Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Evaluación de distintas estrategias de negociación en el entorno de JGomas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Caamaño Panadero, Daniel

Tutor: Julián Inglada, Vicente Javier

Cotutor: Carrascosa Casamayor, Carlos

2015 - 2016

Resumen

El objetivo de este proyecto se resume en el diseño, implementación y validación de un módulo adicional para la versión en Jason de la herramienta didáctica JGOMAS, desarrollada por el grupo de Tecnología Informática - Inteligencia Artificial de la Universitat Politécnica de València, la cuál sea capaz de dotar a la misma, de un protocolo de subastas fiable y robusto, basado en los conceptos de las subastas clásicas, extendiendo y mejorando así la funcionalidad de la herramienta en su futuro uso, tanto con fines didácticos, como orientados a la investigación.

Palabras clave: Sistemas Multiagente, Inteligencia Artificial, subastas, JGOMAS, JASON

Abstract

The objective of this project can be summarized in the design, implementation and validation of an additional module for a Jason version of the didactic tool JGOMAS, developed by the Computer Tecnology - Artificial Intelligence Group of the Universitat Politècnica de València, which is capable to provide to the tool a reliable and robust auction protocol, based on the classic auction concepts, spreading and improving the usefulness of the tool in future use, both for didactical uses as research-oriented.

Keywords: Multiagent Systems, Artificial Intelligence, auctions, JGOMAS, JASON

Tabla de contenidos

Contenido

Índice de Ilustraciones

Índice de Tablas

1. Introducción

1.1 - Motivación

En la actualidad, la utilización de técnicas de inteligencia artificial dentro de una amplia variedad de campos, es cada vez mayor. En concreto, el uso de agentes inteligentes como paradigma de programación, ofrece ventajas únicas respecto al resto de técnicas, ya que proporciona y facilita la creación de programas autónomos y flexibles dentro de entornos distribuidos y complejos, con gran capacidad de interacción y capaces de reaccionar y actuar, de manera rápida y eficaz, a estímulos y cambios en el entorno.

Dentro de este contexto, existen gran cantidad de técnicas para la comunicación y coordinación de agentes, siendo una de las más utilizadas la negociación automática, y de manera más específica, los sistemas de subastas clásicas, donde los agentes negocian, de manera independiente, con el objetivo de obtener el bien subastado al mejor precio posible. Este concepto puede ser extrapolado a cualquier contexto en el que los agentes necesiten realizar una negociación y obtener un recurso, desde técnicas para la distribución de memoria compartida a subastas propiamente dichas, donde los bienes subastados representan procesos logísticos o materiales utilizables por empresas o corporaciones.

Por todo ello, la negociación entre agentes es un tema actualmente en desarrollo, pudiendo encontrar ya sistemas capaces de proporcionar toda la infraestructura necesaria para la realización de este tipo de negociaciones.

En el contexto concreto de este proyecto, se hacía necesaria la implementación de un módulo adicional para la herramienta didáctica Jason JGOMAS, utilizada para facilitar la comprensión y estudio de las técnicas de inteligencia artificial basadas en agentes, el cual carecía de la funcionalidad necesaria para realizar este tipo de negociación basado en subastas. Con la introducción de este módulo adicional se pretende ampliar así su funcionalidad, lo que proporcionaría ventajas tales como:

- Incorporar un recurso docente a la herramienta JGOMAS que permitirá trasladar fácilmente los conceptos teóricos de la negociación entre agentes al terreno práctico, facilitando así su comprensión y permitiendo trabajar con ellos.

- Desde un punto de vista más científico, disponer de una herramienta fiable donde poder evaluar y comprobar distintas técnicas de subastas en un entorno simulado.

1.2 - Objetivos del proyecto

El objetivo principal de este proyecto es ampliar la funcionalidad de la herramienta didáctica Jason JGOMAS, dotándola de un módulo adicional, el cual se encargue de manera independiente de administrar un protocolo de subastas entre los agentes inteligentes participantes en la partida. Se pretende conseguir así, mejorar el uso de la misma en un contexto didáctico, pudiendo ser utilizada como un recurso más en el aprendizaje y comprensión del funcionamiento e implementación de los protocolo clásicos de subastas.

Por otra parte, el uso de la herramienta no queda limitado solo al ámbito didáctico, ya que será requisito necesario la creación de un registro de datos, el cual pueda ser analizado y tratado para su posterior uso con fines académicos o de investigación. Para ello se diseñará una aplicación capaz de procesar la información resultante y mostrarla al usuario de la manera más amigable y fácil de analizar. Por otra parte la aplicación será capaz de almacenar está información de forma permanente en algún tipo de formato para su posterior análisis.

Las subastas planteadas inicialmente para el proyecto son, la subasta de sobre cerrado de primer precio, la subasta de sobre cerrado de segundo precio, la subasta holandesa o a la baja, y la subasta japonesa o por rondas de negociación. Será necesario dotar al módulo de un API intuitivo y flexible, el cual facilite el uso de las subastas dentro y fuera del entorno JGOMAS, permitiéndole recibir una cantidad de parámetros variables en función de las necesidades específicas de cada momento, ya sea el tipo de subasta, el parámetro solicitado en la misma, las rondas de negociación o la cantidad de agentes participantes.

De igual manera será necesaria la validación y comprobación del correcto funcionamiento de las diferentes subastas, incluyendo un análisis estadístico de la cantidad de mensajes enviados, cantidad de información enviada, tiempo empleado, información procesada y carga de los agentes, para cada una de las diferentes subastas utilizadas. De manera paralela al sistema de subastas, se implementará en el mismo módulo un pequeño sistema de perfiles de apostador para los agentes, extendiendo aún más la funcionalidad y dando pie a su posible reutilización para posteriores estudios en el comportamiento de los agentes en subastas, pudiéndose extender fácilmente con procesos de aprendizaje automático en función de las subastas ganadas o perdidas y la efectividad del agente en la partida.

Todas estas tareas deben llevarse a cabo teniendo en cuenta los principios básicos de cualquier aplicación informática, minimizar los costes temporales y espaciales de los procesos implicados, minimizar el intercambio de información, ya sea en memoria compartida o a través de paso de mensajes, ofrecer fiabilidad y robustez en los resultados obtenidos, ofrecer al usuario una interfaz amigable e intuitiva y por supuesto ya que los sistemas MAS son por definición sistemas concurrentes, evitar los problemas derivados de utilizarlos, tales como interbloqueos o sobre-escritura de variables.

Junto al trabajo propiamente técnico, se realizará un documento con carácter informativo de la herramienta, y las funcionalidades que en ella se incluye, para facilitar su posterior uso y mejora, además de incluir una serie de propuestas o escenarios en los cuales puede ser beneficioso el uso de subastas dentro del entorno de juego JGOMAS.

Finalmente, una vez terminada la implementación y validación del módulo, se realizarán una serie de propuestas con carácter didáctico, mediante las cuales se podrá enseñar de manera fácil y rápida el uso de la herramienta, y el posterior análisis de las subastas realizadas con la misma.

Los objetivos específicos del proyecto son:

- Obtener un API flexible e intuitivo
- Crear un protocolo fiable de encuestas
- Obtener un sistema de subastas concurrente
- Obtener un registro de datos del proceso de subastas

- Obtener un registro para el análisis de perfiles
- Minimizar el impacto del módulo original JGOMAS
- Minimizar los costes temporales del proceso de subastas
- Minimizar los costes espaciales del proceso de subastas
- Crear un protocolo fiable de selección de líder
- Maximizar el número de parámetros utilizables en las subastas
- Obtener la mayor flexibilidad entre los parámetros y los tipos de subasta
- Evitar a toda costa posibles problemas de concurrencia
- Minimizar el número de mensajes intercambiados entre los agentes
- Minimizar el tamaño de los mensajes enviados por los agentes
- Facilitar la comprensión del código implementado
- Facilitar nuevas implementaciones de tipos de subastas
- Facilitar el uso del sistema de perfiles a posteriori
- Facilitar el uso como herramienta didáctica
- Diseñar distintos escenarios para su posible uso en el juego
- Diseñar pruebas didácticas para su posible uso
- Se mantendrán las fechas límites de entrega del proyecto

Los límites para este proyecto son los siguientes:

- No se implementará aprendizaje automático para los perfiles de apostador
- No se implementaran solicitudes de subasta concurrentes pero sí peticiones
- No se contemplaran grandes cantidades de datos a analizar por la aplicación
- No se implementarán módulos adicionales de apoyo a la subasta
- No se crearán versiones alternativas en otros lenguajes de programación
- No se utilizará un generador automático de documentación
- No se realizaran análisis estadísticos del desarrollo de los perfiles de apostador
- No se incluirán métodos de selección de líder en base a la carga del agente

2. Sistemas Multiagente

Para aportar una base teórica del contexto en el que se desarrolla el proyecto es necesario describir los fundamentos básicos en los cuales se basa todo el desarrollo de la aplicación JGOMAS y por tanto del módulo de subastas que se propone.

Dentro del marco de la inteligencia artificial clásica, donde se pretende emular el comportamiento del cerebro humano y los razonamientos lógicos propios del mismo, se desarrolla, en la década de los 90, una nueva aproximación más encaminada a lo que se conoce como la inteligencia colectiva o de enjambre, un término por el cual de describe el comportamiento social similar al de las poblaciones de insectos o de algunas aves, las cuales crean sujetos autónomos que actúan de manera independiente, pero que, por otra parte, son capaces de coordinarse formando una inteligencia colectiva, a partir de las interacciones que se producen entre ellos.

De esta manera se obtienen sistemas con los cuales se puede representar la interacción de miles de individuos que trabajan de manera independiente, con unos intereses y objetivos propios, y que son capaces de percibir y de reaccionar en consecuencia a estímulos exteriores, ya sea en ambientes simulados virtualmente, como en el propio mundo real.

La definición comúnmente más aceptada de un agente inteligente software es "un programa informático auto-contenido que puede controlar sus propias acciones, basado en sus percepciones del entorno" (Huhns & Singh, 1997). En la Figura 1 puede verse un ejemplo gráfico esquemático del funcionamiento básico de un agente.

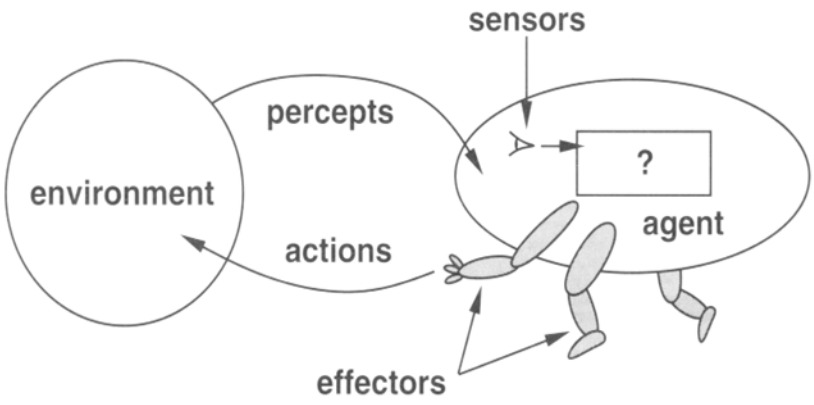


Fig. 1: Agente Inteligente

La mayoría de autores considera que los agentes son entidades AIRP (Wooldridge & Jennings, 1995) y deben cumplir una serie de requisitos para que se puedan considerar como tales:

- Autónomos: Pueden operar de manera independiente sin el control humano o la ayuda de otros agentes.
- Interactivos: Capaces de relacionarse con su entorno y con habilidad social para relacionarse con otros agentes.
- Reactivos: Son capaces de percibir el entorno que les rodea, ya sea virtual o real, y actuar en consecuencia a ello.

- Proactivos: Son capaces de tomar la iniciativa y realizar acciones en base a sus propios objetivos, o los objetivos adquiridos durante su ciclo de vida.

En general, los comportamientos que presentan los agentes están basados en la percepción de su entorno de manera local, no conocen la totalidad del mundo que les rodea, ni tienen por qué conocerlo. Pueden ser programados para realizar procesos de adaptación al entorno, aprendizaje, planificaciones muy sofisticadas e incluso adquirir lenguajes de comunicación propios.

Esta amplia diversidad y flexibilidad de la que hacen gala, hace de ellos una herramienta muy útil en el modelado de poblaciones sociales, pudiendo representar cualquier tipo de entidad humana individual o colectiva, como grupos de familias, empresas, entidades bancarias, países, etc. De igual manera pueden ser usados para representar entes abstractos, u objetos inanimados.

Los primeros modelos de simulaciones eran generalmente más apropiados para recrear sistemas rígidos, poco flexibles, normalmente condicionados a leyes físicas. Por el contrario, las simulaciones con sistemas MAS están más capacitados para representar los escenarios en los que existe una alta variabilidad de poblaciones, con un alto grado de localización y distribución, como podría ser el caso actual de las redes sociales, un hecho que les está llevando a encontrarse entre los sistemas con mayor potencial de desarrollo de los últimos años, siendo usados en infinidad de campos.

No hay que olvidar que estos modelos provienen del campo de la inteligencia artificial distribuida, usados para la resolución de problemas específicos. Ésta se inspira en la aproximación general de la "inteligencia colectiva", en la que "numerosos agentes simples intercomunicados generan comportamientos y soluciones altamente satisfactorias a partir de reglas de comportamiento individual relativamente simples" (Schut, 2007).

