Manual de usuario y desarrollador.

(3865) - Programación Orientada a Agentes

Convocatoria de **Julio**

Curso Académico **2018/19**

Prof. **Pablo Campillo Sánchez**

López Rodríguez, Alejandro Ramón

alejandroramon.lopezr@um.es

23953600C

ÍNDICE:

1. - Ejecución y seguimiento del SMA

La clase encargada de lanzar la plataforma y sus diferentes agentes es *ScenarioLauncher.java*. Esta lanza cada agente y le pasa su archivo de configuración para que se pueda inicializar. Para ello debe haber un primer archivo de configuración que establezca los parámetros del escenario y que se pase como argumento al lanzar la aplicación. En mi caso este archivo se encuentra en *configs/scenario.yaml*. En eclipse es tan sencillo como añadir esa ruta como argumento de programa (*Program Arguments*). Además se debe añadir *-Djava.util.logging.config.file=logging.properties* como argumento de la máquina virtual (*VM Arguments*).

Una vez lanzada la aplicación tenemos varias formas de seguir su ejecución:

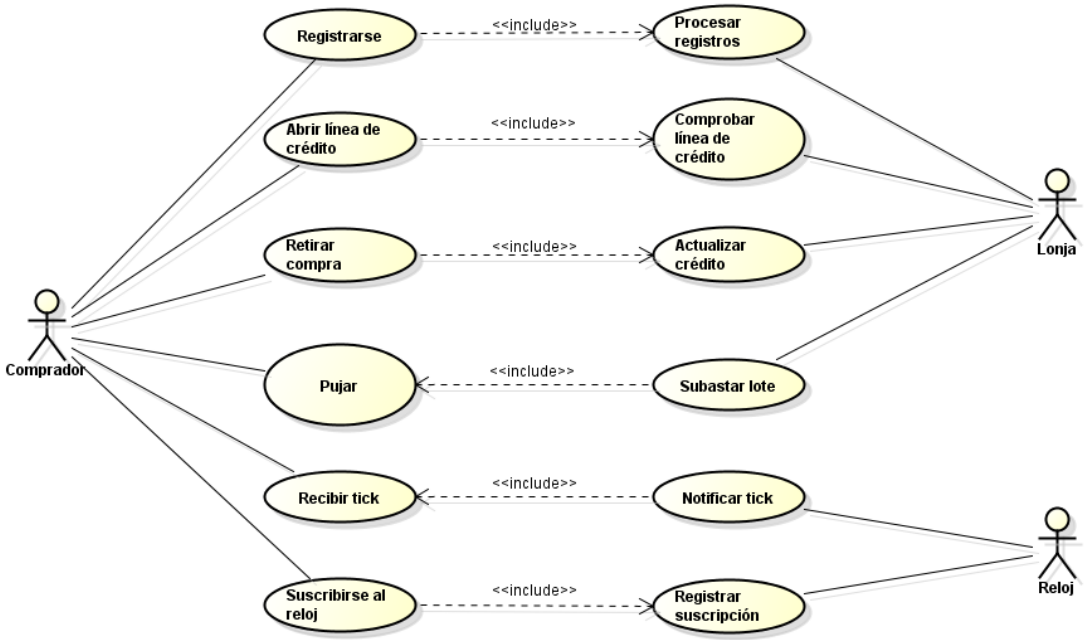
- Terminal: Esta traza salta a la vista cuando ejecutamos la aplicación. Sobre la consola de Eclipse nos irán apareciendo las diferentes transacciones de los agentes. Tiene la desventaja de que es difícil de seguir para la vista, ya que es monocolor y con poca separación, pero aporta un poco más de información que el resto de alternativas ya que en consola se muestran los parámetros de configuración de los diferentes agentes y del escenario.
- Logger Global: La información de las transacciones, los ticks de reloj y otra información es tratada por el logger de cada agente además de mostrarse en pantalla. Cada logger redirige cada registro a los handlers que tiene añadidos. Uno de estos handlers es un archivo común para todos los agentes que permite almacenar toda la información de la simulación en un mismo lugar, en forma de tabla y con diferentes colores. Esta es la principal alternativa para seguir la traza de la simulación de forma ordenada y con toda la información relevante. El archivo en cuestión se puede encontrar en *logs/"nombre_escenario".html*.
- Logger Privado: Cada agente tiene además un handler privado añadido a su logger. De esta forma toda la información que un agente aporta al handler global la aporta también al suyo propio, por lo que para cada agente existe un archivo que almacena sus transacciones. Esto es muy útil cuando buscas información concreta de agentes o de protocolos entre agentes. Estos archivos se encuentran en *logs/"nombre_agente".html*.

En resumen tenemos la información de la consola (vistazo rápido e información de inicialización), el logger global (simulación completa) y el logger privado de cada agente (punto de vista del agente sobre la simulación).

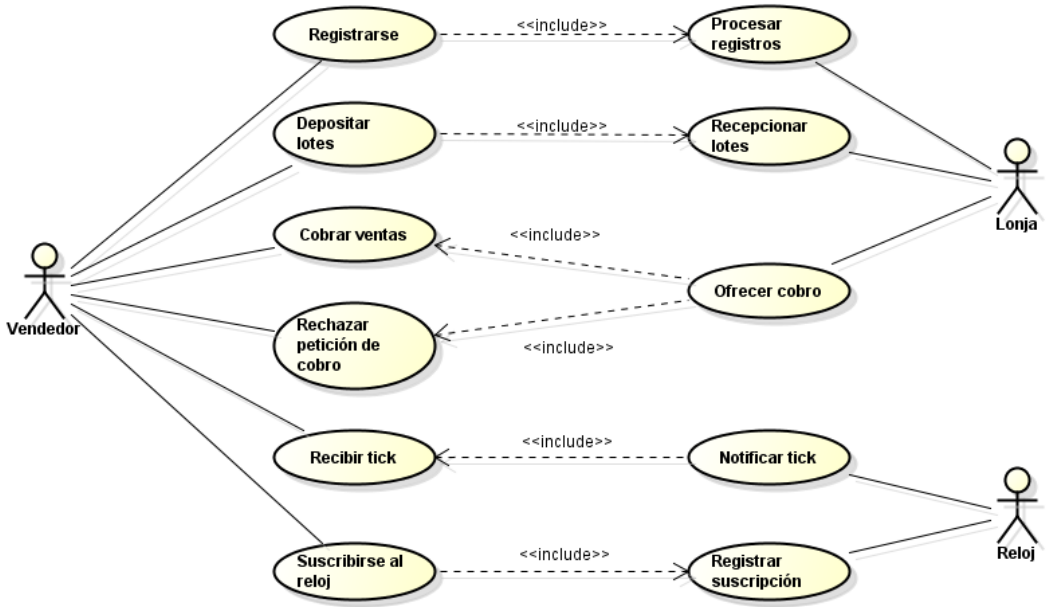
2. - Análisis y diseño del SMA

2.1 - Diagramas de Casos de Uso

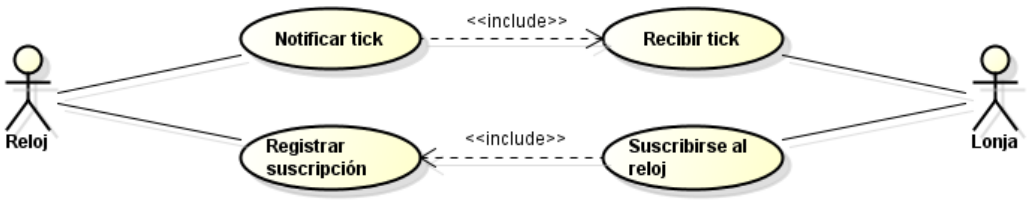
El primer paso para entender la funcionalidad de los agentes es diferenciar sus casos de uso.



