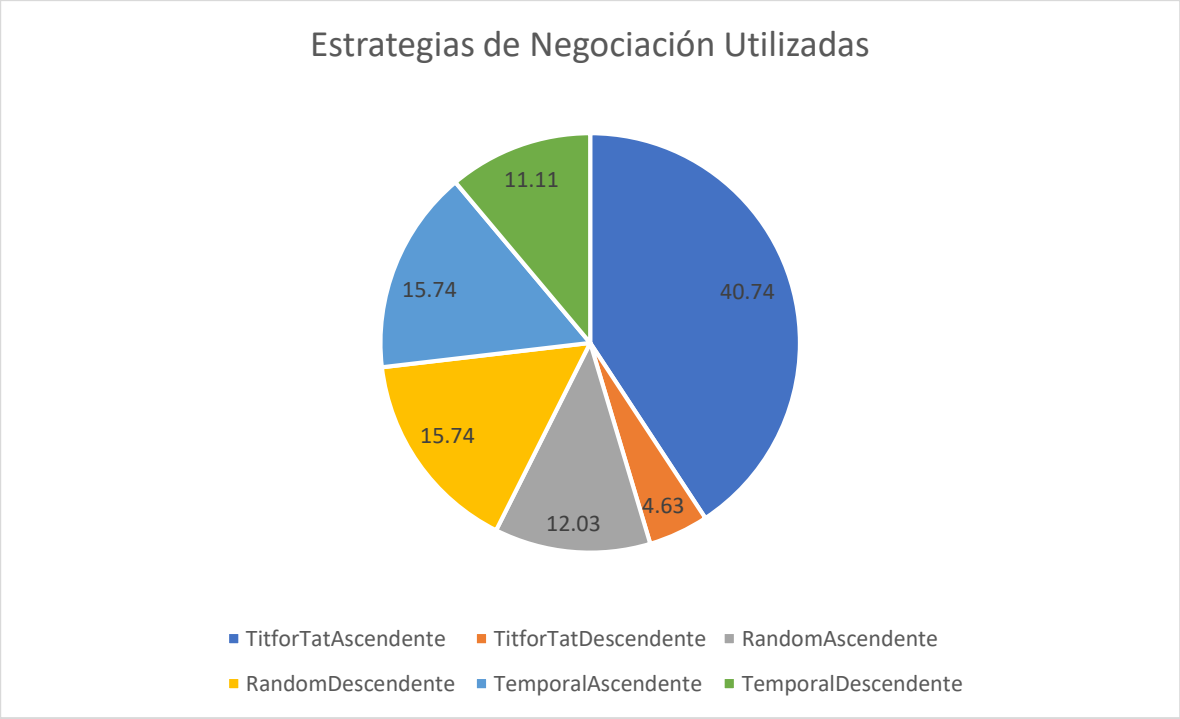

Ilustración 50. Porcentaje de aceptación.

Al realizar los experimentos fue posible apreciar que conforme el número de agentes en el universo aumentaba, cuando finalizaba la ejecución del archivo de configuración (fichero properties), algunos procesos referentes al software no finalizaban, por lo cual era necesario comprobar que no existiera ningún otro proceso ejecutándose, y si lo hacía finalizarlo de forma manual, para poder ejecutar otro archivo de configuración, se menciona esto debido a que es un aspecto que puede afectar la escalabilidad del sistema.

Después de finalizar la realización de los experimentos, es posible concluir que el funcionamiento del prototipo es el adecuado y que la estrategia de negociación con mayor grado de aceptación es la de TitforTatAscendente.

6. Conclusiones y Trabajos Futuros

En este capítulo se detallarán las conclusiones obtenidas al finalizar el trabajo, así como los trabajos futuros que se plantean.

6.1. Conclusiones

El presente trabajo ha detallado el proceso que se llevó a cabo, para realizar el diseño y la implementación de un mercado genérico basado en tecnología multiagentes, logrando el cumplimiento satisfactorio de los objetivos planteados inicialmente.