Para ello es necesario crear un conjunto de agentes que representen una "sociedad virtual colectiva", los cuales, mediante interacciones entre ellos, con el entorno que les rodea, y en igual medida, con sus propias capacidades individuales para desenvolverse dentro del ambiente, sean capaces de conseguir que este proceso desemboque en la resolución del problema que se ha pretendido modelar con la simulación en cuestión.

Para un correcto funcionamiento de todo este sistema, es necesario establecer una ontología, la cual determine como representar ese tipo de conocimientos específicos, que van a guiar al agente en el proceso de desarrollo a lo largo de su ciclo de vida. Para ello, es necesario que el agente sea capaz de determinar la situación que le rodea a cada momento, y en consecuencia, acceder a los posibles comportamientos u objetivos que pueden darse en esa situación específica. También es necesario que el agente sea capaz de almacenar esa información, y recordarla más adelante para posibles usos, además de acceder a un conjunto de reglas que determine qué acciones deben ser realizadas cuando se tiene un objetivo a cumplir.

Estas características hacen de los SMA unos modelos únicos de simulación, los cuales son capaces de simular múltiples agentes de diversos tipos y dotarlos de características y reglas de decisión diferentes, sin necesidad de una racionalidad perfecta. Además permiten ser distribuidos de manera espacial dentro del entorno virtual o el mundo real, incorporar mecanismos de aprendizaje, modelar las propias interacciones entre los agentes y además, estudiar la aparición de comportamientos agregados o emergentes, resultando estos últimos muy

interesantes debido a que no están concebidos ni programados en el comportamiento predefinido inicialmente.

Actualmente no existe ningún acuerdo o estándar sobre la correcta construcción de un SMA, debido al amplio espectro que abarca y las complejidades que pueden alcanzar las relaciones que se establecen en él. Tan solo en lo que se refiere a los objetivos que intenta perseguir el sistema, se establecen ciertas preferencias respecto a qué tipos de metodologías utilizar para su implementación.

Aun así, podemos definir un conjunto de elementos básicos los cuales deben formar parte de un sistema SMA:

- Percibir el estado del entorno.
- Almacenar inputs y acciones pasadas.
- Planificar acciones futuras.
- Llevar a cabo acciones sobre el entorno.
- Generar resultados.

Existen múltiples alternativas técnicas computacionales para la implementación de agentes software, que poseen las cualidades mencionadas previamente. Entre otras podemos encontrar la programación orientada objetos, las redes neuronales artificiales o sistemas de reglas de producción, las cuales pueden ser utilizadas de manera conjunta dentro de un mismo sistema SMA, sin tener porque ser excluyentes entre ellas.

En general, las dos arquitecturas más habituales para la construcción de agentes software se corresponden con el paradigma simbólico de la inteligencia artificial clásica junto a los algoritmos clásicos de resolución de problemas, basados en sistemas de producción y, por otra parte, con las aproximaciones no simbólicas, basadas en técnicas de aprendizaje automático como las redes neuronales, redes neuronales profundas, algoritmos perceptron o algoritmos genéticos.

2.1 – Agentes BDI

Ya que este proyecto utiliza la primera de las arquitecturas mencionadas en el apartado anterior, correspondiente al paradigma simbólico de la inteligencia artificial clásica, es necesario exponer una síntesis de los conceptos básicos que la conforman.

El diseño de agentes BDI se basa en el concepto de crear entidades que actúan usando procesos lógicos similares a los que utilizaría el ser humano, dotándolos de un conjunto de creencias (Beliefs), un conjunto de deseos (Desires) u objetivos, y de un conjunto de intenciones (Intentions) o planes realizables. Esta arquitectura se corresponde con un sistema clásico de producción, una técnica computacional simple y eficaz construida a partir de 3 componentes básicos, un conjunto de reglas, una memoria de trabajo y un intérprete de reglas.

Reglas

Dentro del conjunto de reglas, cada una de ellas está compuesta de dos partes, una condición o causa, la cual define la situación específica donde la regla será disparada por el sistema, y una acción o consecuencia, que representa la lista de acciones que ejecutará el agente, ya sean la ejecución de planes, la obtención de nuevos objetivos, la búsqueda de creencias en su memoria, etc.

Creencias

La memoria de trabajo se estructura a alto nivel en forma de hechos o creencias, los cuales son almacenados por el agente para su posterior uso. La estructuración de la creencias en los lenguajes propios del paradigma lógico suelen estar representadas como sentencias lógicas, con variedad respecto a su aridad en los parámetros que la definen.

Intérprete de Reglas

La función del intérprete de reglas es considerar todas estas reglas, resolver conflictos entre ellas, disparar aquellas que cumplan las condiciones específicas y que se den en el entorno en ese momento, y realizar las acciones pertenecientes a ellas.

Para una comprensión clara de lo que se expone en el texto se incluye en la Figura 2 un ejemplo sencillo de cómo funciona este sistema.

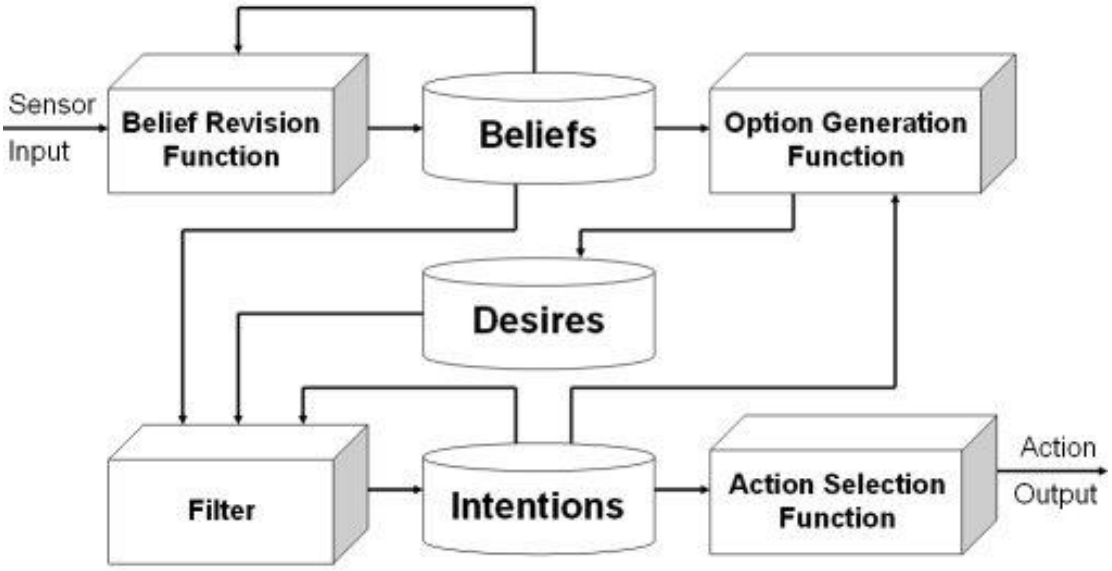


Fig. 2: Funcionamiento BDI

3. Negociación con agentes

Dentro del paradigma de la programación con agentes inteligentes, uno de los factores más relevantes a tener en cuenta son los sistemas y las técnicas de comunicación y coordinación automáticas empleadas para la correcta interacción de los mismos. Este tipo de sistemas son cada vez más utilizados en multitud de campos, siendo especialmente relevantes hoy en día en el mundo de los videojuegos, particularmente en plataformas multijugador, en las simulaciones sociales, donde su uso está ampliamente extendido, y con especial interés, en el mundo empresarial, donde la negociación toma su carácter más literal, al competir por la adquisición de un bien o servicio, mediante el uso de subastas como técnica principal para su realización.

Dado que este proyecto está centrado en el uso de las subastas como técnica de coordinación, a continuación se realiza una exposición más detallada de su utilización dentro del ámbito empresarial, en concreto en la formación y creación de corporación empresariales.

El uso de agentes inteligentes en el ámbito de las corporaciones empresariales está en auge. Debido al ritmo frenético en el que se desarrollan los negocios hoy en día, las empresas deben hacer frente a términos muy duros en cuanto a competitividad se refiere. Este ritmo conlleva, junto a la rápida creación de sectores muy especializados dentro de campos muy variados, cambios drásticos en las situaciones del mercado. Estas situaciones fuerzan a las empresas a mantener cooperaciones incluso con sus competidores más directos, lo que da lugar a una nueva forma de ver los negocios, actualmente en el punto de mira. Según varios expertos, este nuevo concepto, conocido como corporación virtual (Virtual Enterprise, VE), se presenta como el modelo de cooperación del futuro para las empresas (Byrne, 1993).

Este término es comúnmente atribuido a Mowshowitz (Mowshowitz, 1986), quien estableció un paralelismo entre el mismo, y el término memoria virtual, usado en el contexto de las tecnologías de la información. Byrne define una corporación virtual como una red temporal de varias compañías independientes, donde se comparten habilidades, costos y se tiene acceso a los mercados de cada una de las empresas participantes.

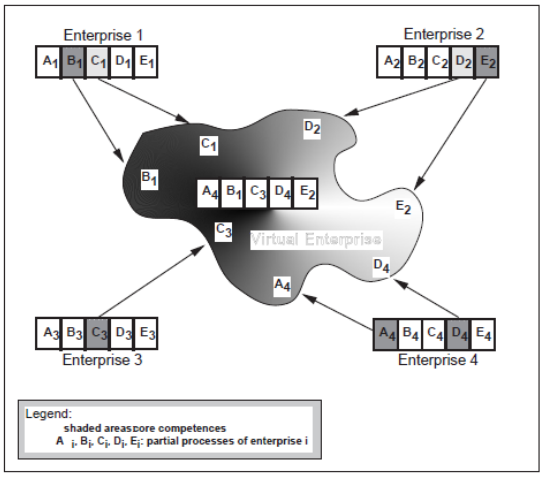


Fig. 3: Esquema Corporación Virtual

En la Figura 3 se puede observar de manera gráfica el funcionamiento de una corporación virtual. En ella se pueden ver los diferentes subprocesos de cada una de las empresas individuales, los cuales son producidos utilizando las competencias, capacidades y tecnologías que aportan todas las empresas en conjunto dentro de la red. De esta manera se obtienen productos mucho más competitivos y exitosos.

Debido a que las empresas participantes solo aportan eslabones a la cadena de producción del producto, el mayor problema derivado de la creación de una corporación virtual es la realización del mapeado de los procesos parciales de los que se ocupa cada una de las empresas, y que intervendrán en esta red. Para ello, se realiza un proceso de diseño dividido en dos fases, una consistente en un proceso de partición de los negocios que intervendrán, y otra encargada de instanciar esa asociación.

En esta segunda fase es donde los participantes de la red corporativa son seleccionados, y a su vez, puede ser dividida en 4 sub-fases:

- Identificación de los compañeros
- Generación de mapeados alternativos
- Evaluación de los intereses estratégicos y los riesgos
- Finalización del producto o servicio

Es en esta fase donde la tecnología basada en agentes inteligentes puede ser utilizada con resultados satisfactorios. Esto se debe a que las compañías no pueden saber en todo momento dónde o cómo pueden contribuir en una red corporativa, debido a que la información actual que ofrece el mercado no está lo suficientemente estructura y resumida. Este fenómeno se da especialmente en lo referente a mercados virtuales en la red, donde la cantidad de información es enorme, y se hace necesario el desarrollo y uso de herramientas capaces de soportar y manejar toda esta información, y usarla para el diseño de redes corporativas funcionales.

Como ya se han descrito las características y capacidades de la tecnología basada en agentes inteligentes en apartados anteriores, aquí únicamente se hará especial hincapié en una arquitectura específica para el diseño de agentes, concretamente InterRRaP, la cual fue desarrollada en el DFKI, y que permite crear agentes autónomos con una gran capacidad de interacción.

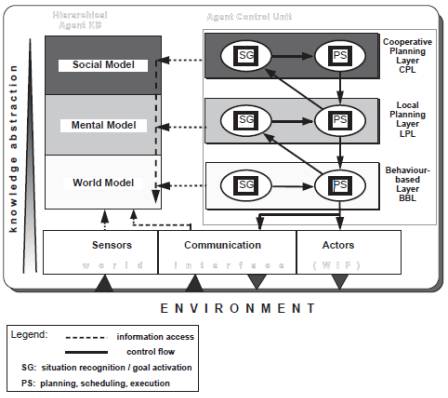


Fig. 4: Arquitectura InterRRaP

En la Figura 4, es posible ver de manera gráfica como se estructura está arquitectura y su funcionamiento. Básicamente consiste en la creación de 3 capas diferenciadas: la capa de comportamiento básico, la capa de planificación local y la capa de planificación cooperativa, las cuales determinarán el comportamiento del agente en cuestión.

Estas 3 capas se corresponden con 3 estructuraciones o modelos distintos en la base de conocimientos del agente, el modelo del mundo, el modelo mental y el modelo social. Esto permite al agente ser más flexible a la hora de tomar deliberaciones, dando normalmente prioridad a la capa encargada del comportamiento básico, frente a las demás, pudiendo así realizar deliberaciones más complejas y elaboradas cuando se usan las capas de planificación o la de cooperación, y reaccionar rápidamente a estímulos directos del entorno, sin realizar procesos complejos de deliberación.

Para utilizar este tipo de tecnología es necesario el uso de servidores de información dedicados que solventan los problemas de recolección y filtrado de la información de las distintas empresas que participan en una VE. En estos servidores se anuncia y obtiene la información relevante de cada compañía, y se realizan funciones tales como:

- Recolección de objetivos y ofertas
- Respuesta a solicitudes de oferta.
- Automatizar la búsqueda de posibles solicitudes con ofertas.