En los DCU podemos ver, además de las responsabilidades del agente, con que otros agentes interacciona. En este caso el agente comprador está fuertemente relacionado con el agente lonja y el agente reloj.



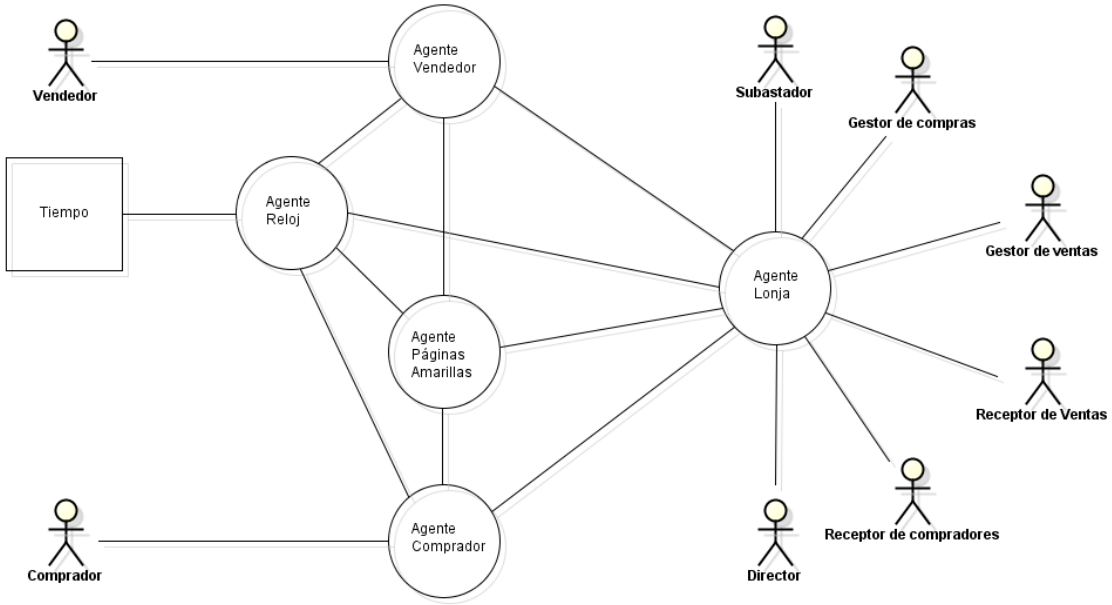
Se puede ver que el DCU del vendedor es muy similar al del comprador.



Por último, el agente reloj interacciona con la lonja del mismo modo que con los otros agentes. Hay que aclarar que el reloj tiene más casos de uso pero que ya se muestran en los DCU del comprador y del vendedor. Del mismo modo no voy a exponer un DCU para la lonja porque sus casos de uso están expuestos ya en el resto de DCUs.

2.2 - Diagrama de Agentes

Una vez hemos visto las responsabilidades que los agentes tienen en común podemos hacer un diagrama donde se represente que agentes interaccionan entre sí.



Vemos que prácticamente todos los agentes interaccionan con todos. El reloj está relacionado con todos al tener que notificarles del tiempo, el DF también, puesto que los agentes acuden a él en busca de servicios, y la lonja por ser el agente principal que coordina el SMA. En cambio el comprador y el vendedor no interaccionan entre ellos, a pesar de que uno compra lotes que el otro vende, esto se produce por medio de la lonja, que actúa de intermediario. Además están representados los roles de los agentes en forma de persona y los recursos usados por los agentes en forma de cuadrado.

2.3 - Responsabilidades y protocolos de interacción de los agentes

En la práctica se definen los siguientes protocolos:

Nº	Protocolo
1	protocolo-registro-vendedor
2	protocolo-deposito
3	protocolo-cobro
4	protocolo-registro-comprador
5	protocolo-apertura-credito
6	protocolo-retirada-compras
7	protocolo-subasta
8	protocolo-notificación (TickerBehaviour)

Si nos fijamos en el protocolo 4, el nombre que se le ha dado al registro del comprador es protocolo-registro-comprador. En el enunciado este aparece como protocolo-admisión-comprador pero he elegido cambiarle el nombre por uno más fácil de comprender. Además el último protocolo no aparece en el enunciado, se trata del servicio de notificación del reloj (que es un *TickerBehaviour* y que ya estaba implementado en el esqueleto de la práctica), lo he añadido para relacionar las responsabilidades que se muestran a continuación, que han sido extraídas de los casos de uso.

Agente	Responsabilidades
Vendedor	1. - Suscribirse al reloj (8) 2. - Registrarse en la lonja (1) 3. - Depositar capturas a vender (2) 4. - Aceptar cobro por parte de la lonja (3) 5. - Rechazar cobro por parte de la lonja (3) 6. - Recibir tick (8)
Comprador	7. - Suscribirse al reloj (8) 8. - Registrarse en la lonja (4) 9. - Abrir línea de crédito (5) 10. - Retirar compra (6) 11. - Pujar (7) 12. - Recibir tick (8)
Lonja	13. - Suscribirse al reloj (8) 14. - Procesar registros (1) 15. - Recepcionar lotes (2) 16. - Ofrecer cobro a vendedor (3) 17. - Abrir línea de crédito a comprador (5) 18. - Actualizar línea de crédito de comprador (6) 19. - Subastar lote (7) 20. - Recibir tick (8)
Reloj	21. - Registrar suscripción (8) 22. - Notificar tick (8)

El número representado entre paréntesis después de cada responsabilidad corresponde al protocolo al que hace referencia. Estos protocolos por debajo están implementados por los protocolos de interacción que estandariza FIPA, los cuales se exponen en las siguientes tablas:

Interacción	Resp.	Protocolo	Rol	Agente	Cuando	Plantilla
Suscribirse al reloj.	1	FIPA Subscribe	I	Agente Vendedor	Startup()	Perf = SUBSCRIBE
Registro de vendedor.	2	FIPA Request	I	Agente Vendedor	El vendedor llega a la lonja.	Perf = REQUEST
Deposito de capturas a vender.	3	FIPA Request	I	Agente Vendedor	Cada vez que llegan lotes de un vendedor a la lonja.	Perf = REQUEST
Aceptar cobro.	4	FIPA Request	R	Agente Vendedor	La lonja propone un cobro con valor superior a 500 o cuando el vendedor no tiene lotes pendientes de entregar ese día.	Perf = AGREE
Rechazar cobro.	5	FIPA Request	R	Agente Vendedor	El vendedor tiene lotes pendientes de entregar ese día y el cobro tiene un valor inferior a 500.	Perf = REFUSE

El agente vendedor usará un protocolo FIPA-Subscribe para recibir las notificaciones del reloj y protocolos FIPA-Request como Initiator o como Responder para el resto de sus interacciones con la lonja.

Interacción	Resp.	Protocolo	Rol	Agente	Cuando	Plantilla
Suscribirse al reloj.	7	FIPA Subscribe	I	Agente Comprador	Startup()	Perf = SUBSCRIBE
Registro de comprador.	8	FIPA Request	I	Agente Comprador	El comprador llega a la lonja.	Perf = REQUEST
Abrir línea de crédito.	9	FIPA Request	I	Agente Comprador	El comprador se ha registrado en la lonja.	Perf = REQUEST
Retirada de compra.	10	FIPA Request	I	Agente Comprador	Cada medio día si el comprador tiene lotes comprados sin retirar.	Perf = REQUEST
Pujar por lote.	11	FIPA Contract Net	R	Agente Comprador	El subastador inicia una subasta.	Perf = PROPOSE si el lote le interesa al comprador o REFUSE en caso contrario.

