UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Un sistema multi-agente para mercados de logística inversa

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Adrián Martínez Carreres

Tutor: Vicente Juan Botti Navarro
Adriana Giret Boggino

Curso 2017-2018

Agradecimientos

Quisiera agradecer en primar lugar al grupo de investigación GTI-IA por brindarme la oportunidad de desarrollar junto a ellos este trabajo, pero especialmente a Adriana Giret y Vicente Botti por ser mis tutores y guiarme a lo largo de este trabajo.

Por otra parte, agradecer también a mi familia por apoyarme en todas mis decisiones y por enseñarme que con esfuerzo y trabajo todo se puede conseguir.

Por último a mis amistades, tanto a las forjadas durante la carrera como a los que siempre han estado ahí desde mi infancia, por todos los buenos momentos que hemos vivido juntos y por estar siempre ahí cuando se les necesitaba.

Resum

L'objectiu d'aquest TFG es desenvolupar un sistema multi-agent per a donar suport a mercarts virtuals en l'àmbit de les cadenes de subministrament verd. Utilitzar sistemes multi-agent en aquests entorns facilitarà a les diverses empreses involucrades en la logística inversa l'accés a aquests tipus de mercats, gracies a l'ús de servicis intel·ligents encapsulats en els agents que participen en els mercats virtuals. A més, en aquest TFG es treballarà en la composició d'ítems de negociació de manera distribuïda. Sent aquest últim problema un tema encara obert en l'àrea de la negociació automàtica.

Paraules clau: Sistema multi-agent, logística inversa, cadena de subministrament verd, ontologia de productes, negociació automàtica

Resumen

El objetivo de este TFG es desarrollar un sistema multi-agente para dar soporte a mercados virtuales en el ámbito de las cadenas de suministro verde. Utilizar sistemas multi-agente en estos entornos facilitará el acceso a este tipo de mercado a las distintas empresas involucradas en la logística inversa, mediante servicios inteligentes encapsulados en los agentes que participan en los mercados virtuales. Además, en este TFG se trabajará en la composición de ítems de negociación de manera distribuida. Siendo este último problema un tema abierto en el área de la negociación automática.

Palabras clave: Sistema multi-agente, logística inversa, cadena de suministro verde, ontología de productos, negociación automática

Abstract

The aim of this project is to develop a multi-agent system, which acts as a virtual market in the green supply chain field. The usage of the multi-agent system in such environments would ease the access of business involved in reverse logistics to this kind of markets, by means of intelligent systems encapsulated in the different agents who take part in the virtual market. Moreover, the item composition for negotiating in a distributed way will also be approached. This is still an open topic in the automatic negotiation field.

Key words: Multi-agent system, reverse logistics, green supply chain, item ontology, automatic negotiation

Índice general

Índice de figuras

Índice de tablas

Índice de algoritmos

CAPÍTULO 1

Introducción

En este capítulo se expone la motivación del trabajo realizado, los objetivos que se plantearon antes de realizarlo y, por último, la estructura que sigue la memoria.

1.1 Motivación

La gran cantidad de desechos que se generan en los países desarrollados es uno de los grandes problemas a los que nos enfrentamos en este era [1]. Diversos gobiernos han aprobado leyes tanto para controlar la cantidad de residuos que se pueden verter como para obligar a las empresas a reciclar [2].

Es por ello que en los últimos años ha surgido un gran interés por estudiar distintas formas de reciclar los residuos de manera eficiente, pero sin que ello conlleve un aumento del coste final de los productos para los usuarios. Una de las áreas con mayor auge es la logística inversa, la cual consiste en recuperar los productos de los clientes, una vez se haya terminado su vida útil, para extraer su materia prima y volver a usarla para producir nuevas unidades.

El interés que existe desde el ámbito informático es el de proponer diversas soluciones para solventar este problema de manera que todo se gestione de manera automática pero primando los intereses particulares de cada empresa. Es por ello que solucionar el problema mediante un sistema distribuido puede ser un enfoque válido ya que, de esta manera, los diversos actores que participen en la solución mantendrán sus preferencias.

1.2 Objetivos

El objetivo principal que tiene este trabajo es el de proponer y desarrollar una solución multi-agente que pueda ser utilizada para gestionar la logística inversa de distintas empresas.

En cuanto a los objetivos específicos:

- Estudiar y analizar los requisitos de un mercado virtual para la logística inversa.
- Definir un modelo de infraestuctura de un sistema multi-agente para mercados virtuales de logística inversa
- Proponer el diagrama de flujo de interacción que seguirán los agentes para interactuar con el mercado.
- Definir estrategias y protocolos de negociación para la composición y descomposición de objetos.
- Validar la solución propuesta mediante el uso de situaciones verosímiles sobre un sector en particular.
- Desarrollar una interfaz gráfica que integre el mercado propuesto para simplificar y hacer más atractivo el uso del propio mercado.

1.3 Estructura de la memoria

La estructura que se va a seguir a lo largo de este trabajo es la siguiente: primero se explicará qué es la logistica inversa y cómo se gestiona en la actualidad, seguidamente se describirán las bases de los sistemas multi-agente y se debatirán las distintas soluciones *software* que se podrían utilizar para implementar el mercado propuesto. A continuación se explicará cómo es el mercado que se propone y se describirán todas las etapas para obtener un producto *software*, que son análisis, diseño, implementación y verificación. En último lugar se expondrán los resultados y conclusiones obtenidos gracias a la herramienta desarrollada.

CAPÍTULO 2

Logística inversa

En las páginas siguientes se explica en detalle en qué consiste la logística inversa y se exponen los avances que existen hasta le fecha. Se mencionan también algunos escenarios donde se aplique, así como algunas de las limitaciones que actualmente existen en los mismos. Por último, se explica cómo es el mercado que se ha implementado, que es la principal motivación de este trabajo.

2.1 Definición

Para poder entender en que consiste la logística inversa, primero se va a explicar qué se entiende por logística dentro del ámbito empresarial y, posteriormente, se definirá la logística inversa.

Según el *Council of Supply Chain Management Professionals* (2014), «Logística es aquella parte de la Gestión de la Cadena de Suministro (SCM), que planifica, implementa y controla el flujo directo e inverso y el almacenaje efectivo y eficiente de bienes y servicios, con toda la información relacionada desde el punto de vista de origen al punto de vista de consumo, para poder cumplir con los requerimientos de los clientes».

En la definición se aprecia como la logística se divide en dos secciones, la directa o logística tradicional y la inversa. A continuación se va a detallar en que consiste cada una.

2.1.1. Logística

La logística tradicional se encarga de todos los procesos necesarios para producir un bien o servicio. Abarca todas las áreas, desde la recolecta de materias primas hasta la entrega al usuario final. Tiene como principal objetivo conseguirlo con el menor coste posible para aumentar su margen de beneficios.

A grandes rasgos, las etapas por las que debe pasar son:

- **Recolecta de materias primas**: extraer las materias primas con las que se producirán los objetos o servicios con los se negociará.
- **Entrada de materias primas**: transportar dicha materia prima al punto donde se trabajará con ellas. Tiene en cuenta también el coste que conlleva almacenar los materiales por parte de la empresa.
- **Producción de componentes**: usar los materiales que se tienen para producir las distintas partes del producto.
- **Montaje del producto**: obtener el producto listo para ser usado por el cliente final.
- **Embalaje**: preparar un lote de productos finales para que pueda ser transportado.
- **Distribución**: transportar dichos lotes a los puntos de venta donde el cliente final podrá adquirirlos.

Cabe mencionar que una empresa en particular puede externalizar algunas de las etapas y centrarse únicamente en algunas de ellas. Sin embargo, esto queda fuera del ámbito del trabajo.

Figura 2.1: Flujo de la logística directa

Como se aprecia en 2.1, el flujo de la logística directa es totalmente secuencial, es decir, para poder realizar una etapa hace falta que todas las anteriores se realicen.

2.1.2. Logística inversa

Por otra parte, la logística inversa trata el flujo inverso. Desde el cliente final hacia la cadena de montaje de la fábrica. En los últimos años ha habido un auge en el uso de dicho flujo, ya sea para proporcionar un servicio postventa o para recuperar los objetos después de su vida útil. Dicha recuperación puede ser con el objetivo de recuperar su valor o para su correcta eliminación.

El uso de la logística inversa para ofrecer un servicio postventa al cliente es algo muy extendido en el mundo empresarial ya que actualmente se puede ver como una gran cantidad de grandes almacenes ofrecen servicios de reparación. Sin embargo, el uso de la logística inversa para recuperar los objetos y, por lo tanto, conseguir reciclarlos para reintroducirlos de nuevo en la cadena de montaje es algo no tan extendido.

En la siguiente figura, 2.2 , se puede apreciar como funciona el flujo de la logística inversa:

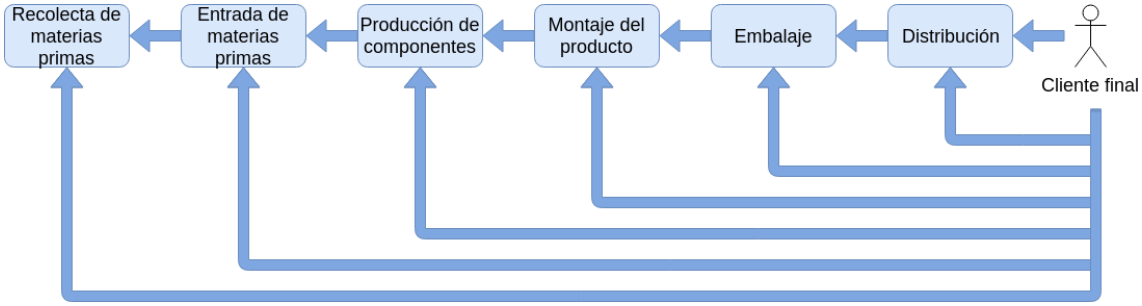


Figura 2.2: Flujo de la logística inversa

Cabe destacar que, al contrario que en la logística directa, el flujo no es completamente secuencial. En otras palabras, el producto devuelto por el cliente puede incorporarse directamente a una de las etapas de la logística directa. Para ilustrar esto se van a proporcionar diversos ejemplos:

- **Del cliente a la distribución**: el cliente devuelve un producto sin que este haya sido utilizado ni extraído de su embalaje original, por lo tanto se puede vender de nuevo.
- **Del cliente al embalaje**: el cliente devuelve un producto no utilizado pero sin su embalaje original, dicho producto se insertará directamente en la fase de embalaje debido a que ya se tiene el producto final.
- **Del cliente al montaje del producto**: el cliente devuelve el producto debido a que tiene un fallo en algunos de sus componentes. Dicho producto se tiene que montar nuevamente pero reemplazando el componente defectuoso.
- **Del cliente a la producción de componentes**: el cliente devuelve una pequeña parte de un producto debido a que ya no se puede utilizar más. Ejemplos de dicha situación son los consumibles como cartuchos de tinta o cables.
- **Del cliente a la entrada de materias primas**: el producto devuelto por el cliente puede ser utilizado directamente como materia prima. Este sería el caso de las botellas de plástico o del vidrio, debido a que el proceso que se debe realizar es similar independientemente de la naturaleza del material (reciclado o recién obtenido).

El gran auge que ha habido en los últimos años, y en el que se centra este trabajo, es en conseguir recuperar los objetos para reciclarlos usando el concepto de la logística inversa. Si se realiza correctamente se conseguiría:

- **Abaratar costes**: no hará falta conseguir las materias primas si no que estas se consiguen de productos desechados.
- **Dar una imagen de empresa concienciada con el medio ambiente**: los clientes observarán que la empresa tiene en cuenta los efectos nocivos que suponen sus actividades e intentan remediarlos. Esto también puede conllevar conseguir nuevos consumidores atraídos por el nuevo modelo productivo de la empresa.
- **Nuevas maneras de conseguir materiales**: los materiales que se necesitan se podrán conseguir únicamente de productos en circulación.

Actualmente este proceso es gestionado por las tiendas y grandes almacenes que ponen en contacto al cliente final con el fabricante del producto. Es por ello que, únicamente si el fabricante dispone de dicho servicio, el cliente podrá ser capaz de usarlo. Además suele existir una cierta compensación por parte de la empresa hacia el cliente, esto sirve como incentivo para que el cliente participe en este proceso. Dicha recompensa puede ser una cierta cantidad de dinero o un descuento en la siguiente compra.

Fijar la cantidad del incentivo es una tarea complicada relacionada con económicas, por lo que queda fuera del ámbito de este trabajo. Sin embargo, cabe mencionar que para fijar dicha cantidad se debe contemplar el importe que conlleva transportar el producto desechado desde el cliente a la empresa, el gasto que implica extraer lo que interese y el coste que supone obtener el material extraído de la fuentes tradicionales. Esto es así ya que a una empresa solo le interesa participar si no obtiene perdidas, aunque si esas pérdidas no son muy importantes puede interesar, debido a que virar hacia un ecosistema más sostenible puede ser visto como una campaña de *marketing*.

Como conclusión cabe destacar que el flujo inverso complementa al directo. Esto es así ya que el flujo inverso no podría existir si no existe el directo debido a que los productos recuperados se insertan en un nuevo ciclo de producción.

2.2 Aplicaciones actuales de la logística inversa

Tal y como se recoge en el estudio [10], actualmente la logística inversa es aplicada en varios tipos de industria. En dicho estudio se analizan distintas decisiones que se deben tomar en cada una de las áreas estratégicas de la empresa cuando se quiere aplicar la logística inversa, Sin embargo, se pueden destacar una serie de patrones que siempre se cumplen, como lo son:

- **Crear una red entre empresas y clientes**: para que puede existir la logística inversa se debe crear una red donde hayan entidades que necesiten los productos usados para reciclarlos o reutilizarlos y entidades que proporcionen esos productos. Dicha red se crea antes de que haya un flujo de productos, la cual puede ser:
  - **Red privada**: un conjunto de empresas con intereses similares se unen para cooperar con el fin de conseguir que los productos desechados por algunas de ellas sean recicladas por otras. De esta forma consiguen reducir la contaminación y, posiblemente, abaratar costes.
  - **Red pública**: la red se crea a nivel regional, nacional o internacional con el fin de proporcionar un sistema donde las empresas de una determinada área se puedan adherir (o estar obligadas por ley a estarlo) para que puedan depositar los productos desechados y el sistema se encarga de reciclarlos correctamente. En algunos casos, y dependiendo de a que sector esté dedicado el sistema, puede que las propias empresas que forman la red sean las que se encarguen de reciclar los productos.
- **Establecer relaciones**: es necesario establecer un cierto incentivo para estimular a las empresas a participar en el proceso o para hacer que los clientes devuelvan los productos, una vez haya terminado su vida útil, a empresas que participan en el

proceso. Los incentivos deben de ser apropiados para ambas partes, con el fin de crear una situación *win-win*. Dependiendo del tipo de producto, se establece un tipo de incentivo u otro, entre los más extendidos están:

- **Tasa de depósito**: el usuario paga por separado el producto en sí y el objeto usado para su distribución, pudiendo recuperar el dinero del segundo devolviéndolo al distribuidor. Dicho modelo se aplica en ciertos países cuando se venden bebidas, el usuario paga por separado el envase y el contenido del mismo para poder recuperar el dinero pagado para el envase una vez lo devuelva a la tienda.
- **Retirada del producto**: la empresa ofrece un servicio de retirada de productos por un precio bajo o se le ofrece cuando el usuario quiere reemplazarlo por uno nuevo. Con este método, se garantiza que la empresa recupera el producto y, posteriormente, lo reciclará correctamente gracias a la logística inversa. Se suele aplicar con ciertos electrodomésticos en grandes almacenes.
- **Descuento**: similar al anterior pero cuando el producto no necesita transporte, consiste en ofrecer al cliente un descuento al renovar un producto. De esta forma, se asegura que el viejo producto sea reciclado correctamente. Útil para dispositivos electrónicos como teléfonos o reproductores de música.

Gracias a los incentivos, el cliente final tiende a participar ya que le resulta rentable. Por otra parte, la empresa a la que acude le realiza un pago pero que será menor que lo que ella ganará al aplicar la logística inversa sobre dicho producto.

Cabe destacar un caso diferente donde el producto en cuestión no puede ser reciclado, dicho proceso se aplica para asegurarse de que sea correctamente almacenado para su eliminación. En este caso, la empresa lo realiza porque está obligada por ley ya que no le sale rentable.

- **Planificación y control del inventario**: cuando una empresa decide participar en la logística inversa, requiere que revise su planificación y control del inventario. Esto es así ya que ahora cuando una fábrica necesite nuevas materias primas para producir, ya no vendrán directamente, si no que habrá que procesar o reciclar los productos que reciba lo que implica más tiempo. Por otra parte, las empresas encargas de recolectar los productos deberán de tener en cuenta el coste adicional que supone almacenar dichos productos hasta que sean reciclados correctamente.

2.2.1. Crítica al modelo actual de la logística inversa

Se aprecia como el proceso más costoso de realizar es el de crear la red de cooperación entre empresas y clientes. Esto es así ya que requiere que se establezcan relaciones entre diversas empresas, cada una con sus intereses particulares, para que todas trabajen conjuntamente. Además, realizar el diseño es algo costoso, tanto económicamente como de tiempo, por lo que una empresa en particular puede no ser capaz de impulsarlo. Ello conlleva a que las redes privadas que se crean sean de un ámbito muy pequeño donde, en la practica, solo acaban participando la empresa fabricante y la empresa distribuidora.

Por otra parte, cuando la red es pública pueden participar todas las empresas que quieran o pueden estar obligadas a participar. Sin embargo no hay siempre una red pú-

blica que cumpla las necesidades de una determinada empresa ni hay una red para todos los sectores.

Respecto al segundo punto, los incentivos son algo necesarios para conseguir que sea atractivo participar en el sistema de la logística inversa. Sin embargo, si se consigue una red más competente se podría aumentar los incentivos dados y, de esta forma, conseguir que se participe más.

Y, por último, la planificación y control es algo que no se puede variar demasiado debido a que los procesos físicos que se realizan, como la transformación, almacenamiento y transporte, se deberán de seguir haciendo.

2.3 Propuesta de un mercado para gestionar la logística inversa

Como se ha podido destacar en la sección 2.2, la logística inversa es un concepto que está extendido pero que es algo costoso de incorporar entre varias empresas dispuestas a ello.

No obstante, hay ciertos estudios, como los recogidos en [11], donde se destaca que si se elimina esa barrera, un mayor número de empresas estarían interesadas en participar en el reciclaje debido a que se pueden incorporar o retirarse del sistema cuando les parezca conveniente.

En dicho artículo se mencionan los componentes principales del mercado, que serían:

- **Compradores**: empresas que desean participar en el ecosistema con el objetivo de comprar productor para reciclarlos y reincorporarlos en su cadena de montaje.
- **Vendedores**: empresas o particulares que quieren desechar un producto, el precio por el que lo venden sería el incentivo que se ha mencionado en el apartado 2.1.2.
- **Negociaciones**: el mercado estará dotado de diversos procedimientos para llevar a acabo una transacción. Dichos procedimientos tendrán en cuenta la naturaleza de la transacción ya que no todas será iguales, se pueden clasificar en:
  - **Negociación simple**: un comprador compra directamente lo que vende un vendedor.
  - **Negociación compuesta**: diversas entidades cooperan para completar el intercambio.

Para que puedan existir las diversas negociaciones, el mercado debe ser capaz de responder a las ofertas que hagan tanto compradores como vendedores con el fin de que cada uno sepa con que otras entidades existir una oportunidad de acuerdo.

El mercado que se propone, cuyos detalles de diseño e implementación pueden ser consultados en la sección 4, parte de esta idea pero haciendo especial hincapié en las negociaciones entre diversos agentes y en las negociaciones de objetos similares, donde hace falta aplicar algún tipo de transformación para que se produzca el intercambio.

Un ejemplo de esto último sería una situación en la que haya un comprador que quiera acero y un vendedor que venda un coche. Hay un interés de negociación ya que a partir del coche se puede obtener acero, pero puede que el comprador no sea capaz de extraerlo directamente. Es por ello que podrá entrar en escena una tercera entidad que se encargue de comprar el coche, extraer el acero y vender el acero al comprador original.

CAPÍTULO 3

Sistemas multi-agente

En este capítulo se explica en detalle en que consiste el paradigma de programación multi-agente. Dicho paradigma ha sido el usado para implementar el mercado. Se debate también el porqué se ha seleccionado este paradigma y distintas plataformas con las que se podrían implementar el diseño propuesto en el capítulo 4.

3.1 Fundamentos de los sistemas multi-agente

En un sistema multi-agente habitan diversos agentes que interactúan entre ellos. La relación que se establecen entre los mismos pueden ser tanto de cooperación, para obtener un bien mayor, o de competición, para que cada uno obtenga el mejor beneficio. Es por ello que la parte más importante de estos sistemas son los agentes, que, a grandes rasgos, se pueden resumir como un programa *software* que realiza un razonamiento inteligente y ejecuta una serie de acciones.

Sin embargo, los sistemas multi-agente no solo están compuestos por agentes, si no que también por un sistema de comunicación que permita la interacción entre ellos y un lenguaje de comunicación que permita a los agentes entenderse mutuamente.

3.1.1. Agentes

En el ámbito de la informática, un agente se define como un sistema informático capaz de percibir del entorno (mediante observación o mensajes) información para posteriormente razonarla y responder a ella intentado maximizar el resultado que se obtenga. Dependiendo de como sea el razonamiento que siga el agente, dicho agente se le considera más inteligente o menos tal y como se clasificará más adelante.

Como dichos agentes se incorporarán en un sistema multi-agente, las percepciones serán, en su gran parte, mensajes recibidos de otros agentes y sus respuestas serán también mensajes. Sin embargo, las acciones podrían ser también modificar el entorno (lo que causa que el resto de agentes lo perciban). En el caso de que los agentes se integren en robots para dotarles de inteligencia, los humanos veríamos las acciones que ejecuta-

sen para interactuar con el entorno, que serían la forma que tienen de interactuar con nosotros.

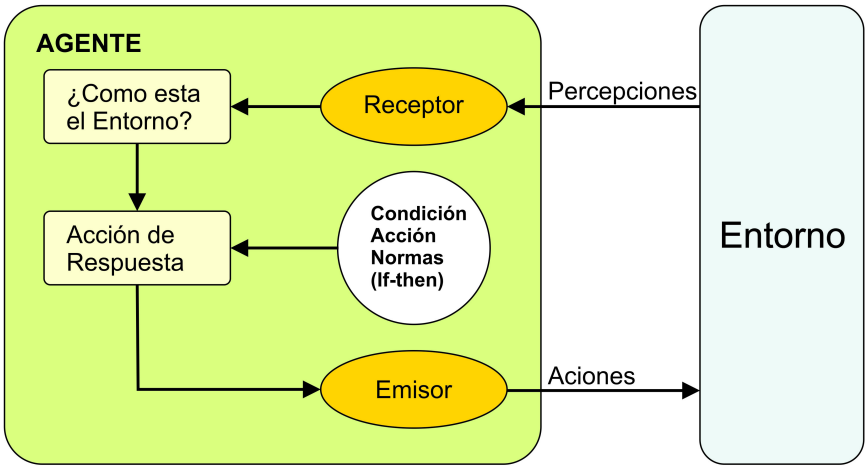


Figura 3.1: Esquema básico de un agente inteligente

Se distinguen cinco clasificaciones para un agente según la inteligencia y las capacidades que demuestre, estas clases son [13]:

- **Agentes reactivos**: agentes con la inteligencia más simple. Perciben a partir de sus sensores y reaccionan en función de las reglas que tengan, sin tener en cuenta percepciones previas ni ningún tipo de razonamiento. Están, por lo tanto, basados en reglas tipo *if A then B*. Funcionan bien cuando el agente puede observar completamente todo el entorno.
- **Agentes reactivos basados en modelo**: son un avance de los previos para entornos donde el agente solo puede observar una porción del mismo. Se distinguen del modelo anterior en que pueden almacenar como era una parte del mundo, el cual no puede ser observada ahora, y como ha podido cambiar en función del modelo del mundo que tenga. En otras palabras, para obtener la acción que tiene que hacer usa tanto información percibida como percepciones previas, las cuales están almacenadas.
- **Agentes basados en objetivos**: extensión de los agentes reactivos basados en modelo donde se tienen unos objetivos que se pretenden alcanzar. Un objetivo es un estado deseable en el que se quiere que el agente se encuentre. Es por ello, que el agente se basará en las percepciones que tenga del entorno (tanto actuales como previas) y del resultado de las distintas acciones que puede realizar para seleccionar la acción que más le aproxime al estado deseado.
- **Agentes basados en utilidad**: si se sigue el enfoque de agentes basados en objetivos, todos los estados en los que se puede encontrar el agente se catalogan como estado objetivo o estado no objetivo. Sin embargo, puede existir un gran abanico de posibilidades entre esas dos clasificaciones. Es por ello que en este enfoque se propone una función de utilidad que sirva para medir cuánto de deseado es el estado en el que se encuentra el agente o cuánto lo será si realiza una cierta acción de la que se conocen sus consecuencias.

- **Agentes que aprenden**: estos agentes aprenden conforme interaccionan con el entorno en el que se encuentran. El aprendizaje lo realizan mediante un proceso de retroalimentación, modificando la acción que se ha realizado en función de lo que se haya conseguido. Estos agentes se vuelven más inteligentes conforme más se relacionen.

3.1.1.1. Arquitectura de los agentes

Las clasificaciones expuestas anteriormente se pueden implementar siguiendo distintas arquitecturas. Las arquitecturas se pueden dividir [14] en:

- **Agentes basados en lógica**: se realizan una serie de deducciones lógicas para deliberar que decisión debe ejecutar. Se emplea una representación simbólica del entorno y de sus objetivos.
- **Agentes reactivos**: la implementación consiste en ejecutar una acción u otra en función de lo que se percibe. Un agente que siga esta arquitectura puede ser visto como una función $f : percepciones \rightarrow acciones$.
- **Agentes BDI (*Belief-Desire-Intention*)**: el agente dispone de una base de datos donde almacena sus creencias, deseos e intenciones. Las creencias representan el conocimiento que dispone el agente a partir del cual podrá razonar, las creencias vienen del entorno, de deliberaciones del agente o a través de las comunicaciones si se trata de un sistema multi-agente. Por otra parte, el agente cuenta con deseos que representan las estados que desea alcanzar. Y, por último, el agente cuenta también con las intenciones que ilustran los deseos que el agente ha decidido alcanzar, de entre todos sus estados que desea alcanzar.
- **Agentes con capas**: en dicha arquitectura, el razonamiento del agente está distribuido a lo largo de una serie de capas. Cada capa se encarga de una naturaleza distinta y tiene prioridades distintas. Por ejemplo, un agente encargado de recoger basura, puede contar con dos capas, una para asegurarse de la integridad del agente, evitando que se caiga o que se choque, y otra capa para detectar la suciedad. Cuando la capa encargada de la seguridad razone que se debe mover a la izquierda para no caerse, el agente realizará ese movimiento aunque la capa de menor prioridad diga de moverse recto ya que ha detectado suciedad.

3.1.2. Sistema de comunicación

Tal y como se ha puesto de manifiesto en el apartado 3.1.1 un agente puede interactuar con el sistema enviado mensajes a otros agentes o realizando una acción que modifique el entorno en el que se encuentra.

La naturaleza de la implementación del sistema dependerá de:

- **Los agentes son simulados por ordenador**: en tal caso todas las acciones e intercambios de mensajes se gestionan mediante *software*. Habrá una distinción si la simulación se hace en el mismo ordenador, donde se hará uso de protocolos como el

IPC para que se pongan en contacto los distintos procesos donde cada uno alberga a un agente; o entre varios ordenadores conectados en red, en dicho caso hará falta que haya comunicación entre ellos usando los protocolos *TCP* o *UDP* para el envío de mensajes.

- **Los agentes se integran en robots físicos**: en este caso los agentes, que son programas *software*, deben controlar al robot, *hardware*. Las acciones que impliquen modificar el entorno, que en este caso será el mundo real, se realizarán mediante el movimiento del robot como mover una caja, coger un objeto... Por otra parte, las acciones que impliquen un intercambio de mensajes con otros agentes se implementarán con diversos estándares para intercambiar información como puede ser el *Wi-Fi* o el *Bluetooth* entre otros.