En la Figura 5 puede verse de manera gráfica el funcionamiento de este tipo de sistemas, donde se hace necesario el uso de agentes inteligentes como asistentes de las decisiones humanas tomadas en él. Es necesario destacar que estos sistemas deben ser diseñados teniendo en cuenta que los humanos pueden estar envueltos en los procesos de negociación, y por lo tanto deben ser capaces de entender los mensajes introducidos por los mismos en el sistema.

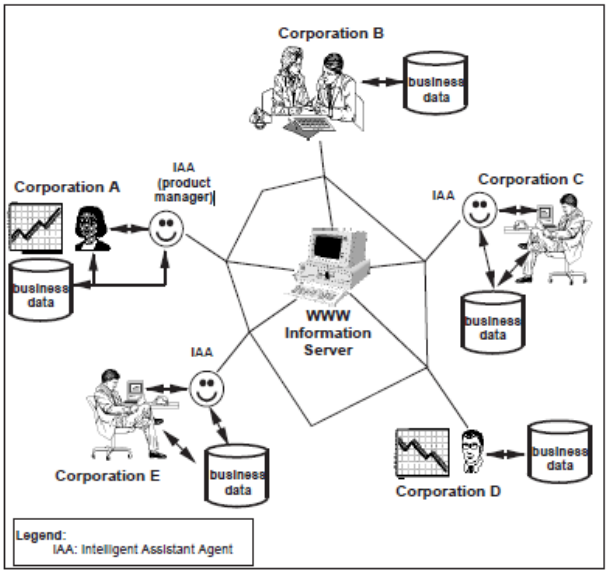


Fig. 5: Sistema Corporación Virtual

Una figura importante en la creación de una VE es el administrador del producto, encargado de la coordinación y asignación de los compañeros de los procesos que se realizan. Esta actividad requiere altos niveles de coordinación y comunicación con el resto de participantes de la VE. Esta actividad puede ser asistida mediante el uso de agentes inteligentes, encargados de automatizar la búsqueda de compañeros, recayendo finalmente en el administrador la tarea de actuar como subastador.

Finalmente, el subastador ofrece diferentes procesos a realizar en la VE y las distintas empresas o colaciones pueden apostar para adquirirlos y realizarlos.

3.1 – Agentes inteligentes como asistentes en VE

La idea principal del uso de agentes inteligentes para la formación de redes corporativas virtuales es tomar el proceso como un problema de decisión multi-agente. Esta solución se realiza de forma distribuida mediante 4 fases diferenciadas:

- Especificación de los objetivos:
  Descripción del producto o servicio que se quiere obtener con la formación de la VE. Son atributos relevantes, el coste, la calidad, el precio, el tiempo de entrega. Esta labor es una tarea experta difícil de formalizar, que puede ser asistida por IAAs, los cuales ayudan al administrador del producto a definir estos objetivos basadas en la cooperación con el resto de empresas implicadas.

- Descomposición:
  Mapeado de los objetivos en sub-procesos, esta tarea consiste en la descomposición del objetivo principal en subprocesos relevantes. Puede ser realizada por humanos asistidos con IAAs, aunque la tendencia actual es intentar automatizar el proceso.

- Asignación:
  Selección de los posibles compañeros en la instanciación de los sub-procesos. Este proceso puede ser asistido por IAAs, los cuales tienen como tareas la recolección de información relevante, la formulación de nuevas ofertas, la asistencia en el proceso de subastas y la realización de negociaciones autónomas entre ellos.

- Síntesis:
  Instanciación del proceso global mediante los sub-procesos realizados. El mayor problema derivado de esta fase es la localización física de los recursos del sistema.

3.2 – Mecanismos de subastas en la formación de VE

En el contexto de la formación de VEs, las subastas pueden ser utilizadas para implementar y resolver el problema planteado en la fase de asignación, donde las distintas empresas apuestan por contribuir en los sub-procesos ofertados por el administrador del producto, el cual toma el rol de subastador, mientras que los sub-procesos toman el rol del bien ofertado.

En la literatura, existen múltiples propuestas para la realización de subastas (e.g.. [Murnighan 91]), generalmente siendo divididas en dos grandes grupos. Las subastas abiertas, donde todos los participantes de la subastas conocen el valor apostado por los demás participantes, y las subastas cerradas, donde únicamente el subastador conoce el valor apostado por los

participantes. Las subastas abiertas más importantes son la inglesa y la alemana. En el grupo de las subastas cerradas se encuentran la subasta de sobre cerrado de primer precio, y la subasta de sobre cerrado de segundo precio.

- Subasta Inglesa: Subasta a la alza, donde el mayor apostador obtiene el bien ofertado. La subasta termina cuando no se realizan nuevas apuestas que superen la última realizada.

- Subasta Alemana: Subasta a la baja, donde el subastador ofrece un precio de inicio alto, y va descendiendo. El primer participante que acepte el precio ofertado es el ganador.

- Subasta de sobre cerrado de primer precio: En esta subasta cada participante realiza una apuesta de manera individual, sin conocer las del resto de participantes. El participante que realiza la oferta mayor es el ganador.

- Subasta de sobre cerrado de segundo precio: En esta subasta se realiza el mismo proceso que en la subasta de primer precio, siendo el precio a pagar por el ganador el ofertado por el segundo precio más alto ofrecido.

Teóricamente no existen diferencias a la hora de seleccionar un tipo u otro de subasta que beneficien al subastador. Aunque se ha comprobado de manera experimental que con participantes que tienen perfiles de apostador no arriesgados o cobardes, la subasta alemana o la de sobre cerrado de primer precio generan valores de venta mayores. Por otra parte, normalmente las subastas de primer precio generan valores de venta mayores que las de segundo precio. Pero por normal general, los resultados de las subastas dependen de la situación del mercado en el momento en que se realizan.

Un fenómeno a tener en cuenta cuando se realizan subastas, es la formación de asociaciones de apostadores o cárteles, los cuales, acuerdan precios de puja, obteniendo así el bien subastado por un valor mucho más bajo del que se obtendría en una subasta habitual, para después ser re-subastado de manera interna, por los miembros de la asociación. Estas prácticas pueden reducirse estableciendo precios de reserva lo suficientemente altos como para no incentivar la formación de asociaciones.

Este apartado ha sido extraído del artículo "*IA in Virtual Enterprises"*, (Klaus Fischer, 1996).

4. Jason JGOMAS

Una vez presentados los fundamentos teóricos del paradigma de programación basado en agentes inteligentes, se pasa a realizar un resumen explicativo de la herramienta JGOMAS en su versión actual implementada bajo JADE y Jason. JGOMAS (Game Oriented Multi-Agent System), es un sistema Multi-Agente desarrollado inicialmente para servir de punto inicial para un estudio de la viabilidad de la integración de SMA y realidad virtual, permitiendo también ser usada como herramienta de validación de SMA.

Para simular el entorno de juego se seleccionó un juego clásico del tipo capturar la bandera, donde dos equipos compiten por obtener la bandera del oponente y devolverla a su base inicial. Este tipo de juego se ha convertido casi en un estándar en casi todos los juegos multi-jugador actuales.

En la versión de JGOMAS, los dos equipos representan dos bandos, el eje y los aliados, encargados de desplegarse y capturar la bandera enemiga y devolverla a su propia base en el caso de los aliados, y de defenderla en el caso del eje. Las partidas tienen una duración determinada tras la cual, una vez transcurrido este tiempo, se finaliza la partida con el bando del eje como ganador del juego.

4.1 – Arquitectura de JGOMAS

La arquitectura de Jason-JGOMAS está compuesta principalmente por dos subsistemas, los cuales trabajan de manera coordinada. Por una parte existe un MAS, con dos clases diferentes de agentes, que está implementado como una capa que trabaja por encima de una plataforma de sistemas multi-agentes, de la cual aprovecha los recursos y servicios que ofrece. Este sistema es concretamente JADE.

Una de las dos clases de agentes mencionadas anteriormente es la que se encarga de controlar la lógica del propio juego, comúnmente conocido como manager, o administrador del sistema. La otra clase de agentes es la que participa activamente y de forma completa en el juego, y pueden dividirse en 3 clases diferenciadas, los agentes que actúan como soldados de asalto, los que proporcionan apoyo en forma de munición y los médicos de campo, encargados de proporcionar suministros médicos a los agentes que los necesiten. En la Figura 6 puede verse una representación gráfica del sistema en cuestión.

Por otro lado, el otro subsistema que conforma la herramienta JGOMAS es un visor grafico (render engine) diseñado ad-hoc para visualizar un entorno 3D virtual. Debidos a los altos requerimientos computacionales de este tipo de sistemas, se diseñó como un módulo externo y no como un agente más del sistema, utilizando la biblioteca grafica OpenGL implementada en C++. Este visor grafico se comunica directamente con el agente manager encargado del control del juego, el cual hace de interfaz para el propio visor.

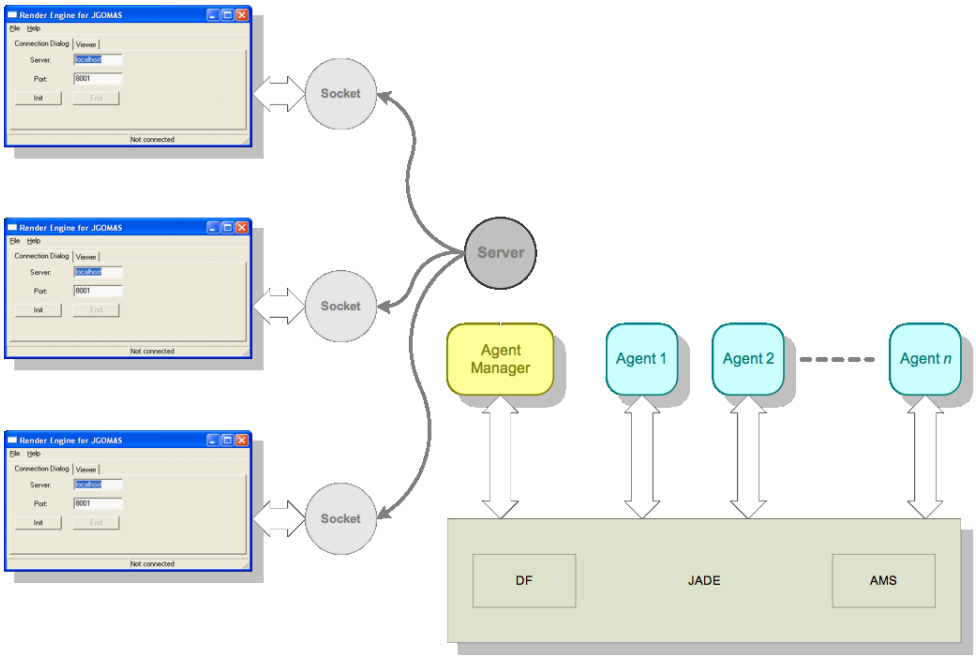


Fig. 6: JADE y JASON

4.2 – Taxonomía de agentes en JGOMAS

Los agentes inteligentes utilizados en JGOMAS pueden ser clasificados siguiendo un criterio en base a su funcionalidad en el juego:

Agentes Internos:

Son propios de la gestión de la plataforma JGOMAS, sus comportamientos están preestablecidos y el usuario no puede modificarlos.

- Manager: Encargado de la coordinación del juego en tiempo real, además de ser capaz de recibir peticiones del resto de agentes implicados en la partida. También se encarga de actuar como interfaz del visor grafico para la visualización del juego.
-
- Pack: En JGOMAS existen 3 tipos diferenciados de packs de apoyo, los medic packs, usados para restaurar la salud de los agentes heridos, los ammo packs, los cuales recargan la munición de los agentes, y los objective pack, estos últimos representados específicamente como la bandera que los agentes tienen que defender o capturar acorde a su bando. Estos 3 tipos de agentes pack son creados y destruidos de manera dinámica durante el transcurso de la partida, a excepción de la bandera que no puede destruirse.

Agentes Externos:

Son los agentes que representan a los participantes del juego actual. Tienen un conjunto de comportamientos básicos ya predefinidos, los cuales pueden ser modificados por el usuario, o incluso se puede extender su funcionalidad añadiendo nuevos comportamientos. A diferencia de

los agentes anteriores, estos agentes han sido desarrollados en Jason, y solo pueden adquirir un único rol específico en la misma partida. Existen ya 3 roles predefinidos, y el usuario puede seleccionar al inicio de la partida cuantos agentes de cada tipo quiere en su equipo o puede incluso desarrollar nuevos roles propios. Los 3 tipos de roles que pueden adoptar los agentes de manera predefinida son:

- Soldado: Proporciona un servicio de apoyo al resto de participantes
- Medico: Encargado de suministrar los botiquines a sus compañeros
- Apoyo: Se ocupa del reparto de municiones de su equipo

Este sistema permite de una forma sencilla y entretenida establecer un campo de pruebas para la realización y testeo de algoritmos y métodos de optimización en cada uno de los agentes de manera individual, además de permitir diseñar estrategias cooperativas de coordinación para los compañeros del mismo bando. De igual manera, al tener a los agentes situados en un entorno virtual, se deberán tener en cuenta las dificultades y variaciones del terreno que los rodea, incluyéndose así un factor de variabilidad a la hora de probar múltiples partidas.