En el caso del agente comprador encontramos una diferencia respecto al vendedor. El comprador usará un protocolo FIPA-ContractNet (IteratedContractNet en realidad) para recibir ofertas de subastar y pujar por ellas.

Interacción	Resp.	Protocolo	Rol	Agente	Cuando	Plantilla
Suscribirse al reloj.	13	FIPA Subscribe	I	Agente Lonja	Startup()	Perf = SUBSCRIBE
Procesar registros.	14	FIPA Request	R	Agente Lonja (RRV/RRC)	Un comprador o un vendedor solicita registrarse y no lo estaba ya.	Perf = AGREE
Recepcionar lotes.	15	FIPA Request	R	Agente Lonja (RRV)	Un vendedor pretende depositar sus lotes y está registrado en la lonja.	Perf = AGREE
Ofrecer cobro a vendedor.	16	FIPA Request	I	Agente Lonja (RGV)	Cada medio día si el vendedor tiene lotes sin cobrar.	Perf = REQUEST
Abrir línea de crédito a comprador	17	FIPA Request	R	Agente Lonja (RGC)	Un comprador pretende abrir su línea de crédito y está registrado en la lonja.	Perf = AGREE
Actualizar línea de crédito de comprador.	18	FIPA Request	R	Agente Lonja (RGC)	Un comprador desea retirar su compra.	Perf = AGREE
Subastar lote.	19	FIPA Contract Net	I	Agente Lonja (RS)	Se recibe un lote en la lonja.	Perf = CFP

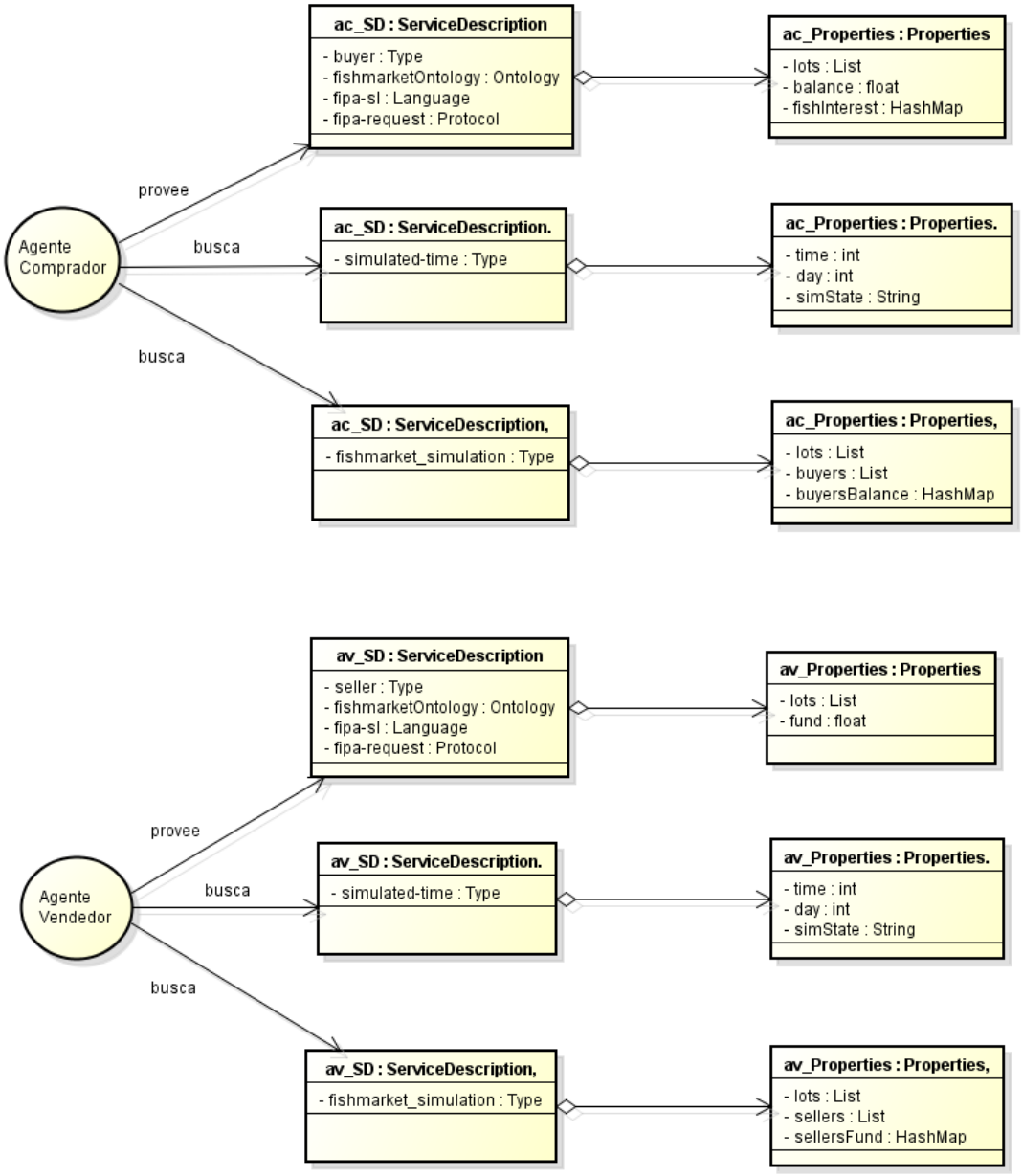
En la tabla del agente lonja se muestran, además, los roles implicados en la interacción.

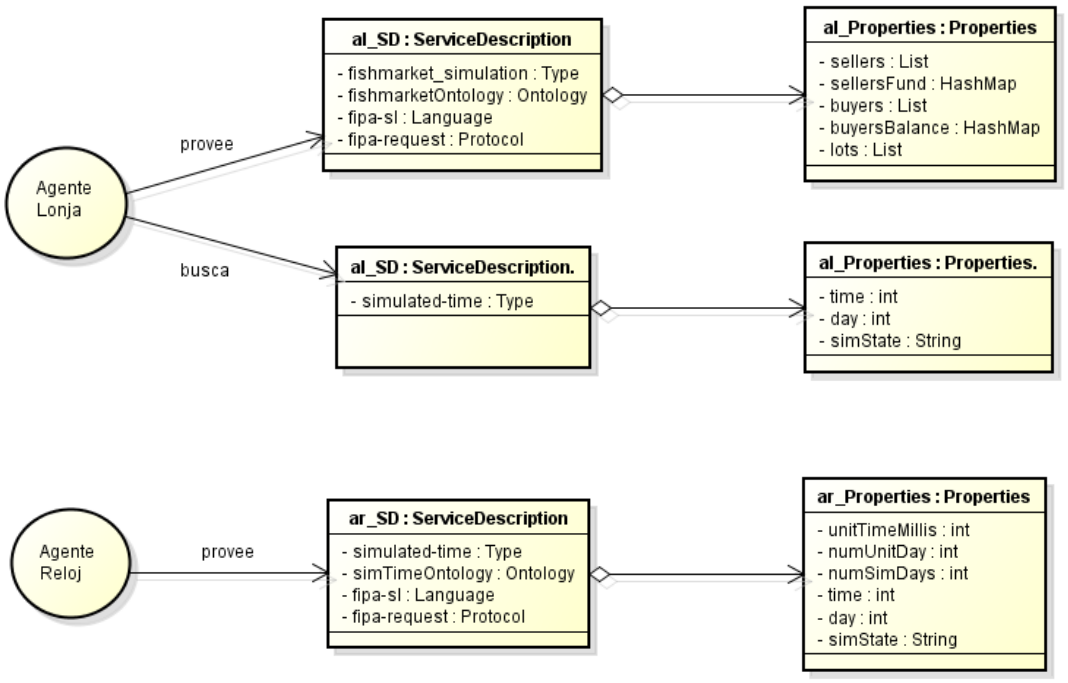
Interacción	Resp.	Protocolo	Rol	Agente	Cuando	Plantilla
Registrar suscripción.	21	FIPA Subscribe	R	Agente Reloj	Un agente se quiere suscribir al reloj.	Perf = AGREE
Notificar tick de reloj.	22	FIPA Subscribe	R	Agente Reloj	Ha pasado un tick de reloj.	Perf = INFORM

Por último tenemos las interacciones del reloj, que se limitan a utilizar el protocolo FIPA-Subscribe en el rol de Responder para notificar a los agentes suscritos el estado de la simulación.