3.1.3. Lenguaje de comunicación

Para que un conjunto de agentes se considere un sistema multi-agente debe de existir una coordinación entre ellos. Es por ello que hay un gran interés en desarrollar un lenguaje de comunicación entre los agentes que sea los más estándar posible. La ventaja que tendría dicho estándar sería que el lenguaje que se usa sería de propósito general y no solo útil para un contexto determinado.

Por lo tanto, cuando se implementa un sistema multi-agente se disponen de dos alternativas a la hora de tratar la comunicación entre agentes:

- **Desarrollar un lenguaje propio para el ámbito en el que se está trabajando**: hará falta desarrollar completamente el lenguaje, pero puede ser algo sencillo y, por lo tanto, más eficiente, ya que solo se quiere que sea funcional para un contexto muy específico.
- **Usar un lenguaje estándar**: lenguaje de propósito general que podrá ser usado para intercambiar cualquier tipo de información. Sin embargo, puede resultar complejo de usar cuando la información que se desea mandar es simple y no se distinguen acciones, es decir, los mensajes son solo informativos entre los agentes, no hay peticiones de acción o de información.

En cuanto a los estándares que existe, caben destacar:

- ***Knowledge Query and Manipulation Language* (KQML)**: fue el primer estándar en ser desarrollado. Es un lenguaje de comunicación y protocolo orientado al intercambio de información entre agentes gracias al envío de mensajes. Un mensaje en este lenguaje está formado por:
  - **Performativa o acto del habla**: propósito general del mensaje. Como puede ser, entre otros, *tell* y *untell* para decir a un agente que algo es cierto o falso, *ask-if* para hacer preguntas o *achieve* y *unachieve* para dar una orden a otro agente.
  - **Parámetros con su valor**: algunos ejemplos son el contenido propio del mensaje, el receptor, la ontología usada, etiqueta que el receptor debe usar si quiere contestar a dicho mensaje...

Este lenguaje cuenta con lo básico para que el desarrollador puede implementar cualquier protocolo, para ello se tendrá que diseñar el flujo de mensajes y la respuesta que debe de hacer cada agente a cada uno de los mensajes en función del contenido. Al no contar con performativas específicas, como sí que contiene su extensión, todo se gestionará generando o eliminando creencias y deseos de los agentes.

- ***Foundation for Intelligent Physical Agents - Agent Communication Language* (FIPA-ACL)**: consiste en una extensión del KQML. Sus mensajes tienen la misma estructura que en KQML pero dispone de más actos del habla. Dicho aumente se debe a que cuando se diseñó, los creadores contemplaban que sería usado para emplear algunos de los protocolos que ellos mismos habían planteado. Dichos protocolos son el *FIPA-request* para que un agente pida a otro hacer cierta acción, *FIPA-query* para solicitar información o *FIPA-contract-net* para instar a diversos agentes a realizar cierta acción. Las performativas que se incluyen en esta extensión son, por ejemplo, para los casos en los que un agente no entienda la solicitud que le ha llegado *NOT-UNDERSTOOD* o para aceptar y rechazar una propuesta como es el caso de *REJECT-PROPOSAL* y *ACCEPT-PROPOSAL* respectivamente.

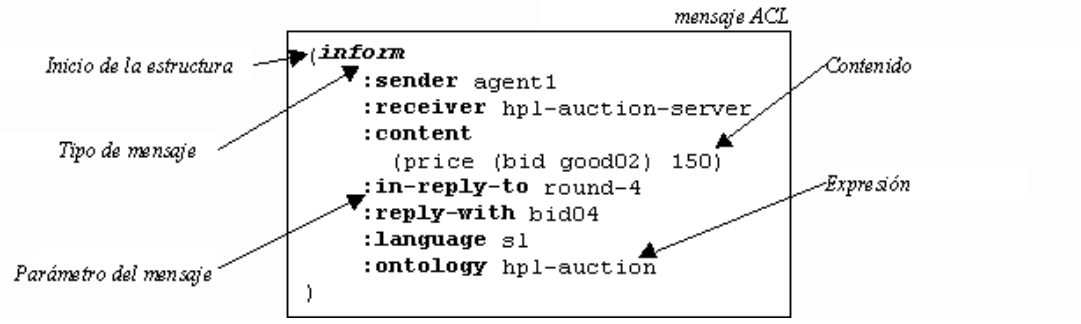


Figura 3.2: Estructura de un mensaje usando FIPA-ACL

Ambos estándares usan una ontología. En el ámbito de la inteligencia artificial, la ontología establece las relaciones entre los conceptos con los que trabajan los agentes y lo que se expresa en realidad. Es decir, sirve para saber cómo se le ha de especificar a un agente sus conocimientos, contempla todo el vocabulario que el agente es capaz de entender. En el caso del ejemplo de la figura 3.2, la ontología, identificada por el nombre *hpl-auction*, sirve para que el agente sea capaz de saber que la creencia *(price (bid good02) 150)* significa que la oferta llamada *good02* tiene un precio de *150*.

3.2 Ventajas de los sistemas multi-agente frente a otras técnicas

Como se ha destacado previamente, dichos sistemas están compuestos por varios fragmentos de código que se ejecutan todos concurrentemente y se relacionan entre ellos. Es por ello que la computación está completamente distribuida frente a otros paradigmas donde la computación es secuencial (aunque algunas fracciones de un algoritmo se pueda paralelizar).

Los sistemas multi-agente se distinguen por el hecho de que los agentes son:

- **Autónomos**: los agentes son capaces de llevar a acabo ciertas acciones sin que haya ningún intermediario.
- **Descentralizados**: ningún agente controla al resto. Sin embargo, como se verá más adelante, pueden existir ciertos agentes con alguna función muy específica.

Existen ciertos problemas que se podrán resolver gracias a este paradigma pero no con otros. Lo son, sobre todo, problemas donde existe una gran cantidad de entidades todas primando su interés particular. Usando esta técnica, y en particular el enfoque de agentes basados en objetivos expuesto en el apartado 3.1.1, se pueden resolver dichos problemas debido a que cada agente tendrá unos deseos que querrá conseguir.

3.3 Elección del sistema multi-agente como paradigma para el mercado

Para implementar el objetivo de este trabajo, el mercado para gestionar la logística inversa expuesto en el apartado 2.3, se ha decidido utilizar un paradigma multi-agente. Esto es así ya que el mercado debe de tener una serie de características que ofrece este paradigma como por ejemplo:

- **Diversas entidades que cooperan pero primando sus intereses**: el mercado deberá de contar con una serie de compradores y vendedores, pero cada uno deberá de ser guiado por sus intereses propios, no por el bien común. Por lo tanto, los agentes basados en objetivos o en una función de utilidad son los más apropiados.
- **Negociaciones entre las entidades**: para que se lleve a cabo un intercambio, ya sea simple o compuesto, hace falta que exista una serie de negociaciones entre las entidades. Dichas negociaciones se implementarán como comunicación entre los agentes.
- **Priorizar las ofertas según el beneficio a obtener**: una demanda de un comprador se puede satisfacer comprando el producto necesario a diversos vendedores que lo ofrecen. Sin embargo, en función a una serie de parámetros, el agente deberá razonar para saber cuál, de entre todas las opciones disponibles, le interesa más.

Sobre la arquitectura que se empleará, se utilizará la arquitectura BDI debido a que las distintas entidades del mercado tienen una serie de productos con los que se quiere negociar y tienen como objetivo cumplir con sus peticiones, por lo tanto, dicha arquitectura es la más apropiada ya que existe una correspondencia entre las partes del BDI y las exigencias del mercado:

- **Creencias (*Beliefs*)**: productos con los que se quieren negociar. En el caso de los vendedores serán los productos que se quieren vender y en el caso de los compradores los que se quieren comprar. El agente sabrá el precio dispuesto a pagar o mínimo que le paguen, su composición y cantidad.

- **Deseos (*Desires*)**: los diversos agentes querrán, en el caso de los vendedores conseguir un comprador para cada uno de sus productos y en el caso de los compradores querrán conseguir todos los productos que desean por un cierto precio.
- **Intenciones (*Intentions*)**: producto y agente con el que querrá empezar una negociación. Cuando un agente sepa con cuales puede empezar una negociación, tendrá que deliberar para saber la opción que más le interesa.

3.4 *Frameworks*

Cuando se quiere desarrollar un sistema multi-agente se debe seleccionar una plataforma. Entre todas las disponibles vamos a centrarnos en las que sigan una arquitectura BDI para el agente, como lo son:

- ***JADE***: este *framework* está implementado sobre *Java*, se diseñó para que cumpla con las especificaciones FIPA [4] donde se especifica el ciclo de vida de un agente, el cual se divide en una serie de estados (inicialización, activación, suspensión, espera, eliminación y transición). Por otra parte, se define también el concepto de comportamiento, que sirve para responder a un evento.

  Al tener que programar el agente usando un lenguaje de programación de propósito general, se escribirá más código ya que el lenguaje no ha sido diseñado para que únicamente realice un pequeño conjunto de instrucciones, y, por lo tanto, el programador no se puede centrar únicamente en el comportamiento del agente. Sin embargo, se podrá programar todo lo que se desee y se tendrá un control completo de todos los procesos que realiza el agente.
- ***JASON***: es una extensión del lenguaje de programación *AgentSpeak*, el cual tiene como principal objetivo ser utilizado para programar el comportamiento individual de los agentes. Dichos agentes pueden cooperar entre ellos mediante el envío de mensajes o modificando el entorno. Cuenta con algunas funciones básicas implementadas para el tratamiento de los datos, pero permite al desarrollador poder añadir nuevas o modificar alguno de los componentes actuales. Destaca frente al resto por ser agente orientada, es decir, el programador solo se centra en el razonamiento lógico del agente, abstrayéndose de la arquitectura, la cual se maneja internamente por el mismo *framework*.
- ***JACK***: dicha plataforma ha desarrollado su propio lenguaje que permite especificar el comportamiento del agente, acciones, eventos... Dispone también de una serie de comandos ya desarrollados y dota al programador de la posibilidad de usar *Java* para definir los suyos propios. Cuenta además con una interfaz gráfica lo que facilita la depuración del software desarrollado. Sin embargo, la licencia de dicho *Framework* no es pública.

3.4.1. Elección de un *Framework*

Para el desarrollo del prototipo, el cual se explica en el apartado 4, se ha seleccionado el *framework JASON* ya que nos permite centrarnos más en el desarrollo de la lógica y

razonamiento del agente sin tener que preocuparnos por la arquitectura del agente, es decir, de la inicialización, envío y recepción de mensajes, finalizado de un agente, entre otros, se encarga la propia plataforma, aunque si que permite la capacidad de poder modificarla. Además, dicho *Framework* es de licencia pública, frente a *JACK* que dispone de una licencia de *software* privativo.

Por otra parte, el lenguaje de programación con el que trabaja, *AgentSpeak*, es un lenguaje de programación lógico que permite expresar muy cómodamente las creencias del agente, así como también los deseos. Es por ello que dispone de, entre otros, comandos para añadir, eliminar o modificar una creencia o deseo, una sintaxis sencilla para expresar la respuesta de un agente frente a un evento, que es el hecho de que una creencia o deseo sean añadidas o eliminadas, o se reciba un mensaje de otro agente.

Sin embargo, habrán algunos procedimientos que no se podrán programar tan fácilmente usando *AgentSpeak*, como por ejemplo, ordenar una lista. Para conseguirlo, se podrán definir una serie de acciones internas para extender el conjunto de ordenes proporcionadas. Dicho nuevos comandos se programarán usando *Java*.

CAPÍTULO 4

Desarrollo del mercado propuesto

En esta unidad del trabajo se propone una posible solución para el problema de logística inversa expuesto en el capítulo 2, para ello se utilizan los sistemas multi-agente explicados en la sección 3. En este capítulo se atraviesan todas las etapas que posee el desarrollo de *software*.

Para desarrollar el mercado primero se ha hecho un análisis detallado de todo lo que necesita el *software* para que pueda ser considerado un mercado virtual útil para este problema. Seguidamente se ha realizado un diseño que cumpla todos los requisitos expuestos en la fase de análisis. A continuación se ha implementado el diseño que se ha obtenido y, por último, se han considerado diversas pruebas de validación para comprobar que la aplicación funciona tal y como se ha diseñado.

4.1 Análisis

El *software* que se desea desarrollar debe de seguir la idea propuesta en [11] y explicada en el apartado 2.3. Dicha aplicación debe de servir para que una serie de compradores y vendedores puedan satisfacer sus necesidades, pero haciendo especial hincapié en que los diferentes productos con los que se comercialice se puedan componer y descomponer. Además, se pretende desarrollar también una interfaz gráfica para simplificar su uso. Para ello, primero vamos a extraer los requisitos que se deben cumplir.

4.1.1. Mercado

- Debe existir una entidad que corresponda con los compradores.
- Ha de haber una entidad que represente a los vendedores.
- Los diferentes productos con los que se negocien deben de ser descritos según una ontología para que se sepa lo que representan y a los agentes a los que les podrían interesar.
- El sistema debe de identificar las oportunidades de negocio y transmitirlas para que se empiece a negociar. Se establecerán negociaciones entre los compradores y

vendedores en forma de puja para que el precio dependa de la oferta y demanda que exista. Dichas negociaciones tienen que cubrir los tres escenarios posibles[1]:

- **Correspondencia directa**: un vendedor dispone exactamente del objeto que desea un comprador, la compra se realiza entre ellos de forma inmediata.
- **Alianza necesaria**: hace falta que varios vendedores se alíen para satisfacer la demanda de un comprador. Este caso ilustra situaciones donde un comprador desea una cantidad muy elevada de un cierto objeto, pero los vendedores solo poseen porciones más pequeñas, hace falta, por lo tanto, varios vendedores para satisfacer una única demanda.
- **Descomposición requerida**: existe un comprador al que le interesa un componente de un objeto que tiene un vendedor, en esta situación hará falta descomponer el objeto para obtener todos sus componentes y luego negociar con los distintos componentes.

- Al acabar la ejecución se deben saber las negociaciones satisfactorias que han habido, así como estadísticas de ganancias y contaminación[2].
- Cada entidad será parametrizable para que pueda tener en cuenta diversos factores durante su negociación. Dichos factores serán por una parte para primar las ganancias, primar la baja contaminación o ambas, y por otra parte para que cada agente tenga una estrategia distinta a la hora de realizar las diferentes ofertas.

4.1.2. Interfaz gráfica

- Se debe poder especificar la ontología que se usará para describir los distintos objetos con los que las diferentes entidades podrán negociar.
- Se podrán crear diferentes compradores y vendedores y, para cada uno de ellos, se tiene que poder determinar los objetos que cada uno de ellos tenga.
- Cada agente se podrá paremetrizar para primar sobre el factor económico, medioambiental o ambos, y para estipular la estrategia de negociación que va a seguir.
- Las estadísticas proporcionadas por el mercado se visualizarán de manera gráfica para poder comparar diferentes situaciones.

4.2 Diseño

En este apartado se describen y argumentan todas las decisiones de diseño, tanto del mercado como de la interfaz, que se han tomado como paso previo a la implementación del proyecto.

[1] Aunque pueden existir más situaciones, las descritas son las más comunes.

[2] La contaminación se mide en función de la distancia a la que están el comprador y vendedor, debido a que la contaminación se producirá durante el transporte del producto. Los procesos de transformación también pueden contaminar pero, para simplificar el trabajo, no se han tenido en cuenta. Sin embargo, no sería complejo añadirlos una vez dicha información esté definida.

4.2.1. Mercado

Tal y como se ha explicado a lo largo del capítulo 3, pero especialmente en el apartado 3.3, para desarrollar este *software* se ha razonado hacerlo mediante un sistema multi-agente. Para ello, en esta sección se definen los roles que se han identificado, la ontología para definir a las ofertas y peticiones, el diagrama de clases completo del mercado, el procedimiento de razonamiento estudiado para implantar distintas estrategias y los diagramas de interacción para que cada agente se pueda comunicar con el resto y sea contestado.

Cabe destacar que el diseño y procedimientos explicados a lo largo de esta sección no son los únicos, se va a proponer un diseño válido que cumpla todos los requisitos extraídos de la fase de análisis, diseño al que se lo podrían cambiar cada uno de sus componentes en el futuro.

4.2.1.1. Agentes

En este apartado se definen los roles que se han identificado en el mercado junto a las creencias, objetivos y tareas que tienen cada uno de ellos. Cabe mencionar que al identificar y tratar individualmente cada una de las tareas particulares que posee cada rol, se está proponiendo un diseño modular de los agentes. Lo que conlleva que se pueda alterar un cierto comportamiento sin que haya que modificar completamente al agente.

Para cumplir con los requisitos expuestos, se han identificado 4 roles distintos:

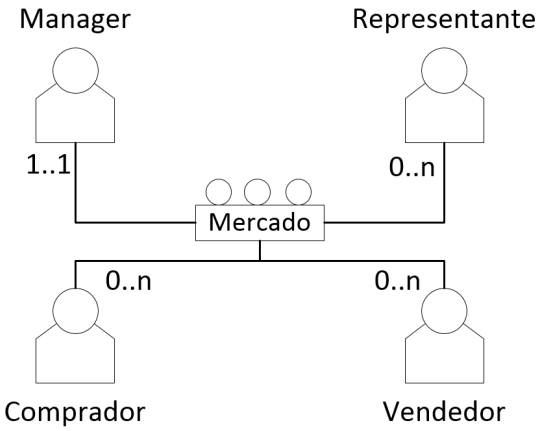


Figura 4.1: Diagrama de los roles del sistema multi-agente

- ***Manager***: agente que representa al mercado, su principal objetivo es el de coordinar y el de poner en contacto a los distintos agentes para que se realicen las negociaciones. Este agente interviene únicamente en una serie de situaciones:
  - **Identifica las coincidencias**: Recibe los productos que tienen los *vendedores* y los deseos de los *compradores* para poder realizar el proceso de comparación y, posteriormente, notifica a los *vendedores* sobre los *compradores* que estarían interesados en sus productos.

- **Intermediario para formar una alianza**: cuando un *vendedor* quiere intentar formar una alianza para satisfacer la demanda de un *comprador*, se lo notificará al *manager*, el cual se encargará de formarla cuando hayan los suficientes miembros para componer la petición que se pretende conseguir. Como se explica más adelante, cuando haya una posible alianza, se creará un nuevo agente, el *representante*, de esta manera, el nuevo agente será neutral y no primará sobre los intereses particulares de ningún agente.

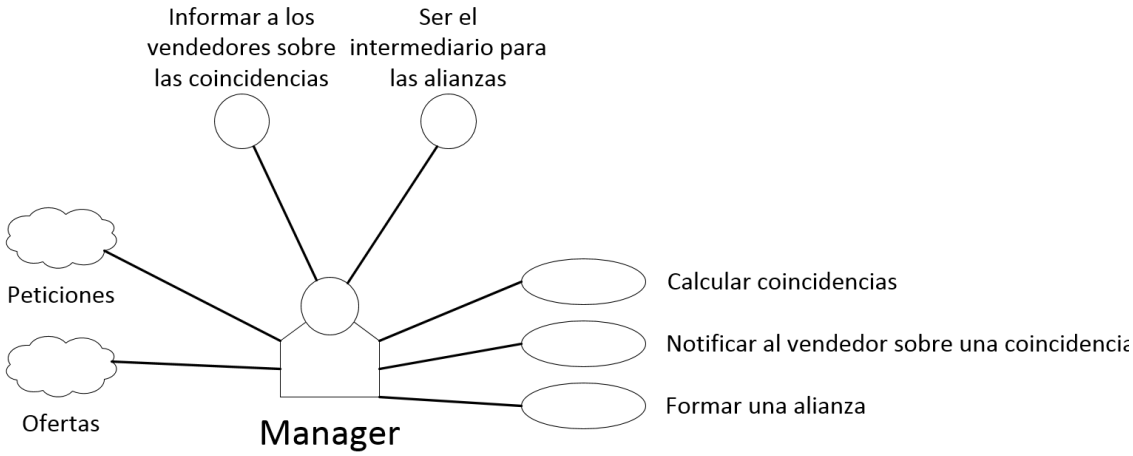


Figura 4.2: Esquema del rol *Manager*

- **Comprador**: este tipo de agente tendrá una serie de objetos que quiere conseguir, para ello, serán capaces de recibir una oferta y responder a ella, en función a las ofertas previas que le hayan hecho. El agente primará pagar lo menos posible por una oferta, para ello se pueden seguir diversos procedimientos, para este diseño se ha decidido hacer uso de la puja inglesa, es decir, cuando el agente reciba dos ofertas para un mismo objeto, se quedará con la que le cueste menos dinero pero notificará a la oferta no ganadora una contra-oferta, dándole la oportunidad de poder efectuar otra oferta en el futuro. El proceso se realizará iterativamente hasta que transcurra un cierto tiempo sin recibir ninguna oferta mejor, en dicho caso, el agente decidirá que la oferta más ventajosa hasta el momento es la ganadora final.

  Sin embargo, para que las ofertas no sea infinitas y para que se asemeje más a situaciones reales, el *comprador* marcará un número máximo de ofertas que cada *vendedor* le puede hacer para cada una de sus peticiones. De esta forma, la estrategia que siga cada *vendedor* será crucial para saber la cuantía de dinero que tiene que ofertar en cada ronda.

  Por otra parte, al inicio de la ejecución, el agente tendrá que notificar al *manager* sobre los objetos que desea satisfacer con el objetivo de que el mercado sea capaz de identificar las coincidencias entre *compradores* y *vendedores*.

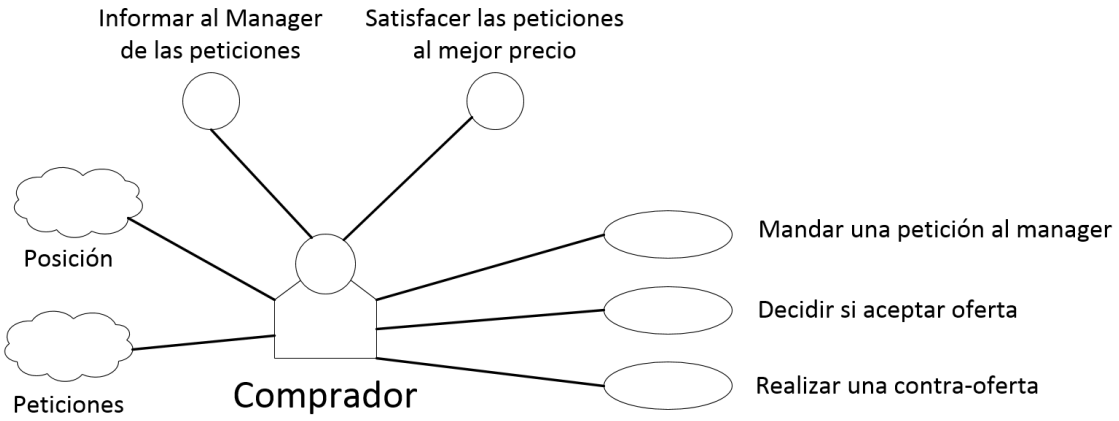


Figura 4.3: Esquema del rol *Comprador*

- **Vendedor**: es la entidad que se encarga de empezar todas la negociaciones. Primero, al igual que el *comprador*, debe notificar los productos que tiene al *manager*. Posteriormente, recibirá del mismo todas las peticiones de los *compradores* con las que puede negociar y, seguidamente, el agente tendrá que ordenarlas según una prioridad, tema que se profundizará en el apartado de 4.2.1.5, para saber cual es la negociación que le es más ventajosa según unos factores.

  Se seleccionará la oferta que el *vendedor* considere mejor y empezará la negociación con ella. En este punto, la negociación que está a punto de comenzar se clasifica en uno de los tres grupos mencionados en el apartado 4.1.1:

  - **Correspondencia directa**: el *vendedor* negocia directamente con el comprador. Le hace una oferta inicial y el *comprador* le puede contestar:
    - **Oferta ganadora**: el *comprador* adquiere el producto del *vendedor*, por lo tanto, el *vendedor* ya no podrá negociar más con él.
    - **Hay una oferta mejor**: el *comprador* le notifica sobre la oferta que, por el momento, es la ganadora. El *vendedor* tendrá que reevaluar sus posibilidades de negocio para saber si puede hacer una nueva oferta, pero pidiendo menos dinero, o debe probar con otro *comprador*. Este procedimiento, la estrategia de negociación, en particular se explica con más detalle en la sección 4.2.1.4.
    - **Producto ya conseguido**: el *comprador* le informa de que el producto que le ha ofrecido ya no le interesa debido a que, previamente, otro *vendedor* se lo ha ofrecido y lo ha comprado.
  - **Alianza necesaria**: el *vendedor* necesita aliarse con otros para componer una oferta que pueda ser adquirida por un *comprador*. Para ello, notificará al *manager* de ello, el cual se encargará de llevar a acabo la alianza. En este caso, el *manager* podrá responder:
    - **Alianza satisfactoria**: el *manager* le notifica que la alianza que se ha formado con el producto del *vendedor* ha conseguido vender el producto compuesto. Por lo tanto y al igual que ocurría en el caso de la *Correspondencia directa*, el *vendedor* ya no podrá negociar con él debido a que se acaba de vender.
    - **Alianza imposible**: el *comprador* ha rechazado todas las alianzas que se han formado con las exigencias de precio del *vendedor*. Debido a ello, y

de manera análoga al caso previo de *Hay una oferta mejor*, el *vendedor* debe volver a evaluar si le conviene intentar formar una alianza pero pidiendo menos o si debería intentar negociar con otro agente.

- **Producto ya conseguido**: el deseo del *comprador* ya ha sido satisfecho, así que el *comprador* ya no quiere otro producto similar.

- **Descomposición requerida**: en este caso, el *vendedor* posee un objeto que se puede descomponer y el *manager* le ha dicho que hay uno o más *compradores* interesados en al menos uno de los componentes del producto. El *vendedor* deberá de razonar si descompone el producto, lo que conlleva un precio, para poder negociar con sus partes. En el caso de que decida hacerlo, la negociación de cada uno de los componentes se efectuará según el grupo en el que se clasifique dicha parte *Correspondencia directa* o *Alianza necesaria*

Siempre que el agente tenga que reevaluar una cierta petición de un *comprador* para saber el dinero que le debe de ofrecer en la siguiente ronda, tendrá que tener en cuenta el dinero mínimo que quiere conseguir, el dinero máximo que estima por el cual puede vender su oferta, la ronda en la que se encuentra y el número de ofertas distintas que permite el *comprador* para esa petición. Este tema se profundiza en el apartado 4.2.1.4.

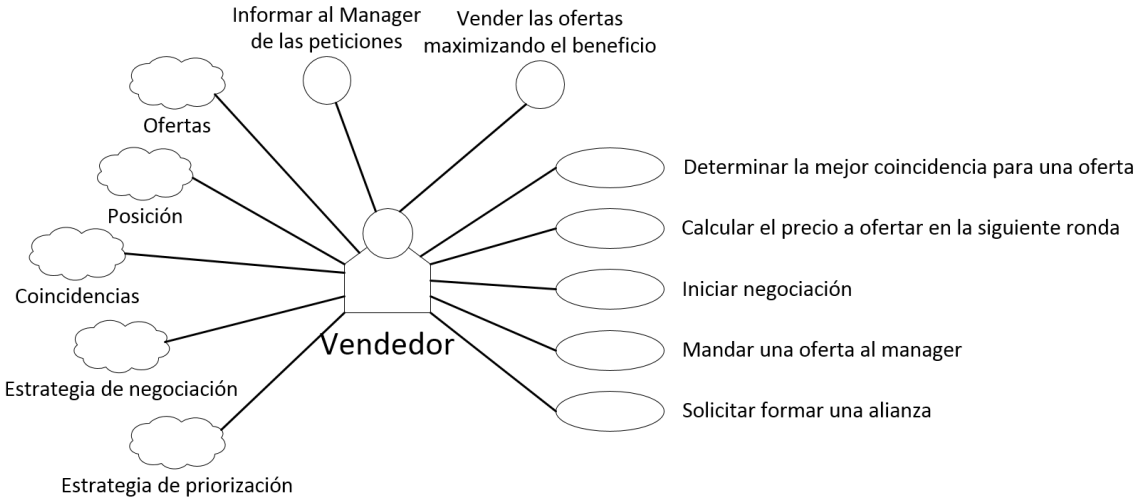


Figura 4.4: Esquema del rol *Vendedor*

- **Representante de alianza**: es creado por el *manager*, tiene una vida muy corta y una función muy concreta. El *representante* se crea cada vez que el *manager* identifica que diversos *vendedores* interesados en una misma petición se pueden agrupar en una alianza. El *representante* sabrá la petición que debe comprar y la cantidad de dinero que tiene que ofrece, que será la suma de lo que piden cada uno de los miembros de la alianza.