Queda mencionar que toda la comunicación que se realiza entre los agentes que componen el sistema en su totalidad se realiza por medio de paso de mensajes según los protocolos establecidos por FIPA ACL.

4.3 – Mapas

Como se ha mencionado anteriormente, JGOMAS utiliza un entorno virtual para definir el campo de juego, y este entorno viene definido por mapas. Por defecto, estos mapas son de tamaño 256x256, de esta manera la posición de los agentes tendrá que estar determinada por unas coordenadas espaciales x, y, z, que tomarán valores entre 0 y 255. Ya que no existen diferentes alturas la variables Y, siempre tomará el valor 0.

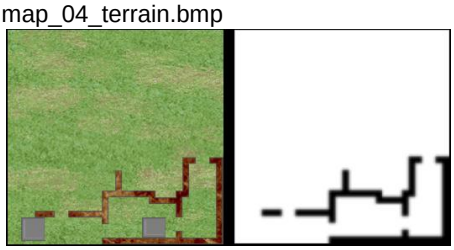
Los agentes participantes no conocen la extensión del mapa, ni su totalidad, ya que solamente pueden percibir los objetos que se encuentran dentro de un campo de visión definido por un "cono de visión".

Los mapas se almacenan dentro del proyecto, y pueden ser modificados o creados por los propios usuarios. Para definirlos se utilizan 3 tipos de archivos: uno en formato de texto, el cual define la configuración del mapa, e incluye las variables necesarias para el correcto funcionamiento del sistema, mientras que los otros dos archivos son imágenes en formato mapa de bits, que representan el modelado artístico del mismo.

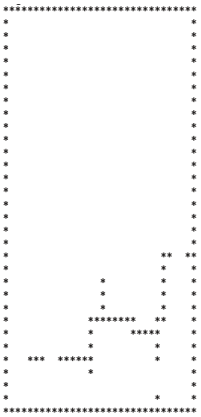
En la Figura 7 se muestra como ejemplo uno de los mapas que utiliza el sistema, y las variables que incluye.

```
[JADE]
JADE_OBJECTIVE:28 28
JADE_SPAWN_ALLIED:2 28 4 30
JADE_SPAWN_AXIS: 20 28 22 30
JADE_COST_MAP: 32 32
map_04_cost.txt
```

map_04_terrain.bmp



map_04_cost.bmp



map_04_cost.txt

- JADE_OBJECTIVE: Localización inicial de la bandera.
- JADE_SPAWN_ALLIED: Localización de la base Aliada.
- JADE_SPAWN_AXIS: Localización de la base del Eje.
- JADE_COST_MAP: Tamaño y nombre del fichero de costes.
- RENDER_ART_MAP: Tamaño y nombre del fichero de arte.
- RENDER_COST_MAP: Tamaño y nombre del fichero de arte de los costes.
- RENDER_HEIGHT_MAP: Tamaño y nombre del fichero de arte de las alturas.

Fig. 7: Mapas JGOMAS

4.4 – Tareas

El funcionamiento de los agentes está determinado mediante tareas predefinidas, las cuales deben realizarse una posición concreta del entorno virtual. Estas tareas representan acciones que los agentes pueden llevar a cabo, siendo las principales:

• TASK_GIVE_MEDICPAKS: Un médico debe generar paquetes de medicina en un lugar determinado (la posición del agente que se lo ha solicitado y al que ha accedido ir a dárselos).

• TASK_GIVE_AMMOPAKS: Un fieldops debe generar paquetes de munición en un lugar determinado (la posición del agente que se lo ha solicitado y al que ha accedido ir a dárselos).

• TASK_GIVE_BACKUP: Un soldado debe ir a ayudar a un compañero a un lugar determinado (la posición del agente que se lo ha solicitado y al que ha accedido ir a dárselo).

• TASK_GET_OBJECTIVE: El agente, del equipo atacante o ALLIED, tiene que ir a la posición inicial de la bandera a por ella. Si logra coger la bandera, esta tarea se transforma en ir hasta su base de origen.

• TASK_GOTO_POSITION: El agente debe ir a una posición concreta.

Cada agente almacena estas tareas en una lista de tareas, las cuales son ordenadas mediante un identificador de prioridad asignado a cada una de ellas. Además de su tipo y su prioridad, una tarea tiene asociados el agente que la provocó y algún posible contenido adicional. El usuario puede modificar estas prioridades, e incluso modificar la propia lista de tareas, pero una vez seleccionadas, el sistema es el encargado de llevarlas a cabo.

Estas tareas son añadidas mediante el uso de un plan predefinido, el cual se encarga de incluirlas en la lista de tareas activas del agente.

4.5 – Bucle de ejecución

El comportamiento de los agentes en JGOMAS está controlado mediante una RMA o máquina de estados, la cual se mantiene en constante ejecución mientras el agente siga vivo. En la Figura 8 es posible ver de manera gráfica el comportamiento de esta máquina de estados, la cual puede representarse como un autómata finito.

Esta máquina se compone de 3 estados diferentes:

• STANDING: El agente no tiene ninguna tarea lanzada.

• GO_TO_TARGET: El agente ha lanzado una tarea y está moviéndose hacia la posición en la que debe realizarla.

• TARGET_REACHED: El agente ha alcanzado la posición en la que debe realizar la tarea lanzada y está realizando las acciones indicadas en la tarea.

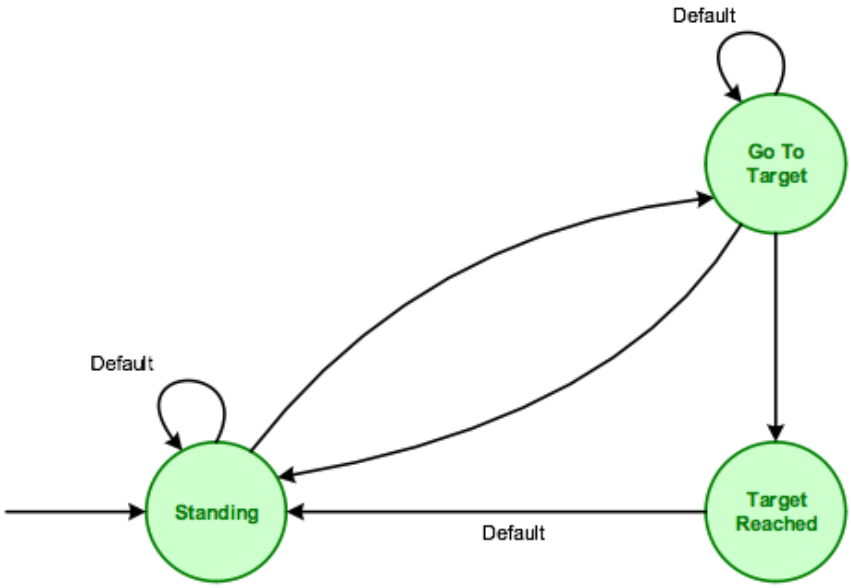


Fig. 8: RMA JGOMAS

4.6 – Interfaz (API)

La interfaz en JGOMAS está compuesta por los ficheros .asl escritos en Jason, mediante los cuales el usuario puede acceder y modificar los distintos planes que definen los comportamientos del agente en cuestión. Estos archivos incluyen el fichero original jgomas.asl en el cual vienen definidos los comportamientos no modificables por el usuario. Existen 6 tipos de ficheros .asl, los cuales definen cada uno de los tipos de agentes participantes en la partida, de igual manera, el usuario puede definir nuevos tipos de agente, con la creación de nuevos ficheros, y su posterior inclusión en el sistema, de manera rápida y sencilla. Estos 6 ficheros básicos son:

- jasonAgent_ALLIED.asl

- jasonAgent_ALLIED_MEDIC.asl
- jasonAgent_ALLIED_FIELDOPS.asl
- jasonAgent_AXIS.asl
- jasonAgent_AXIS_MEDIC.asl
- jasonAgent_AXIS_FIELDOPS.asl

En cuanto a las creencias que poseen los agentes inicialmente, las principales son:

- tasks(task_list): Contiene la lista de tareas activas del agente.
- fovObjects(object_list): Contiene la lista de objetos que ahora mismo ve el agente. La estructura de un objeto es [#, TEAM, TYPE, ANGLE, DISTANCE, HEALTH, POSITION].
- state(Estado_actual): Esta creencia se utiliza para indicar el estado del agente en su máquina de estados: standing, eligiendo que tarea hacer o esperando; go_to_target, acudiendo a su próximo objetivo; target_reached, ha llegado al objetivo; quit, debe terminar.
- my_health(X): Guarda la salud del agente. El valor inicial y máximo es 100, cuando llega a 0, el agente se muere.
- my_ammo(X): Guarda la cantidad de balas que dispone el agente. El valor inicial es 100.
- my_position(X, Y, Z): Guarda la posición última conocida por el agente.

Los diferentes planes modificables con los que cuentan los agentes son los siguientes:

- !perform_look_action

Este objetivo se invoca cuando se ha mirado alrededor y se supone que se ha actualizado la lista de objetos alrededor fovObjects(L).

- !perform_aim_action

Si hay un enemigo al que apuntar se lanza este objetivo, el cual puede servir para tomar alguna decisión respecto a qué hacer.

- !get_agent_to_aim

Este objetivo se invoca después del !perform_look_action, se utilizaría para decidir si hay algún enemigo al que apuntar.

- !perform_no_ammo_action

Este objetivo se lanza cuando el agente dispara y no le quedan balas.

- !perform_injury_action

Este objetivo se lanza cuando el agente es disparado. Sería necesario implementar el plan asociado a la creación de este evento para tomar una decisión, por ejemplo huir si hay poca vida.

- !performThresholdAction

Este objetivo se lanza cuando el agente dispone de menos vida o balas que los umbrales definidos en my_ammo_threshold(X) y en my_health_threshold(X). Una implementación sencilla del plan asociado ya está disponible, la cual siempre pide ayuda a médicos o a fieldops de su equipo.

- !setup_priorities

Este objetivo se lanza en la inicialización del agente para fijar las prioridades de las tareas del agente. Cada agente puede tener sus propias prioridades. Una implementación sencilla del plan asociado ya está disponible.

- !update_targets

Este objetivo puede utilizarse para actualizar las tareas y sus prioridades. Se invoca cuando el agente pasa a estado standing y debe elegir nueva tarea entre las disponibles.

4.7 – Comunicación

La comunicación en JGOMAS es uno de los factores más importantes del sistema, y puede dividirse a grandes rasgos en dos fases, el registro y la coordinación.

- Registro: Cada agente que participa activamente en la partida debe registrarse previamente mediante una función interna, indicando su tipo y equipo al DF (también conocido como páginas amarillas). De esta manera se mantiene el rol establecido durante toda la partida para ese agente, aplicando las reglas necesarias acorde a él. De igual manera un agente puede solicitar información de que otros agentes han sido registrados previamente en la partida, mediante una solicitud al DF.

- Coordinación: La coordinación entre los agentes en JGOMAS puedes realizarse de dos maneras distintas:

  Sin comunicación explicita: La cual implica utilizar únicamente la percepción del agente dentro del entorno virtual, ya sea viendo a otro agente enemigo o aliado, al haber visualizado el objetivo, o cualquier otra situación similar.

  Con comunicación explicita: Los agentes pueden comunicarse directamente mediante el envío de mensajes, un servicio proporcionado por la plataforma en la que se ha implementado, y el cual es sencillo y rápido de manejar.
  Este servicio de mensajería es capaz de transmitir todo tipo de información entre los agentes, ya se trate de creencias, objetivos o planes completos. Una parte importante del módulo de subastas implementado en este trabajo utiliza este sistema de mensajería.

4.8 – Interactividad

Finalmente la última versión de JGOMAS incluye la posibilidad de que el usuario interactúe directamente con el sistema, tomando el control de los soldados y permitiendo ordenarles directamente las acciones en lugar de realizar su comportamiento prefijado. Para ello es necesario el uso en conjunto de la herramienta grafica diseñada en Unity para su visualización.

5. Metodología

Debido a que la parte técnica del proyecto está dividida en dos partes, la propia implementación del módulo en JASON y la aplicación encargada del análisis de los registros, implementada en JAVA, las metodologías utilizadas en los dos casos han sido totalmente diferentes.

Ya que las dos implementaciones por separado no requerían la creación de un gran proyecto, ni habría un grupo de trabajo dedicándose al mismo, no fue necesario el uso de herramientas software de control de versiones, ni se necesitó un análisis exhaustivo de los requisitos de la aplicación o las necesidades del cliente. En lugar de ello se planteó un análisis sencillo de los requisitos del módulo y de la aplicación y se desarrollaron en consecuencia.

Para la implementación del módulo en JASON se utilizó un enfoque de desarrollo de software que podría incluirse en lo que se conoce como un modelo iterativo o creciente, el cual se basa en el concepto de planificar distintas fases en el proyecto e ir refinándolas de manera incremental, incluyendo pruebas de validación en cada una de las iteraciones. De esta manera se consigue que los componentes logren evolucionar el resultado final, agregando nuevas características y funciones a cada paso, y evitando así una acumulación de errores de fases anteriores. En la Figura 9 se muestra un gráfico representativo de este tipo de metodología para el diseño de productos software.