Al finalizar el trabajo podemos concluir lo siguiente:

- Actualmente existen varias plataformas virtuales que permiten la interacción entre compradores y vendedores, las plataformas comerciales requieren que los actores de cada interacción se encuentren de manera presencial para que esta pueda llevarse a cabo.
- Existen diversos proyectos creados utilizando tecnología MultiAgente, siendo la mayoría plataformas que permiten la creación de mercados virtuales y agentes autónomos que funcionan basándose en los deseos de un usuario común.
- La plataforma virtual Magentix2 posee todos los servicios y herramientas necesarios para el diseño y creación de mercados virtuales, permitiendo la correcta interacción entre compradores y vendedores, por medio de la tecnología MultiAgente.
- El prototipo realizado es un diseño completo y estructurado, que permite la creación de un mercado genérico, en el cual los usuarios podrán vender y comprar bienes y servicios, sin necesidad de encontrarse de manera presencial al momento de la negociación.

6.2. Trabajos Futuros

El trabajo realizado puede ser extendido de las siguientes maneras:

- Crear una plataforma web que permita optimizar la interacción entre compradores y vendedores, esta plataforma deberá utilizar bases de datos para almacenar la información de cada usuario, así como todos los parámetros necesarios para el correcto funcionamiento del mercado virtual.
- Utilizar la herramienta Hibernate para facilitar el mapeo de atributos entre la base de datos creada y el modelo de objetos del prototipo.
- Utilizar algoritmos evolutivos para optimizar la búsqueda de propuestas, con el fin de perfeccionar la capacidad de negociación de los agentes.

7. Bibliografía

Alberola, J. M.-F. (2008). Magentix: a multiagent platform integrated in linux. *Proceedings of the sixth European workshop on multi-agent systems EUMAS*.

Aumann, R., & Hart, S. (1994). *Handbook of Game Theory with Economic Applications, Volumen 2.* Netherlands : North Holland.

Brenner, W. Z. (1998). *Intelligent Software Agents: Foundations and Applications.* New York: Springer-Verlag.

Chavez, A. &. (1996). Kasbah: An agent marketplace for buying and selling goods. *Proceedings of the first international conference on the practical application of intelligent agents and multi-agent technology Vol. 31*, 40.

Cuní, G. E. (2004). MASFIT: Multi-Agent System for Fish Trading. *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI),August 22-27, Valencia, Spain.* .

e-Commerce: aplicación y desarrollo. (2010). Editorial Vértice.

FARIÑO, G. (2011). *Modelo espiral de un proyecto de desarrollo de software.*

Ferber, J. (1999). *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence.* Addison Wesley.

Florea, C. C. (2004). Autonomy in Multi-agent Systems: A Classification Attempt. *Lecture Notes in Computer Science*.

Friedman, J. (1991). *Teoría de Juegos con Aplicaciones a la Economía.* Alianza Universidad.

Giunchiglia, F. M. (2002). The Tropos Software Development Methodology:. *Proceedings of the AAMAS'02 Workshop on Agent Oriented Software Engineering (AOSE-2002)*, 63-74.

Goyal, M. &. (2012). A fuzzy attitude based bidding strategy in continuous double auctions. *Web Intelligence and Agent Systems: An International Journal*, 10(1), 65-74.

GTI, I. (2015). *Grupo de Tecnología Informática – Inteligencia Artificial*. Obtenido de http://www.gti-ia.upv.es/sma/tools/magentix2/

Guttman, R. M. (1999). Agents as mediators in electronic commerce. *Intelligent Information Agents, Springer, Berlin, Heidelberg.* , 131-152.

H., M. (1996). Negotiation principles. *Foundations of distributed artificial intelligence*, 211 - 229 .

Huang, S. L. (2005). Designing intelligent sales-agent for online selling. *Proceedings of the 7th international conference on Electronic commerce*, 279-286.

Jaso, C. (2004). El Impacto de Compranet en el Gobierno Electrónico en México. *EU-LAT Workshop on e-Government and e-Democracy,. Santiago de Chile, May.*, 24-27.

K Potiron, A. S. (2013). *From fault classification to fault tolerance for multi-agent systems.* Springer.

Kraus, S. (2001). *Strategic Negotiation in Multiagent Environments.* Cambridge, MA: MIT Press.

KROKOU, D. (2015). *ESTRATEGIAS DE ENTRADA EN EL EMERGENTE MERCADO ELECTRÓNICO CHINO.* Babelcube Inc.

Krulwich, B. (1996). Information integration agents: BargainFinder and NewsFinder. *nternet-Based Information Systems: Papers from the 1996 AAAI Workshop, Vol 9*.

Lomuscio, A. R. (2001). Automated negotiation:Prospects, methods and challenges. *Internat. J. Group Decision Negotiation*, 199–215.