  Como la alianza se ha formado para poder satisfacer una petición concreta de un *comprador*, el *representante* negocia con un objeto que se corresponde directamente con lo que pide el comprador. Así que, el representante negocia de manera similar a como lo hace el vendedor en el caso de *Correspondencia directa*, sin embargo, en el caso de *Hay una oferta mejor*, el representante no puede rebajar su precio, si no que se debe disolver la alianza notificándolo al *manager*, el cual se encargará de

transmitírselo a sus miembros y, si todos sus miembros razonan que sigue siendo la mejor opción, se volvería a crear la misma alianza pero pidiendo menos dinero.

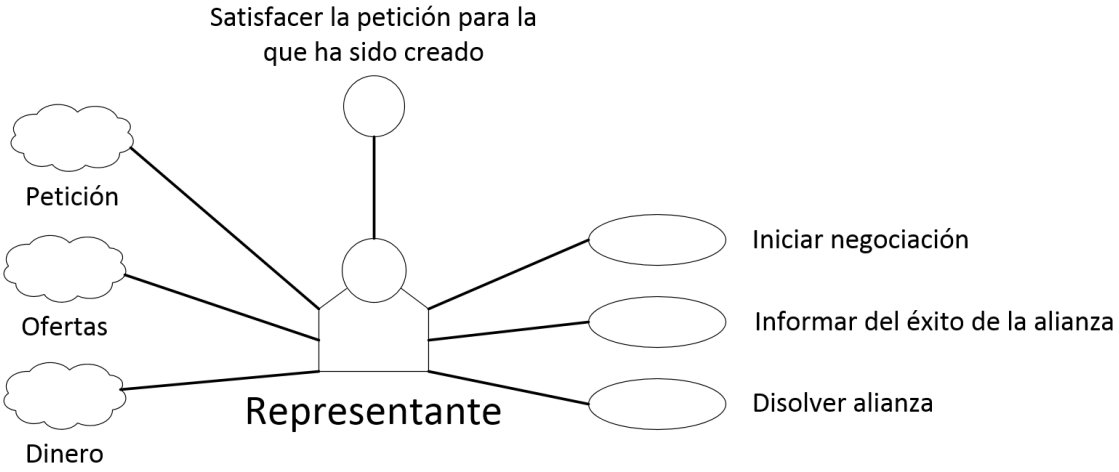


Figura 4.5: Esquema del rol *Representante*

4.2.1.2. Ontología: descripción de los productos

Se requiere definir una ontología para que todos los mensajes y conocimiento que se transmita entre los diferentes agentes sean completamente entendidos. Uno de los puntos clave de este proyecto es la definición y caracterización de cada uno de los diferentes productos o peticiones que pueden existir. En este subapartado se expone el diseño que se ha decidido para representar a los distintos objetos, el cual se usa en la fase de implementación.

Los productos con los que puede negociar un *vendedor* se componen por una serie de atributos, pero distinguiendo entre objeto sencillo y objeto a descomponer:

- **Objeto simple**: como objeto simple se entiende a todas aquellas materias primas que son de interés directo para algún *comprador*. Se caracterizan por:
  - **Material**: identificador de un material al que representa. Los posibles identificadores deben de estar definidos previamente, según el ámbito de uso donde se usa el mercado, para que sean conocidos por *vendedores* y *compradores*.
  - **Cantidad**: gramos del material.
- **Objeto compuesto**: son objetos con los que no se puede negociar directamente, hará falta descomponerlos en objetos simples, con los que ya se podrá negociar. Un objeto compuesto viene definido por:
  - **Ítem**: identificador que corresponde a un objeto compuesto, del que se conocen los diferentes objetos simples puede generar. Al igual que con el material, los posibles identificadores deben de estar definidos previamente.
  - **Unidades**: número de objetos representados por el ítem que forman el objeto compuesto.

Cabe mencionar que las peticiones de los *compradores* tendrán la forma de objeto simple. Ya que para cada petición habrá que definir el material y la cantidad que se quiere comprar, al igual que en la definición de un objeto simple.

En último lugar cabe mencionar a las *transformaciones*, conocimiento que dispone el *manager* para saber los objetos simples en los que se puede descomponer cada objeto compuesto. Cada transformación está compuesta por:

- **Ítem**: identificador que será usado por los objetos compuestos para saber lo que representan.
- **Coste**: cantidad de dinero que debe abonar el *vendedor* para que se le aplique la transformación.
- **Objetos simples**: lista de los objetos simples (formados por el material y la cantidad) en los que se podrá transformar el objeto compuesto en cuestión.

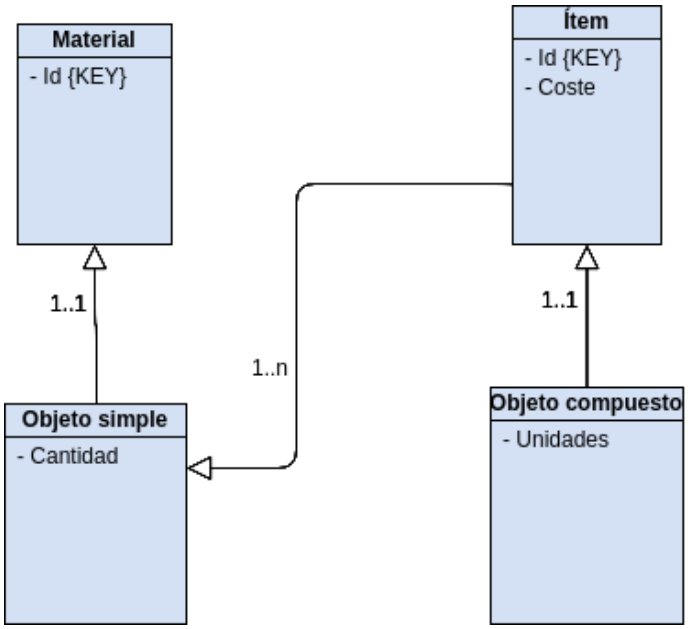


Figura 4.6: Diagrama de las clases empleadas para representar a la ontología

El esquema de la figura 4.6 representa el conjunto de clases necesarias para poder almacenar correctamente una producto según la ontología representada en este apartado. El diagrama en cuestión se extiende en el siguiente apartado junto a todas las clases necesarias para representar a las ofertas, peticiones y los distintos agentes.

4.2.1.3. Diagrama de clases

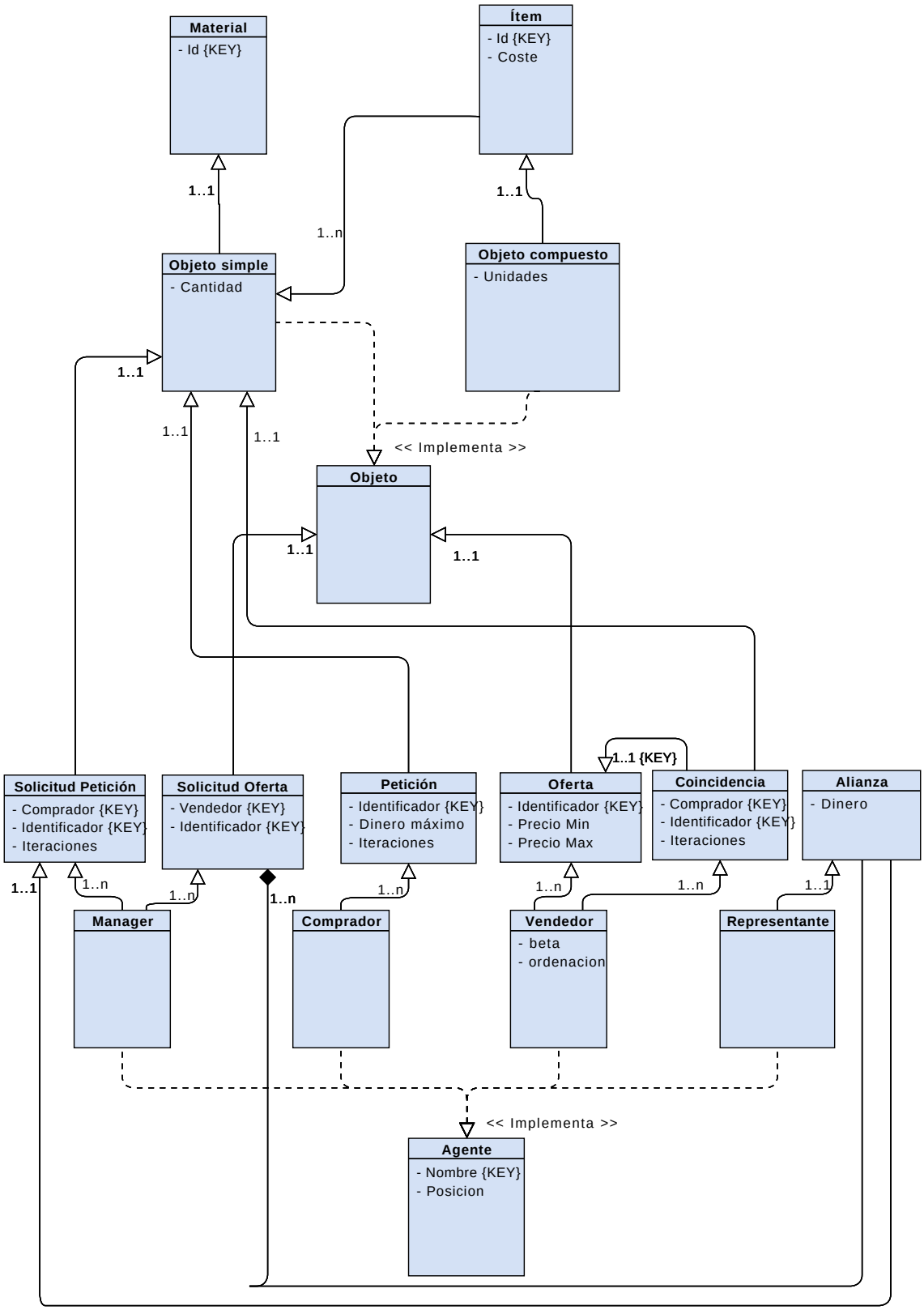


Figura 4.7: Diagrama de las clases empleadas por el mercado

La figura 4.7 corresponde al diseño de clases que tiene el mercado. Como se aprecia, cada clase representa a un posible agente del mercado o a las distintas ofertas y peticiones con las que negocia. Su clases son:

- **Material**: representa a cada una de las materias primas que pueden existir en el mercado, tiene asociado un identificador para poder distinguirlos.
- **Ítem**: sirve para caracterizar a cada uno de las entidades que se podrán descomponer, tiene vinculado un identificador para distinguirlos, una lista de *Objetos simples* en los que se descompone y el precio de dicha descomposición
- **Objeto**: interfaz utilizada para englobar a los *Objetos simples* y a los *Objetos compuestos*.
- **Objeto simple**: cada uno de los objetos con los que se podrá negociar directamente, compuestos por el material que son y su cantidad.
- **Objeto compuesto**: simboliza a un objeto que se tiene que descomponer, tiene asociado un identificador de la clase *Ítem* para saber en lo que se puede descomponer (gracias a la ontología definida) y las unidades individuales que forman ese conjunto.
- **Petición**: representa lo que desea conseguir el *comprador*, esta formado por el identificador por el que el *comprador* distingue las peticiones, un objeto de la clase *Objeto simple* para describir su petición, el número máximo de ofertas distintas que permite por cada *vendedor* y el dinero máximo que está dispuesto a pagar.
- **Oferta**: de manera análoga, los productos vendidos por un *vendedor* se almacenan en esta clase. Compuesta por el identificador para diferenciar unas ofertas de las otras, un objeto de la clase *objeto* (ya sea simple o compuesto) y las cantidades mínima y máxima de dinero que pretende conseguir, parámetros que junto a β del *vendedor* será usados para calcular la oferta a realizar, explicado en el apartado 4.2.1.4.
- **Solicitud Petición**: clase utilizada por el *manager* para almacenar cada una de las peticiones que reciba de cada *comprador*. En ella se tiene que almacenar el *Objeto simple* que representa la petición del agente en sí, el *comprador* para que al calcularse las coincidencias y se las envíe al *vendedor*, este sepa a que agente se debe dirigir; por el identificador por el que el *comprador* en cuestión distingue a sus distintas peticiones y, por último, el número de iteraciones que permite como máximo el *comprador* para que el *vendedor* lo sepa de antemano y no le tenga que pedir dicha información al *comprador*.
- **Solicitud Oferta**: similar a la clase previa, sirve para almacenar las ofertas recibidas por un *vendedor*. Compuestas por el *Objeto* que lo forma (ya sea simple o compuesto) junto al *vendedor* y al identificador para que el *manager* sepa al agente al que le tiene que enviar las coincidencias y para que el *vendedor* conozca la oferta a la que se está refiriendo.
- **Coincidencia**: los objetos de la clase en cuestión se almacenan por el *vendedor* cada vez que el *manager* le notifique de que ha calculado una coincidencia. Esta clase está formada por un objeto de la clase *Oferta* el cual representa el objeto que le va a intentar vender al *comprador*, el *comprador* y el identificador que se usa para saber al agente y la petición en concreto a la que se tiene que dirigir, el número de iteraciones máximas que permite el agente y un objeto de *Objeto simple* que sirve para describir

la petición del *comprador* con la que se ha encontrado una coincidencia, interesa de ella saber la cantidad para compararla con la cantidad de la oferta con el fin de saber si hace falta una alianza o no, este procedimiento se explica en profundidad a lo largo del apartado 4.2.1.5.

- **Alianza**: sirve para representar la alianza a la que representa el *Representante*, esta formada por la cantidad de dinero por la que debe vender el objeto que representa, un conjunto de *Solicitud Oferta* de los objetos que componen la alianza y una *Solicitud Petición* para saber la oferta que debe intentar satisfacer.
- **Agente**: interfaz de agente, contiene todos los métodos y atributos necesarios para que dicha clase se comporte como un agente autónomo. Además dispone del atributo de *posición* que es usado por los agentes para representar su ubicación.
- **Manager**: agente encargado de encontrar las coincidencias y de formar las alianzas. Posee dos listas: una lista de objetos tipo *Solicitud Petición* donde almacena todos las peticiones enviadas por un *comprador* y una lista de *Solicitud Oferta* para guardar los objetos remitidos por cada *vendedor*.
- **Comprador**: agente que tiene como objetivo conseguir un producto que satisfaga cada una de sus peticiones, esta formado, por lo tanto, por una lista de objetos de la clase *Petición*
- **Vendedor**: agente que debe vender todos los productos que tiene (tantos simples como compuestos), por lo que tiene una serie de objetos de tipo *Oferta*. Además, para satisfacer dichas ofertas, recibe del *Manager* una serie de coincidencias que guarda en una lista de objetos de la clase *Coincidencias*. Dispone también del parámetro β usado para saber la estrategia de negociación que va a seguir, tal y como está explicado en el apartado 4.2.1.4. Por último, cuanta también con una variable que determina la ordenación que sigue al priorizar las distintas coincidencias que encuentre, dicho procedimiento se explica en profundidad en la sección 4.2.1.5, en concreto en el diagrama *oferta de un vendedor para correspondencia directa*.
- **Representante**: agente que simboliza a una alianza cuyo principal objetivo es satisfacer una única petición mediante la composición de una serie de ofertas de distintos vendedores. Por lo tanto, tiene un único objeto de la clase *Alianza*.

Cabe destacar, que tanto las peticiones como las ofertas almacenadas por el *manager* son distintas a las almacenadas por los *compradores* y *vendedores* respectivamente. Esto es así ya que el dinero exigido u ofertado es conocimiento que solo conoce el propio agente y por otra parte, se debe almacenar el agente al que pertenece cada solicitud para que el *manager* se pueda poner en contacto con ellos.

En el diagrama de clases se han omitido las operaciones que disponen las clases de tipo agente ya que se explican en la subsección 4.2.1.5.

4.2.1.4. Razonamiento del precio a ofrecer por los vendedores

Como ya se ha mencionado en el apartado 4.2.1.1, todas las negociaciones las inician los *vendedores* así que ellos son los que tienen que razonar para saber cuanta cuantía deben ofrecer en cada iteración. Cada agente sabe para cada objeto que disponga cual es la

cantidad mínima de dinero por el que lo quiere vender y el dinero estimado máximo que puede ganar, que será su valor de partida. Si cada *vendedor* pudiera hacer un número infinito de ofertas, cuando fracase en una ronda, en la siguiente pedirá la misma cantidad pero rebajada en una unidad.

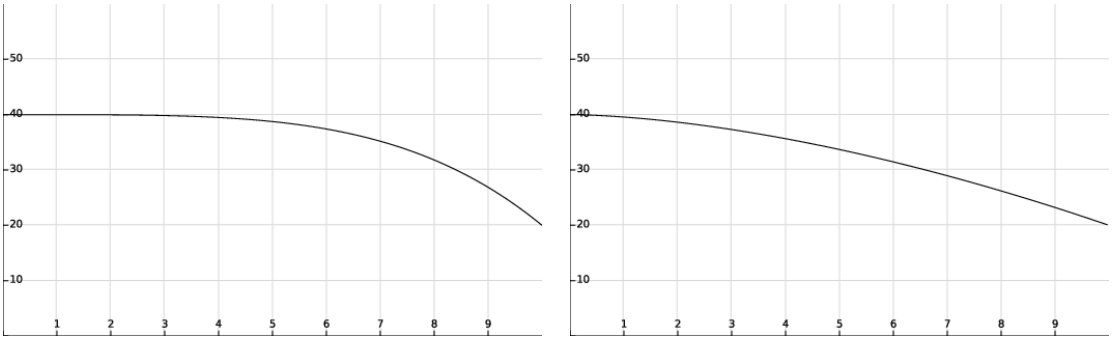
Sin embargo, para evitar dicho comportamiento y para que se asemeje más a situaciones reales, se permite que el *comprador* determine un número máximo de rondas, es decir ofertas distintas, en las que podrá participar cada *vendedor* para cada una de sus peticiones. De esta forma, el decremento de la cuantía exigida ya no será de una unidad. Hecho que permite que los agentes tengan un comportamiento específico.

Se define una función que calcule la variación de las ofertas de los agentes en cada ronda. Dicha función hará que la estrategia del agente sea distinta en base a un parámetro. La función en cuestión está extraída de [15]:

$$PMIN + (1 - s(t)) * (PMAX - PMIN) \qquad s(t) = (\frac{t}{T})^{\frac{1}{\beta}}$$

Donde:

- $PMIN$: cantidad mínima por la que el *vendedor* está dispuesto a ofrecer su producto.
- $PMAX$: cantidad de dinero máxima que el *vendedor* estima que podrá ganar al vender su objeto.
- t: número de la ronda en la que se encuentra el *vendedor*.
- T: número de ofertas distintas que el *comprador* permite para la petición en cuestión procedentes del mismo *vendedor*.
- β (valor que define el comportamiento): en función de su valor, el comportamiento del agente será:
  - $\beta = 1$**:** comportamiento lineal, los descensos entre iteraciones son siempre los mismos.
  - $0 < \beta < 1$: comportamiento exigente, el agente varía poco su oferta en las primeras iteraciones para intentar ganar lo máximo posible pero en las últimas rondas desciende más rápido para acercarse a la cantidad mínima.
  - $1 < \beta < \infty$: comportamiento benévolo, el agente baja mucho su oferta en las primeras rondas con el fin de ofrecer su producto de manera más económica y, por lo tanto, ser el ganador. Sin embargo, como ya ha descendido mucho su oferta, en las siguientes rondas varía poco el precio.



(a) $\beta = 0{,}25$ **(b)** $\beta = 0{,}6$

Figura 4.8: Ejemplos de la función con un valor de comportamiento exigente para β

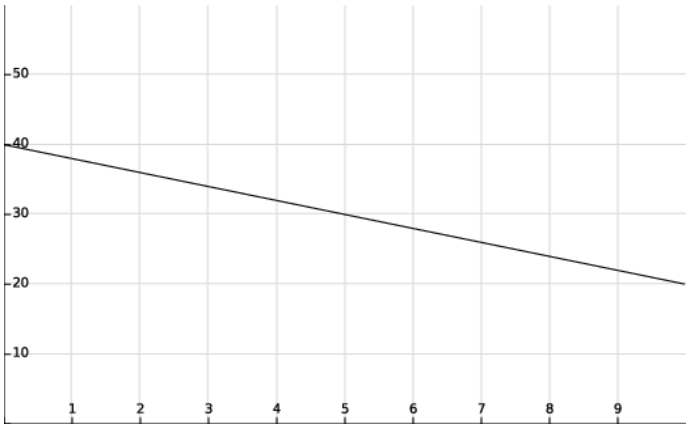
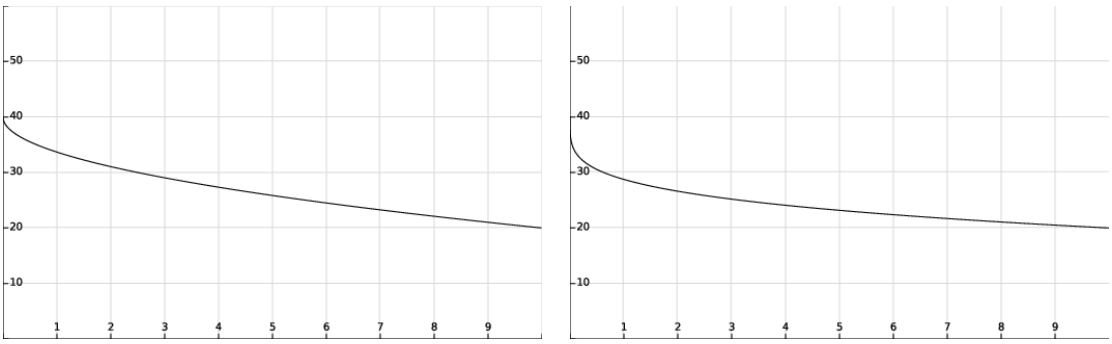


Figura 4.9: Ejemplos de la función con un valor de comportamiento lineal, $\beta = 1$



(a) $\beta = 2$ **(b)** $\beta = 4$

Figura 4.10: Ejemplos de la función con un valor de comportamiento benévolo para β

En las imágenes podemos ver como un *vendedor* varía el precio que pide, sabiendo que como mínimo va a exigir 20 y como máximo estima que puede ganar 40, valores que conoce mediante parametrización. Para esta petición, se supone que el número de rondas determinadas por el *comprador* es de 10. Se aprecia como en función del valor que adquiera β, el comportamiento del agente será más exigente o más benévolo.

4.2.1.5. Diagramas de interacción

En este apartado se muestran y comentan las distintas interacciones que se han planteado para que el mercado funcione según lo explicado:

- **Añadir las ofertas de los vendedores**

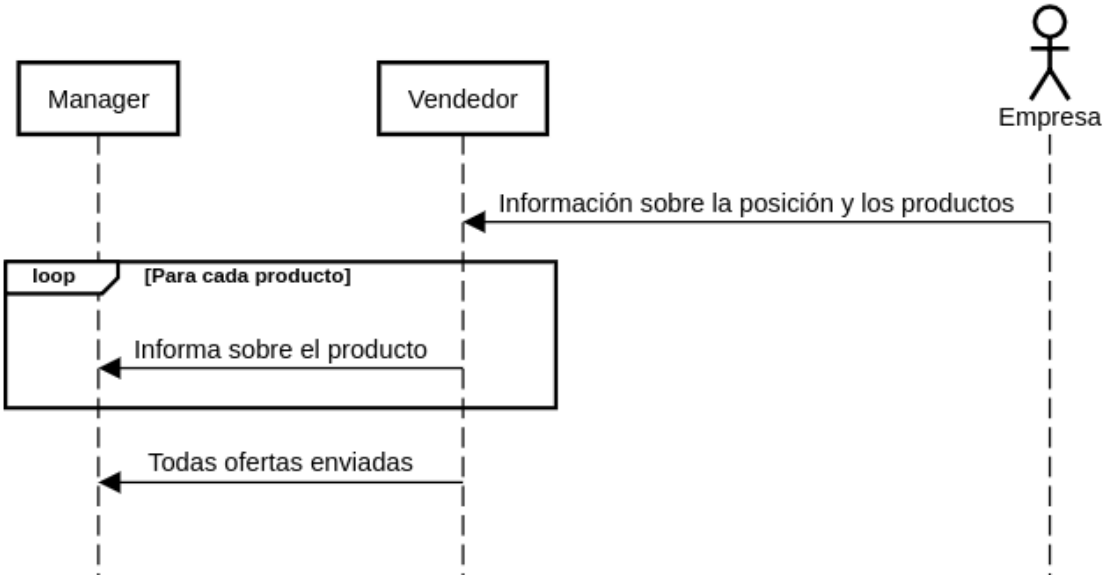


Figura 4.11: Diagrama sobre añadir las distintas ofertas de un vendedor

Una de las primeras interacciones que deben haber en el mercado es la de informar al *manager* de todos los productos que poseen los distintos *vendedores*, gracias a lo cual, el *manager* podrá calcular las distintas coincidencias que se producen.

El vendedor es en sí una *empresa*, que será representada en el mercado con un agente tipo *vendedor*. En primer lugar la empresa parametriza a su agente para que sepa los productos con los que debe negociar y la posición en la que se encuentra. Seguidamente, el *vendedor* informa al *manager* sobre cada producto y, por último, le indica que el agente ya ha terminado la fase de envio de productos y pasa a la espera de recibir las peticiones de los compradores para empezar la fase de negociación.

El envío de un producto será distinto en función de la naturaleza del producto. Si es un producto simple se envía su material y cantidad y si es un compuesto, se envía el identificador y las unidades, tal y como se ha explicado en la sección 4.2.1.2 y 4.2.1.3. En ambos casos, se envía también el identificador que usa el *vendedor* para identificar a cada uno de sus productos, de esta forma, cuando el *manager* le conteste sabrá el producto al que se está refiriendo fácilmente.

- **Añadir las peticiones de los compradores**

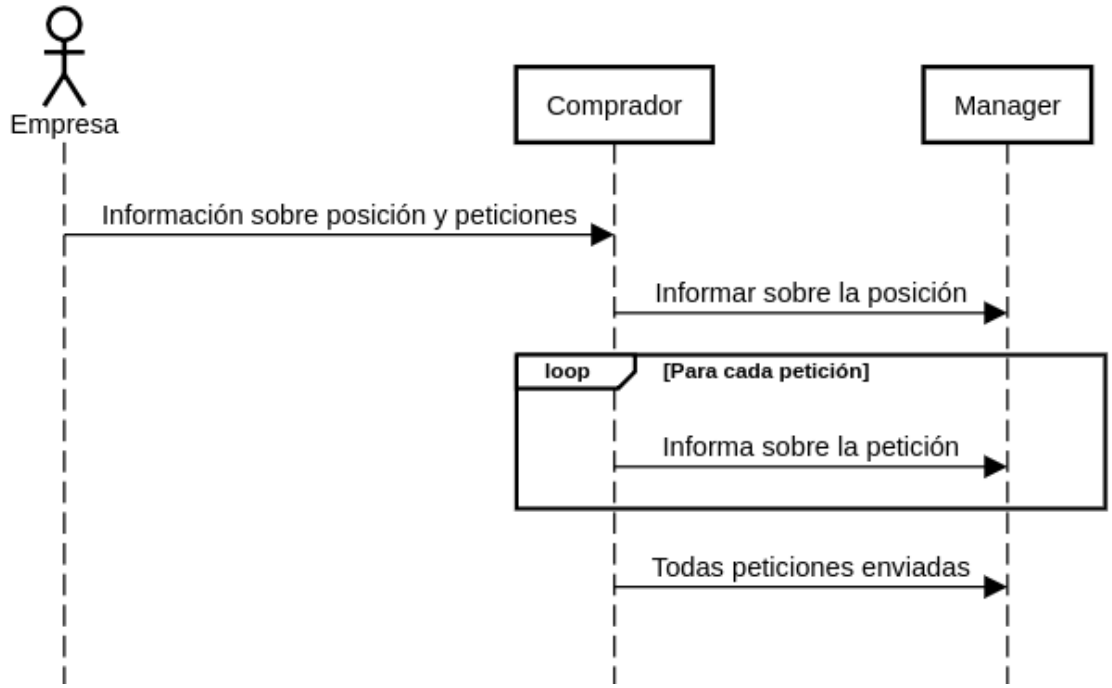


Figura 4.12: Diagrama sobre añadir las distintas peticiones de un comprador

Las empresas que quieren comprar unos productos tendrán que proceder análogamente. De esta forma, el *manager* ya dispondrá de toda la información necesaria para calcular las coincidencias.