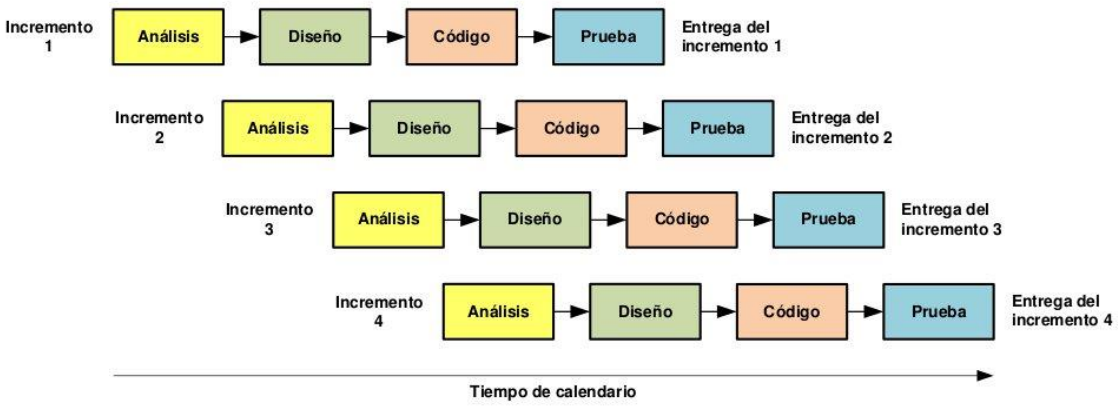


Fig. 9: Metodología incremental

En el apartado de implementación incluido más adelante se describe el proceso completo de desarrollo del módulo; pero a rasgos generales, se plantearon una serie de hitos o fases en el proyecto, y en base a ellos se fueron refinando de manera iterativa, consiguiendo así unos buenos resultados finales. Como en todo proceso de desarrollo, cuando más tiempo y recursos puedan emplearse para llevarlo a cabo, se acabará obteniendo un mejor producto final. En el caso específico de este proyecto, se establecieron unos límites representativos de hasta dónde se quería llegar, y cundo se llegó a ellos se dedicó el resto del tiempo a refinar y perfeccionar los procesos implicados, depurando errores y asegurando la robustez final del módulo.

Para la creación de la aplicación dedicada al tratamiento de los resultados para su posterior uso, se utilizó una metodología mucho más directa y sencilla, ya que la complejidad del proyecto podía reducirse a algo muy simple. De igual manera esta aplicación podría haber tenido un enfoque mucho más costoso, si tenemos en cuenta que podría usarse para el procesamiento de cantidades enormes de datos, analizando registros de miles o millones de subastas. Ya que en este proyecto no se contemplaba el análisis de tal cantidad de datos, no se usaron técnicas para la mejora del rendimiento del análisis de textos, aunque podrían tenerse en cuenta para futuras mejoras del proyecto.

Finalmente para el diseño de la aplicación de análisis del registro de subastas se utilizó una metodología que podría incluirse en la categoría RAD (Rapid Application Development), desarrollada inicialmente por James Martin en 1980, que implica el desarrollo iterativo y la construcción de prototipos de igual manera. El objetivo clave de esta metodología es el rápido desarrollo y entrega de productos de una alta calidad con un coste de inversión bajo. Este proceso hace hincapié en el cumplimiento de la necesidad comercial, y en el cumplimiento de los plazos establecidos. Con esta aplicación lo que se pretendía era obtener unos resultados claros y rápidos con un mínimo esfuerzo de implementación, lo que se ajusta a la metodología mencionada.

La metodología incremental proporciona productos altamente refinados, y combinada con el sistema de prototipado, el cual proporciona aplicaciones que permiten probar su funcionalidad en fases muy tempranas del proyecto, se obtuvo en poco tiempo una herramienta capaz de ayudar a la propia depuración del módulo de negociación en JASON, consiguiendo así un mayor rendimiento a la hora de implementarlo y mejorarlo.

El proceso típico de prototipado consiste en realizar versiones rápidas y sencillas del proyecto, las cuales no tienen que incluir necesariamente toda la lógica o características del modelo terminado. Otra de las ventajas de este modelo es que permite a los diseñadores y desarrolladores saber si se están cumpliendo las expectativas del proyecto. En la Figura 10 se puede observar de una manera gráfica como funciona este tipo de metodología.

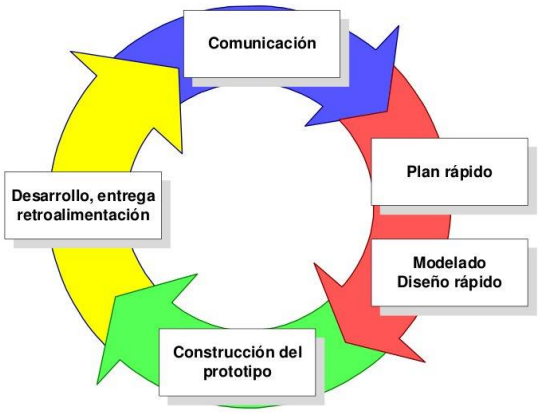


Fig. 10: Prototipado

Una vez que las dos partes del proyecto, el propio módulo en JASON y la aplicación en JAVA, comenzaron a ser mínimamente funcionales, su desarrollo se continuó de manera paralela, ya que, una vez obtenido el resultado final, deberían trabajar de manera coordinada.

Para ello se continuó con la metodología incremental anteriormente mencionada, mejorando y refinando cada vez el resultado final del trabajo conjunto de los dos sistemas desarrollados, hasta que se consideró que se habían alcanzado todos los objetivos del proyecto, y se dio por terminada la implementación del mismo en su totalidad.

En lo referente al análisis de los datos obtenidos, era necesario comprobar el correcto funcionamiento de los procesos implicados en las subastas y los resultados devueltos por ellas. Para la comprobación de los datos devueltos, no existía margen de error posible, ya que solo era necesario determinar que los valores devueltos por los agentes eran correctamente recibidos y filtrados en función del tipo de subasta seleccionada.

Para la comprobación del correcto funcionamiento de los procesos, era necesario determinar de manera empírica si las distintas subastas funcionaban de manera similar en cuanto a costes espaciales y temporales respecto a las medidas teóricas esperadas y, por tanto, validar que cada ronda de negociación tenía unas duraciones en tiempo similares, sin mostrar variaciones significativas que pudieran indicar errores en el funcionamiento. Para determinar esto se utilizaron test de hipótesis, mediantes los cuales se pudo determinar si existían diferencias significativas a la hora de utilizar distintas subastas, números de rondas o parámetros diferentes.

Este tipo de test es utilizado para determinar si las diferencias entre los resultados obtenidos al realizar un estudio son significativas, dentro del contexto del problema analizado. De esta manera se pueden extraer conclusiones respecto al correcto funcionamiento de los procesos aun cuando exista una variabilidad en las mediciones realizadas, determinando si estas variaciones son realmente significativas, o se tratan de fluctuaciones producidas por procesos aleatorios o por el azar. Si llevamos esta idea al ámbito de los procesos informáticos, y del proyecto en cuestión, podemos, utilizando este tipo de test, determinar si las subastas están funcionando de un modo correcto, aun cuando se vean afectadas por la carga del sistema o en el caso de que nos encontrásemos en un sistema MAS localizado en diferentes máquinas, donde los agentes tuviesen que realizar el paso de mensajes a través de mensajería y no mediante memoria compartida, pudiendo ser afectados por las fluctuaciones en la red utilizada.

Para la realización de estas pruebas de plantean dos hipótesis iniciales, una nula, la cual afirma que no existen diferencias significativas en las muestras, y otra alternativa, afirmando que sí se puede considerar significativa la diferencia. Una vez planteado esto, es necesario obtener valores como la proporción global que existe entre las muestras analizadas, para finalmente realizar una prueba Z, con la que se podrá validar una de las dos hipótesis, siempre teniendo en cuenta un margen de confianza de un 95%, el cual corresponde a un valor de X igual a 1.96.

Por lo tanto, si en las pruebas se obtienen resultados de Z inferiores a 1.96, aceptaremos que no existen diferencias significativas en nuestras muestras, mientras que en el caso contrario, existirían pruebas empíricas de que alguna de las subastas no presenta un funcionamiento correcto.

Para una mejor comprensión del método utilizado se incluye un ejemplo explicativo de cómo funciona el mismo:

- Nh = tamaño de la muestra subasta X
- Nm = tamaño de la muestra subasta Y

- Xh= número total de subastas X
- Xm= número total de subastas Y

- Proporción global:

  P = (Xh+Xm)/(Nh+Nm)

- Proporción de cada subasta:

  Ph = Xh/Nh

  Pm = Xm/Nm

- Planteamiento de las hipótesis:

  H0 - (hipótesis nula) = no hay diferencias entre las muestras, la diferencia observada se debe al azar.

  H1 - (hipótesis alternativa) = la diferencia es estadísticamente significativa, ambas muestras realmente son diferentes en relación a la variable estudiada.

- Calculo de la Z-Prueba:

$$Zprueba = \frac{P_h - P_m}{\sqrt{P(1-P)(\frac{1}{N_h} + \frac{1}{N_m})}}$$

Finalmente en lo referente a las comunicaciones del sistema, se realizó un estudio básico de su correcto funcionamiento, comparando los resultados obtenidos con los esperados de manera teórica.

6. Implementación del proyecto

La idea inicial es sencilla, crear un módulo adicional para JGOMAS, que pueda usarse o no a conveniencia, para que los agentes sean capaces de realizar subastas de manera independiente, pudiendo llevar un registro completo de todas las acciones realizadas, envíos de mensajes, costes espaciales, temporales, y cualquier dato extraíble para su posterior uso en análisis estadísticos y comparativos, tanto a nivel académico como para su uso en investigación.

Ya que la implementación del proyecto está pensada desde un principio para integrarla como una parte más de JGOMAS, y dado que todo el mismo se resume en un módulo con formato *.asl*, no se planteó la posibilidad de utilizar un sistema de control de versiones para ir integrando las diferentes mejoras y cambios que se efectuaban en el mismo a lo largo del tiempo. En lugar de eso se planteó la realización de distintas versiones del proyecto, para realizar copias de seguridad en cada paso y seguir trabajando a partir de ahí, enriqueciendo y mejorando las capacidades y opciones que ofrece el modulo.

La mayor dificultad del proyecto en sí recaía en superar los problemas de concurrencia que surgían al trabajar en el mismo, debido a estar limitados al control que hace JADE sobre los agentes y las variables que se manejan de manera concurrente. Al no poder usas bloqueos, barreras y demás estrategias de coordinación de manera directa en JASON se ha tenido que rediseñar varias veces la estructura de coordinación para evitar ese tipo de conflictos.

6.1 – Versión 1: Selección de líder, comunicaciones y subastas cerradas

En la primera fase del proyecto se implementó la estrategia básica de coordinación necesaria en cualquier subasta, una selección de líder, en este caso, de subastador, que será el encargado de atender la petición del solicitante o solicitantes de la subasta. En este momento se planteó una elección que complicó bastante el proyecto a posteriori, que el mismo agente que solicita la encuesta sea el encargado de realizarla o que se realice una auténtica selección de líder, el cual pueda seleccionarse en función de la carga de computo que tenga en ese momento o demás factores posibles a tener en cuenta, aumentando así la eficiencia del sistema de subastas para su posible uso en investigación. Finalmente se optó por realizar la selección de líder, el cual debe ser capaz de aceptar peticiones de subasta, computarlas y devolverlas al agente que ha realizado la petición, pudiendo realizar todo este trabajo de manera concurrente como si se tratase de un bróker o distribuidor de peticiones típico para un servicio web.

Dada la naturaleza de JGOMAS, existen 3 tipos de agentes que solo se diferencian por su función y su equipo, además de poseer un identificador único. Con esto en mente se implementó un protocolo mediante el cual, cuando un agente quiere realizar la subasta, envía una petición del tipo alcanzar un objetivo a todos los agentes de su equipo, para que lancen el plan de negociación, encargado de seleccionar al subastador, además de lanzar también su propio plan de negociación, ahorrando así un mensaje del proceso. De esta manera todos los agentes inician un proceso el cual debe terminar con un resultado idéntico para todos ellos, sin dejar margen de error posible.

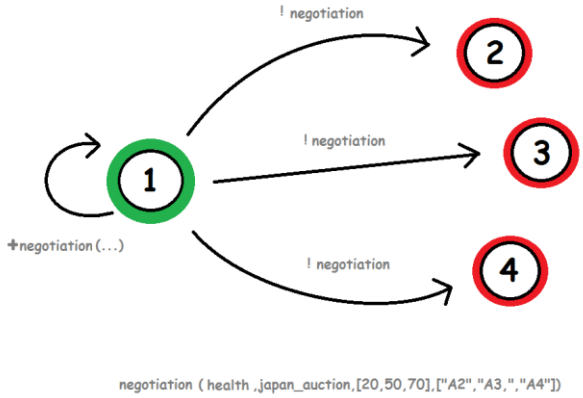


Fig. 11: Proceso negociación

Para la selección de líder se utilizaron los propios identificadores de cada agente, los cuales se obtienen realizando una petición al manager. Ya que la lista de identificadores devuelta no incluía al propio agente solicitante, era necesario incluir al mismo en la lista y a partir de ahí seleccionar a uno de ellos. En este caso de realiza una selección simple, convirtiéndolos a cadenas de caracteres y utilizando una ordenación natural de los identificadores, seleccionando siempre el mínimo de ellos. De esta manera conseguimos un resultado único siempre y, ya que en la lista de identificadores devuelta por el manager solo aparecen agentes vivos, no se seleccionará nunca un líder que ya haya sido eliminado del sistema. En este paso de la negociación se identifica también si existen agentes suficientes como para realizar la subasta, ya que, si el solicitante es el único agente vivo de su equipo, no hay opción a una posible negociación, haciendo que el plan falle y termine. Una vez que se ha realizado la selección de líder, el agente seleccionado como tal es el único que continua con el proceso de subasta, los demás agentes terminan su plan de negociación y vuelven a su comportamiento habitual, hasta que se les vuelve a solicitar.