2.4 - Uso de las Páginas Amarillas (Directory Facilitator)

Los agentes deben registrar sus servicios en el DF para que otros puedan encontrarlos.

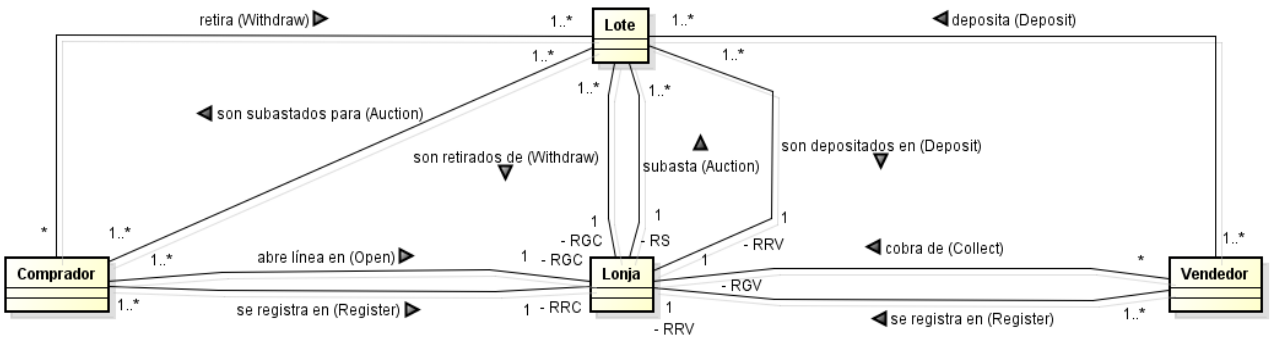




Podemos ver que todos los agentes buscan los servicios del reloj para estar al tanto del tiempo de la simulación. Tanto el comprador como el vendedor buscarán a la lonja para la primera interacción: el registro. En ese momento la lonja almacenará en su base de datos al agente que realizó la petición, es por esto que la lonja nunca tendrá que buscar en el DF a estos agentes, ya que nunca empieza ella la conversación.

2.5 - Modelo de la Ontología

Los agentes se relacionan entre ellos mediante acciones y conceptos que están descritos en la ontología de la aplicación, la cual voy a describir a continuación:



En el diagrama se encuentran todos los agentes menos el reloj, que no necesita una ontología específica para comunicarse con los demás agentes, simplemente pasa el tiempo de la simulación a sus suscriptores. Además de los tres agentes representados como clases en el diagrama tenemos una clase “Lote”, que representa un concepto de la ontología. El concepto tiene sus propios atributos y es usado en algunas acciones. El

resto de relaciones entre las clases del diagrama representan las acciones posibles. Cada una representa a un protocolo de la práctica y serán usadas en dichos protocolos para dar al mensaje que se manda el significado que le corresponde. Para la implementación de la práctica son necesarias 6 acciones: Registrar, Depositar, Cobrar, Retirar compras, Subastar y Abrir crédito.

3. - Implementación del SMA

3.1 - Esqueleto de la práctica y modificaciones

Para la implementación del sistema he usado el esqueleto facilitado aún que he realizado algunas modificaciones que creo relevante comentar.

Clase POAAgent: El método *setup()* se ha ampliado para añadir el handler privado del agente al logger. De igual forma en el *takeDown()* se cierra este handler privado.

Clase ClockAgent: El agente reloj debe añadir el handler global a su logger como hacen el resto de agentes. Además debe cerrarlo en su método *takeDown()* ya que nadie más lo va a usar.

Clase ScenarioLauncher: No veía necesario que el logger se crease en el launcher ya que quienes lo necesitan son los agentes así que el escenario lo que hace en esta nueva versión es crear el handler global y pasarlo por parámetro al resto de agentes. Ellos crearán su logger y añadirán tanto el handler privado como el recibido del launcher. Por lo tanto se suprime el método *initLogging(String fileName)* que ya no es necesario.

Clase AgentLoggerManager: Ahora cuando se crea el logger se llama a LogManager y se añade, puesto que esto ya no ocurre en el launcher y necesitamos añadir loggers para cada agente. Además en el método *info(String behaviour, String msg)* se ha añadido una conversión de cadena estándar para la salida por consola (con nueva línea *\n*) a cadena con formato html para los logs (con nuevas línea *
*). Por último se ha modificado el método *close()* para que solo cierre el handler indicado. Esto es útil para que los agentes cierren solo su handler privado, ya que cerrar el global supone que el resto de agentes no puedan escribir en él, incluso si no han terminado de escribir. El handler global lo cerrará el reloj por ser el último agente que lo usa.

Clase AgentLoggingHTMLFormatter: En esta nueva versión se recibe en el constructor el nombre del agente o del escenario para indicarlo en el título del log.

Clase LogAgentFilter: Se ha suprimido esta clase al no haberle encontrado una utilidad necesaria para el correcto funcionamiento o seguimiento de la práctica.

3.2 - Lenguaje y Ontología

Como ya se ha visto en el modelo de la ontología, esta es el conjunto de conceptos y acciones que relacionan a los agentes y que crean un contexto uniforme sobre el que todos pueden entenderse. En mi caso la ontología de la lonja está definida en el paquete *fishmarketOntology* cuya clase principal es *FishmarketOntology.java*. En ella se definen estas acciones, conceptos y sus atributos, y se añaden todos a la única instancia de la ontología, ya que sigue un patrón Singleton. Esta forma de hacer la ontología ha sido sacada de los ejemplos de la propia página de Jade y es muy importante no confundirla con la clase *SimTimeOntology* del paquete *agents.clock*. Esta clase en realidad no es la ontología de la lonja o de la simulación vista de forma global, si no la estructura de datos creada para albergar todos los datos del tiempo de simulación, es decir, es más bien una ontología del tiempo de simulación. El uso de la palabra ontología en el nombre puede ser confuso ya que, de hecho, no es una clase que herede de *Ontology.java*, pero he mantenido el nombre que tenía esa clase en el esqueleto de la práctica. Por lo tanto los agentes deberán crear objetos pertenecientes a la ontología (acciones o conceptos), enviarlos indicando en el mensaje que la ontología es la de la lonja (*setOntology()*) y el receptor deberá extraer el contenido y convertirlo otra vez en el objeto de la ontología conveniente. Para lo que se puede apreciar que no se hace ningún uso de *SimTimeOntology.java*. Además en el mensaje hay que indicar que el lenguaje usado es SL por ser el códec que he elegido usar al ser mucho más legible para el humano que Leap, lo cual facilita la depuración.