El comprador es en este caso también una *empresa*, que se verá identificada en el mercado con un agente de tipo *comprador*. Primero se parametriza con la lista de productos a conseguir y la posición de la empresa representada con sus coordenadas, luego se le informa al *manager* de la posición y de cada uno de los ítems a obtener. Finalmente le indica al mismo que ya le ha transmitido todas las peticiones y pasa a la espera para recibir ofertas de los *vendedores*.

La petición de una compra se caracteriza por el material, la cantidad y el identificador que tendrán que usar los distintos *vendedores* para referirse a dicha petición.

La posición de la empresa en este caso sí que hay que mandarsela ya que, como se explica posteriormente, el *manager* se la enviará a cada *vendedor* para que puedan calcular el coste del transporte. Se ha diseñado de esta forma ya que todas las negociaciones que se producen en el mercado las inicia el *vendedor*.

Se podría haber realiza de otra forma: que el *comprador* se la transmita al *vendedor* cada vez que el último se la pida. Sin embargo, se ha decidido no desarrollarlo así ya que es más simple de la forma explicada, aunque un *vendedor* pueda recibir la posición de un *comprador* con el que nunca se va a establecer una negociación.

- **Cálculo de las coincidencias**

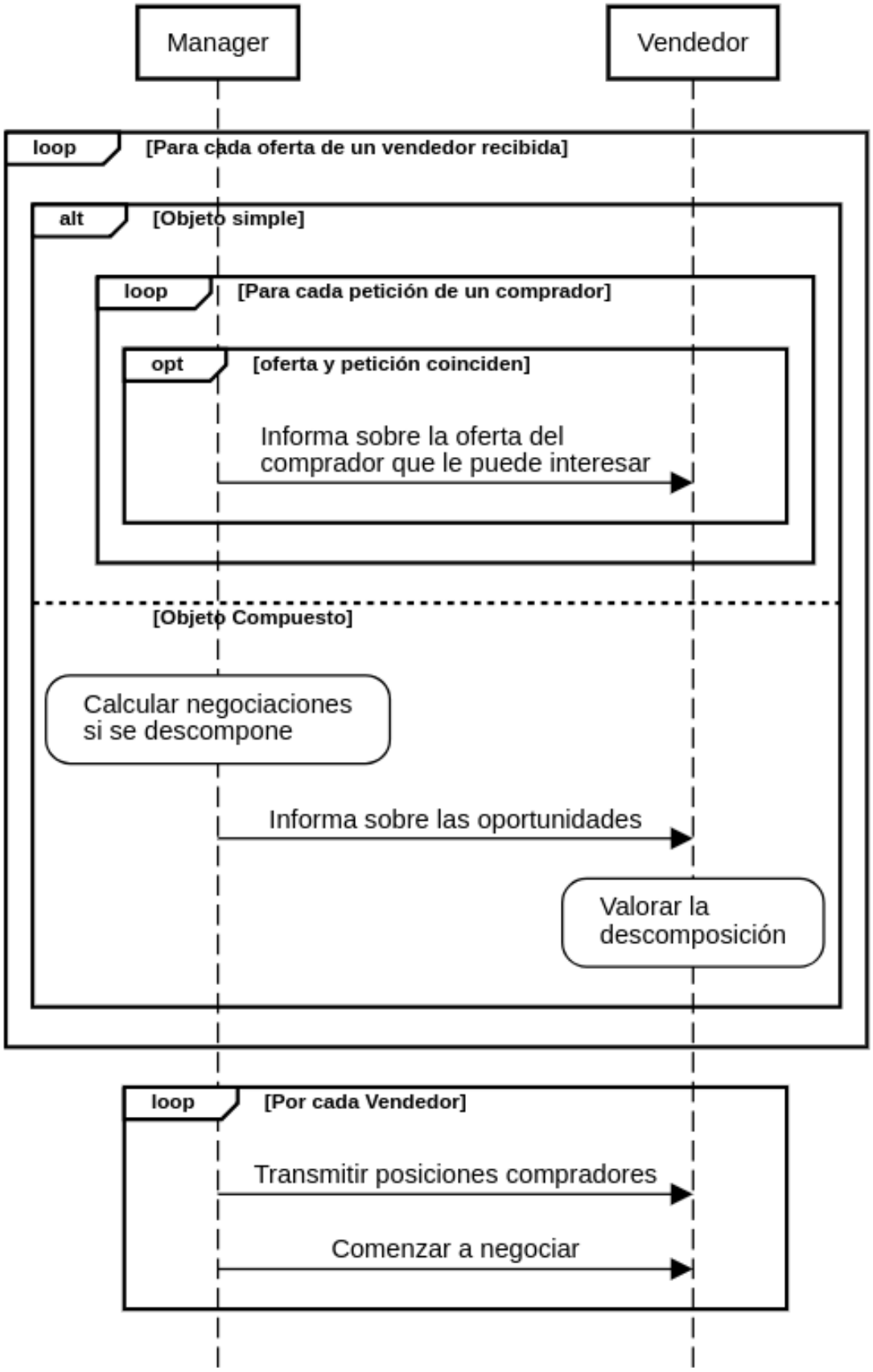


Figura 4.13: Diagrama sobre informar a los vendedores sobre las coincidencias

Una vez el *manager* ya disponga de todos los productos que tienen los *vendedores* y de todas las peticiones de los *compradores*, se pueden transmitir a los distintos *vendedores* las oportunidades de negocio que se identifiquen.

El modo de proceder que se ha diseñado es el siguiente: recorremos todas las ofertas de los distintos *vendedores* y para cada una de ellas, dependiendo de la naturaleza del objeto:

- **Objeto simple**: en dicho caso, el *manager* informa al *vendedor* sobre la oferta del *comprador* si el par oferta-petición coinciden.

  Como coincidir se entiende que el material del objeto sea el mismo. En tal caso, se podrá negociar directamente o mediante la formación de una alianza (si el *comprador* exige más de lo que oferta el *vendedor* en cuestión).
- **Objeto compuesto**: cuando ocurre esta situación, el *manager* debe calcular todas las posibles negociaciones que pueden aparecer si se decide descomponer y, seguidamente, le pasa dicha información al *vendedor* para que razone si le interesa efectuar la descomposición.

  Para calcular todas las posibles negociaciones, el *manager* calcula todos los componentes que se obtendrían y calcula para cada uno de ellos las peticiones de los *compradores* con las que coincide. Eso se realiza de manera similar a lo descrito anteriormente ya que, ahora, cada componente se puede considerar como si fuese un *objeto simple*.

  Después de informar al *vendedor*, el agente debe valorar si acepta la transformación. El razonamiento más simple, pero efectivo, es el de realizarla si existe al menos una oportunidad de negocio. En caso de aceptarla, dispondrá de nuevos objetos simples con los que podrá empezar una negociación.

En ambos casos, la oferta de un *comprador* está descrita con la tupla *(identificador, comprador)*. De esta forma, se podrá referir a un oferta de manera directa y sencilla, sin tener que referirse a ella mediante su composición. Además, junto a las coincidencias, el *vendedor* recibe también la cantidad que desea el *comprador* para identificar rápidamente si se requiere o no una alianza.

Una vez el *manager* haya procedido con todas las ofertas de los *vendedores* que se encuentran en el marcado, les manda a cada uno de ellos las posiciones de cada *comprador* y la señal para que pasen a la fase de negociación. Se ha decidido que los agentes se tengan que esperar a recibir la señal (en lugar de empezar a negociar cuando reciban una coincidencia) para que el *vendedor* posea todas las coincidencias antes de tomar una decisión, pero, sobre todo, para que todos los agentes comiencen a la vez.

- **Oferta de un vendedor para correspondencia directa**

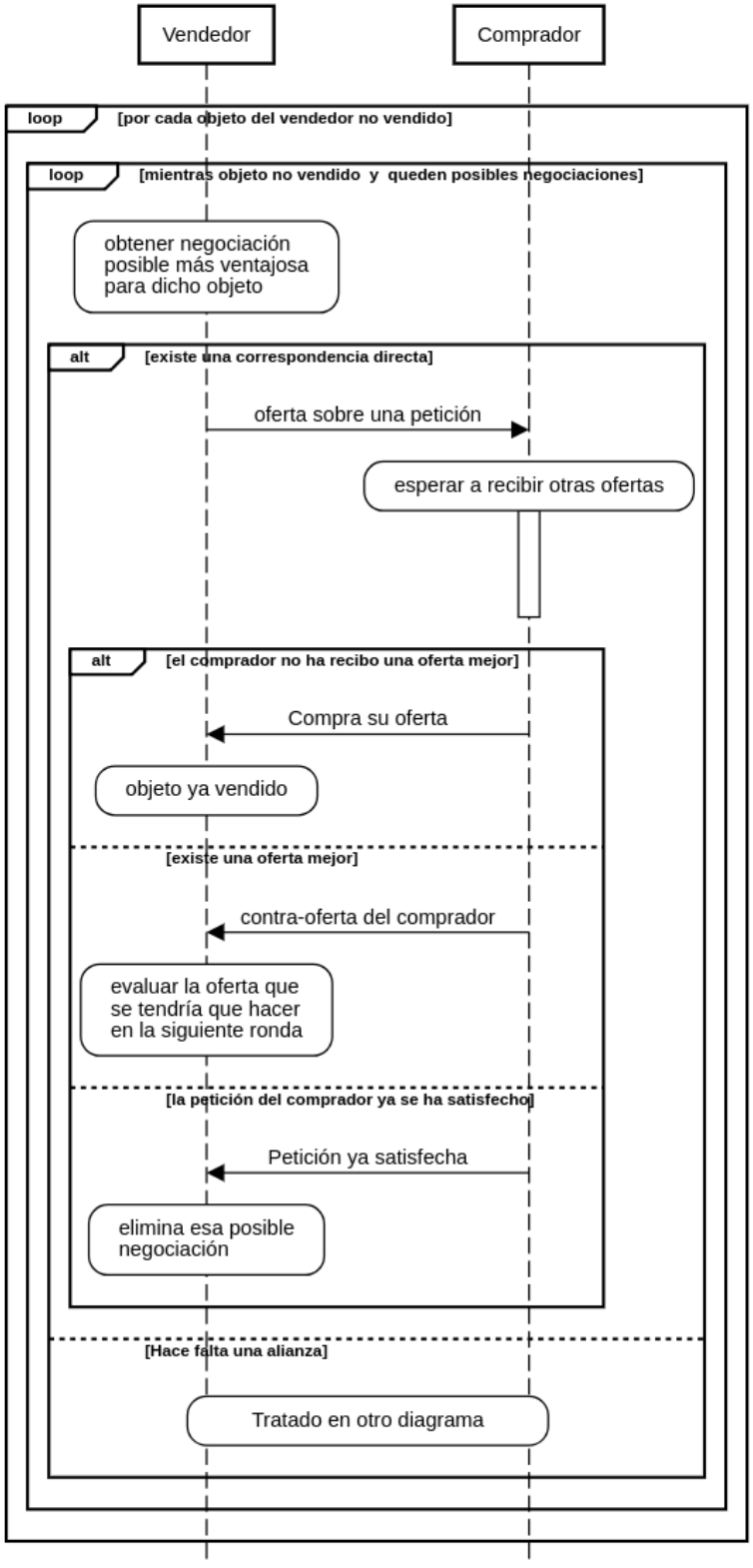


Figura 4.14: Diagrama sobre la oferta de un vendedor

Tras recibir la señal de *Comenzar a negociar*, el *vendedor* pasa a la fase de negociación. Tendrá disponible un listado de las peticiones de los compradores, con las que existe una correspondencia con uno de sus productos, junto a la cantidad que exigen para que el *vendedor* sepa si hace falta o no una alianza.

En el diagrama expuesto como se puede ver, se ejecutará por cada objeto que no se haya vendido siempre y cuando aún queden posibles negociaciones. Se seleccionará la coincidencia más ventajosa en función de cómo se haya paremetrizado el agente:

- **En función del precio**: se seleccionará la opción que el agente crea que es con la que más dinero puede conseguir.
- **En función de la contaminación**: se guiará únicamente por el coste de transporte, el cual dependerá de la distancia entre el *comprador* y el *vendedor*.
- **En función del precio y de la contaminación**: la selección se basará tanto en el precio como en la contaminación, el agente dispone de un parámetro para saber cuánto peso darle a cada medida.

  Para ello, el *vendedor* valorará, mediante un parámetro c, la cuantía de dinero en la que valora la contaminación producida por cada kilómetro. Por lo tanto, la ganancia realizada es de: $posible\_dinero\_a\_ganar - c * distancia$. Cabe destacar que cuando $c = 0$, estamos en el caso de *en función del precio*.

Con la coincidencia ganadora se procederá a negociar y, en función de la cantidad del objeto a negociar y de la requerida en la petición del *comprador*, el *vendedor* sabrá si hace falta una alianza o no. El caso de que haga falta se trata en el siguiente apartado.

Si la correspondencia es directa, la negociación con el *comprador* la empieza el *vendedor* mediante el envío del identificador de la petición y de la cantidad de dinero por la que se la vende. En este punto, el *comprador* contestará, transcurrido un tiempo, con su respuesta:

- **Comprador no ha recibido una oferta mejor**: cuando esto ocurra, el *vendedor* tiene que anotarse que su objeto ya se ha vendido, por lo tanto, no podrá negociar más con él. Por otra parte, a partir de este punto, el *comprador* responderá a las demás ofertas diciendo que su petición ya ha sido satisfecha.
- **Existe una oferta mejor**: el *comprador* le notifica al *vendedor* de que no está interesado en la oferta realizada. Así que, el *vendedor* tendrá que razonar cuánto dinero le ofrecerá en el siguiente intento, siempre y cuando se siga razonando que la petición de dicho *comprador* es la más ventajosa.

  Tal y como se ha explicado en la sección 4.2.1.4, la cuantía de dinero que un *vendedor* ofrece en cada ronda depende de una serie de parámetros, entre ellos de β, gracias a lo cual, se permite que los agentes sigas diferentes estrategias dependiendo del tipo de comportamiento que se desee conseguir.
- **Petición ya satisfecha**: el *vendedor* es informado por el *comprador* de que esa petición ya ha sido satisfecha, por lo tanto, el *vendedor* se lo anota para no volver a intentar negociar con él.

Como la recepción de la respuesta del comprador puede tardar un tiempo, el agente puede empezar a razonar qué hacer con su siguiente objeto no vendido. De esta forma, el *vendedor* negociará de manera concurrente con todos sus objetos.

En la implementación se tendrá que tener en cuenta el caso en el que hayan dos objetos del *vendedor* que coincidan con una única petición de un *comprador* para que el *vendedor* no compita contra sí mismo.

- **Oferta de un vendedor cuando hace falta una alianza**

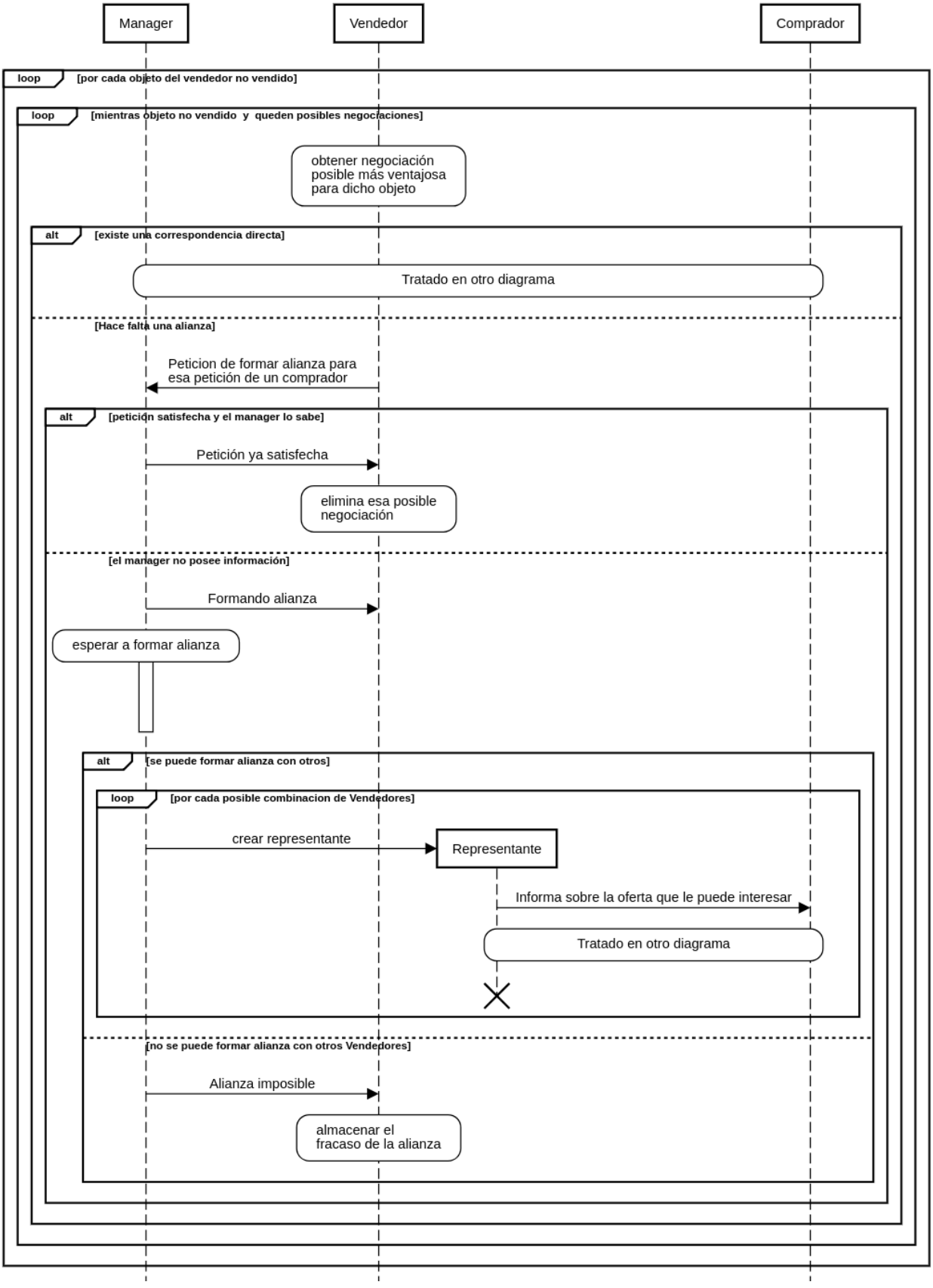


Figura 4.15: Diagrama sobre la creación de alianzas de vendedores

Siguiendo la lógica del diagrama anterior 4.14, en este diagrama se contempla el caso en el que el *vendedor* razona que una alianza es necesaria. Cuando esto ocurre, delegará dicho deseo en el *manager*, el cual formará la alianza cuando hayan suficientes *vendedores* y se cumplan una serie de restricciones.

La solicitud de formar una alianza está formada por la tupla *(comprador, identificador)* para saber la petición en particular sobre la que se quiere crear la alianza. Además el *vendedor* también le dirá la cantidad de dinero que pide y la oferta que está intentado vender para que el *manager* puede saber la cantidad que dispone al formar las distintas alianzas.

En el momento en el que el *manager* reciba la petición de formar una alianza, pueden ocurrir dos situaciones:

- **Petición satisfecha y el *manager* lo sabe**: en este caso, el *manager* contesta directamente informando al *vendedor* de que la petición ya ha sido satisfecha, mensaje que es tratado de la misma forma que si el emisor es un *comprador* según lo explicado en el diagrama de la figura 4.14.

  Al proceder de esta forma, el *software* evitará crear agentes cuando ya se sabe que todos fracasarán, por lo tanto se ahorra tiempo y recursos.

- **El *manager* no posee información**: en esta situación, el *manager* se esperará a poder formar una alianza con un subconjunto de los *vendedores* que quieren aliarse para una cierta petición. Por cada posible alianza que se detecte, se creará un agente de tipo *representante*, agente que sabrá la petición en particular que debe satisfacer y la cantidad de dinero que debe ofrecer cuando inicie su negociación.

  Para la detección de las distintas alianzas se propone el siguiente pseudocódigo de un algoritmo de *backtracking*:

```
/*
*  Metodo llamado cuando el manager recibe una peticion para formar una alianza
*/
formar_alianza((identificador_peticion, comprador), (identificador_objeto, vendedor, oferta)){
  // Obtenemos los aliados que hay para la peticion
  aliados = obtener_aliados_actuales(identificador_peticion, comprador);

  alianzas = backtracking((identificador_peticion, comprador), aliados, [(identificador_objeto, vendedor, oferta)]) ;

  // Instanciamos las alianzas
  for(alianza in alianzas){
    crear_alianza(alianza);
  }

  // Anyadimos al vendedor para la formacion de futuras alianzas
  anyadir_aliado(aliados, (identificador_objeto, vendedor, oferta));
}

/*
* Metodo empleado para formar todas las alianzas posibles
*/
```

```
22 backtracking((identificador_peticion, comprador), aliados, miembros)
       {
23   sol = [];
24   cantidad_a_conseguir = obtener_cantidad(identificador_peticion,
       comprador);
25   cantidad_actual = 0
26   for (miembro in miembros){
27     cantidad_actual += miembro.cantidad;
28   }
29
30   if(cantidad_a_conseguir == cantidad_actual){ // Alianza encontrada
31     sol.append(miembros);
32   }
33   else{ // Seguir buscando
34     aliado1 = aliados[0];
35     aliados.pop(); // Eliminamos el primer aliado de la lista
36
37     // Aplicamos backtracking metiendo a dicho aliado (si se puede)
38     if(cantidad_actual + aliado1.cantidad <= cantidad_a_conseguir){
39       sol.append(backtracking((identificador_peticion, comprador),
           aliados, [miembros, aliado1]));
40     }
41
42     // Aplicamos backtracking sin meterlo
43     sol.append(backtracking((identificador_peticion, comprador),
         aliados, miembros));
44   }
45   return sol;
46 }
```

Algoritmo 4.1: Formación de alianzas

Como se aprecia, de esta forma se crearán todas las alianzas posibles para una petición en particular. El algoritmo propuesto trabaja de la siguiente forma: el *manager* recibe la petición de formar una alianza por parte de un *vendedor* e intentará formar todas las alianzas posibles donde dicho *vendedor* participe, al hacerlo así, nos aseguramos de que cada posible alianza solo se crea una única vez ya que todas las alianzas que se creen en una iteración no se habrán podido crear en las anteriores. Seguidamente buscamos e instanciamos todas las alianzas posibles y, por último, añadimos al *vendedor* que ha hecho la petición a la lista de posibles aliados del *manager* para futuras peticiones de otros vendedores.

Sobre el método *backtracking*, dicho método recibe la petición que debe satisfacer, los posibles aliados con los que puede formar una alianza y los miembros actuales de una alianza parcial o total. Al ser llamado este método dándole el valor del *vendedor que ha hecho la petición* al último parámetro, dicho agente siempre participará en todas las alianzas que se formen.

El método comprobará si con los miembros actuales ya se puede formar una alianza. En caso negativo, explorará todo el posible árbol de búsqueda podando en el que caso en el que al añadir un nuevo miembro, la cantidad total ofertada sea mayor que la que pide el *comprador*, ya que en dicha rama del espacio de búsqueda no se podrá encontrar una alianza.

El coste de este algoritmo es exponencial pero, al crearse todas las alianzas posibles y como compiten entre ellas, se asegura que gane la posible alianza que haga la oferta más económica.

Para intentar contrarrestar el elevado coste, se propone una medida adicional para podar el algoritmo de backtracking expuesto: se sabe que si se crean dos alianzas, el *comprador* se quedará con la que le oferte el producto por menos dinero, por lo tanto solo se creará una nueva alianza si esta ofrece una cantidad de dinero menor de la ya ofertada previamente por otra alianza. De esta forma se evitará llamar en exceso a la instrucción *Crear representante*, la cual es muy costosa comparadas con el resto, pero asegurándose de que la optimalidad del algoritmo no se ve sacrificada. El pseudo-código quedaría por lo tanto:

```
/*
* Metodo empleado para formar todas las alianzas posibles
*/
backtracking((identificador_peticion, comprador), aliados, miembros) {
  sol = [];
  cantidad_a_conseguir = obtener_cantidad(identificador_peticion, comprador);
  cantidad_actual = 0
  mejor_oferta = obtener_cantidad(identificador_peticion, comprador); // Infinito si no hay
  oferta_actual = 0;
  for (miembro in miembros){
    cantidad_actual += miembro.cantidad;
    oferta_actual += miembro.oferta;
  }

  if(mejor_oferta > oferta_actual){ // La alianza puede ser mejor que las ya existentes
    if(cantidad_a_conseguir == cantidad_actual){ // Alianza encontrada
      sol.append(miembros);
    }
    else{ // Seguir buscando
      aliado1 = aliados[0];
      aliados.pop(); // Eliminamos el primer aliado de la lista

      // Aplicamos backtracking metiendo a dicho aliado (si se puede)
      if(cantidad_actual + aliado1.cantidad <= cantidad_a_conseguir){
        sol.append(backtracking((identificador_peticion, comprador), aliados, [miembros, aliado1]));
      }

      // Aplicamos backtracking sin meterlo
      sol.append(backtracking((identificador_peticion, comprador), aliados, miembros));
    }
  }
  return sol;
}
```

Algoritmo 4.2: Formación de alianzas podando

Cabe mencionar también el caso en el que no se haya podido crear ninguna alianza, porque no se cumplen las condiciones necesarias, y que el *manager* no tenga infor-

mación sobre la petición en cuestión. En esta situación, después de haber transcurrido un cierto tiempo, se le informará al *vendedor* de que la alianza ha sido imposible, por lo que dicho agente se anotará dicho fracaso e intentará iniciar otra negociación.