Una vez terminada la fase anterior se pasó a implementar el plan que se ocupa del protocolo de comunicación encargado de solicitar y recibir la información de los agentes para cada subasta, es decir la encuesta. Para su primera implementación se utilizó un único valor, rápido de obtener y transmitir por los agentes, su propia cantidad de vida restante. De esta manera, seleccionando el parámetro en la petición de subasta, el líder debía ser capaz de solicitar este parámetro a todos los agentes, y recibir las respuestas de todos ellos. Basando la implementación en las comunicaciones que ya se realizan en el módulo JGOMAS, se creó rápidamente un protocolo de mensajes, los cuales el líder recibía y almacenaba en una lista. En este momento fue donde aparecieron los primeros problemas de concurrencia. Al utilizar el sistema de mensajería que ofrece JASON, si enviamos todos los mensajes a la vez e intentamos almacenarlos en un mismo hecho o variable predefinida como una lista de resultados, se producen problemas de lectura y escritura de una misma variable. En una primera implementación se optó por realizar un bucle de espera, en el que cada vez que se enviaba una petición a un agente, se esperaba su respuesta mediante un plan que se lanzaba a si mismo cada cierto tiempo, y cuando la respuesta era recibida se enviaba la petición al siguiente agente de manera secuencial. La otra alternativa que se probó para realizar el envío de mensajes concurrente fue diseñada de tal manera que el

subastador almacenaba las respuestas con el identificador del agente que respondía, y comprobaba cada vez que llegaba un mensaje, si habían llegado todos los demás mensajes de los participantes de la subasta. En este caso también se producían errores de concurrencia fácilmente detectables al comprobar que se lanzaba varias veces la respuesta mediante la cual se informaba de que todos los mensajes habían llegado, o en caso contrario no lanzándola ninguna vez. Finalmente se realizó una implementación del bucle de espera más refinada, utilizando el propio sistema de suspensión de procesos que ofrece JASON, mediante el comando *.wait*, el cual nos permite poner en espera al plan correspondiente a la espera de recibir un hecho con un valor concreto o la llegada de un hecho en particular, retomando así el plan anterior. De esta manera se estableció un protocolo de encuestas robusto y fiable, donde los mensajes se distribuyen de manera secuencial a cada agente y se espera su respuesta.

```
////////////////////////////////
// Negotiation
////////////////////////////////


+!survey(Parameter,Type,ID)
    <-
        //El valor parametro usado se transmite en la cadena de mensajes
        ?agentsList(ListAgents,ID);
        //Se inician las comunicaciones
        //Se envia el parametro preguntado a todos los agentes
        .concat("nego_ask(",Parameter,",",Type,",",ID,")",Msg);
        ?debugN(Mode); if (Mode>2) {.println("Negotiation: ID ",ID," The Leader sends requests for all"); }
        for(.member(A,ListAgents)){
            .send_msg_with_conversation_id(A, tell, Msg, "ASK");
            ?debugN(Mode); if (Mode>2) {.println("Negotiation: ID ",ID," The Leader sends ask for ",A); }
            .wait({+nego_aux(_,ID,_)});
            //si la subasta es abierta se envia lo apostado por el agente  a todos
            ?nego_aux(Val,ID,Ag);
            if(Type==english_auction | Type==deutch_auction | Type==japan_auction){
            .concat("nego_open(",Val,",",ID,",",Ag,")",Msg2);
            .send_msg_with_conversation_id(ListAgents, tell, Msg, "OPN");
            }
        }
.
```

Fig. 12: Código encuesta

Llegados a este momento se realizaron las primeras pruebas del protocolo, y se obtuvieron los primeros resultados erróneos. El agente subastador encargado de procesar las subastas no era capaz de recibir varias peticiones de manera concurrente ya que todas las subastas compartían las mismas variables y planes, dando de nuevo lugar a errores de concurrencia. Este comportamiento también fue fácilmente detectable al comprobar que cuando varios agentes lanzaban subastas de manera concurrente, solo una de ellas se procesaba, mientras que las otras fallaban. En este momento se pasó a una nueva fase del proyecto, la más costosa de todas, conseguir que el agente subastador se comportase como un bróker web con peticiones concurrentes.

En esta fase lo más costoso fue, una vez detectado el problema, saber por qué se producía, principalmente, y posteriormente darle una solución. Tras su detección y algo de investigación se llegó a la conclusión de que las distintas subastas usaban de manera concurrente los mismos hechos para almacenar y procesar los datos y que, por tanto, se producían errores de sobre-escritura. Para solucionarlo, fue necesario introducir un identificador para cada subasta, haciendo que, de esta manera cuando el líder recibiera las peticiones de cada agente, generase un código aleatorio basado en el milisegundo que llegaba la petición, multiplicada por un número aleatorio. En base a esto, los hechos almacenados en la base de hechos podían diferenciarse para cada una de las subastas.

Esta fue la primera elección de implementación, aunque más adelante se optó por utilizar los propios identificadores de los agentes que solicitaban la subasta, utilizando un contador de subastas realizadas por el propio agente y de esta manera evitando los problemas de concurrencia que se crearían si el líder mantuviese el registro de subastas realizadas en una única variable o hecho, al recibir varias peticiones de subasta de forma concurrente.

Una vez terminada esta fase y comprobando su correcto funcionamiento se pasó a elaborar los diferentes tipos de subastas que podría realizar el módulo una vez finalizado.

Para la realización de las subastas cerradas se utilizó la subasta de sobre cerrado de primer precio, la de segundo precio y la subasta a la baja. Estas 3 subastas son lanzadas por el subastador en el plan encargado de realizar la subasta y se seleccionan en base al parámetro que haya seleccionado el agente que solicita la misma. Más adelante fue necesario transmitir este valor a todos los agentes que participan en la subasta, para poder adaptar correctamente su perfil de apostador al tipo de subasta utilizado. El plan de subasta es lanzado por el agente subastador una vez terminado el plan de encuestas, utilizando de nuevo los recursos que JADE y JASON nos ofrece, y consiguiendo así que la subasta nunca empiece hasta que se hayan obtenido todas las respuestas de los agentes participantes. Una vez seleccionado el tipo de subasta, se realiza un filtrado de los datos de respuesta, obteniendo así el ganador de la subasta, y el valor correspondiente a su apuesta. Cuando tenemos estos datos, es necesario enviárselos al propio solicitante de la encuesta, para que pueda usarlo acorde a su propia implementación. De esta manera se completa el ciclo para la realización de una subasta de una sola ronda, y tras realizar las pertinentes comprobaciones de su correcto funcionamiento se pasó a implementar la siguiente fase, donde se dota al módulo de la capacidad de realizar varias rondas de subastas.

6.2 – Versión 2: Subastas abiertas con rondas, japonesa, inglesa y alemana

La implementación de las rondas de subasta era en principio un proceso sencillo, que finalmente resultó más costoso de lo esperado, debido a errores en el procesamiento de los datos por parte del subastador, cuando se introducían varias rondas de negociación en el proceso. Este nuevo concepto fue necesario para determinar la ronda de negociación en la que se encontraba la subasta, y así utilizar los hechos correspondientes a ella para el procesamiento de los datos. Este identificador de ronda también debía ser enviado en el protocolo de encuesta, incrementando así el tamaño de los mensajes enviados y complicando el proceso. Tras varios intentos fallidos, se optó por una implementación más sencilla, donde el propio agente solicitante de la subasta se encarga de hacer el control de las rondas de subasta, enviando al subastador un identificador diferente en base a su contador de subastas realizadas, su nombre y la ronda en que se encuentra la subasta. De esta manera para el subastador todas las subastas realizadas son diferentes, y el propio agente solicitante es el encargado de realizar las comparaciones pertinentes y el filtrado de datos necesario mediante el cual se decide cual es el agente ganador de todas las rondas de las que consta la subasta.

En esta fase se introdujeron más parámetros seleccionables por el agente solicitante de la encuesta, tales como la munición y la distancia al objetivo. Ya que la distancia al objetivo de los agentes era el parámetro más variable dentro de las diferentes partidas de prueba fue el que se usó para la mayoría de pruebas realizadas para comprobar el correcto funcionamiento del filtrado de datos y selección de ganadores y valores ganadores.

Para la implementación de las subastas abiertas, inglesa, alemana y japonesa, se modificó el comportamiento del agente solicitante de la encuesta, y del subastador. Esto fue debido a la implementación de la subasta japonesa, donde en cada ronda es necesario filtrar los participantes que podrán apostar en la siguiente ronda de negociación, en base a un límite fijado en la misma ronda. Estos valores son fijados por el propio solicitante, y deben ser transmitidos al subastador, aumentando la cantidad de información de los mensajes en el proceso. De esta manera se consigue que el subastador sea capaz de realizar las diferentes rondas de negociación de la subasta únicamente con los agentes que han devuelto un valor dentro de los límites fijados con anterioridad.

En la implementación de este proceso se incluyó una nueva funcionalidad en el módulo. Dado que era necesario enviar la lista de los participantes en cada ronda de negociación, se amplió el diseño original, incluyendo la lista de participantes de la subasta como un nuevo parámetro seleccionable a la hora de lanzar la encuesta. De esta manera puede realizarse una selección inicial de los agentes participantes seleccionados por el usuario para que intervengan en el proceso de subastas.

En este tipo de subasta es donde se hace más patente el hecho de que, los agentes participantes en ella que tengan unos valores idénticos o muy similares entre sí, siempre participaran en las mismas rondas de negociación, obteniéndose siempre unas resultados muy similares en todas las subastas realizadas, haciendo que el proceso sea más parecido a un sistema de mensajería o paso de mensajes, que una subasta propiamente dicha. Con esto en mente se desarrolló el concepto de perfiles de agente, mediante los cuales se confiere a cada agente un perfil de apostador, que influye en su respuesta y que fueron implementados en la tercera versión del proyecto.

6.3 – Versión 3: Perfiles de apostador y registro de datos

Como ya se ha comentado, para dotar a cada agente de un perfil diferente de apostador que influya a la hora de devolver su respuesta en la subasta, fue necesario desarrollar un nuevo plan mediante el cual, solamente una vez al inicio de cada partida, todos los agentes participantes en la misma seleccionen de manera aleatoria un perfil de apostador, acompañado de un valor numérico decimal entre 0 y 1, equivalentes al porcentaje con el que cada agente al recibir un mensaje solicitándole un valor para una subasta, modificaría el valor de respuesta, influenciando así las respuestas y los resultados de las subastas realizadas acorde a su perfil de apostador.

```
//////////////////////
/// Agent Profile ////
//////////////////////


+!agentProfile
<-

    //Seleccion aleatoria segundo intento
    .random(X);
    if(X>=0.6){
        if(X>=0.8){+profType("Reckless");+profValue(1);}
        else{+profType("Sincere");+profValue(0.8);}
    }else{
        if(X>=0.4){+profType("Bidder");+profValue(0.6);}
        else{
            if(X>=0.2){+profType("Safe");+profValue(0.4);}
            else{+profType("Coward");+profValue(0.2);}
            }

        }
    ?profType(Profile);
    ?profValue(Value);
    ?debugN(Mode); if (Mode>0) {.println("Negotiation: My Profile is ",Profile," with the value ",Value); }
    //refrescas tu perfil
    !!perfiles;
.
```

Fig. 13: Plan Perfil

En un primer momento se introdujeron 5 perfiles diferentes de jugador, con unos valores comprendidos entre 0.2 y 1, lo que definirían un desde un perfil cobarde, donde el apostador ofrece mucha menos cantidad de la que realmente posee, y un perfil valiente, o arriesgado, donde el agente apuesta todo lo que posee en el momento de la subasta. Un detalle de implementación que se tuvo que tener en cuenta en este momento fue que en las subastas básicas de sobre cerrado de primer y segundo precio, los agentes con perfiles cobardes seguirían el patrón fijado y apostarían mucho menos de lo que poseen, mientras que en las subastas a la baja, los agentes seguirían apostando un valor muy inferior, dando así un comportamiento cobarde para esa subasta. Igualmente los agentes con perfiles valientes actuarían de manera contraria a su comportamiento habitual en las subastas a la baja, actuando de forma cobarde. Por lo tanto fue necesario transmitir también a todos los agentes participantes en la subasta el parámetro que indicaba en qué tipo de subasta básica se estaba participando y modificar así su respuesta en función del perfil de apostador y el tipo de subasta que se estaba solicitando.

Una vez implementados los perfiles básicos de agente y comprobado su correcto funcionamiento en las subastas, se pasó a implementar el sistema, mediante el cual, los agentes serían capaces de modificar su propio perfil, en función de las subastas que hubiesen ganado o aquellas que hubieran perdido, incrementando o disminuyendo de esta manera el valor representativo asociado a su perfil de apostador.