3.3 - Conversaciones y protocolos

Cada agente tiene una serie de comportamientos que corresponden a los protocolos de la práctica. Estos comportamientos en realidad no hacen otra cosa que comprobar cuando debe actuar el comportamiento, lanzar un protocolo de interacción FIPA y sobrescribir sus métodos para tratar los mensajes de respuesta. Describiré en detalle los pasos de la conversación para cada protocolo.

protocolo-registro-vendedor:

Paso Nº	Agente	Comportamiento	Acción
1	Vendedor	SellerRegisterB	Añadir un comportamiento AchieveREInitiator para envíar un Request a la lonja con contenido Register seller.
2	Lonja	ReceiverAnd ManagerB	Añadir un AchieveRE Responder para recibir el Request.
3a	Lonja	ReceiverAnd ManagerB	Si el vendedor no estaba registrado ya se envía un AGREE.
3b	Lonja	ReceiverAnd ManagerB	Si ya estaba registrado se devuelve un REFUSE.
4a	Lonja	ReceiverAnd ManagerB	Si se envió un AGREE entonces se procede a registrar al vendedor y se devuelve un INFORM.
5a	Vendedor	SellerRegisterB	Al recibir el INFORM añade los comportamientos DepositInitiatorB y CollectResponderB.

protocolo-registro-comprador:

Paso Nº	Agente	Comportamiento	Acción
1	Comprador	BuyerRegisterB	Añadir un comportamiento AchieveREInitiator para envíar un Request a la lonja con contenido Register buyer.
2	Lonja	ReceiverAnd ManagerB	Añadir un AchieveRE Responder para recibir el Request.
3a	Lonja	ReceiverAnd ManagerB	Si el comprador no estaba registrado ya se envía un AGREE.
3b	Lonja	ReceiverAnd ManagerB	Si ya estaba registrado se devuelve un REFUSE.
4a	Lonja	ReceiverAnd ManagerB	Si se envió un AGREE entonces se procede a registrar al comprador y se devuelve un INFORM.
5a	Comprador	BuyerRegisterB	Al recibir el INFORM añade el comportamiento OpenBalanceB.

protocolo-deposito:

Paso Nº	Agente	Comportamiento	Acción
1	Vendedor	DepositInitiatorB	Busca si han llegado nuevos lotes del vendedor comprobando el momento de llegada de cada lote y el tiempo actual.
2a	Vendedor	DepositInitiatorB	La lista de nuevos lotes está vacía, el comportamiento acaba y se reinicia.
2b	Vendedor	DepositInitiatorB	La lista de nuevos lotes no está vacía, se añade un comportamiento AchieveREInitiator para envíar un Request a la lonja con contenido Deposit <lotes>.
3ba	Lonja	ReceiverAnd ManagerB	Si el vendedor está registrado se envía un AGREE.
3bb	Lonja	ReceiverAnd ManagerB	Si no está registrado se devuelve un REFUSE.
4ba	Lonja	ReceiverAnd ManagerB	Si se envió un AGREE entonces se procede a depositar los lotes y se devuelve un INFORM.
N	Vendedor	DepositInitiatorB	Cuando al vendedor no le quedan lotes por llegar el comportamiento acaba.

protocolo-apertura-credito:

Paso Nº	Agente	Comportamiento	Acción
1	Comprador	OpenBalanceB	Añadir un comportamiento AchieveREInitiator para envíar un Request a la lonja con contenido Open <saldo>.
2	Lonja	ReceiverAnd ManagerB	Añadir un AchieveRE Responder para recibir el Request.
3a	Lonja	ReceiverAnd ManagerB	Si el comprador está registrado y tiene un saldo mayor que 0 se envía un AGREE.
3b	Lonja	ReceiverAnd ManagerB	En otro caso se devuelve un REFUSE.
4a	Lonja	ReceiverAnd ManagerB	Si se envió un AGREE entonces se procede a abrir la línea de crédito del comprador y se devuelve un INFORM.
5a	Comprador	BuyerRegisterB	Al recibir el INFORM añade los comportamientos BidderB y WithdrawInitiatorB.

protocolo-subasta:

Paso Nº	Agente	Comportamiento	Acción
1	Lonja	AuctioneerB	Si hay algún lote sin subastar, la cinta está libre y ha pasado el periodo de latencia, se busca un lote a subastar.
2	Lonja	AuctioneerB	Añadir un ContractNetInitiator para enviar un CFP a cada comprador con el contenido Auction <lote> <precio>.
3	Comprador	BidderB	Añadir un ContractNetResponder para recibir los CFP.
3a	Comprador	BidderB	Si al comprador le interesa el lote devuelve un PROPOSE.
3b	Comprador	BidderB	Si no le interesa devuelve un REFUSE.
4aa	Lonja	AuctioneerB	Establecer al primer PROPOSE con dinero para pagar el lote como ganador y devolverle un ACCEPT_PROPOSAL.
4ab	Lonja	AuctioneerB	A los que no ganaron devolverles un REJECT_PROPOSAL.
4ba	Lonja	AuctioneerB	Si nadie envía un PROPOSAL el lote baja de precio. Si no se alcanza el precio de reserva se vuelve al paso 2.
4bb	Lonja	AuctioneerB	Si se alcanza el precio de reserva la subasta acaba y el lote se desecha.
5aa	Comprador	BidderB	Reducir el saldo local del comprador (sirve para saber el saldo real que tiene en la lonja) y devolver un INFORM.
6aa	Lonja	AuctioneerB	Establecer el comprador del lote y decrementar su saldo.

protocolo-retirada-compras:

Paso Nº	Agente	Comportamiento	Acción
1	Comprador	WithdrawInitiatorB	Si hay lotes sin retirar y ha pasado el tiempo necesario (medio día o que sea el último tick de simulación) se añade un comportamiento AchieveREInitiator para envíar un Request a la lonja con contenido Withdraw.
2	Lonja	ReceiverAnd ManagerB	Añadir un AchieveRE Responder para recibir el Request.
3a	Lonja	ReceiverAnd ManagerB	Si el comprador está registrado y tiene un algún lote sin retirar se envía un AGREE.
3b	Lonja	ReceiverAnd ManagerB	En otro caso se devuelve un REFUSE.
4a	Lonja	ReceiverAnd ManagerB	Si se envió un AGREE entonces se devuelve un INFORM cuyo contendido es Withdraw <lotes>.
5a	Comprador	WithdrawInitiatorB	Extraer los lotes.

protocolo-cobro:

Paso Nº	Agente	Comportamiento	Acción
1	Lonja	CollectInitiatorB	Cada medio día o en el último tick de la simulación y para cada vendedor con fondos se crea un Request con contenido Collect <fondos> y se envía.
2	Vendedor	CollectResponderB	Añadir un AchieveRE Responder para recibir el Request.
3a	Vendedor	CollectResponderB	Si el vendedor tiene lotes pendientes de llegar en el día actual y los fondos son menores de 500 se devuelve un REFUSE.
3b	Vendedor	CollectResponderB	En otro caso de devuelve un AGREE.
4a	Lonja	CollectInitiatorB	Se confirma la respuesta del vendedor y se vuelve al paso 1.
4b	Vendedor	CollectResponderB	Si se envió un AGREE entonces se incrementan los fondos del vendedor y se devuelve un INFORM.
5b	Lonja	CollectInitiatorB	Al recibir el INFORM retira los fondos del vendedor de la lonja, confirma la respuesta del vendedor y vuelve al paso 1.