- **Oferta de un representante**

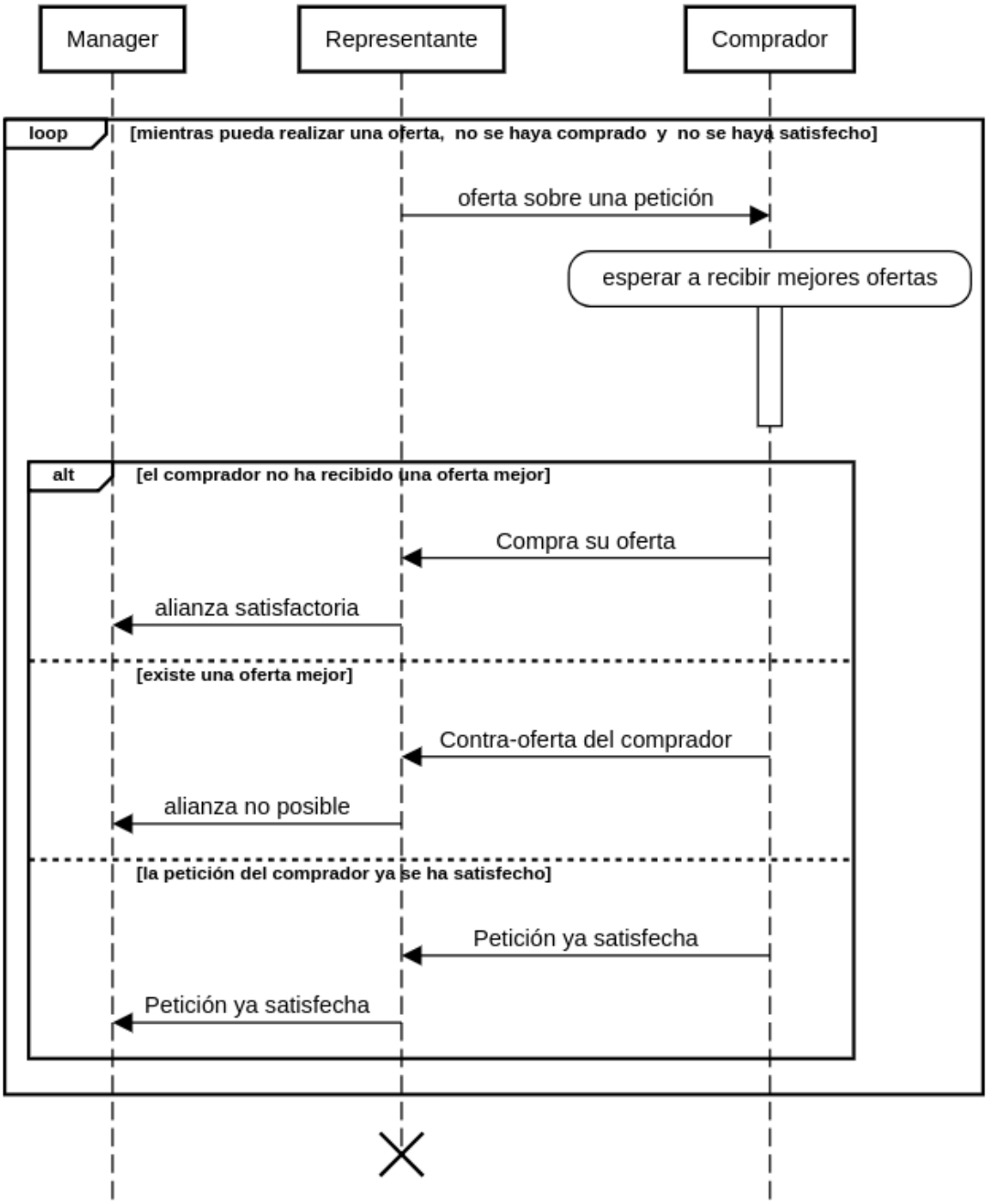


Figura 4.16: Diagrama sobre la oferta de un representante

Una vez el *manager* ya haya creado a un *representante*, este ya sabrá el dinero que como mínimo tiene que conseguir para satisfacer a todos sus miembros y la petición de un *comprador* en particular que tiene que intentar satisfacer. Cabe mencionar que al unirse varios *vendedores* cuya cantidad era menor, ahora la cantidad del producto que dispone el *representante* es la misma que la exigida por el *comprador*, por lo tanto, estamos ante el caso de *correspondencia directa*. Así que, el *comprador* no distingue

entre si al que está comprando es un *vendedor* o un *representante*, por lo que todas sus respuestas serán las mismas que en el caso descrito del diagrama de la figura 4.14.

El comportamiento del agente se repetirá siempre y cuando pueda realizar la oferta que ha razonado, no se lo haya comprado en la ronda anterior y no haya sido satisfecha la petición del *comprador* por otro agente. En tal caso, el *representante* realizará su oferta, la cual, después de un tiempo, será respondida por el *comprador* diciendo:

- **Comprador no ha recibido una oferta mejor**: el *comprador* adquiere el producto que es representado mediante la alianza. Por lo tanto, el *representante* informará al *manager* de dicho exito, el cual tendrá que procesar esa respuesta, como se ve en el diagrama de la figura 4.17. Por otra parte, y al igual que en el caso previo, a partir de este punto, el comprador responderá a las demás ofertas diciendo que su petición ya ha sido satisfecha.
- **Existe una oferta mejor**: el *representante* es informado del fracaso de su oferta debido a que propone una cantidad de dinero demasiado elevada. Por lo tanto, tiene que disolver la alianza que representa e informar al *manager* de ello. Posteriormente, se explica el procedimiento que sigue el *manager* para informar a los *compradores* sobre el fracaso, en caso de que haga falta.
- **Petición ya satisfecha**: en este caso, el *comprador* ya le ha comprado previamente el objeto a otro agente, así que su petición ya está satisfecha. El *representante* informará al *manager* de ello, el cual tendrá que informar a los distintos miembros de la alianza y anotarse que dicha petición ya ha sido satisfecha para no intentar crear otras alianzas para ella.

Por último, el agente se destruirá debido a que su vida útil ya ha llegado a su fin. Cuando esto ocurra significará que la alianza ha conseguido comprar o no, en ambos casos, el *manager* será informado de ello y se encargará de efectuar lo necesario para que los respectivos *vendedores* puedan seguir con su funcionamiento.

- **Representante informa sobre la alianza**

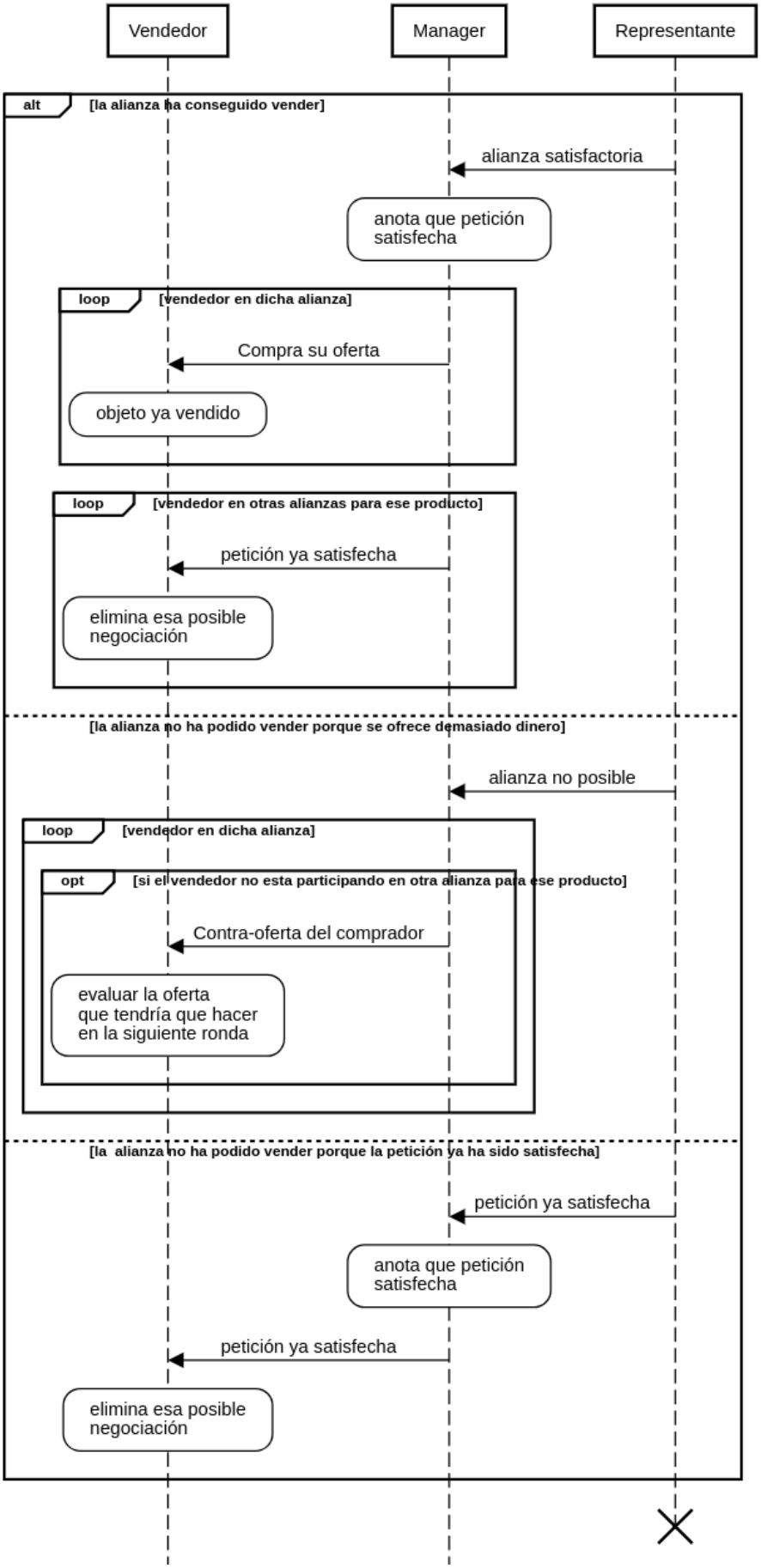


Figura 4.17: Diagrama sobre el representante informando sobre la alianza

Para acabar, queda contemplar el protocolo de lo que hay que hacer cuando el *representante* informa al *manager* sobre el éxito o fracaso de una alianza. El *representante* podrá contestar con tres mensajes distintos y, cada uno de ellos se tratará de manera diferente:

- **La alianza ha conseguido vender**: la alianza encarnada por el *representante* ha conseguido vender su objeto. Por lo tanto, debe informar a todos sus integrantes de dicho éxito, además informará también a todos los *vendedores* que formen parte de una alianza similar para intentar satisfacer la misma petición pero que no formen parte de la alianza ganadora de que ellos no han conseguido vender. En ambos casos, el *vendedor* tratará la respuesta de la misma forma que si le contestase un *comprador*, tal y como se ha explicado en el diagrama de la figura 4.14.

  El *manager* se anotará el hecho de que dicha petición ya ha sido satisfecha para no intentar formar otra alianza para la misma petición. De esta forma, la respuesta la da directamente el *manager* sin que se tenga que crear una alianza que se sabe que va a fracasar.
- **La alianza no ha podido vender porque se ofrece demasiado dinero**: el *representante* se disuelve porque no ha podido realizar una oferta mejor, por lo que todos sus integrantes deben de ser informados para razonar si deben hacer una oferta mejor. Solo se le informará del fracaso a los *vendedores* que no formen parte de otra alianza en dicho momento, esto debe de ser así ya que, siguiendo el diseño propuesto en el diagrama de la figura 4.15, un *vendedor* puede formar parte de más de una alianza para una misma petición debido a que solo la mejor será la ganadora.

  De manera análoga a como ocurría en el caso anterior, la respuesta dada por el *manager* de *contra-oferta del comprador* será tratada de la misma forma que si fuese la respuesta de un *comprador* según el diagrama de la figura 4.14. Es decir, el *vendedor* contará que ha pasado una iteración y realizará la oferta que calcule en base a la estrategia que siga.
- **La alianza no ha podido vender porque la petición ya ha sido satisfecha**: en el caso en el que el *representante* se disuelva porque la petición ya haya sido satisfecha, el *manager* tendrá que informar de ello a todos los *vendedores* que ha solicitado crear una alianza para dicha petición para indicarles que dicho deseo ya ha sido satisfecho.

  Al igual que pasaba en el primer caso, el *manager* se apuntará que esa petición ya sido satisfecha y evitará formar nuevas alianzas para la misma petición.

Tal y como se ha comentado en el diagrama de la figura 4.16, una vez el *representante* haya informado, este se destruye.

4.2.1.6. Resultados obtenidos

Por último, hace falta que se tenga constancia de todo lo que ocurre en el mercado para sacar estadísticas y medir cual es el comportamiento general del mismo. Para ello, se propone crear a un nuevo agente, el agente *log*, que disponga de la información de

todos los intercambios producidos de tal forma que pueda realizar cálculos sobre los mismos. Para este fin, hace falta notificar a dicho agente en ciertos casos:

- **El *manager* ha calculado todas las coincidencias**: se le notifica del número de ofertas y peticiones que han tenido al menos una coincidencia.
- **Un *vendedor* decide descomponer un objeto compuesto**: se le informa de las nuevas coincidencias que se producen debido a la aparición de nuevos objetos simples.
- **Un *comprador* compra para satisfacer una petición**: se le comunica de que una petición ha sido satisfecha así como del dinero que ha tenido que pagar.
- **Un *vendedor* ha vendido una de sus ofertas**: el nuevo agente es notificado de que un *vendedor* ha conseguido vender una de sus ofertas, se le informa del dinero que ha conseguido y de la distancia entre él y el *comprador*.

Al realizarse el diseño según se ha expuesto a lo largo de este punto, se verifica que se cumplen todos los requisitos expuestos en la fase de análisis. Sin embargo, en la parte de validación se comprueba mediante pruebas unitarias el correcto funcionamiento del *software* diseñado y, posteriormente, implementado.

4.2.2. Interfaz gráfica

El diseño que se ha razonado para la interfaz gráfica del mercado expuesto en la sección 4.2.1 consta de 4 ventanas: la ventana principal, la ventana para especificar las materias primas, una distinta para determinar los ítems y la ventana correspondiente a la salida del mercado. En este apartado se va a mostrar un esquema para cada una de ellas así como las funcionalidades, a grandes rasgos, que proporcionan cada uno de sus elementos.

4.2.2.1. Ventana principal

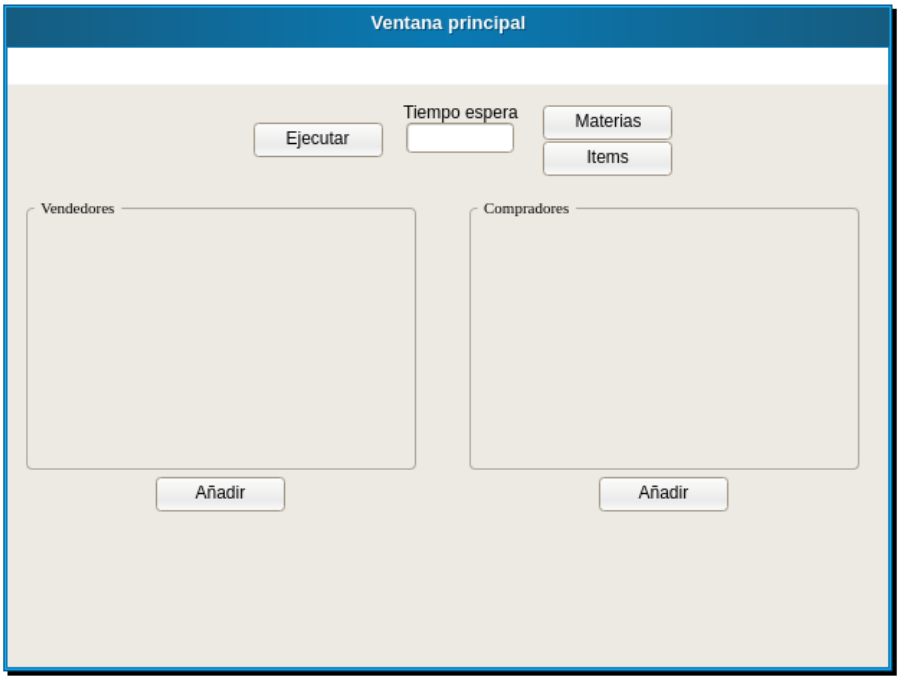


Figura 4.18: Boceto sobre la ventana principal de la aplicación

Los elementos que se aprecian son:

- **Ejecutar**: botón que pone en marcha el sistema multi-agente y muestra en pantalla los resultados que se van produciendo.
- **Tiempo espera**: sirve para especificar la cantidad de tiempo que tiene que pasar sin que un *comprador* reciba ofertas mejores para que adquiera la mejor oferta hasta el momento (procedimiento explicado en la figura 4.14), o para que transcurrido dicho tiempo el *manager* decida que un *vendedor* no ha podido formar alianza si no existen los suficientes aliados (detallado en la figura 4.15).
- **Materias** e **Ítems**: botones que abren la ventana donde se especifican las materias o ítems respectivamente con las que posteriormente los distintos agentes podrán negociar. Es decir, botones que sirven para especificar lo básico de la ontología tal y como se ha explicado en la sección 4.2.1.2.
- **Listado de *Vendedores*** y **Listado de *Compradores***: listados donde se mostraran y desde donde se podrán configurar los distintos agentes del mercado. Además, para los *vendedores* se podrá especificar los objetos simples y compuestos con los que podrá negociar, y, por otra parte, las distintas peticiones para los *compradores*. Los bocetos sobre los distintos agentes, sus productos y sus peticiones se muestran a continuación.
- **Añadir**: botón para añadir un nuevo agente, dependiendo de a que lista haga referencia.

Como se puede ver, en la figura 4.18 se encuentran dos listas, en cada una de ellas al accionar el botón de añadir se agregará un elemento que represente al tipo de agente

en cuestión (*vendedor* o *comprador*), dichos elementos seguirán el diseño de los siguientes bocetos:

- ***Vendedor***

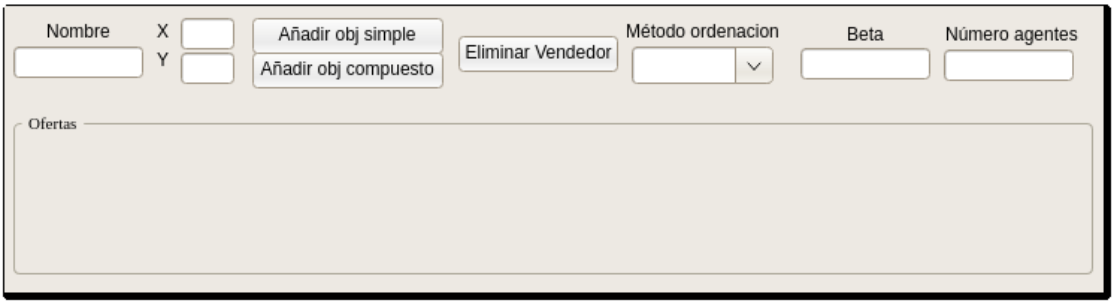


Figura 4.19: Boceto para configurar a un agente tipo *vendedor*

Cuando se añada un agente tipo *vendedor* se le tendrá que configurar con una serie de parámetros, además tendrá una serie de acciones. A continuación se van a detallar cada uno de sus elementos:

- **Nombre**: utilizado para parametrizar el nombre que mostrará el agente en cuestión.
- **X** e **Y**: sirve para determinar la posición según las coordenadas cartesianas.
- **Añadir objeto simple** y **Añadir objeto compuesto**: botones usados para añadir en la lista de *ofertas* los distintos tipos de objetos con los que puede negociar. Dichos esquemas se exponen a continuación.
- **Eliminar vendedor**: botón empleado para eliminar al agente de la lista mostrada en el boceto 4.18.
- **Método de ordenación**: listado para seleccionar el método de ordenación que usará el *vendedor* para ordenar las distintas coincidencias que se produzcan, siguiendo lo explicado en el diagrama 4.14. Las opciones que mostrará son: *por precio*, *por contaminación* y *por precio y contaminación*. En el último caso, se debe mostrar un campo adicional para parametrizar la cuantía de dinero en la que se valora un kilómetro.
- **Beta**: parametro usado para determinar la estrategia que seguirá el agente, según lo explicado en la sección 4.2.1.4.
- **Número de agentes**: campo empleado para permitir varias copias distintas del mismo agente. Es decir, todos estos agentes tendrán los mismos parámetros y objetos con los que negociar.

Por último para este tipo de agente, se van a exponer los bocetos de los dos tipos de objetos con los que puede negociar. Los campos exigidos en ambos, son los ya discutidos en el diagrama de clases de la sección 4.2.1.3. Los bocetos son:

- **Objeto simple**

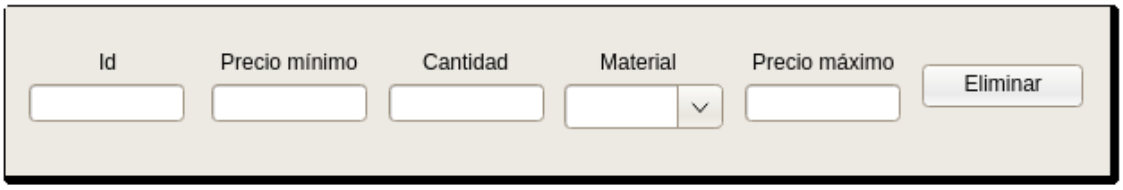


Figura 4.20: Boceto para parametrizar un objeto simple de un *vendedor*

Los elementos que dispone son:

- **Id**: identificador usado por el *vendedor* para distinguir a este objeto del resto.
- **Precio mínimo**: cantidad de dinero mínima por la que el *vendedor* está dispuesto a negociar con este producto.
- **Cantidad**: gramos del objeto simple compuesto por el material.
- **Material**: listado que permite seleccionar la materia en particular de la que está compuesto este objeto simple. Los identificadores de las distintas materias se especifican según lo expuesto en la sección 4.2.2.2.
- **Precio máximo**: cuantía de dinero máximo por la que el *vendedor* estima que podrá vender este objeto.
- **Eliminar**: botón para eliminar a esta oferta del listado del boceto 4.19.

- **Objeto compuesto**

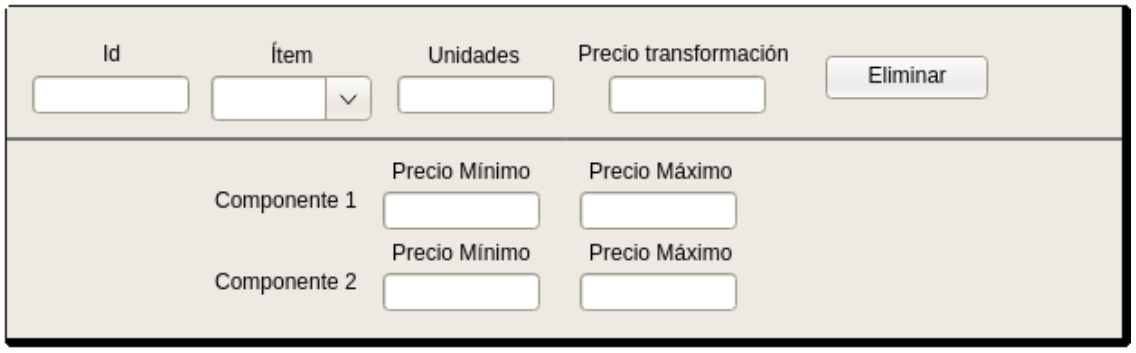


Figura 4.21: Boceto para parametrizar un objeto compuesto de un *vendedor*

Los elementos que se describen son:

- **Id**: identificador empleado por un agente en particular para diferenciar a este objeto frente a otros.
- **Ítem**: listado desde el cual se puede seleccionar el tipo de objeto compuesto al que representa. Los distintos identificadores que conforman esta lista se definen tal y como esta explicado en la sección 4.2.2.3.
- **Unidades**: cantidad de productos individuales del tipo detallado en el campo de *ítem* que conforman este objeto compuesto.
- **Precio transformación**: cuantía de dinero que el *vendedor* está dispuesto a pagar por para aplicar la transformación que le ofrecerá el *manager*.
- **Eliminar**: botón para eliminar a esta oferta del listado del boceto 4.19.
- **Componentes**: cada una de las materias primas en las que se sabe que se descompondrá el objeto compuesto. Para cada una de ellas, y al igual que

en el caso del boceto 4.20, se deberá especificar el precio mínimo y máximo con el que negociará, debido a que si el agente decide descomponer, cada uno de sus componentes serán tratados como objetos simples.

- ***Comprador***

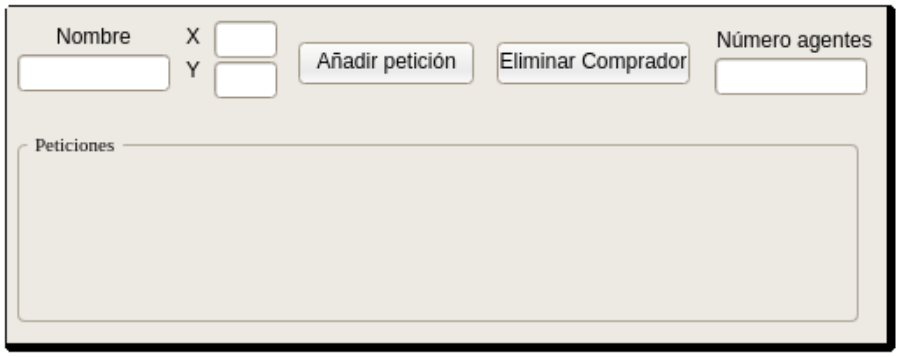


Figura 4.22: Boceto para configurar a un agente tipo *comprador*

Al añadir un *comprador* se debe permitir al usuario parametrizar sus campos y realizar un conjunto de acciones con él. A continuación se va a detallar cada uno de sus elementos:

- **Nombre**: utilizado para indicar el nombre que mostrará el agente en cuestión.
- **X** e **Y**: sirve para determinar la posición según las coordenadas cartesianas.
- **Añadir petición**: botón usado para añadir cada una de las peticiones que pretenderá conseguir en el listado de *peticiones*. El boceto empleado por cada petición se muestra a continuación.
- **Eliminar comprador**: botón empleado para eliminar al agente de la lista mostrada en el boceto 4.18.
- **Número de agentes**: campo empleado para permitir varias copias distintas del mismo agente. Es decir, todos estos agentes tendrán los mismos parámetros y peticiones que conseguir.

Cuando se quiera añadir una petición, se tendrá que rellenar la interfaz representada por el siguiente boceto:

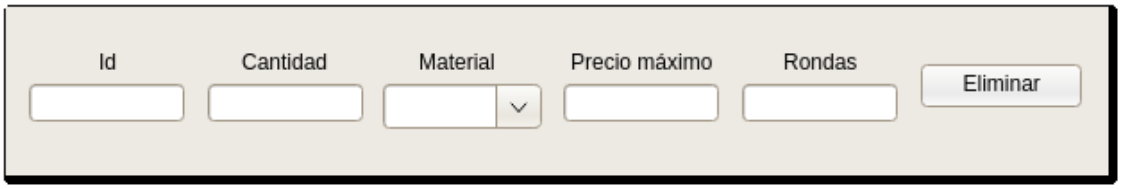


Figura 4.23: Boceto para parametrizar una petición de un *comprador*

Donde los elementos que se aprecian son:

- **Id**: identificador usado por el *comprador* para poder distinguir a las distintas peticiones.
- **Cantidad**: gramos del material especificado que conforman la petición.

- **Material**: listado que permite seleccionar la materia en concreto que se exige en esta petición. Los identificadores de las distintas materias se especifican según lo expuesto en la sección 4.2.2.2.
- **Precio máximo**: dinero máximo por el que el *comprador* en cuestión aceptará una oferta de un *vendedor* para satisfacer esta petición.
- **Rondas**: número de ofertas distintas de un mismo *vendedor* que el *comprador* esta dispuesto a aceptar para esta petición.
- **Eliminar**: botón empleado cuando se quiera eliminar a la petición a la que pertenece de la lista de *peticiones* que se puede ver en el boceto 4.22.

4.2.2.2. Ventana materias

Una vez se accione el botón del boceto *Materias* de 4.18 se mostrará la ventana para tratar sobre las distintas materias primas con las que se podrán negociar posteriormente.

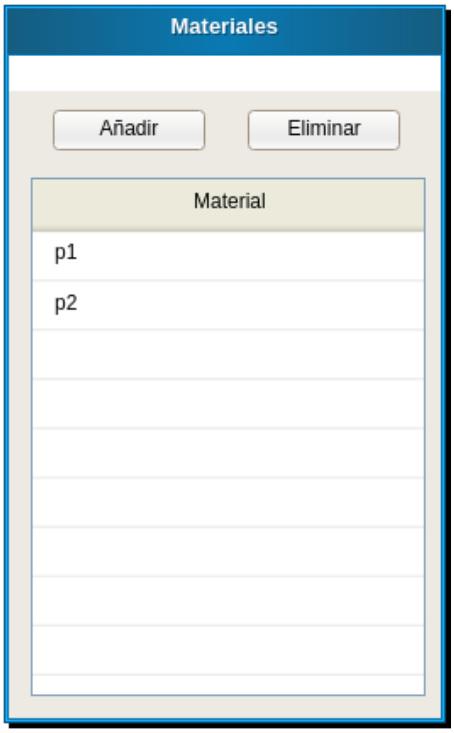


Figura 4.24: Boceto sobre la ventana de materiales

En dicho boceto se aprecian los siguientes elementos:

- **Añadir**: botón que al pulsarlo muestra un diálogo para añadir un nuevo material, el mostrado en la figura 4.25.
- **Eliminar**: botón que sirve para eliminar a un material concreto de la tabla.
- **Tabla de materiales**: tabla que muestra los distintos materiales que ya se han definido y añadido al sistema.