Para ello se decidió que el propio agente solicitante de la subasta y por tanto, el único destinatario de la respuesta fuera el encargado de transmitir el mensaje, con el ganador y el identificador de la subasta, a todos los demás agentes. Lo que en principio era un paso de mensajes sencillo, volvió a presentar problemas de concurrencia, debido a que si un agente recibe varios mensajes para la modificación de su perfil de apostador de manera concurrente, el hecho que almacena este perfil sufrirá errores de sobre escritura, y por lo tanto haciendo que todo el proceso deje de ser fiable y robusto. Para evitar este problema, era necesario encontrar un momento del proceso de negociación donde no hubiese problemas de concurrencia. En un primer momento se pensó en actualizar los perfiles de apostador de cada agente en el momento que solicita una subasta, pero de esta manera un agente que hubiera ganado o perdido varias subastas, pero nunca solicitase ninguna, no actualizaría su perfil de jugador. Así que finalmente, se optó por crear un plan el cual es lanzado al finalizar la selección de perfil del agente, y que de manera recurrente, tras un cierto tiempo de espera definido, se vuelve a lanzar a sí mismo. Este plan, encargado de procesar los hechos que indican si el agente ha ganado o ha perdido una subasta concreta definida por su identificador, permanece activo durante toda la partida mientras el agente esté vivo, modificando así constantemente su perfil. Finalmente esta idea también se desechó, y se sustituyó el tiempo de espera por una suspensión del proceso hasta que se reciba un mensaje de negociación, el cual reciben todos los agentes del equipo en cada ronda de negociación, pudiendo actualizar así los perfiles de forma correcta.

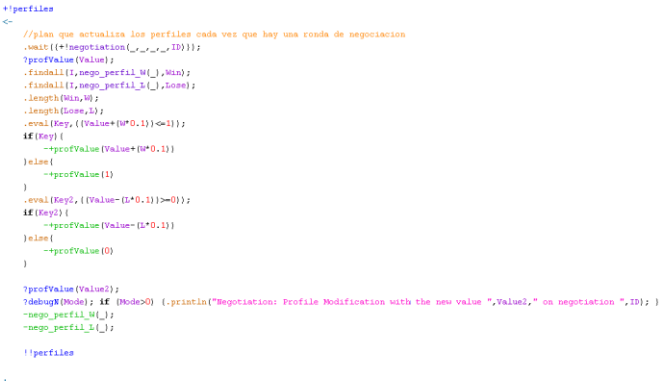


Fig. 14: Plan Actualizar Perfil

Tras la comprobación del correcto funcionamiento de esta fase, se pasó a implementar el log, mediante el cual se podrían extraer los resultados de las subastas realizadas, mensajes enviados y demás información que pudiese ser utilizada posteriormente para su análisis. Aunque durante todo el proceso de desarrollo ya se habían ido introduciendo mensajes con la información necesaria para comprobar el correcto funcionamiento del módulo, fue necesaria una fase propia para su desarrollo y correcto funcionamiento.

Debido a la imposibilidad de escribir ficheros directamente desde JASON y aprovechando que en la propia implementación de JGOMAS ya existe la creación de un fichero de texto el cual almacena todos los mensajes que el programa imprime por la salida estandar, se pensó en la posibilidad de crear una pequeña aplicación la cual fuera capaz de mostrar los datos de las subastas de manera rápida y clara, analizando y procesando el propio fichero de log de JGOMAS, y facilitando así el posterior análisis de los datos. Para la implementación de esta pequeña aplicación, se utilizó el lenguaje de programación JAVA, tanto para el apartado grafico como el de análisis de los datos. De esta manera se obtuvo de forma rápida una herramienta más, que facilitaría mucho el análisis de los datos a primera vista para su uso de forma didáctica, y que paralelamente podría generar datos estadísticos en formato de fichero con la información necesaria para su análisis más exhaustivo. Para el desarrollo de la aplicación fue necesario trabajar teniendo en cuenta solo los mensajes que pertenecían exclusivamente al apartado de negociación y por tanto diferenciándolos del resto de mensajes habituales del sistema. Ya que la aplicación está pensada para su uso conjunto a JGOMAS no se le ha dotado de flexibilidad para poder analizar otros archivos u otros tipos de registro que no sean el asociado a la negociación de JGOMAS. Finalmente ya que las negociaciones y los mensajes ocurren de manera concurrente fue necesario realizar una ordenación de los mismos, para una mayor comprensión a nivel didáctico y para facilitar su tratamiento y análisis a posteriori.

7. Interfaz

7.1 – Módulo de subastas

Debido a que la implementación del módulo adicional para subastas se resume en la utilización de un único plan por parte del usuario, el cual puede ser utilizado en cualquiera de las situaciones en las que un agente se encuentre dentro de la partida, habiéndolo incluido previamente en el código del propio agente utilizado, es necesario exponer los distintos parámetros y funciones que puede ofrecer, para una correcta utilización de la herramienta.

Por otra parte, ya que el modulo está abierto a futuras mejoras o implementaciones, es necesario incluir una descripción de la estructura y planes relevantes mediantes los cuales se desarrolla el proceso completo de subastas.

7.1.1 – Estructura del módulo

El módulo de subastas está dividido básicamente en 3 fases. Cada vez que un agente lanza el plan mediante el cual solicita la subasta, se realizan estas fases de manera sucesiva, siempre en el mismo orden.

- Negociación: Cuando el agente solicita una subasta, se inicia el proceso de negociación mediante el cual se selecciona un líder, o subastador, que será el encargado de realizar esa subasta.
- Encuesta: Una vez seleccionado el líder, este se encargará de solicitar la información a todos los agentes participantes, y de recibir sus contestaciones. Para ello será necesario administrar cada subasta mediante un identificador único, que será proporcionado por el solicitante.
- Subasta: Esta fase se encarga de realizar el filtrado de los datos recibidos por los agentes, en base a la subasta seleccionada y los parámetros asignados a la misma. El subastador será el encargado de realizarla, y de enviar la contestación al agente solicitante.

La fase de negociación engloba a las otras dos, y no finaliza hasta que terminan. De igual manera, las fases de encuesta y subastas están relacionadas entre sí, de manera que la fase de subasta o filtrado de datos no comienza hasta que se da por terminada por completo la fase de encuestas, y se han recibido todos los datos pertenecientes a esa subasta.

En lo perteneciente al apartado de los perfiles de apostador, existe una relación directa con las fases de subastas, ya que al finalizar cualquiera de ellas, se envía la información del ganador a todos los participantes, lo que genera unos hechos que indican la cantidad de subastas que ese agente ha ganado o perdido. Cada vez que se realiza una fase negociación, los agentes actualizan sus perfiles en función de esa información, evitando así posibles errores de concurrencia.

A continuación se presentan las distintas comunicaciones, planes, y objetivos implementados y sus principales funciones, además de las relaciones que presentan entre ellos. Este documento sirve de apoyo para la comprensión del código debidamente comentado ya.

Creencias

leader("none").

Creencia que indica al agente quien es el líder o subastador y que le permite saber si es el que tiene que realizar el proceso de subastas.

debugN(3).

Creencia que indica la cantidad información que queremos que el agente muestre por la salida estándar, su rango de valores va del 0 al 3.

id(1).

Identificador propio del número de subasta solicitada por el agente en la partida.

Para evitar la carga del sistema, se crean una gran cantidad de hechos auxiliares que son utilizados a lo largo de la negociación y subasta, que finalmente son destruidos cuando termina el proceso, por lo tanto no se incluyen en esta documentación.

Planes

!survey_request:

El plan mediante el cual el usuario interacciona con el modulo, es el encargado de realizar las distintas rondas de subasta y de filtrar los parámetros para iniciar los diferentes tipos de negociación. Al hacer de interfaz con el usuario será ampliada su información más adelante.

!negotiation:

Plan que es lanzado por todos los agentes cada vez que se realiza una subasta, o ronda de subasta, engloba los planes !refresh, !survey y !auction. Estos dos últimos solo son ejecutados por el subastador, previamente seleccionado en !refresh.

!survey:

Este plan es lanzado dentro de !negotiation, y es el encargado de realizar el envío y la recepción de los mensajes que se generan entre los participantes y el subastador . El envío se hace de manera secuencial, evitando así problemas de concurrencia en la recepción. Únicamente el subastador realiza este plan.

!auction:

Plan realizado por el subastador únicamente, encargado del filtrado de los datos en función de la subasta que se haya seleccionado. Se lanza de manera secuencial respecto al plan anterior y está englobado dentro del plan !negotiation.

¡refresh:

Plan encargado de la selección de líder propiamente dicha. Los agente sutilizando la lista de participantes de su equipo para su selección. Este plan es el que debería modificarse para incluir mejoras en la selección de líder en función de la carga, u otros factores que se requieran.

!rounds:

Plan encargado de realizar el filtrado de datos de las distintas rondas de negociación dentro de una misma subasta. Se encuentra dentro de !survey_request, y es realizado por el solicitante de la encuesta.

!profiles:

Plan lanzado una única vez por todos los agentes participantes en la partida, independientemente del bando. En este plan se realiza la selección del perfil de apostador de una manera aleatoria, siendo fácilmente ampliable o modificable para una mejora de los perfiles en un futuro.

!agentProfile:

Plan que se lanza cada vez que un agente recibe una solicitud de negociación, y que se mantiene en espera para volver a lanzarse a sí mismo al volver a recibir otra solicitud. Es el encargado de realizar la actualización de los perfiles, en función de la información que tenga el agente de sus victorias o derrotas apostando.

Comunicaciones

Siguiendo el estilo usado en JGOMAS, las comunicaciones se realizan mediante la adquisición de nuevos conocimientos por el agente. Estas acciones se realizan agregar un hecho en su base de conocimientos, ya sea mediante la percepción o por mensajes de otros agentes.

+nego_ask:

Acción que realizan los agentes participantes en una subasta, cuando reciben un mensaje solicitando el valor que van a apostar. Esta acción genera un mensaje de respuesta para el subastador que ha realizado la solicitud.

+nego_resp:

Acción que únicamente realiza el subastador, y que se activa cuando recibe un mensaje de respuesta por parte de alguno de los participantes de la subasta que modera. Es utilizado por el plan encargado de las encuesta como punto de espera con el que el subastador sabe si ha recibido una respuesta y puede enviar una pregunta al siguiente participante.

+nego_win:

Acción que se activa cuando el agente recibe un mensaje con el ganador de la subasta y el valor asociado a ella, y que almacena los resultados recibidos para su posterior uso por el agente solicitante. Esta acción se utiliza como condición para finalizar el plan mediante el cual un agente solicita una subasta.

+nego_jap:

Acción utilizada para el trasiego de información perteneciente a las subastas japonesas y el filtrado de agentes participantes en ella.

+nego_profile:

Acción encargada de almacenar la información de las victorias o derrotas del agente en las distintas subastas, y que posteriormente es utilizada para el refresco de los perfiles de apostador.

+nego_open:

Acción encargada de tratar la información recibida de los valores apostados por los participantes en cada ronda de negociación. Este plan está abierto a futuras implementación para variar la apuesta del agente en función del resto de apuestas.

Interfaz

Listado de los parámetros posibles que puede aceptar la solicitud de subasta:

- ammo = solicitud de la cantidad de munición del agente.
- health = solicitud de la cantidad de vida del agente.
- dist_objective = solicitud de la distancia al objetivo del agente.

Listado de las subastas posibles que puede aceptar la solicitud de subasta:

Cerradas:

- closed_card1 = subasta de sobre cerrado primer precio.
- closed_card2 = subasta de sobre cerrado segundo precio.
- downward_auction = subasta de sobre cerrado a la baja

Abiertas:

- Subasta Japonesa: incluyendo una lista de valores ascendentes:

  !survey_request(ammo, japan_auction,[30,60,100]).

- Subasta Holandesa: incluyendo una lista de valores descendentes:

  !survey_request(ammo, closed_card1,[100,80,10]).

- Subasta Inglesa: incluyendo una lista de valores ascendentes:

  !survey_request(ammo, english_auction,[30,60,100]).

Modos:

- Incluyendo una lista de agentes participantes:

  !survey_request(health, closed_card1,[100,80,10],[“A1”,”A2”,”A3”]).

- Todos los participantes del equipo:

  !survey_request(ammo, closed_card1,[100,80,10]).

- Incluyendo número de rondas deseadas para subastas cerradas:

  !survey_request(ammo, closed_card1,100, [“A1”,”A2”,”A3”])).

Un factor a tener en cuenta es que no puede seleccionar una subasta japonesa, inglesa o alemana al mismo tiempo que se selecciona el número de rondas, ya que por defecto, se toma como número de rondas la cantidad de valores incluidos en la lista de filtros. Por otra parte, la futura ampliación de la funcionalidad puede modificar el comportamiento del interfaz o su significado.

7.2 –Aplicación para el análisis de subastas

Aunque la herramienta incluida para realizar el análisis de los datos generados por el módulo de subastas es bastante simple, se incluye un pequeño esquema de su funcionamiento para facilitar la comprensión y posterior uso de la misma.

La aplicación es capaz de generar informes de datos completos de todas las subastas, pero para ello, es importante recordar que, para poder obtener esta información generada por el módulo, es necesario modificar el valor de la creencia debugN, aumentándolo o disminuyéndolo en función de la cantidad de información que se desee obtener.