3.4 - Otros comentarios

Simplemente justificar el uso de ciertos parámetros de la lonja. Según el enunciado estos parámetros son opcionales si se justifica porque se desaconseja su uso. En mi proyecto tanto la ventana de oportunidad para la puja como el periodo de latencia son totalmente funcionales, ahora bien acojámonos a la definición del enunciado:

Ventana de oportunidad para la puja: tiempo entre dos precios sucesivos emitidos por el rol RS. En la práctica yo he implementado esto usando las ranuras de los mensajes llamadas *replyBy* que permiten establecer un timeout dentro del cual se debe recibir el mensaje. En este caso *replyBy* se establece en el momento de crear el mensaje usando el tiempo actual y sumándole la ventana de oportunidad, el resultado es el tiempo límite en el que puede llegar el mensaje. Esto no es realmente fiel a la descripción del enunciado ya que la ventana de oportunidad no es el tiempo que pasa entre dos precios si no el tiempo **máximo**. Si los agentes contestan en un milisegundo no me parece lógico retenerlos X milisegundos hasta llegar a la ventana de oportunidad, de hecho creo que bastante ineficiente. Así que, en resumen, sí uso la ventana de oportunidad pero en mi práctica se trata de un timeout.

Periodo de latencia: tiempo permitido entre dos rondas sucesivas de venta. Entiendo este parámetro como una forma de retardar las subastas de una misma cinta. Junto con el parámetro anterior podrían hacer una simulación más realista donde los agentes no contestan instantáneamente y se puede seguir la conversación en tiempo de ejecución. A mí no me gusta el uso de este parámetro en particular porque no lo veo acorde al resto de la simulación. En una escala de simulación en la que un día pueden ser perfectamente 10 segundos no parece muy lógico que entre dos subastas pasen 2’5 segundos, por ejemplo. Sería como si en un día natural pasasen 6 horas entre cada subasta a pesar de tener los lotes desde el principio. Además no he leído nada de retardos en el resto de protocolos así que habría protocolos que se ejecutarían de forma instantánea mientras que las subastas tendrían retardos. Por este motivo a mí me gusta más dejar la latencia a 0 y dejar la ejecución de forma natural. Claro está para seguir la simulación es mejor acogerse al algún log ya que por pantalla sale muy rápido la información. A pesar de todo la latencia está implementada y es funcional (al final otro usuario puede tener gustos diferentes a los míos). Básicamente se calcula cuando se puede hacer la próxima subasta (al igual que antes sumando el tiempo actual y el de latencia) y no se deja que empiece una hasta llegar a la marca de tiempo establecida.

4. - Ejemplos de simulación

4.1 - Escenario estándar

Este escenario se puede probar pasándole al launcher el archivo *configs/standardScenario.yaml*. Es una simulación larga y completa, por lo que no voy a detenerme en explicar cada protocolo, solo los aspectos más relevantes.

En este escenario tenemos dos vendedores y tres compradores, durante los tres días de simulación los vendedores depositarán siete lotes.

Los dos primeros lotes llegan en el 0:2.

Sun Jul 07 18:14:56 CEST 2019	Lonja	PROTOCOLO-DEPOSITO	Depositando los lotes: - Lot-0 [day= 0, time=2, kind=sardina, kg=15.0, reservePrice=60.0, startingPrice=120.0, finalPrice=0.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime=null, seller=Vendedor1, buyer=null, auctioned=false] - Lot-1 [day= 0, time=2, kind=pulpo, kg=10.0, reservePrice=140.0, startingPrice=300.0, finalPrice=0.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime=null, seller=Vendedor1, buyer=null, auctioned=false] del Vendedor1.
Sun Jul 07 18:14:56 CEST 2019	Lonja	SUBASTA	Cinta 1: Se abre subasta de lote de sardina por 120.0

La subasta se abre y los compradores pujan por ella.

Sun Jul 07 18:14:56 CEST 2019	Comprador1	SUBASTA	A Comprador1 le interesa el lote de sardina (120.0) [interes = 70.06% / 700.6]
Sun Jul 07 18:14:56 CEST 2019	Comprador3	SUBASTA	A Comprador3 NO le interesa el lote de sardina (120.0) [interes = 26.82% / 53.64]
Sun Jul 07 18:14:56 CEST 2019	Comprador2	SUBASTA	A Comprador2 le interesa el lote de sardina (120.0) [interes = 86.27% / 431.35]
Sun Jul 07 18:14:56 CEST 2019	Comprador3	SUBASTA	A Comprador3 NO le interesa el lote de pulpo (300.0) ya que no tiene fondos suficientes para pagarlo (200.0)
Sun Jul 07 18:14:56 CEST 2019	Lonja	SUBASTA	Cinta 1: Lote de sardina vendido a Comprador1 por 120.0

En este caso al comprador3 no le interesaba el lote porque cree que su precio adecuado es de 53’64. Al comprador1 y al 2 si les interesa pero el 1 pujó primero por lo que obtiene el lote.

Sun Jul 07 18:14:56 CEST 2019	Comprador2	SUBASTA	A Comprador2 NO le interesa el lote de pulpo (300.0) [interes = 15.35% / 76.75]
Sun Jul 07 18:14:56 CEST 2019	Comprador1	SUBASTA	A Comprador1 NO le interesa el lote de pulpo (300.0) [interes = 28.05% / 246.84]
Sun Jul 07 18:14:56 CEST 2019	Lonja	SUBASTA	Cinta 2: Lote de pulpo en subasta por 225.0
Sun Jul 07 18:14:56 CEST 2019	Comprador3	SUBASTA	A Comprador3 NO le interesa el lote de pulpo (225.0) ya que no tiene fondos suficientes para pagarlo (200.0)
Sun Jul 07 18:14:56 CEST 2019	Comprador1	SUBASTA	A Comprador1 le interesa el lote de pulpo (225.0) [interes = 28.05% / 246.84]
Sun Jul 07 18:14:56 CEST 2019	Comprador2	SUBASTA	A Comprador2 NO le interesa el lote de pulpo (225.0) [interes = 15.35% / 76.75]
Sun Jul 07 18:14:56 CEST 2019	Lonja	SUBASTA	Cinta 2: Lote de pulpo vendido a Comprador1 por 225.0

En la subasta del segundo lote podemos ver como los tres compradores rechazan la oferta del lote de pulpo por un precio de 300. El comprador1 acepta la segunda ronda y obtiene el lote.

Sun Jul 07 18:14:59 CEST 2019	clock	ClockTickerBehaviour	day=0, time=5
Sun Jul 07 18:14:59 CEST 2019	Vendedor1	PROTOCOLO-COBRO	Fondos de Vendedor1 incrementados en 311.0 (311.0)
Sun Jul 07 18:14:59 CEST 2019	Lonja	PROTOCOLO-COBRO	Fondos de Vendedor1 retirados de la lonja.
Sun Jul 07 18:14:59 CEST 2019	Comprador1	PROTOCOLO-RETIRADA-COMPRAS	Lotes retirados: - Lot-0 [day= 0, time=2, kind=sardina, kg=15.0, reservePrice=60.0, startingPrice=120.0, finalPrice=120.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime={"simState":"RUNNING","day":0,"time":2}, seller=Vendedor1, buyer=Comprador1, auctioned=true] - Lot-1 [day= 0, time=2, kind=pulpo, kg=10.0, reservePrice=140.0, startingPrice=300.0, finalPrice=225.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime={"simState":"RUNNING","day":0,"time":2}, seller=Vendedor1, buyer=Comprador1, auctioned=true] Cartera de Comprador1 actualizada (655.0)

Cuando pasa medio día la lonja realiza una solicitud de cobro al vendedor1 que es el único que ha incrementado sus fondos y este acepta ya que no tiene más lotes por llegar ese día. De igual manera el comprador realiza al medio día su petición de retirada de compras.

Sun Jul 07 18:15:03 CEST 2019	clock	ClockTickerBehaviour	day=0, time=9
Sun Jul 07 18:15:04 CEST 2019	clock	ClockTickerBehaviour	day=1, time=0
Sun Jul 07 18:15:05 CEST 2019	clock	ClockTickerBehaviour	day=1, time=1

Durante el medio día restante no llega ningún lote, por eso al llegar el 1:0 la lonja no realiza ningún cobro (nadie ha obtenido fondos) y ningún comprador retira sus compras (pues no tienen ningún lote nuevo).