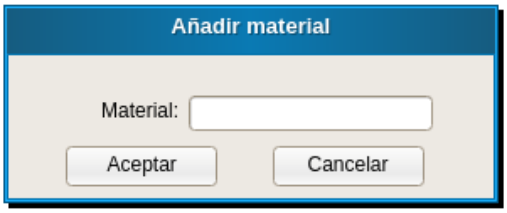


Figura 4.25: Boceto sobre la ventana de añadir un material

En el esquema se pueden visualizar una serie de componentes que son:

- **Campo de material**: campo de texto donde se determina el identificador de un material en particular.
- **Aceptar**: empleado para añadir el material especificado en el campo al sistema.
- **Cancelar**: botón para cerrar el diálogo sin realizar ninguna acción.

4.2.2.3. Ventana ítems

Si se selecciona el botón de *Ítems* de la ventana 4.18 se mostrará la ventana para visualizar y añadir los diferentes ítems compuestos con los que se podrá negociar.

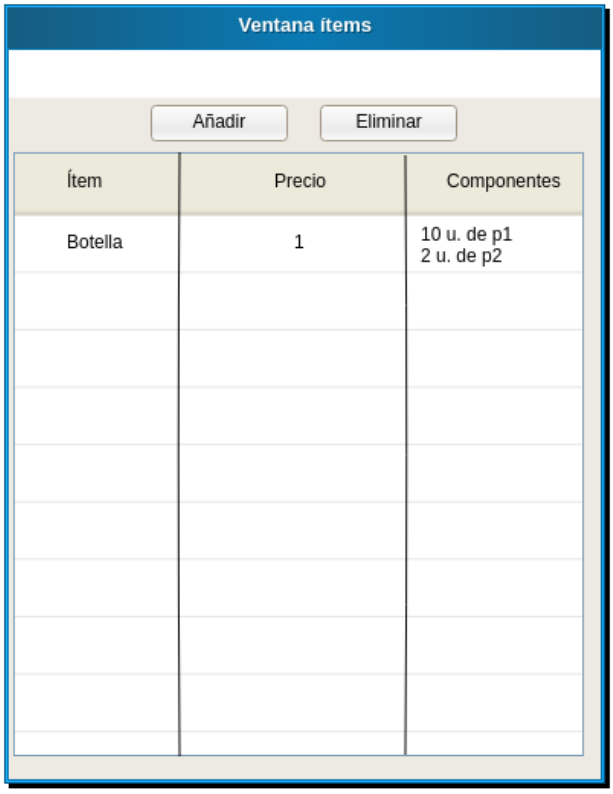


Figura 4.26: Boceto sobre la ventana de ítems

Los componentes que se aprecian en el boceto son:

- **Añadir**: botón empleado para abrir el dialogo, que se explica a continuación en el boceto 4.27, para añadir un ítem nuevo.
- **Eliminar**: permite al usuario eliminar un ítem de la tabla.
- **Tabla ítems**: muestra los ítems que dispone el sistema. Para cada uno de ellos muestra: el identificador que tendrá que ser seleccionado cuando se añada un objeto compuesto a un *vendedor*, el precio de aplicar la transformación a una unidad y un listado de los componentes que son las materias y las cantidades en las que se descompondrá.

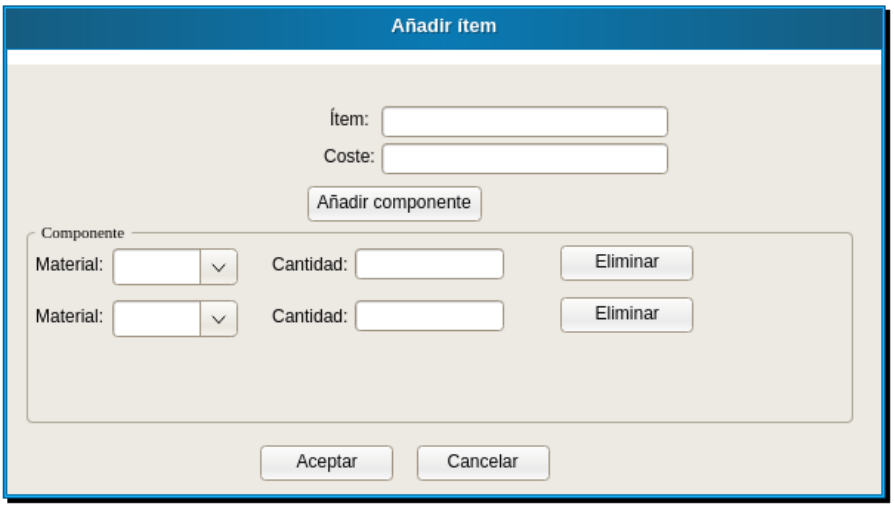


Figura 4.27: Boceto sobre la ventana de añadir un ítem

Por último, tenemos la ventana que se muestra cuando se desea añadir un ítem nuevo. Para ello, tenemos una serie de elementos:

- **Ítem**: campo para definir el identificador por el que se diferenciará el ítem que se va a añadir.
- **Coste**: cantidad necesaria para aplicar la transformación.
- **Añadir componente**: añade un componente más a la lista.
- **Componente**: cada uno de los materiales en los que se transformará. Para cada uno de ellos se especificará el material (procedentes del listado definido en la sección 4.2.2.2) y la cantidad, además tiene un botón de eliminar por si se desea eliminar ese componente.
- **Aceptar**: añade al ítem al sistema.
- **Cancelar**: cierra el dialogo sin modificar el sistema.

4.2.2.4. Salida del mercado

Tal y como se ha explicado en la sección 4.2.1.6, se dispone de un agente que es notificado de todas las transacciones que se van produciendo, es por ello que dicho agente es el encargado de mostrar por pantalla las estadísticas del mercado en cada momento.

Se propone que cada vez que el agente sea informado de un evento, muestre por pantalla un resumen del evento y un resumen del estado general del mercado.

Los eventos que se pueden producir son:

- **El *manager* ha calculado todas las coincidencias**: muestra el mensaje *"Inicio de las negociaciones"* para indicar que ya se han enviado las distintas coincidencias y los distintos agentes van a empezar a negociar para satisfacer sus deseos.
- **Un *vendedor* decide descomponer un objeto compuesto**: indica de dicho hecho mediante el mensaje *"El vendedor NOMBRE ha decidido descomponer su producto ID_OFERTA "*.
- **Un *comprador* compra para satisfacer una petición**: con el mensaje *"El comprador NOMBRE ha satisfecho su petición de ID_PETICION por un precio de PRECIO"* manifiesta que un *comprador* ha comprado una oferta.
- **Un *vendedor* ha vendido una de sus ofertas**: aparece en pantalla el mensaje *"El vendedor NOMBRE ha vendido su oferta de ID_OFERTA por un precio de PRECIO"* para indicar que un *vendedor* ha conseguido vender unas de sus ofertas.

Por otra parte, el estado general del mercado se representa con una serie de campos:

- **Ofertas vendedores**: porcentaje de ofertas vendidas frente al número de ofertas que se pueden vender (que tienen al menos una coincidencia). Por lo tanto *OFERTAS_SATISFECHAS / OFERTAS_CON_COINCIDENCIA*.
- **Peticiones compradores**: porcentaje de peticiones satisfechas en relación al número total de peticiones que pueden ser satisfechas con al menos una oferta. Es decir *PETICIONES_SATISFECHAS / PETICIONES_CON_COINCIDENCIA*.
- **Ganancias compradores**: suma total de lo que gana cada *comprador* con una transacción. La ganancia individual se calcula como el precio máximo que estaba dispuesto a pagar para esa petición menos el precio por el que la ha conseguido.
- **Ganancias vendedores**: cuantía total de los beneficios de los *vendedores* al vender sus ofertas. Cada ganancia de una negociación se define por: el precio por el que ha conseguido vender menos el precio mínimo que exigía para esa oferta.
- **Contaminación**: como la contaminación producida en el transporte de las mercancías desde los *vendedores* a los *compradores* es proporcional a la distancia entre ellos, en este campo se va a mostrar la suma de las distancias que se deben recorrer para cada negociación.

Con el resumen del evento y del estado del mercado se adquiere la información necesaria para analizar la evolución que tiene el mercado.

4.3 Implementación

A lo largo de esta sección del capítulo 4 se va a exponer de manera superficial los aspectos más importantes de la implementación del programa. Como se han desarrollado de manera independiente, se va comentar por separado el mercado y la interfaz gráfica.

4.3.1. Mercado

El mercado, que es el sistema multi-agente, se ha decidido implementar usando el *framework JASON* como se ha explicado en el apartado 3.4. A continuación se va a comentar la estructura que se ha seguido, el envío de mensajes y las acciones internas que se han tenido que desarrollar.

4.3.1.1. Estructura

Para cumplir con el diseño razonado en el punto 4.2.1, se han creado los distintos agentes para cada rol identificado, cada uno de ellos se definen en un archivo *.asl* independiente: *comprador.asl*, *vendedor.asl*, *manager.asl*, *representante.asl* y *log.asl*. Dichos agentes tienen programado el comportamiento y razonamiento expuesto en la etapa de diseño.

Como se explica después, los agentes *comprador.asl* y *vendedor.asl* son únicamente plantillas que se tendrán que adaptar a cada agente que se desee crear. Esto es así ya que no todos los *compradores* tienen las mismas peticiones y todos los *vendedores* poseen las mismas ofertas. De la misma forma, no todos los agentes será parametrizados con los mismos valores, es decir, por ejemplo, los distinos *vendedores* pueden tener diferentes valores para β (variable que define la estrategia de negociación) o diferentes métodos de ordenación de las coincidencias.

Además, se ha creado también el archivo *.mas2j* que es donde se especifican los agentes y el número de cada uno de ellos que se van a desplegar. Dicho archivo será el que se tendrá que interpretar y ejecutar para poner en marcha el sistema.

Los agentes especificados en dicho archivo son agentes (representados en un archivo *.asl*) que disponen tanto de las reglas (procedimientos a ejecutar para responder a un evento) como de las creencias (conocimiento propio sobre el que razonan). Es por ello que, por ejemplo, si se desean poner en ejecución dos *compradores* pero cada uno de ellos con peticiones distintas, se tendrán que crear dos archivos distintos de tipo *.asl* que sean idénticos excepto en el conocimiento que dispongan y, seguidamente, se especifica que se desea ejecutar una copia de cada uno en el archivo *.mas2j*. Este procedimiento se explica con más detalle a lo largo de la sección 4.3.2 debido a que la interfaz gráfica es la encargada de leer del usuario los distintos agentes que se quieren crear, para instanciarlos y, posteriormente, desplegarlos.

4.3.1.2. Envío de mensajes

Al tratarse de un sistema multi-agente, el envío de mensajes para que los distintos agentes se comuniquen entre ellos es crucial. Los mensajes son enviados por los agentes gracias a la instrucción interna ya implementada *.send()*. Dicha función crea en el agente destino las creencias que se pretendan enviar, las cuales se añaden a la base de creencias del agente en cuestión.

Además, el agente receptor debe tener implementadas una serie de reglas que se ejecutan cuando se creen algunas creencias en su base de conocimiento. En dichas reglas, el agente analiza el mensaje recibido, modifica su base de creencias o envía un mensaje, según el comportamiento que se desee implementar.

Por lo tanto, para implementar cada uno de los protocolo explicados con los diagramas de interacción en la sección 4.2.1.5, se hace mediante la creación de creencias y la regla relacionada a ella. Cada par de creencia-regla corresponde a un envío de un mensaje en particular.

Cabe destacar, que las creencias en el lenguaje *JASON* están compuestas por una propiedad y una serie de argumentos. Por ejemplo *oferta(500,plastico)*, donde *oferta* es la propiedad y el resto son argumentos. En este caso, se debe crear una regla que responda a la creencia con la propiedad *oferta* con dos argumentos, los lea y razone en base a ellos. Dicha regla será utilizada para todas las creencias recibidos con dos argumentos y con *oferta* como propiedad.

4.3.1.3. Acciones internas adicionales

Como se ha mencionado en la sección 3.4.1, el *framework JASON* funciona con el lenguaje de programación *AgentSpeak*, el cual es un lenguaje lógico, no de propósito general. Es por ello, que algunas de las funcionalidades que se requieren en el *software* no se pueden realizar con dicho lenguaje.

Sin embargo, el *framework* usado permite implementar unas acciones internas con una serie de argumentos que se llaman desde el código del agente, pero se desarrollan que *Java*, un lenguaje de propósito general.

Las acciones internas que se han tenido que implementar son los tres tipos de priorizaciones que puede usar cada agente en función de lo que se quiera que se tenga en cuenta al elegir la mejor coincidencia para negociar. Los tres tipos de ordenaciones eran: *en función del precio*, *en función de la contaminación* y *en función del precio y de la contaminación*. Por otra parte, se ha creado también una acción interna adicional para calcular el dinero que se debe ofertar en las distintas rondas de una negociación. Cada una de dichas acciones se implementan en un archivo *.java*.

4.3.2. Interfaz gráfica

Para implementar la interfaz gráfica se ha decidido hacer uso de la tecnología *JavaFX*, debido a que las acciones internas se deben programar en *Java* y a que la propia tecnología *JavaFX* es una de las más usadas tanto en el ámbito docente como el mundo empresarial para crear las interfaces con las que el usuario interacciona.

A lo largo de este capítulo se van a mencionar algunas de las características más relevantes que se han tenido que tratar para el desarrollo de la interfaz.

4.3.2.1. Modelo vista-controlador

El concepto MVC (modelo vista-controlador) consiste en un patrón de arquitectura donde los datos, la vista y la lógica están separados. Dicho patrón es el más usado cuando se quiere construir un *software* con una interfaz gráfica. A grandes rasgos se trata de que el código que sirve para representar la información (la vista) y el código encargado de la interacción con el usuario (la lógica de negocio) se albergue en archivos distintos, hecho que aumenta la legibilidad del código y, por lo tanto, su mantenimiento.

Para implementar el diseño explicado en el apartado 4.2.2, se ha seguido el modelo MVC, para ello, se han creado tres archivos *.fxml* correspondientes a las tres ventanas de las figuras 4.18, 4.24 y 4.26, y por otro lado para cada ventana un archivo *.java*, llamado controlador, donde se encuentra la lógica. En los controladores está el comportamiento que debe tener la vista y las acciones que se deben ejecutar cuando se interaccione con cada uno de los elementos.

Por último, cabe mencionar también que se han creado más archivos *.fxml* correspondientes a los vendedores (figura 4.19), ofertas (figuras 4.20 y 4.21), compradores (figura 4.22) y peticiones (figura 4.23). Dichas vistas se insertan dentro de la ventana principal (figura 4.18) según convenga, se ha decido proceder de esta forma para que las distintas vistas que se tienen que generar sean reutilizables y para que el *software* en sí sea más mantenible.

4.3.2.2. Instanciación de los agentes

Como se ha explicado en la implementación del mercado, sección 4.3.1, se debe crear un archivo *.asl* para cada *vendedor* y *comprador* que disponga de una ofertas o peticiones particulares respectivamente. Por lo tanto, se deben recorrer las listas de *compradores* y *vendedores* que el usuario ha insertado en la interfaz en la ventana principal (4.18), extraer los distintos parámetros y creencias para posteriormente generar los archivos *.asl* en particular a través de la plantilla (donde se dispone del código de los agentes correspondiente a las reglas).

Una vez se haya procedido con los agentes, se tendrá que crear el archivo *.mas2j* donde se recogan todos los agentes creados en la etapa anterior junto al número de agentes que se quieren instanciar de cada tipo. Por último habrá que interpretar dicho archivo con las herramientas proporcionadas por *JASON* para poner en ejecución el sistema multiagente.

4.3.2.3. Salida del mercado

Para saber lo que está pasando en el mercado así como para obtener un resumen del estado global del sistema, en la fase de diseño se decidió crear un agente cuyo único propósito es el de notificar sobre los eventos que se van produciendo, el agente *log*. Dicho agente recibe del resto de agentes ciertos eventos y el agente *log* los procesa para realizar una serie de cálculos, tal y como se ha explicado en la sección 4.2.1.6.

Una vez se produzca un evento y siguiendo con lo explicado en la sección 4.2.2.4, el agente muestra por pantalla el evento producido y el resumen del estado gracias a la acción interna *.print()*. Dicha acción muestra en pantalla una terminal donde aparecen los distintos mensajes que el agente *log* genere.

Cabe mencionar que al mostrar el estado de manera centralizada, los eventos no se mostrarán en el orden de ocurrencia, si no en el orden en el que el agente *log* los recibe y procesa. Sin embargo, como los eventos son procesados uno a uno es decir, con exclusión mutua, el resultado que se va creando si que es el correcto, aunque en algunos casos no sea consiste. Esto puede pasar cuando por ejemplo se muestre que un *vendedor* ha vendido pero que no muestre que un *comprador* ha obtenido dicho producto, dicho estado inconsistente se solucionará cuando el agente procese el evento producido por el *comprador* en cuestión.

4.4 Pruebas de validación

En este apartado se verifica que el comportamiento del mercado es el esperado. Para ello, se han creado diversos casos de prueba sencillos que se han ejecutado sobre el *software* desarrollado. En cada caso, se explica la situación, se razona lo que debe ocurrir y, por último, se comprueba si la salida es la esperada.

Cabe destacar que al tratarse de un entorno no determinista, el comportamiento del mercado no es siempre idéntico, pero si que es similar. Este tema se trata y analiza para cada prueba en particular.

Por último, se realiza una prueba de carga para comprobar empíricamente cual es el tiempo que tarda el programa en realizar todas las negociaciones esperadas. Dicha prueba corresponde con la prueba número 06.

Para la elaboración de las pruebas se ha decidido emplear el sector del plástico debido a que es uno de los más atractivos en cuanto al reciclaje. En dicho sector, el reciclaje es algo que ya está extendido y, por lo tanto, existen estándares para la clasificación del mismo, para ello se usa el *PIC* (Código de Identificación del Plástico). En función del grupo al que pertenezca cada objeto, el proceso de reciclaje será distinto, pero todos los objetos formados por un tipo de plástico se procesan de la misma forma.

A continuación se muestra una tabla con los distintos tipos de plásticos, mencionando su nombre y usos más frecuentes. La columna del nombre es la empleada en la interfaz para definir el identificador del material de cada materia prima con la que se puede negociar.

Nombre	Usos
pet	botellas de agua y refrescos.
hdpe	envases para lácteos, zumos, detergentes, perfumes y champús.
pvc	embalaje productos no alimenticios, aislantes de cables y tuberías.
ldpe	bolsas de basura, bolsas de congelado y botellas exprimibles.
pp	utensilios de cocina, industria automovilística y construcción.
ps	protector de material electrónico y juguetes.

Tabla 4.1: Tabla con las materias primas usadas en la validación

Por otra parte, existen en el mercado también una gran cantidad de productos que están compuestos por más de un tipo de plástico, es aquí donde entra el juego el papel de las transformaciones. Seguidamente se expone una de todas las posibles que pueden existir en este sector.

Nombre	Precio	Componentes
Brick leche	2	• 20 de hdpe • 5 de pet

Tabla 4.2: Tabla con las transformaciones usadas en la validación

4.4.1. Prueba 01: Compra directa

La primera prueba que se ha efectuado consiste en comprobar que el sistema detecta una coincidencia y que el comprador acaba satisfaciendo su petición. Se ha contemplado tanto el caso en el que la coincidencia es con un objeto simple como el caso en el que el objeto en cuestión es compuesto y por lo tanto hace falta aplicar una transformación. En ambos casos, la coincidencia se puede satisfacer de manera directa sin necesidad de formar alianzas, por lo tanto, la cantidad pedida y ofertada deben coincidir.

Para ambas situaciones, el conjunto de compradores está formado por:

Nombre	Peticiones				
	Id	**Cantidad**	**Material**	**Precio máximo**	**Rondas**
Comp1	plastico	500	pet	20	5

Tabla 4.3: Tabla de compradores para la prueba 01 (compra directa)

En el primero caso, el grupo de vendedores es:

Nombre	Ofertas				
	Id	**Precio mínimo**	**Cantidad**	**Material**	**Precio máximo**
Vend1	desechos	5	500	pet	15

Tabla 4.4: Tabla de vendedores para la prueba 01 (compra directa) caso sin transformaciones

En este caso, el comprador tiene que acabar comprando el objeto ofertado por el único vendedor. Al no haber competencia, el precio pagado será el que pida el vendedor siempre y cuando el comprador pueda pagarlo. Si el comprador no pudiese, el vendedor variaría su precio entre el *precio máximo* y el *precio mínimo* para intentar vendedor. En ambos casos, el comprador no puede realizar una contra-oferta ya que no tiene alternativa.

Se ejecuta el programa y el resultado es el esperado:

```
[log] ofertas vendedores: 0/1    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El comprador comp1 ha satisfecho su petición de plastico por un precio de 15
[log] ofertas vendedores: 0/1    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend1 ha vendido su oferta de desechos por un precio de 15
[log] ofertas vendedores: 1/1    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 10    contaminacion: 0
```

Figura 4.28: Salida de la prueba 01 caso 1

Se aprecia como en la situación final que se alcanza, todas las ofertas y peticiones se han satisfecho, los compradores han ganado 5 debido a que el único comprador estaba dispuesto a pagar hasta 20 pero le han hecho una oferta de 15. Por otra parte, las ganancias de los vendedores es de 10 porque lo ha vendido por 15 y como máximo estaba dispuesto a rebajarlo hasta 5. Por último, la contaminación es 0 debido a que ambos agentes estaban situados en el punto *(0,0)* y por lo tanto el coste del transporte es 0, aunque esto no se tiene en cuenta para esta prueba.

Por otra parte, en el segundo caso el conjunto de vendedores es:

Nombre	Ofertas						
	Id	Ítem	Unidades	Precio Transformación	Componentes		
					Material	Precio mínimo	Precio máximo
Vend1	pack leche	Brick leche	100	300	2000 de hdpe	350	600
					500 de pet	5	15

Tabla 4.5: Tabla de vendedores para la prueba 01 (compra directa) caso con transformaciones

En esta situación primero se aplica la transformación ya que el vendedor puede pagar el precio de la transformación y posteriormente se sigue el mismo razonamiento explicado para el primer caso.

Al ejecutar el programa, se comprueba que la salida es la razonada:

```
[log] ofertas vendedores: 0/0    peticiones compradores: 0/0    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend1 ha decidido descomponer su producto pack_leche
[log] ofertas vendedores: 0/1    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El comprador comp1 ha satisfecho su petición de plastico por un precio de 15
[log] ofertas vendedores: 0/1    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend1 ha vendido su oferta de pack_lechepet por un precio de 15
[log] ofertas vendedores: 1/1    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 10    contaminacion: 0
```

Figura 4.29: Salida de la prueba 01 caso 2

En este último caso el vendedor no ha conseguido satisfacer todas sus ofertas ya que el objeto compuesto se ha transformado en dos simples, pero de ellos solo se ha vendido uno, sin embargo al no haber ninguna coincidencia para este segundo objeto se omite en la salida del programa. Por lo tanto, la salida del programa sigue el mismo razonamiento explicado para el primer caso.

Para ambos casos, la posición de los distintos agentes, la estrategia de ordenación, la de negociación y las rondas es irrelevante al haber únicamente una sola coincidencia.

4.4.2. Prueba 02: Subasta

En esta prueba se ha validado que el comportamiento de la subasta es el esperado, es decir, que varía entre el *precio máximo* y el *precio mínimo*, y que se tenga en cuenta la estrategia de negociación. Sin embargo, esto último se ha validado en profundidad en la prueba 04.

Para ello, se va a definir el conjunto de compradores como:

Nombre	**Peticiones**				
	Id	**Cantidad**	**Material**	**Precio máximo**	**Rondas**
Comp1	plastico	500	pet	20	5

Tabla 4.6: Tabla de compradores para la prueba 02 (subasta)

Y el de vendedores:

Nombre	**Ofertas**				
	Id	**Precio mínimo**	**Cantidad**	**Material**	**Precio máximo**
Vend1	desechos	5	500	pet	15
Vend2	desechos	10	500	pet	15

Tabla 4.7: Tabla de vendedores para la prueba 02 (subasta)

En esta situación, al haber dos vendedores cuyos únicos objetos coinciden con la petición del único comprador, los dos vendedores compiten por ser el ganador. Ambos realizarán la primera oferta por 15, uno de ellos será el ganador parcial (dicho agente variará entre una ejecución y otra debido a que es una situación no determinista, lo será el primer agente que se ponga en contacto con el comprador) y el otro recibirá un mensaje informándole de que si se lo vende por menos al comprador, él será el ganador. Dicho procedimiento se repetirá hasta que el *Vend2* no pueda rebajar más su precio y la compra se realizará al *Vend1* por la cantidad que haya ofertado.

La cuantía de dinero que cada agente oferte en cada ronda depende de la estrategia de negociación que siga, aspecto que no se tiene en cuenta en esta prueba. Las β, variable que define la estrategia de negociación, de cada vendedor toman el valor de 1, lo que se traduce en un descenso lineal entre el precio máximo y el mínimo.

Al ejecutar la situación descrita, la salida es:

```
[log] ofertas vendedores: 0/2    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El comprador comp1 ha satisfecho su petición de plastico por un precio de 10
[log] ofertas vendedores: 0/2    peticiones compradores: 1/1    ganancias compradores: 10    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend1 ha vendido su oferta de desechos por un precio de 10
[log] ofertas vendedores: 1/2    peticiones compradores: 1/1    ganancias compradores: 10    ganancias vendedores: 0    contaminacion: 0
```

Figura 4.30: Salida prueba 02

Se aprecia como en este caso, se han satisfecho todas las peticiones de los compradores, pero no todas las ofertas de los vendedores debido a que las dos coincidencias que se han producido lo hacían con la misma petición del comprador, el cual solo está interesado en comprar una de ellas. El resto de campos toman los valores esperados aunque no son relevantes para lo que se pretendía validar con esta prueba.

4.4.3. Prueba 03: Alianzas

En esta tercera prueba se ha verificado que el sistema es capaz de detectar una situación en la que es necesaria una alianza, la alianza se formará y venderá el objeto que representa. La alianza se formará tanto por vendedores que tienen objetos simples como por vendedores que tienen objetos compuestos.

El conjunto de compradores está formado por un único comprador que dispone de una petición para la que hace falta una alianza.

Nombre	Peticiones				
	Id	Cantidad	Material	Precio máximo	Rondas
Comp1	plastico	500	pet	20	5

Tabla 4.8: Tabla de compradores para la prueba 03 (alianza)

El grupo de vendedores está divido en 2:

- Vendedores con un objeto simple:

Nombre	Ofertas				
	Id	Precio mínimo	Cantidad	Material	Precio máximo
Vend1	desechos	1	100	pet	3
Vend2	desechos	1	100	pet	3
Vend3	desechos	1	100	pet	3

Tabla 4.9: Tabla de vendedores para la prueba 03 (alianza) con objeto simple, primer caso

- Vendedores con un objeto compuesto:

Nombre	Ofertas						
	Id	Ítem	Unidades	Precio Transformación	Componentes		
					Material	Precio mínimo	Precio máximo
Vend_trans1	pack leche	Brick leche	20	60	400 de hdpe	70	120
					100 de pet	1	3
Vend_trans2	pack leche	Brick leche	20	60	400 de hdpe	70	120
					100 de pet	1	3

Tabla 4.10: Tabla de vendedores para la prueba 03 (alianza) con objeto compuesto, primer caso

En este caso, el conjunto de las ofertas de todos los vendedores componen la petición exigida por el comprador. Por lo tanto, se creará una única alianza que los represente e intente vender la oferta compuesta al comprador. El dinero exigido será la suma del dinero exigido por cada vendedor, en caso de que el comprador no pudiera pagar tanto o de que haya competencia, la alianza se hubiese disuelto informando a sus miembros para que rebajen el precio. Esto último se ilustra con otro ejemplo más adelante.