Luo, X. J. (2003). A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence*, 148(1-2), 53-102.

M. Barbuceanu, W.-K. L. (2000). A multi-attribute utility theoretic negotiation architecture for electronic. *Proceedings of the fourth international conference on Autonomous agents*, 239-246.

María, G. (2002). Concepto de mercado y sus tipos. *https://www.gestiopolis.com/concepto-mercado-tipos/*.

MAS, A. (2005). *AGENTES SOFTWARE Y SISTEMAS MULTIAGENTE CONCEPTOS,ARQUITECTURA Y APLICACIONES.* Madrid : PEARSON EDUCACION, SA.

N. Matos, C. S. (1998). Evolutionary computing and negotiating agents, in: Agent-Mediated Electronic Commerce. *Lecture Notes in Artificial Intelligence, Vol. 1571, Springer, Berlin.*, 126–150.

Ngobye, M. D. (2010). Types and priorities of multi-agent system interactions. . *Interdisciplinary Description of Complex Systems: INDECS, ,* 49-58.

Oliveira, E. &. (2001). Agents advanced features for negotiation in electronic commerce and virtual organisations formation process. *Agent Mediated Electronic Commerce,Springer, Berlin, Heidelberg.*, 78-97.

P. Faratin, C. S. (2002). Using similarity criteria to make issue tradeoffs in automated. *Artificial Intelligence 142 (2)*, 205–237.

P. Maes, R. G. (1999). Agents that buy and sell: Shoppers and sellers alike dispatch them into the digital bazaar to autonomously represent their best interests. *Comm*, 81–91.

Pou, M. A. (2006). *Manual práctico de comercio electrónico.* LA LEY.

R. Das, J. H. (2001). Agent-human interactions in the continuous double auction. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence Vol. 2, Seattle, WA*, 1169–1176.

R. Kowalczyk, V. B. (2000). On constraint-based reasoning in e-negotiation agents. *International Workshop on Agent-Mediated Electronic Commerce, Springer*, 31-46.

Rahwan, I. K. (2002). Intelligent agents for automated one-to-many e-commerce negotiation. *Australian Computer Science Communications Vol. 24*, 197-204.

Richter, C. W. (1998). Genetic algorithm evolution of utility bidding strategies for the competitive marketplace. *IEEE transactions on power systems, 13(1).*, 256-261.

Rosenschein, J. S. (1994). *Rules of encounter: designing conventions for automated negotiation among computers.* MIT press.

S. Kraus, K. S. (1998). Reaching agreements through argumentation: A logical model and implementation. *Artificial Intelligence 104 (1–2)*, 1-69.

S. Parsons, C. S. (1998). Agents that reason and negotiate by arguing. *J. Logic Comput*, 261–292.

S. Paurobally, J. C. (2001). Specifying the processes and states of negotiation. *Agent Mediated Electronic Commercee,Springer, Berlin, Heidelberg*, 61-77.

Sandholm, T. W. (1999). Distributed Rational Decision Making. *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence. The MIT Press*, 201-258.

Sycara, K. P. (1990). Persuasive argumentation in negotiation. *Theory and decision*, 203-242.

Thomson, W. (1997). Cooperative Theory of Bargaining I: Classical. *Hart S., Mas-Colell A. (eds) Cooperation: Game-Theoretic Approaches* , Volumen. F-155 9-24 Springer.

Tsvetovatyy, M. &. (1996). Toward a virtual marketplace: Architectures and strategies. *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM Vol. 96*, 597-613.

Tsvetovatyy, M. G. (1997). Magma an agent based virtual market for electronic commerce. *Applied Artificial Intelligence, 11(6)*, 501-523.

Varian, H. R. (1992). *Microeconomic Analysis.* New York: W. W. Norton.

Walsh, W. E. (1998). A market protocol for decentralized task allocation. *Proceedings of the Third International Conference on Multi-Agent Systems*, 325–332.

Wooldridge, M. J. (1995). Intelligent agents: Theories, Architectures and Languages. *Lecture Notes in Artificial Intelligence vol. 890,*.

Wu, D. J. (2001). Software agents for knowledge management: coordination in multi-agent supply chains and auctions. *Expert Systems with Applications, 20(1)*, 51-64.

Wurman, P. R. (1998). The Michigan Internet AuctionBot: A configurable auction server for human and software agents. *Proceedings of the second international conference on Autonomous agents* , 301-308.