Sun Jul 07 18:15:09 CEST 2019	clock	ClockTickerBehaviour	day=1, time=5
Sun Jul 07 18:15:09 CEST 2019	Lonja	PROTOCOLO-DEPOSITO	Depositando los lotes: - Lot-3 [day= 1, time=5, kind=bacalao, kg=20.0, reservePrice=250.0, startingPrice=350.0, finalPrice=0.0, registerTime={"simState":"RUNNING","day":1,"time":5}, saleTime=null, seller=Vendedor1, buyer=null, auctioned=false] del Vendedor1.
Sun Jul 07 18:15:09 CEST 2019	Lonja	SUBASTA	Cinta 0: Se abre subasta de lote de bacalao por 350.0

En el 1:5 llega un nuevo lote. Podemos apreciar que en este instante el vendedor2 no cobra la venta del lote de lubina llegada en el instante 1:3 puesto que tiene un lote pendiente de llegar para el instante 1:9. La política del vendedor intenta ahorrar transacciones cobrando todos los lotes de un día cuando se han vendido todos, a no ser que el cobro sea de 500 o más.

Sun Jul 07 18:15:09 CEST 2019	Comprador3	SUBASTA	A Comprador3 NO le interesa el lote de bacalao (350.0) ya que no tiene fondos suficientes para pagarlo (200.0)
Sun Jul 07 18:15:09 CEST 2019	Comprador2	SUBASTA	A Comprador2 NO le interesa el lote de bacalao (350.0) [interes = 47.58% / 237.9]
Sun Jul 07 18:15:09 CEST 2019	Lonja	SUBASTA	Cinta 0: Lote de bacalao en subasta por 263.0
Sun Jul 07 18:15:09 CEST 2019	Comprador2	SUBASTA	A Comprador2 NO le interesa el lote de bacalao (263.0) [interes = 47.58% / 237.9]
Sun Jul 07 18:15:09 CEST 2019	Comprador1	SUBASTA	A Comprador1 NO le interesa el lote de bacalao (263.0) ya que no tiene fondos suficientes para pagarlo (205.0)
Sun Jul 07 18:15:09 CEST 2019	Comprador3	SUBASTA	A Comprador3 NO le interesa el lote de bacalao (263.0) ya que no tiene fondos suficientes para pagarlo (200.0)
Sun Jul 07 18:15:09 CEST 2019	Lonja	SUBASTA	Cinta 0: Subasta cerrada, se ha alcanzado el precio de reserva (250.0) Lote de bacalao desechado.

En este caso el lote es desechado ya que a nadie le interesó su precio antes de alcanzar el precio de reserva.

Sun Jul 07 18:15:23 CEST 2019	clock	ClockTickerBehaviour	day=2, time=9
Sun Jul 07 18:15:23 CEST 2019	Lonja	PROTOCOLO-DEPOSITO	Depositando los lotes: - Lot-6 [day= 2, time=9, kind=calamar, kg=15.0, reservePrice=100.0, startingPrice=450.0, finalPrice=0.0, registerTime={"simState":"RUNNING","day":2,"time":9}, saleTime=null, seller=Vendedor1, buyer=null, auctioned=false] del Vendedor1.
Sun Jul 07 18:15:23 CEST 2019	Vendedor1	PROTOCOLO-DEPOSITO	Lotes del Vendedor1 depositados correctamente.
Sun Jul 07 18:15:23 CEST 2019	Lonja	SUBASTA	Cinta 0: Se abre subasta de lote de calamar por 450.0

En el último tick (2:9) llega un lote, se deposita y se subasta (gracias a que no tenemos retardos el lote se subasta a tiempo).

Sun Jul 07 18:15:23 CEST 2019	Lonja	SUBASTA	Cinta 0: Lote de calamar vendido a Comprador2 por 254.0
Sun Jul 07 18:15:23 CEST 2019	Vendedor1	PROTOCOLO-COBRO	Fondos de Vendedor1 incrementados en 229.0 (540.0)
Sun Jul 07 18:15:23 CEST 2019	Lonja	PROTOCOLO-COBRO	Fondos de Vendedor1 retirados de la lonja.
Sun Jul 07 18:15:23 CEST 2019	Comprador2	PROTOCOLO-RETIRADA-COMPRAS	Lotes retirados: - Lot-6 [day= 2, time=9, kind=calamar, kg=15.0, reservePrice=100.0, startingPrice=450.0, finalPrice=254.0, registerTime={"simState":"RUNNING","day":2,"time":9}, saleTime={"simState":"RUNNING","day":2,"time":9}, seller=Vendedor1, buyer=Comprador2, auctioned=true] Cartera de Comprador2 actualizada (246.0)
Sun Jul 07 18:15:24 CEST 2019	clock	ClockTickerBehaviour	Fin de la simulacion.

A pesar de que el lote se subaste en el último instante de la simulación, el comprador consigue retirar el lote obtenido y el vendedor obtiene sus fondos. Esto es posible gracias a la política de ambos protocolos, que tratan de forma especial el último tick. Mientras que en el resto de la simulación las peticiones se realizan una cada medio día, en el último tick se permite que las solicitudes se realicen en bucle durante todo el tick (1 segundo en este caso). Esto permite que en cuanto se cumpla una de las condiciones para realizar el protocolo, este se encuentre preparado para realizarse.

4.2 - Escenario con latencia

Este escenario se puede probar pasándole al launcher el archivo *configs/latencyScenario.yaml*. Cuenta con los tres mismos compradores del anterior y un vendedor que deposita simultáneamente cuatro lotes. Con esto se consigue que una cinta tenga que subastar dos lotes y se vea el tiempo de latencia entre las subastas (en este caso cuatro segundos). Basta con usar un día de simulación para ver esto.

Mon Jul 08 11:37:08 CEST 2019	Lonja	SUBASTA	Cinta 2: Lote de sardina vendido a Comprador1 por 120.0
Mon Jul 08 11:37:08 CEST 2019	Comprador1	SUBASTA	A Comprador1 le interesa el lote de pulpo (300.0) [interes = 92.63% / 815.14]
Mon Jul 08 11:37:08 CEST 2019	Comprador1	SUBASTA	A Comprador1 NO le interesa el lote de bacalao (350.0) [interes = 49.69% / 288.2]
Mon Jul 08 11:37:08 CEST 2019	Lonja	SUBASTA	Cinta 0: Lote de pulpo vendido a Comprador1 por 300.0
Mon Jul 08 11:37:08 CEST 2019	Lonja	SUBASTA	Cinta 1: Lote de bacalao en subasta por 263.0

En esta primera captura vemos que las tres cintas se ocupan y que la primera en acabar es la cinta 2 (11:37:08 AM), por lo que será la que cogerá el cuarto lote para subastar.

Mon Jul 08 11:37:12 CEST 2019	Lonja	SUBASTA	Cinta 2: Se abre subasta de lote de calamar por 450.0

La subasta del cuarto lote se produce a las 11:37:12 AM, es decir, cuatro segundos después de acabar la primera subasta, justo el tiempo establecido como latencia.

4.3 - Escenario de lonja con una sola cinta

Este escenario se puede probar pasándole al launcher el archivo *configs/oneDayScenario.yaml*. En este ejemplo tenemos la misma configuración que en el escenario estándar pero con una sola cinta para las subastas. El proceso sigue el mismo ritmo ya que las subastas no tienen retardos y, aunque solo haya una cinta, esta es capaz de despachar todos los lotes sin problemas. La principal diferencia es que los compradores solo contestan a una subasta ya que no hay ofertas en paralelo.