Al introducir en el programa esta situación y ejecutarlo, el resultado que se obtiene es:

```
[log] ofertas vendedores: 0/3    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend_trans2 ha decidido descomponer su producto pack_leche
[log] ofertas vendedores: 0/4    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend_trans1 ha decidido descomponer su producto pack_leche
[log] ofertas vendedores: 0/5    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El comprador comp1 ha satisfecho su petición de plastico por un precio de 15
[log] ofertas vendedores: 0/5    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend2 ha vendido su oferta de desechos por un precio de 3
[log] ofertas vendedores: 1/5    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 2    contaminacion: 0

[log] El vendedor vend_trans1 ha vendido su oferta de pack_lechepet por un precio de 3
[log] ofertas vendedores: 2/5    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 4    contaminacion: 0

[log] El vendedor vend_trans2 ha vendido su oferta de pack_lechepet por un precio de 3
[log] ofertas vendedores: 3/5    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 6    contaminacion: 0

[log] El vendedor vend1 ha vendido su oferta de desechos por un precio de 3
[log] ofertas vendedores: 4/5    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 8    contaminacion: 0

[log] El vendedor vend3 ha vendido su oferta de desechos por un precio de 3
[log] ofertas vendedores: 5/5    peticiones compradores: 1/1    ganancias compradores: 5    ganancias vendedores: 10    contaminacion: 0
```

Figura 4.31: Salida de la prueba 03, primer caso

Se aprecia como el resultado es el esperado. El conjunto de los vendedores ganan 2 unidades monetarias debido a que lo consiguen vender por 3 y como mucho podían rebajar hasta 1. La alianza vende por 15 ($5 \quad vendedores \quad * \quad 3 \quad unidades\_monetarias$).

Sin embargo, si hubieran habido más vendedores, y por lo tanto competencia, se habrían formado diversas alianzas que compiten entre sí. En este caso el grupo de vendedores está divido en:

- Vendedores con un objeto simple:

Nombre	Ofertas				
	Id	Precio mínimo	Cantidad	Material	Precio máximo
Vend1	desechos	1	100	pet	3
Vend2	desechos	1	100	pet	3
Vend3	desechos	1	100	pet	3
Vend4	desechos	1	100	pet	3

Tabla 4.11: Tabla de vendedores para la prueba 03 (alianza) con objeto simple, segundo caso

- Vendedores con un objeto compuesto:

Nombre	Ofertas						
	Id	Ítem	Unidades	Precio Transformación	Componentes		
					Material	Precio mínimo	Precio máximo
Vend_trans1	pack leche	Brick leche	20	60	400 de hdpe	70	120
					100 de pet	1	3
Vend_trans2	pack leche	Brick leche	20	60	400 de hdpe	70	120
					100 de pet	1	3

Tabla 4.12: Tabla de vendedores para la prueba 03 (alianza) con objeto compuesto, segundo caso

Lo esperado para esta situación es que la alianza venda por la suma del mínimo exigido por cada agente, es decir por 5. Esto es así ya que los agentes habrán ido disminuyendo su cantidad para intentar ser el ganador cada vez que una alianza haya sido rechazada hasta llegar a su mínimo en la última ronda de la negociación. Sin embargo, el agente excluido de la alianza ganadora variará de una ejecución a otra, ya que depende del orden final en el que los distintos agentes interactúen.

```
[log] ofertas vendedores: 0/4    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend_trans2 ha decidido descomponer su producto pack_leche
[log] ofertas vendedores: 0/5    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend_trans1 ha decidido descomponer su producto pack_leche
[log] ofertas vendedores: 0/6    peticiones compradores: 0/1    ganancias compradores: 0    ganancias vendedores: 0    contaminacion: 0

[log] El comprador comp1 ha satisfecho su petición de plastico por un precio de 5
[log] ofertas vendedores: 0/6    peticiones compradores: 1/1    ganancias compradores: 15    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend1 ha vendido su oferta de desechos por un precio de 1
[log] ofertas vendedores: 1/6    peticiones compradores: 1/1    ganancias compradores: 15    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend_trans1 ha vendido su oferta de pack_lechepet por un precio de 1
[log] ofertas vendedores: 2/6    peticiones compradores: 1/1    ganancias compradores: 15    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend3 ha vendido su oferta de desechos por un precio de 1
[log] ofertas vendedores: 3/6    peticiones compradores: 1/1    ganancias compradores: 15    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend4 ha vendido su oferta de desechos por un precio de 1
[log] ofertas vendedores: 4/6    peticiones compradores: 1/1    ganancias compradores: 15    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend_trans2 ha vendido su oferta de pack_lechepet por un precio de 1
[log] ofertas vendedores: 5/6    peticiones compradores: 1/1    ganancias compradores: 15    ganancias vendedores: 0    contaminacion: 0
```

Figura 4.32: Salida prueba 03, segundo caso

Al haber competencia pero solo por parte de los vendedores, es decir, los vendedores solo disponen de una alternativa, las ganancias de los compradores aumentan. Mientras que por otra parte, las ganancias de los vendedores decrecen.

Para ambos casos, la posición de los distintos agentes, la estrategia de ordenación, la de negociación y las rondas es irrelevante al haber únicamente una única coincidencia.

4.4.4. Prueba 04: Variación según la estrategia

A lo largo de esta prueba se ha verificado que el descenso del precio ofertado por parte de los vendedores depende del valor de β, parámetro que define la estrategia de negociación, tal y como se ha explicado en la fase de diseño. Para ello, se va a modificar el código de los compradores para que al recibir una oferta de un agente, muestren por pantalla el nombre del agente en cuestión y el precio exigido en cada ronda.

Recordemos que los valores de β significan:

- $\beta = 1$: comportamiento lineal, los descensos entre cada iteración son siempre idénticos.
- $0 < \beta < 1$: comportamiento exigente, se disminuye poco el precio en las primeras rondas y en las últimas los saltos son más pronunciados.
- $1 < \beta < \infty$: comportamiento benévolo, al principio disminuye el precio bruscamente y luego varía poco el precio hasta llegar el precio mínimo.

Se va a definir un único comprador:

Nombre	**Peticiones**				
	Id	**Cantidad**	**Material**	**Precio máximo**	**Rondas**
Comp	plastico	500	pet	15	5

Tabla 4.13: Tabla de compradores para la prueba 04 (variación según estrategia)

Y diversos vendedores cada uno de ellos con distintos valores de β:

Nombre	β	**Ofertas**				
		Id	**Precio mínimo**	**Cantidad**	**Material**	**Precio máximo**
Vend_0,2	0,2	desechos	5	500	pet	15
Vend_0,5	0,5	desechos	5	500	pet	15
Vend_1	1	desechos	5	500	pet	15
Vend_2	2	desechos	5	500	pet	15
Vend_4	4	desechos	5	500	pet	15

Tabla 4.14: Tabla de vendedores para la prueba 04 (variación según estrategia)

El método de ordenación de los vendedores y las posiciones de los distintos agentes no es relevante debido a que todas las coincidencias producidas suceden con la misma petición del comprador. Además y gracias a ello, se puede verificar si la variación de precio de cada agente en las distintas rondas sigue los valores esperados.

Las distintas ofertas producidas son:

```
[comp] vend_2  15
[comp] vend_02  15
[comp] vend_1  15
[comp] vend_4  15
[comp] vend_05  15
[comp] vend_02  14
[comp] vend_1  11
[comp] vend_4  7
[comp] vend_05  13
[comp] vend_02  14
[comp] vend_1  9
[comp] vend_2  8
[comp] vend_02  11
[comp] vend_05  11
[comp] vend_1  7
[comp] vend_2  7
[comp] vend_02  5
[comp] vend_1  5
[comp] vend_05  8
[comp] vend_4  6
[comp] vend_2  6
[comp] vend_05  5
[comp] vend_4  5
[comp] vend_4  5
[comp] vend_2  5
[log] El comprador comp ha satisfecho su petición de plastico por un precio de 5
[log] ofertas vendedores: 0/5    peticiones compradores: 1/1    ganancias compradores: 10    ganancias vendedores: 0    contaminacion: 0

[log] El vendedor vend_02 ha vendido su oferta de desechos por un precio de 5
[log] ofertas vendedores: 1/5    peticiones compradores: 1/1    ganancias compradores: 10    ganancias vendedores: 0    contaminacion: 0
```

Figura 4.33: Salida de la prueba 04

Se aprecia como cada agente desciende su precio de una forma distinta, para una mayor comprensión se ha sintetizado dicha información en una gráfica.

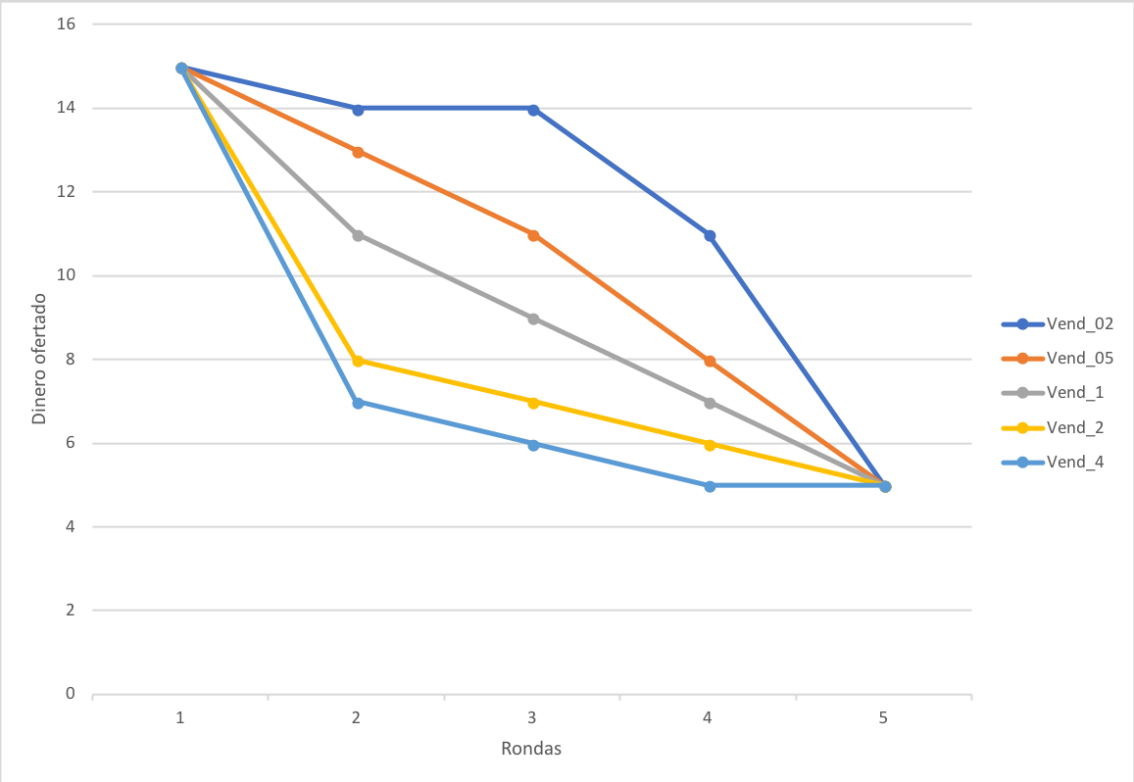


Figura 4.34: Gráfica sintetizando los datos de la prueba 04

Se observa como el resultado es el esperado. En función del valor de β la curva tiene una forma u otra, pero siguiendo los 3 comportamientos identificados: *lineal*, *exigente* y *benévolo*.

4.4.5. Prueba 05: Variación según la ordenación

A continuación se ha verificado que los distintos vendedores priorizan su coincidencias en función de sus parámetros.

Primero se valida el funcionamiento de la ordenación *en función del precio* y de la de *en función de la contaminación*. Para ello, se define:

- **Conjunto de compradores**

Nombre	Posición(X,Y)	Peticiones				
		Id	Cantidad	Material	Precio máximo	Rondas
Comp_cerca_1	(1,1)	plastico	500	pet	5	5
Comp_cerca_2	(1,1)	plastico	500	pet	5	5
Comp_lejos_1	(10,10)	plastico	500	pet	10	5
Comp_lejos_2	(10,10)	plastico	500	pet	10	5

Tabla 4.15: Tabla de compradores para la prueba 05 (variación según la ordenación), primer caso

- **Conjunto de vendedores**

Nombre	Ofertas				
	Id	Precio mínimo	Cantidad	Material	Precio máximo
Vend_beneficio	desechos	5	500	pet	15
Vend_contaminacion	desechos	5	500	pet	15

Tabla 4.16: Tabla de vendedores para la prueba 05 (variación según la ordenación), primer caso

En este caso, los dos vendedores están ubicados en la posición *(0,0)*. Por otra parte, la estrategia de negociación se configura a $\beta = 1$ para que el descenso sea lineal.

Se aprecia que disponemos de 2 compradores de cada tipo para que cada vendedor pueda ofertar su producto al comprador que razone que es el mejor y dicho agente se lo pueda comprar, sin que haya competencia entre los vendedores.

En esta situación cada vendedor seguirá un razonamiento distinto:

- ***vend_beneficio***: dicho agente estima al inicio que la mejor oferta para las cuatro coincidencias que ha recibido es intentar venderlo por 15, sabe también que cada comprador permite hasta 5 rondas y que puede descender hasta 5 (los precios ofertados serán por lo tanto 15, 12, 9, 6, 5). Cada vez que fracase una negociación, el agente elegirá la coincidencia con la que estima un mayor beneficio.

  El procedimiento que sigue el agente es el siguiente: le realiza la oferta de por 15 a cualquiera de los agentes (esto es no es determinista y por lo tanto variará de una ejecución a otra al estimar el mismo beneficio para todas las opciones que tiene). Al ser rechazado procederá de manera similar con el resto de agentes y, una vez todos los intentos de venderlo por 15 hayan fracasado, intentará venderlo por 12. Al fracasar también, procederá a realizar la oferta pidiendo 9, en este caso, se lo

venderá al primero agente del tipo *comp_lejos* con el que contacte debido a que son los únicos capaces de pagar ese precio.

Cabe destacar que el descenso de la cantidad ofertada (15, 12, 9, 6, 5) podría haber sido distinta para cada coincidencia. Por ejemplo, si el *comp_lejos* hubiese permitido 10 rondas para su objeto, las cantidades ofertadas realizadas a dicho agente habrían sido: 15, 14, 13, 12, 11, 10 y al llegar a 10 se lo habría comprado debido a que es la cantidad que el agente está dispuesto a pagar, y debido también al hecho de que no existe una oferta mejor.

- ***vend_contaminacion***: en este caso, la ordenación de las coincidencias se realiza únicamente por la distancia entre la posición del vendedor que razona y el comprador con el que existe cada una de las coincidencias.

  Por lo tanto, mientras no se hayan consumido todas las rondas permitidas, intentará vendérselo al *comp_cerca*. Dicho agente se lo compra una vez el agente lo rebaje a 5, debido a que es la cantidad máxima que el comprador está dispuesto a pagar. En este caso, al ser el precio máximo, el mínimo y la estrategia de negociación igual que en el caso anterior, las distintas ofertas que se realizarían son: 15, 12, 9, 6, 5.

  Cabe mencionar que si el *comp_cerca* no hubiera podido pagar el precio mínimo del vendedor, el agente hubiera procedido a negociar con el *comp_lejos* debido a que sería la única alternativa que dispone.

Al ejecutar el programa, la salida producida es la siguiente:

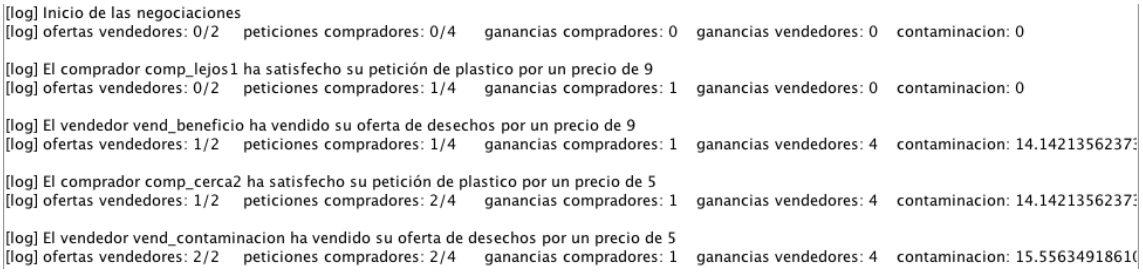


Figura 4.35: Salida de la prueba 05, primer caso

Se aprecia como ocurre lo esperado, el vendedor que ha ordenado por beneficio se lo vende al comprador ubicado más lejos, ya que es el que mayor beneficio le aporta, pero también una mayor contaminación. Por otra parte, el vendedor que ordena por distancia se lo ha vendido al comprador que está situado más cerca pero por una cantidad menor, por lo tanto, reduciendo su beneficio.

Dichos eventos se aprecian en la salida, cuando vende el *vend_contaminación* sube poco la contaminación pero no se producen ganancias. Mientras que cuando vende el *vend_beneficio* la contaminación aumenta más que en el caso anterior, pero el beneficio si que crece.

Por último en esta prueba, se ha verificado el funcionamiento del método de ordenación *en función del precio y de la contaminación*. Para ello, hace falta definir la variable *c*, que es la que valora la cuantía de dinero con la que se valora la contaminación producida por cada kilómetro. Lo esperado es que cuanto mayor sea la *c*, el vendedor valore más el

coste del transporte, por lo tanto, preferirá comprar a agentes más cercanos (para que el coste de la contaminación sea menor), aunque ello implique un menor beneficio.

Para este fin se van a definir:

- **Conjunto de compradores**

Nombre	Posición(X,Y)	Peticiones				
		Id	Cantidad	Material	Precio máximo	Rondas
Comp_2	(2,0)	plastico	500	pet	5	10
Comp_4	(4,0)	plastico	500	pet	8	10
Comp_8	(8,0)	plastico	500	pet	12	10
Comp_16	(16,0)	plastico	500	pet	16	10
Comp_32	(32,0)	plastico	500	pet	20	10

Tabla 4.17: Tabla de compradores para la prueba 05 (variación según la ordenación), segundo caso

- **Conjunto de vendedores**

Nombre	Posición(X,Y)	Ofertas				
		Id	Precio mínimo	Cantidad	Material	Precio máximo
Vend	(0,0)	desechos	5	500	pet	20

Tabla 4.18: Tabla de vendedores para la prueba 05 (variación según la ordenación), segundo caso

La estrategia de negociación se configura a $\beta = 1$ para que el descenso sea lineal. Por otra parte, el método de ordenación es *por precio y distancia* y el valor de la variable *coste de contaminación*, la *c*, se ha ido variando.

Para realizar esta prueba, se han lanzado una serie de ejecuciones variando únicamente la *c* del vendedor entre cada una de ellas. Los datos obtenidos han sido interpretados y recogidos en una tabla.

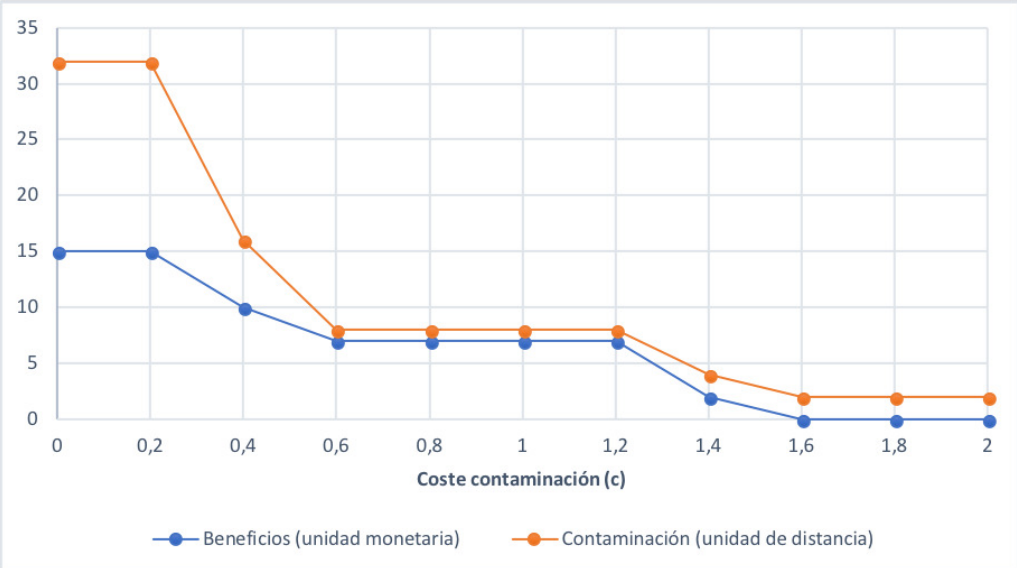


Figura 4.36: Gráfica representado las salidas de la prueba 05, segundo caso

Se observa lo esperado, conforme aumenta la c, el vendedor razona que es mejor vender a los compradores que están ubicados más cerca con el objetivo de contaminar menos aunque ello implique una bajada en los beneficios potenciales obtenidos por el vendedor.

Recordemos que la fórmula usada para saber cual es la coincidencia más ventajosa es: $posible\_dinero\_a\_ganar - c * distancia$. Se puntúan todas ellas y se elige la de mayor beneficio.

4.4.6. Prueba 06: Carga

En esta última prueba se ha comprobado empíricamente la cantidad de agentes que soporta el mercado y que a su vez, el comportamiento sea el esperado. Para ello se han lanzado una serie de ejecuciones para medirlo.

El ordenador con el que se han ejecutado dispone de:

- **Procesador**: 2,6 GHz Intel Core i5
- **Memoria RAM**: 8Gb 1600 MHz DDR3
- **Tarjeta gráfica**: Intel Iris 1536MB

Se crea para medirlo un comprador y un vendedor de los cuales se lanzan varias copias idénticas:

Nombre	Peticiones				
	Id	Cantidad	Material	Precio máximo	Rondas
Comp	plastico	500	pet	20	5

Tabla 4.19: Tabla de compradores para la prueba 06 (carga)

Nombre	Ofertas				
	Id	Precio mínimo	Cantidad	Material	Precio máximo
Vend	desechos	5	500	pet	15

Tabla 4.20: Tabla de vendedores para la prueba 06 (carga)

Las posiciones de los agentes se ha configurado en *(0,0)*, la ordenación se ha seleccionado que prime el beneficio y, por último, $\beta = 1$, lo que se traduce en una estrategia lineal para las negociaciones.

Se han lanzado en todas las pruebas el mismo número de vendedores y de compradores, de tal manera que todos los agentes puedan quedar satisfechos. Sin embargo, los agentes deben de negociar para encontrar la situación en la que todos los vendedores ofertan su producto al comprador con el que obtienen el mayor beneficio posible.

Para esta prueba en particular, se ha modificado el código del agente *log* para que calcule el tiempo que transcurre desde que se pone en marcha hasta que todas las coincidencias hayan sido satisfechas. Sin embargo, la función implementada para medir el tiempo tiene la precisión de segundos, por lo que para pruebas con pocos agentes, el valor no será significativo.

Se han lanzado diversas pruebas con distintos números de agentes y se han medido los tiempo de ejecución para cada situación. Como el tiempo de ejecución puede variar según otros procesos que se estén ejecutando en el ordenador, se va a lanzar cada prueba 5 veces y en la tabla se pondrá el valor de la mediana. Sobre el rango, se va a comenzar con 20 agentes de cada tipo y se irá incrementado en 20 unidades hasta llegar a una configuración que tarde más de 10 minutos (600 segundos).

Número de *compradores* y *vendedores*	**Tiempo transcurrido (segundos)**
20	6
40	7
60	9
80	11
100	15
120	20
140	32
160	43
180	68
200	90
220	120
240	165
260	217
280	279
300	343
320	405
340	547

Tabla 4.21: Tabla con los tiempos de las ejecuciones de la prueba 06 (carga)

Cabe destacar que el tiempo con el que se ha configurado el tiempo de espera para que una negociación sea satisfactoria es de 5 segundos.

Para ver de una forma más visual la correlación entre el tiempo de ejecución hasta encontrar la situación en la que todos los agentes hayan satisfecho sus exigencias y el número de agentes, se ha realizado una gráfica que sintetice dichos datos:

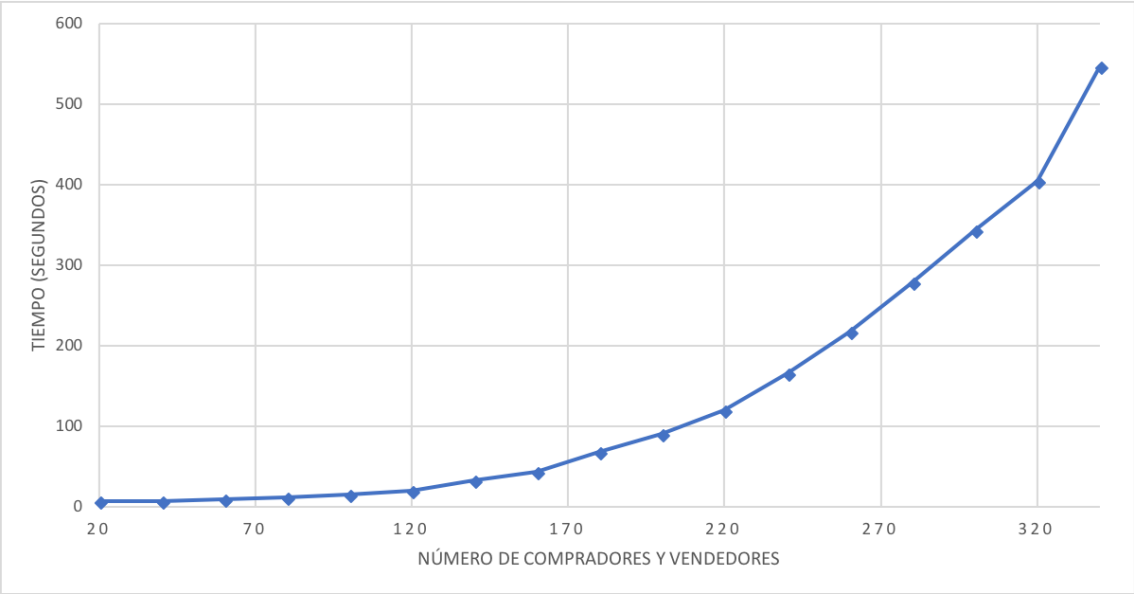


Figura 4.37: Gráfica sintetizando el tiempo de la pruba 06

Se aprecia como para la situación planteada, conforme aumentan el número de agentes, el tiempo necesario para llegar a la situación final crece exponencialmente. Esto se debe principalmente al hecho de que un único agente, el *manager*, debe calcular todas las coincidencias para posteriormente mandárselas al resto de agentes.

CAPÍTULO 5

Conclusiones

A lo largo del capítulo 4 se ha explicado como se han abordado cada una de las etapas de desarrollo del *software* haciendo uso del conocimientos adquirido a lo largo de la carrera junto a la diversa documentación consultada. Lo que ha permitido realizar cada uno de los pasos de manera competente y similar a como se haría en el mundo de la investigación o en el empresarial.

En el trabajo realizado se ha expuesto un primero enfoque al mercado propuesto en [11]. Dicho enfoque ha consistido en el diseño de la arquitectura del sistema multi-agente, las interacciones de los agentes y las estrategias que rigen su comportamiento. El planteamiento es válido, tal y como se ha podido observar en la sección 4.4, para todos los requisitos extraídos durante el análisis, sin embargo algunos de los procedimientos implementados se podrían estudiar con una mayor profundidad.

Este primer prototipo puede usado en dos ámbitos distintos:

- **Controlar de forma automática un sistema de logística inversa**: este proyecto puede ser el punto de partida para realizar un *software* usado por las distintas empresas de un sector en concreto para que se gestione la logística inversa de forma totalmente automática a partir de las peticiones y desechos generados por cada una de ellas.
- **Analizar el mercado frente a ciertos cambios**: el proyecto puede ser usado, por otra parte, como un simulador para calcular la reacción del mercado frente a la aparición de algún impuesto adicional, leyes medioambientales o algún tipo de restricción en particular.

Dicha propuesta fue desarrollada en colaboración con el grupo de investigación GTI-IA de la UPV para ser enviada a *The Agreement Technologies Conference (AT 2017)* ([9]), la cual fue posteriormente aceptada. A fecha de la escritura de la memoria (julio del 2018) se encuentran en edición todos los artículos aceptados y serán publicados en el futuro. El artículo que trata sobre la propuesta de la que trata este TFG es [12].

Para la realización del proyecto se ha seguido una metodología ágil donde semanalmente se fijaban nuevos objetivos, lo que conllevaba cambiar alguna de las etapas de desarrollo. Con ello, se ha comprobado empíricamente como el desarrollo del *software*

no es algo lineal si no que es un proceso iterativo donde hay que revisar constantemente las distintas fases, principalmente la de diseño, para que cumplan con los objetivos definidos. Posteriormente se procedió a la escritura de la memoria.

5.1 Futuras líneas de trabajo

Este trabajo propone el punto de partida para desarrollar extensiones en el futuro. A lo largo de la memoria se han propuesto procedimientos específicos para cada una de las partes del sistema multi-agente. Sin embargo, cada parte puede ser modificada o ampliada para dotar a los agentes de un razonamiento más inteligente.

A continuación se van a detallar las distintas funciones de los agentes que podrían ser alteradas:

- **Formación de alianzas**: actualmente la formación de las alianzas lo realiza el *manager* a petición de los *vendedores*. Se podría apostar por un enfoque más distribuido donde los *vendedores* se encarguen de formar las alianzas, lo que disminuiría considerablemente la carga que tiene el *manager* en el diseño propuesto.

  No obstante y tal y como se ha mencionado en la sección de diseño, dicho procedimiento alternativo no es neutral ya que cada agente buscará lo mejor para él. Es por ello que se debería encontrar un punto intermedio donde un único agente no se tenga que encargar de todo el cálculo, pero que al mismo tiempo las alianzas creadas no primen a ningún agente en particular, si no al bien común de todos sus integrantes.

- **Variación de precio**: en el diseño propuesto se ha decidido que la variación del precio entre las distintas rondas de una negociación siga lo explicado en [15]. Sin embargo, se podrían seguir otras funciones o hacer que la función utilizada considere más información para realizar los cálculos. Por ejemplo, se puede tener en cuenta el beneficio ya obtenido con otras negociaciones lo que permitiría rebajar aún más el precio ofertado; o que la estrategia de negociación (parámetro que actualmente se fija al inicio de la ejecución) se adapte en función de las ganancias que se hayan obtenido, es decir, que al inicio tenga una estrategia más benevolente para intentar obtener beneficios aunque sean bajos y, cuando ya se hayan obtenido algunos, variar la estrategia hacia una más exigente para intentar ganar lo máximo posible en las siguiente negociaciones.

- **Elección de la mejor coincidencia**: se ha decido que la priorización de las distintas coincidencias pueda seguir 3 funciones: *en función del precio*, *en función de la contaminación* o *en función del precio y la contaminación*. No obstante, se pueden estudiar otras formas de ordenarlas para que se tengan en cuenta más factores o profundizar más en los factores que ya tiene en cuenta.

  Se podría tener en cuenta, por ejemplo, el coste monetario que conlleva el transporte, el salario de los trabajadores y algún posible impuesto que pueda existir. Por otra parte, la función usada para estimar la distancia que actualmente calcula la distancia en línea recta entre ambas posiciones, se podría extender para que tenga

en cuenta las carreteras, y el hecho de tener que usar o no barco o avión, así como también el tipo o tipos de vehículos usados.

- **Descomposición de los productos**: las transformaciones pensadas suponen casos ideales donde un producto se transforma en una o varias materias primas. Sin embargo, esto puede que no sea así en todos los casos, si no que por ejemplo un producto compuesto se puede transformar en otros compuestos los cuales sí que se pueden transformar en materias primas. Por lo tanto se podría plantear que a un producto le hagan falta una serie de transformaciones para poder negociar con él en lugar de solo una, cada una de ellas tendrá ligado un coste y solo se realizará si el agente razona que es rentable.

  Por otra parte, para realizar la transformación hace falta que el producto sea transportado a una fábrica para ser procesado. Dicho transporte conlleva un coste que dependerá de la posición de la fabrica y de la posición del *vendedor*. Así que el dinero del transporte y la contaminación producida también se podría tener en cuenta por parte del agente para decidir si le conviene transformarlo para poder negociar con él o no.

Bibliografía

[1] Artículo de *Eurostat* sobre los desechos generados por los países europeos. http://ec.europa.eu/eurostat/statistics-explained/index.php/Waste_statistics.

[2] Directiva europea (2008/98/EC) relativa a residuos. http://eur-lex.europa.eu/eli/dir/2008/98/oj.

[3] Página web oficial de la plataforma JADE. http://jade.tilab.com/.

[4] Página web oficial de FIPA (*Foundation for Intelligent Physical Agents*). http://www.fipa.org/.

[5] Página web oficial de la plataforma JASON. http://jason.sourceforge.net.

[6] Página web oficial del *software* empleado para la realización de los bocetos. https://pencil.evolus.vn/.

[7] Página web oficial del *software* empleado para la realización del diagrama de clases. https://www.draw.io/.

[8] Página web oficial del *software* empleado para la realización de los diagramas de secuencia. https://sequencediagram.org/.

[9] Página web oficial de la conferencia sobre tecnologías de acuerdo a la que fue enviada la propuesta. https://at2017.ibisc.univ-evry.fr.

[10] Marisa P. de Brito, Rommert Dekker y Simme Douwe P. Flapper. *Reverse Logistics: A Review of Case Studies Distribution Logistics. Lecture Notes in Economics and Mathematical Systems*, vol 544. Springer, Berlin, Heidelberg.

[11] Adriana Giret y Miguel A. Salido *A Multi-agent Approach to Implement a Reverse Production Virtual Market in Green Supply Chains*. *Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing. APMS 2017. IFIP Advances in Information and Communication Technology*, vol 514. *Springer*, Cham.

[12] Adriana Giret, Adrian Martinez y Vicente Botti *A Multi-agent approach for composing negotiation items in a reverse logistic virtual market*. *The Agreement Technologies conference (AT 2017), Springer* (en edición).

[13] Stuart J. Russell y Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, tercera edición, 2009.

[14] Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Paperback, 2000.

[15] Vicente Botti. Apuntes asignatura Agentes Inteligentes (AIn). ETSInf (UPV), 2016.

APÉNDICE A

Configuración del sistema

El proyecto desarrollado a lo largo de esta memoria puede ser consultado, tal y como se explica en el apéndice B. Para poder ejecutarlo hace falta tener instalado y configurado correctamente:

- ***Java jre***: necesario para poder compilar y ejecutar proyectos usando el lenguaje de programación *Java*.
- ***Ant***: herramienta usada por *JASON* al compilar los agentes.
- ***Eclipse***: entorno de desarrollo con el que se ha realizado el proyecto, es necesario para poder compilarlo y ejecutarlo. Sin embargo, se podría usar otro entorno.

  Se debe configurar el entorno en cuestión para que al compilar use la variable *PATH* definida por el sistema, para ello, clicamos en la barra de herramientas de *eclipse* la opción *Run>Run configurations...* y en el dialogo mostrado *Environments*. Una vez allí, se añade la variable *PATH*.
- ***e(fx)clipse***: es un *plugin* para el entorno *eclipse* que proporciona todas las librerías necesarias para poder compilar y ejecutar código de *JavaFX*, usado para las interfaces.
- ***JASON***: plataforma de desarrollo multi-agente con la que se han desarrollado los diversos agentes. Es necesaria para poder ejecutarlos.

  Una vez descargado el *software* desde [5], se debe configurar indicándole donde se encuentra instalado *Java*, para ello se debe ejecutar el archivo *jason-*.jar* que se encuentra dentro de la carpeta *libs* del *software* descargado.

APÉNDICE B

Código del proyecto

Todo el código desarrollado para este proyecto se encuentra disponible en: https://github.com/admarcar/TFG.

En dicho repositorio podemos encontrar la siguiente estructura dentro de la carpeta *src*:

- *agents*: se encuentra el código de los agentes junto al de las acciones internas que se han tenido que desarrollar.
- *application*: se encuentra el archivo que es el punto de inicio del *software*, es el encargado de lanzar la ventana principal de la aplicación.
- *clases*: dispone de todas las clases *Java* necesarias para albergar la ontología al ser introducida por el usuario.
- *controller*: se hayan los controladores de cada una de las distintas ventanas que componen la aplicación.
- *css*: se encuentra el archivo de estilo usado a lo largo de la aplicación.
- *scripts*: se hayan los distintos *scripts* proporcionados por *JASON* para poner en marcha el sistema multi-agente a partir del archivo *.mas2j*.
- *tmp*: ejemplo de los agentes finales generados después de que el usuario haya insertado una situación y la haya ejecutado.
- *view*: se encuentran los distintos archivos *.fxml* que albergan las distintas vistas que forman la aplicación.

APÉNDICE C

Guía de usuario

Este anexo consiste en una guía para explicar el uso de la aplicación desarrollada. Para ello, se muestra paso por paso como poner en ejecución una de las situaciones utilizadas durante la validación del *software*, en concreto se ha usado el primer caso planteado en la prueba 03, la cual estaba formada por:

- Materias primas:

Nombre
pet
hdpe

Tabla C.1: Tabla con las materias primas usadas para la *Guía de usuario*

- Transformaciones:

Nombre	Precio	Componentes
Brick leche	2	• 20 de hdpe • 5 de pet

Tabla C.2: Tabla con las transformaciones usadas para la *Guía de usuario*

- Conjunto de compradores:

Nombre	Peticiones				
	Id	Cantidad	Material	Precio máximo	Rondas
Comp1	plastico	500	pet	20	5

Tabla C.3: Tabla de compradores usadas para la *Guía de usuario*

- Conjunto de vendedores: grupo dividido a su vez en:
  - Vendedores con un objeto simple:

Nombre	Ofertas				
	Id	Precio mínimo	Cantidad	Material	Precio máximo
Vend1	desechos	1	100	pet	3
Vend2	desechos	1	100	pet	3
Vend3	desechos	1	100	pet	3

Tabla C.4: Tabla de vendedores con objeto simple usadas para la *Guía de usuario*

- Vendedores con un objeto compuesto:

Nombre	Ofertas						
	Id	Ítem	Unidades	Precio Transformación	Componentes		
					Material	Precio mínimo	Precio máximo
Vend_trans1	pack leche	Brick leche	20	60	400 de hdpe	70	120
					100 de pet	1	3
Vend_trans2	pack leche	Brick leche	20	60	400 de hdpe	70	120
					100 de pet	1	3

Tabla C.5: Tabla de vendedores con objeto compuesto usadas para la *Guía de usuario*

Al ejecutar el proyecto, se visualiza la siguiente ventana:

Figura C.1: Ventana principal del programa

Lo primero que se debe hacer es rellenar el campo de *Tiempo de espera* para introducir el tiempo que debe transcurrir sin que un *comprador* reciba una oferta mejor para aceptar la mejor oferta hasta el momento. Dicho parámetro es global para todo el mercado.

A continuación se introducen las materias primas con las que los agentes podrán negociar, para ello se clica en el botón *Materiales*, el cual abre una tabla con las materias primas actuales:

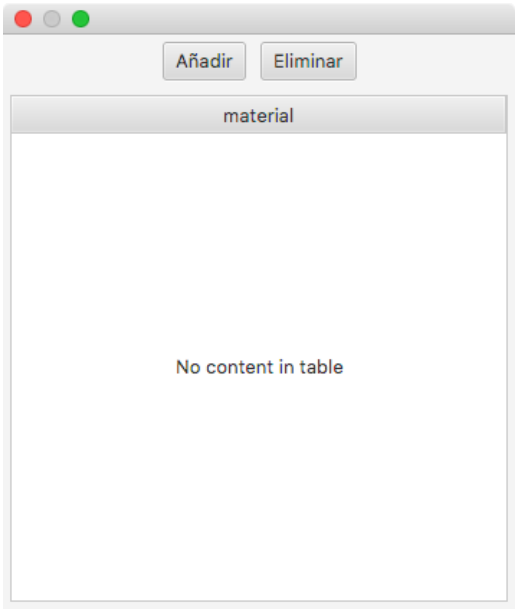


Figura C.2: Ventana del listado de materiales actuales

Clicando en el botón de *Añadir* muestra un diálogo para añadir una materia más:

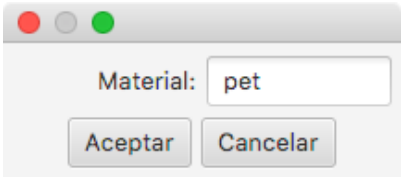


Figura C.3: Ventana del dialogo para añadir un material

Una vez se hayan añadido las dos materias primas, el listado de materiales quedará:

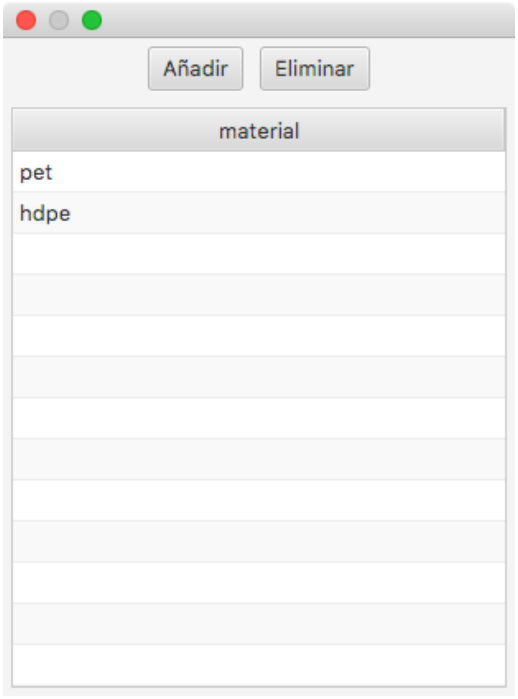


Figura C.4: Ventana del listado de materiales completa

Seguidamente se debe proceder de manera similar pero con las transformaciones, para ello, se clica en el botón *Transformaciones* de la ventana principal con el que se abre una tabla similar a la anterior:

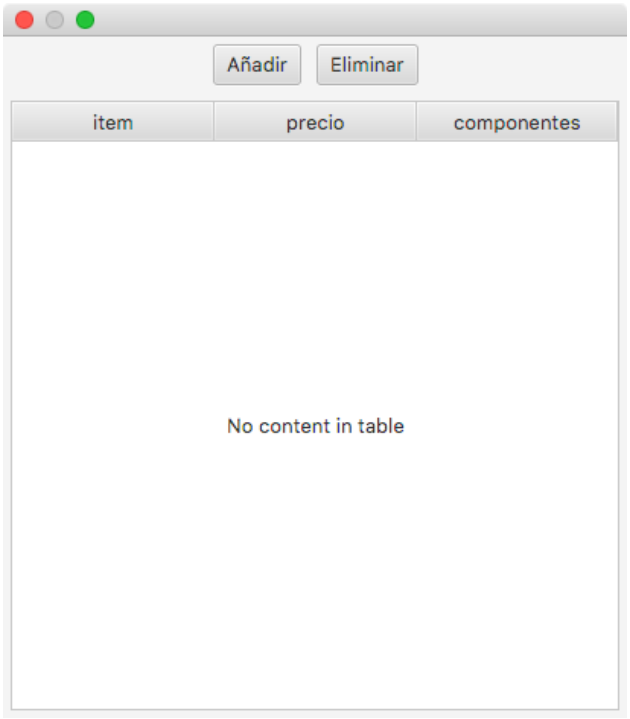


Figura C.5: Ventana del listado de transformaciones actuales

Cuando se pincha en el botón de *Añadir*, muestra un diálogo para añadir una transformación. El campo de *Ítem* será el identificador del producto que se puede transformar

y los distintos componentes son las materias primas (definidas en el paso anterior) que se obtienen al aplicar la transformación.

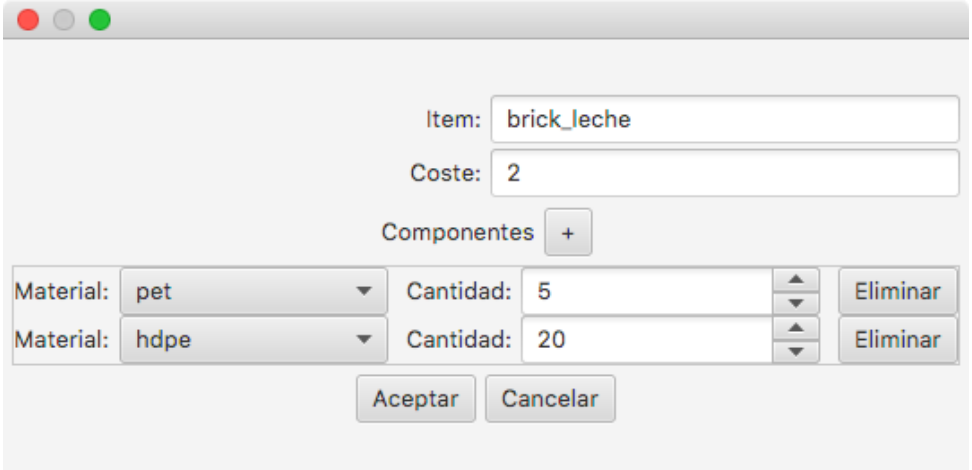


Figura C.6: Ventana del dialogo para añadir una transformación

Después de realizar dicho paso, el listado de transformaciones queda:

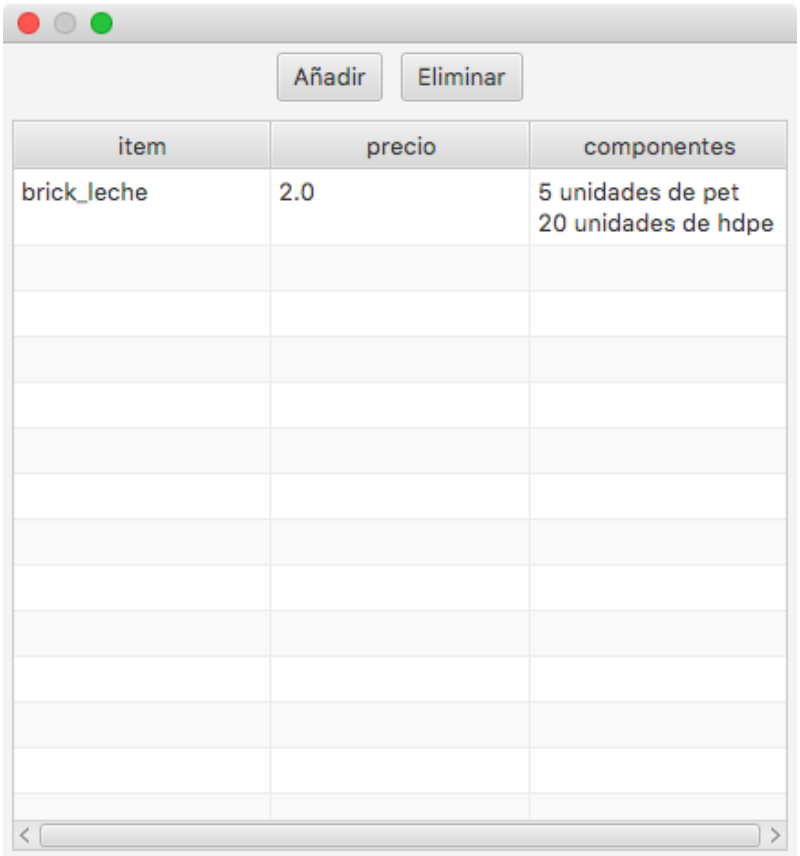


Figura C.7: Ventana del listado de transformaciones completa

Una vez ya se tenga la ontología introducida, se procede a introducir a los diversos agentes. Se aprecia como aunque hayan 5 vendedores definidos en las tablas, solo hay 2 tipos de vendedores ya que los 3 en la tabla C.3 y los 2 en C.5 negocian con el mis-

mo producto y las mismas exigencias de precio. Por lo tanto se definen ambos tipos de vendedores, pero indicando que se quiere más de una copia para cada uno de ellos.

Sobre el métodos de ordenación se elige primar el beneficio y para la estrategia seguida se elige una $\beta = 1$ lo que se traduce en un comportamiento lineal, tal y como está explicado en la sección 4.2.1.4.

A continuación se añade primero al comprador, para ello se clica en *Añadir Comprador* y se rellenan los campos exigidos, tanto los del comprador como los de la petición.

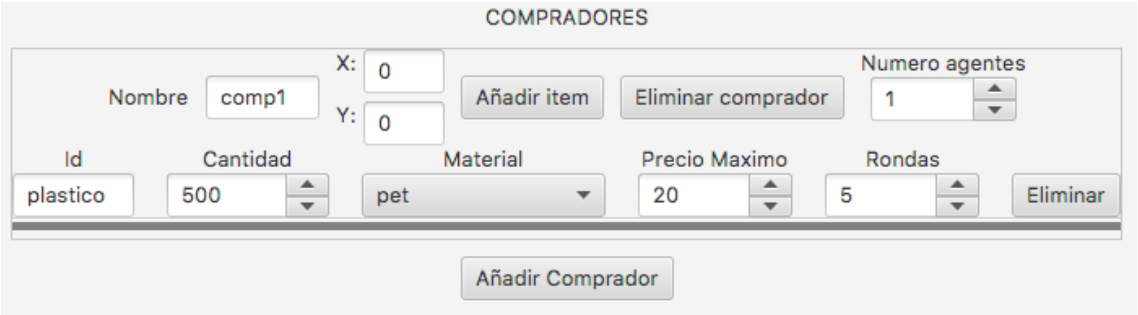


Figura C.8: Ventana principal después de añadir los datos del comprador

Posteriormente se clica 2 veces en el botón de *Añadir Vendedor* para añadir los 2 tipos de agentes. En uno de ellos se clica en el botón *Añadir ítem* y en el otro *Añadir compuesto*, seguidamente se rellena cada agente junto a su oferta. Se indica también el número de agentes que se desea de cada tipo.

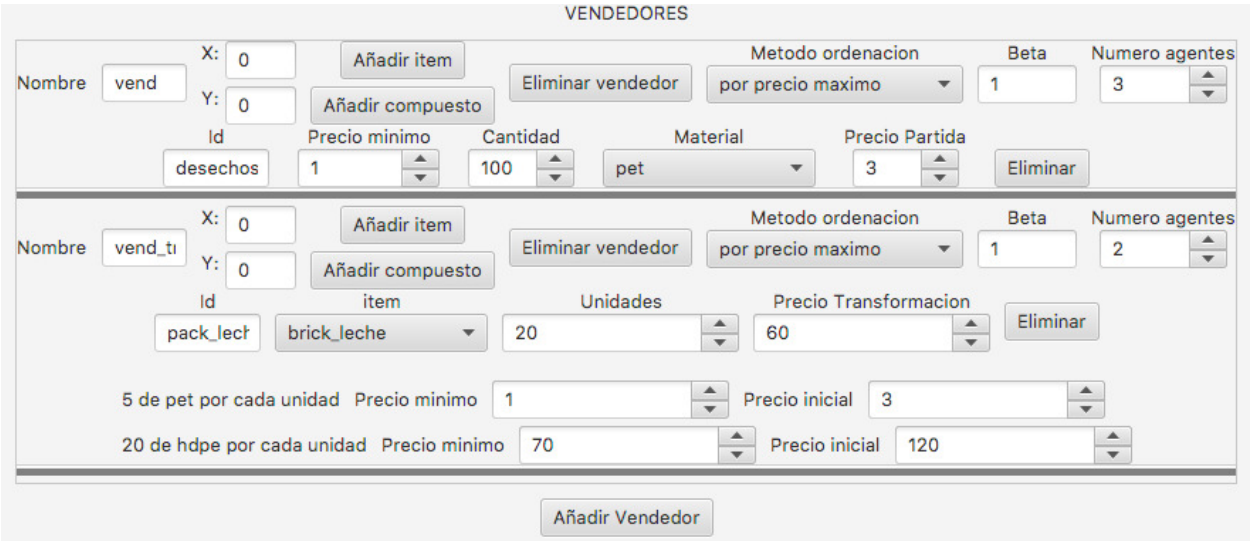


Figura C.9: Ventana principal después de añadir los datos del vendedor

Por último se lanza la ejecución pinchando el botón de *Ejecutar*. Cuando el sistema se haya ejecutado se mostrará una ventana adicional donde irán apareciendo los distintos eventos que se vayan produciendo junto al estado general del mercado.

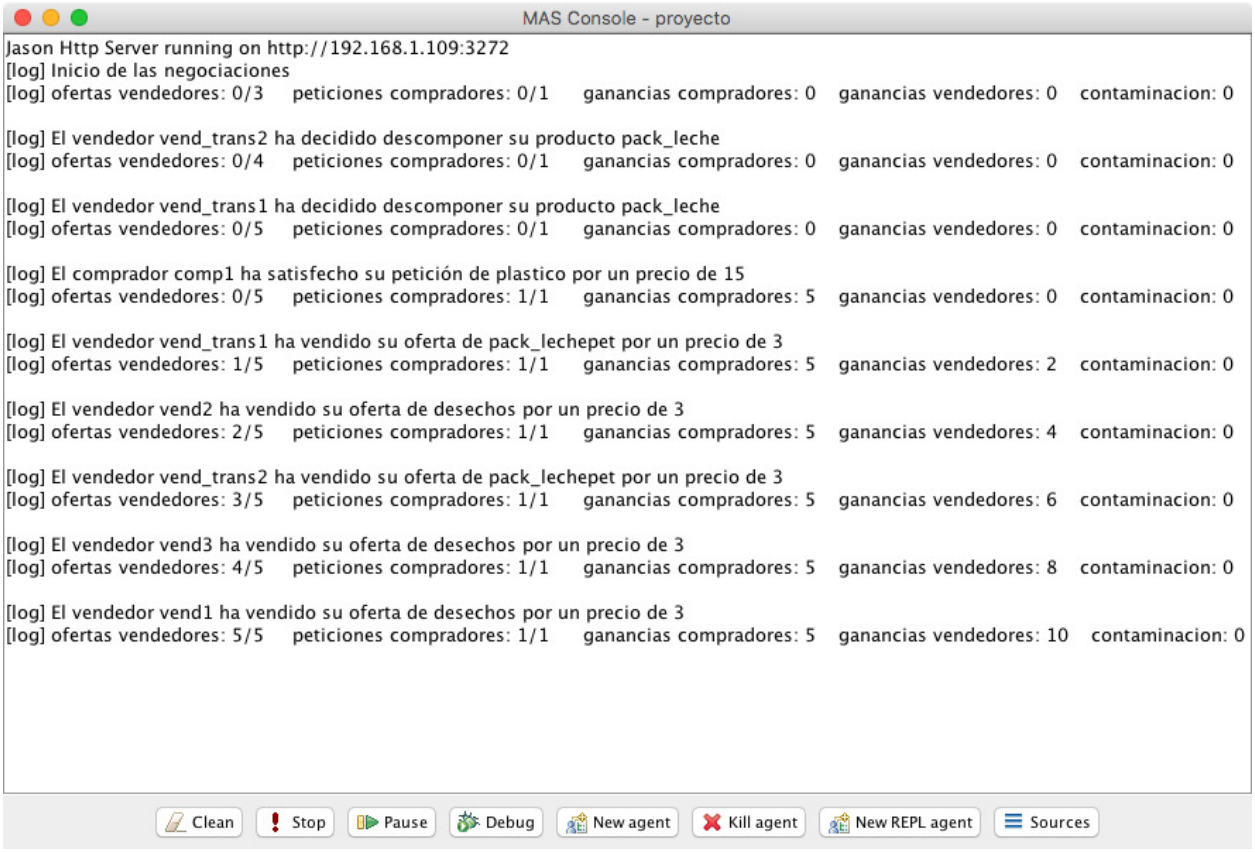


Figura C.10: Salida del programa