4.4 - Escenario con mucha demanda/poca oferta

Este escenario se puede probar pasándole al launcher el archivo *configs/muchDemmandScenario.yaml*. Para este ejemplo tenemos un vendedor que deposita cuatro lotes y cinco compradores con una cartera inicial de 1000. Aunque el interés es una probabilidad es bastante común que los compradores quieran el lote, ya que tienen mucho dinero respecto al precio de los lotes. Esta situación de mucha demanda y poca oferta produce que todos los lotes se consigan vender y que algunos compradores no consigan comprar nada.

Mon Jul 08 13:13:37 CEST 2019	Comprador3	takeDown	- Lotes adquiridos: - Lot-0 [day= 0, time=2, kind=sardina, kg=15.0, reservePrice=60.0, startingPrice=120.0, finalPrice=120.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime={"simState":"RUNNING","day":0,"time":2}, seller=Vendedor1, buyer=Comprador3, auctioned=true]. - Cartera: 880.0 Comprador3 finalizando...
Mon Jul 08 13:13:37 CEST 2019	Lonja	takeDown	- Ingresos de Lonja: 110.0 - Lonja finalizando...
Mon Jul 08 13:13:37 CEST 2019	Comprador5	takeDown	- Lotes adquiridos: - Lot-1 [day= 0, time=2, kind=pulpo, kg=10.0, reservePrice=140.0, startingPrice=300.0, finalPrice=300.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime={"simState":"RUNNING","day":0,"time":2}, seller=Vendedor1, buyer=Comprador5, auctioned=true] - Lot-2 [day= 1, time=5, kind=bacalao, kg=20.0, reservePrice=250.0, startingPrice=350.0, finalPrice=350.0, registerTime={"simState":"RUNNING","day":1,"time":5}, saleTime={"simState":"RUNNING","day":1,"time":5}, seller=Vendedor1, buyer=Comprador5, auctioned=true]. - Cartera: 350.0 Comprador5 finalizando...

Mon Jul 08 13:13:37 CEST 2019	Comprador4	takeDown	- Lotes adquiridos: Ninguno. - Cartera: 1000.0 Comprador4 finalizando...
Mon Jul 08 13:13:37 CEST 2019	Vendedor1	takeDown	- Ingresos de Vendedor1: 998.0 - Vendedor1 finalizando...
Mon Jul 08 13:13:37 CEST 2019	Comprador2	takeDown	- Lotes adquiridos: - Lot-3 [day= 2, time=9, kind=calamar, kg=15.0, reservePrice=100.0, startingPrice=450.0, finalPrice=338.0, registerTime={"simState":"RUNNING","day":2,"time":9}, saleTime={"simState":"RUNNING","day":2,"time":9}, seller=Vendedor1, buyer=Comprador2, auctioned=true]. - Cartera: 662.0 Comprador2 finalizando...
Mon Jul 08 13:13:37 CEST 2019	Comprador1	takeDown	- Lotes adquiridos: Ninguno. - Cartera: 1000.0 Comprador1 finalizando...

4.5 - Escenario con poca demanda/mucha oferta

Este escenario se puede probar pasándole al launcher el archivo *configs/fewDemmandScenario.yaml*. Para este ejemplo tenemos dos vendedores que deposita siete lotes y un comprador con una cartera inicial de 1000. Esta configuración permite que el comprador pueda comprar sin competencia, es decir, siempre que le interese un lote lo obtendrá por ser siempre el primero en pujar por él. Además podrá rechazar y bajar el precio del lote sabiendo que no hay otro comprador que quiera comprarlo más caro. Es por esto que podría conseguir comprar un lote de lubina (580) por 185.

Mon Jul 08 13:38:31 CEST 2019	Comprador1	SUBASTA	A Comprador1 le interesa el lote de lubina (185.0) [interes = 36.24% / 210.19]
Mon Jul 08 13:38:31 CEST 2019	Lonja	SUBASTA	Cinta 0: Lote de lubina vendido a Comprador1 por 185.0

O un lote de calamar (450) por 191.

Mon Jul 08 13:38:36 CEST 2019	Comprador1	SUBASTA	A Comprador1 le interesa el lote de calamar (191.0) [interes = 57.61% / 227.56]
Mon Jul 08 13:38:36 CEST 2019	Lonja	SUBASTA	Cinta 0: Lote de calamar vendido a Comprador1 por 191.0

El resultado final es que el comprador obtendrá muchos lotes y su cartera bajará considerablemente.

Mon Jul 08 13:38:37 CEST 2019 Comprador1 takeDown

- Lotes adquiridos:
- Lot-0 [day= 0, time=2, kind=sardina, kg=15.0, reservePrice=60.0, startingPrice=120.0, finalPrice=120.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime={"simState":"RUNNING","day":0,"time":2}, seller=Vendedor1, buyer=Comprador1, auctioned=true]
- Lot-1 [day= 0, time=2, kind=pulpo, kg=10.0, reservePrice=140.0, startingPrice=300.0, finalPrice=300.0, registerTime={"simState":"RUNNING","day":0,"time":2}, saleTime={"simState":"RUNNING","day":0,"time":2}, seller=Vendedor1, buyer=Comprador1, auctioned=true]
- Lot-5 [day= 2, time=4, kind=lubina, kg=8.0, reservePrice=180.0, startingPrice=580.0, finalPrice=185.0, registerTime={"simState":"RUNNING","day":2,"time":4}, saleTime={"simState":"RUNNING","day":2,"time":4}, seller=Vendedor2, buyer=Comprador1, auctioned=true]
- Lot-6 [day= 2, time=9, kind=calamar, kg=15.0, reservePrice=100.0, startingPrice=450.0, finalPrice=191.0, registerTime={"simState":"RUNNING","day":2,"time":9}, saleTime={"simState":"RUNNING","day":2,"time":9}, seller=Vendedor1, buyer=Comprador1, auctioned=true].
- Cartera: 204.0
Comprador1 finalizando...

4.6 - Conclusión de los ejemplos.

En resumen podemos ver que el SMA funciona de forma adecuada en escenarios complejos, con una o más cintas, para uno o más días, con mucha o poca demanda, con mucha o poca oferta, con o sin latencia e incluso, cuando los lotes llegan juntos o en el último tick de reloj. También hay que dejar claro que la simulación varía en gran medida en cada ejecución independientemente de la configuración debido a las probabilidades usadas para la puja de los compradores. Esto deriva en que los resultados que yo he comentado en este documento serán distintos a los producidos en ejecuciones posteriores. Recomiendo seguir la explicación de cada escenario con las capturas y no viendo el log producido en la ejecución, porque probablemente sea similar pero diferente, y ver el log a parte.

5. - Conclusión

En definitiva creo que esta práctica me ha sido muy útil para explorar el mundo de los sistemas multiagente y la plataforma de Jade. Han sido muchos meses trabajando solo en el desarrollo de la misma y después del esfuerzo puedo decir que creo que realmente he aprendido cómo funciona el modelo de agente, las conversaciones, los protocolos de interacción, etc. Es verdad que al principio me desesperé bastante porque es difícil empezar a trabajar con una librería desconocida, que no sabes bien como funciona, pero cuando coges el ritmo no me parece una práctica que suponga un esfuerzo abusivo. Para otros años creo que se podría mantener como está, ya que creo que cumple la relación de esfuerzo y aprendizaje adecuada. También quería valorar la entrega a los alumnos de un esqueleto base para la práctica, que es enormemente útil para no empezar de cero, creo que te da una dirección en la que empezar a investigar y a explorar para continuar con el desarrollo del SMA.