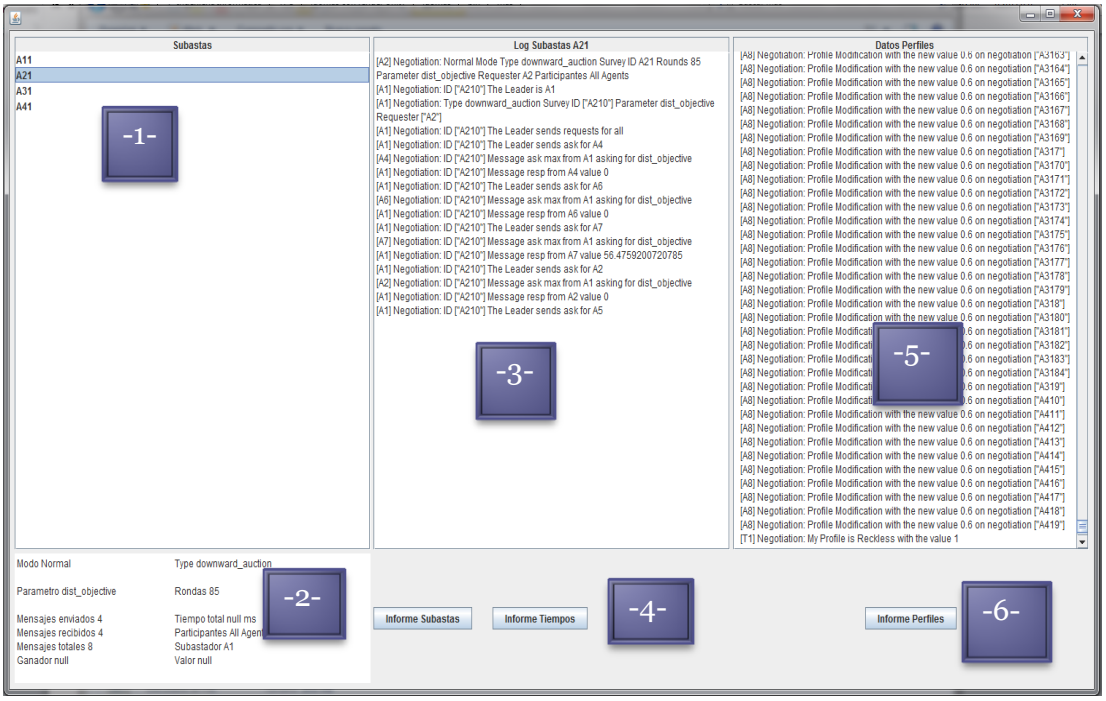


Fig. 15: NegoLog.java

1. Panel de Subastas: En él se presentan todas las subastas realizadas en la partida analizada. Cada una de ellas está representada por su identificador. Cuando se selecciona una de estas subastas, se presenta su información en el resto de campos.
2. Panel de información: En esta sección aparece la información más relevante de la subasta seleccionada, su tipo, número de rondas, numero de mensajes intercambiados, etc.
3. Panel de registro: En este panel se presenta toda la información perteneciente a la subasta seleccionada, de manera ordenada. Se pueden observar las rondas de negociación paso a paso, los resultados de cada ronda, el filtrado de datos o los agentes participantes,
4. Generación de informes de subasta: Estos dos botones se encargan de la generación de archivos útiles para su análisis posterior. Pueden generarse informes de todas las subastas, y de manera independiente generar otro registro únicamente con los tiempos de todas las subastas.
5. En esta sección se presenta de manera ordenada la evolución de los perfiles de subastador de cada uno de los participantes de las subastas realizadas en la partida analizada. Esta información no se modifica en función de la subasta seleccionada en el panel de subastas, ya que es global a toda la partida.
6. Para finalizar, existe también un generador de informes exclusivamente para los perfiles de apostar, independizándolos así de los registros de las subastas.

8. Validación

8.1 – Subastas cerradas

Para la validación del módulo de subastas se han realizado diversas pruebas en función del coste temporal asociado a cada una de las distintas variaciones de subastas de tipo cerrado que se pueden utilizar. Para ello se ha utilizado la herramienta JGOMAS como plataforma de evaluación para realizar las pruebas pertinentes y extraer los datos asociados a ellas.

Es relevante decir que los tiempos obtenidos están condicionados a las respuestas que los agentes proporcionan en la ronda de encuestas, las cuales, tienen implementadas una función de espera, haciendo que estos retrasen la respuesta de manera aleatoria, en un intervalo comprendido entre 0 y 1 segundos. Esta variación permite que las respuestas a las subastas tengan una cierta variabilidad, que no se obtendrían sin ella. Por otra parte, esta función introduce un retraso importante a la hora de realizar las subastas, donde el tiempo en el que se completa la subasta pasa de unos pocos milisegundos, a varios segundos. Aun así, ya que dentro del entorno JGOMAS se obtienen mejores resultados usando esta técnica, las mediciones realizadas y los resultados obtenidos en el análisis del módulo tienen en cuenta este factor.

Para este análisis se han realizado 85 subastas, utilizando 4 agentes diferentes realizando solicitudes de subastas, con un total de 8 participantes del mismo equipo, dentro de una partida estándar. Es importante señalar que los 4 agente solicitantes también participan en las negociaciones, y que de los 8, uno de ellos es seleccionado como líder, recayendo en él el mayor porcentaje de carga computacional. En cada experimento se ha utilizado uno de los 3 tipos básicos de subasta, pudiendo así analizar si existen diferencias significativas entre ellos en sus tiempos de ejecución. Debido al alto coste computacional del proceso, y al tiempo requerido para realizar las pruebas, se han tenido que limitar las subastas a esta cantidad.

La primera subasta analizada es la variante de sobre cerrado de primer precio con 1 sola ronda. En esta subasta el análisis de los datos se limita a la búsqueda del valor máximo dentro de la lista de resultados obtenidos, de un tamaño igual al número de agentes participantes. Este hecho implica que el coste asintótico temporal del proceso tenga una relación directamente lineal con el número de agentes participantes en la subasta. Los resultados obtenidos se muestran de manera gráfica representados mediante un histograma en la Figura 16.

Como se puede observar la mayor parte de los datos se concentran en la franja entre los 4000 ms y los 4500 ms, un hecho razonable dado que si tenemos 8 participantes, que tardan medio segundo de media en contestar, la mayoría de las subastas tardarán aproximadamente 4000 ms. Por otra parte, hay que tener en cuenta los costes asociados al tratamiento de los datos, al coste de las comunicaciones y a la carga computacional que en ese momento tengan los agentes, principalmente el agente que en ese momento tenga asociado el rol de subastador.

Para el segundo experimento se han mantenido las mismas condiciones anteriores, variando únicamente el tipo de subasta empleada. En este caso la subasta que se utiliza es la modalidad de sobre cerrado de segundo precio. Aunque es prácticamente idéntica a la anterior, su coste computacional debería ser algo mayor, debido a que el filtrado de los datos requiere alguna comprobación adicional al modelo anterior. Aunque a nivel teórico este debería ser al resultado

esperado, el análisis empírico de los datos no revela grandes diferencias respecto al tipo de subasta anterior. En la Figura 17 es posible ver una representación gráfica de los resultados obtenidos en forma de histograma.

ms	*Frecuencia*
0	0
1000	0
1500	0
2000	0
2500	3
3000	4
3500	11
4000	26
4500	15
5000	16
5500	5
6000	4
y mayor...	0

Tabla 1: cc1

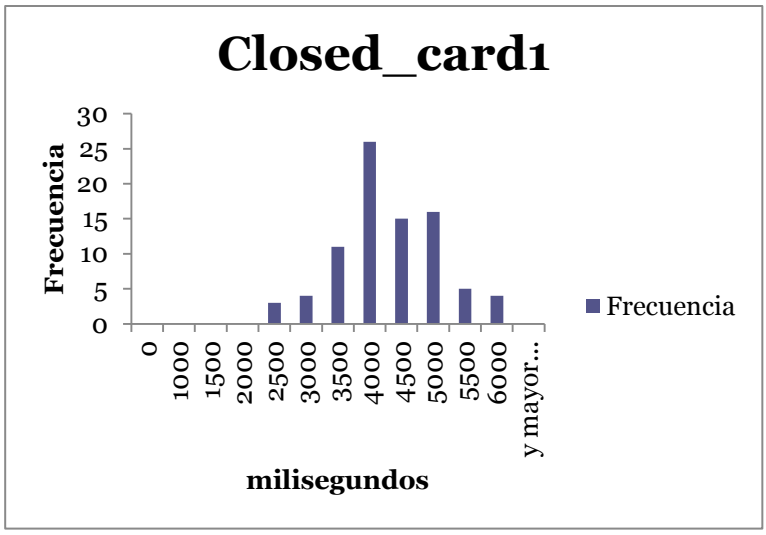


Fig. 16: Histograma cc1

ms	*Frecuencia*		
1500	0	4000	26
2000	0	4500	16
2500	3	5000	10
3000	9	5500	9
3500	7	6000	4
		y mayor...	0

Tabla 2: cc2

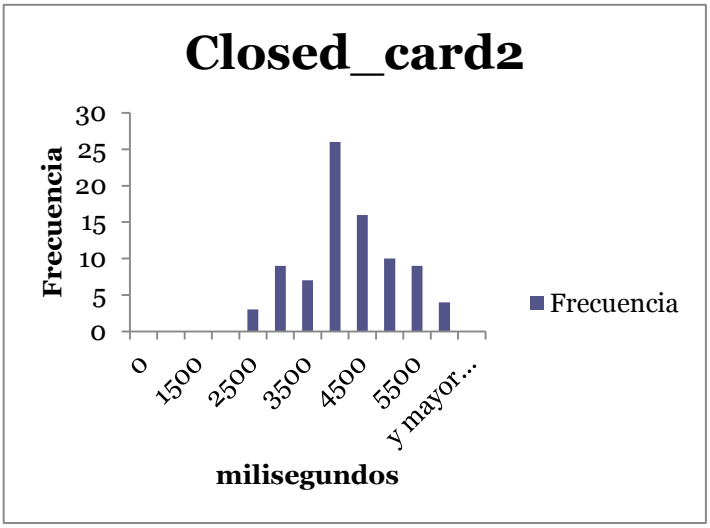


Fig. 17: Histograma cc2

Para el último tipo de subasta básica analizada, la subasta a la baja, se repiten de igual manera a las anteriores las condiciones en las que se desarrolla el experimento, variando únicamente el tipo de subasta. En este caso la subasta a la baja requiere el mismo análisis de los datos que la subasta de sobre cerrado de primer precio, solo que en esta variante se busca el valor mínimo de la lista de resultados. Esto implica una relación lineal en el coste asintótico de la subasta con respecto al número de participantes de la misma. En la Figura 18 puede observarse de manera gráfica los resultados obtenidos.

De igual manera que en los anteriores experimentos, la mayor parte de los datos se concentran en la franja entre los 4000 ms y los 4500 ms. Aunque en este caso se aprecia claramente la misma cantidad de resultados en la franja entre los 4500 ms y los 5000 ms, lo que indica que podemos tomar los 4500 ms como valor medio de los tiempos de cada subasta. Un valor razonable teniendo en cuenta el tiempo medio de respuesta comentado anteriormente, más el incremento derivado del tratamiento de los datos, el envío de mensajes, y la carga de los agentes en ese momento.

Si tenemos en cuenta las sumas totales de las 85 subastas realizadas, obtenemos unos valores promedio que oscilan entre 33000 ms y los 34000 ms en las 3 variantes posibles. Estos datos son útiles para la comprobación del buen funcionamiento del método usado para la realización de distintas rondas dentro de una misma subasta.

ms	*Frecuencia*
0	0
1000	0
1500	0
2000	0
2500	3
3000	7
3500	13
4000	18
4500	18
5000	13
5500	7
6000	5
y mayor...	0

Tabla 3: da

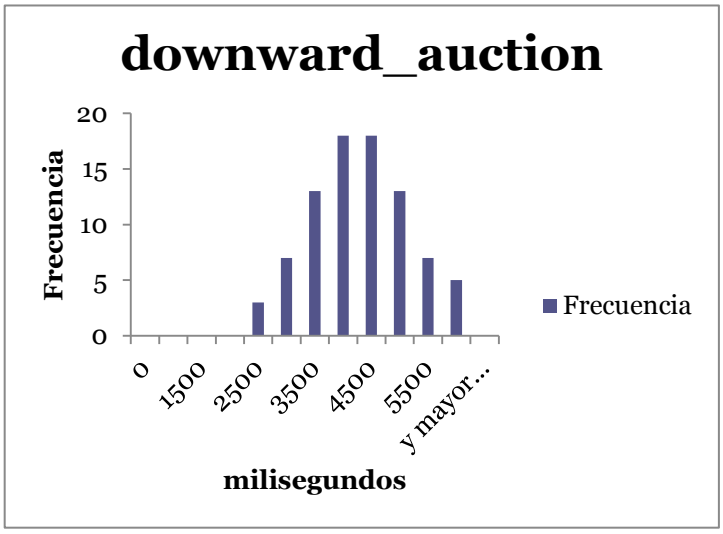


Fig. 18: Histograma da

Para ello, se realizaron 2 subastas de sobre cerrado, utilizando 85 rondas de negociación, las cuales, dada la implementación, tienen un coste asintótico muy similar a la realización de 85 subastas, siendo la única diferencia sustancial el hecho de no realizar la fase de selección de líder en cada una de las rondas, lo cual debería aligerar el coste computacional del proceso, al reducirse el número de comunicaciones realizadas, y el coste de inicializar cada vez todo el proceso completo de subastas.

Los resultados obtenidos en este caso fueron de 34171 ms y 35618 ms, en el primer caso siendo más rápido que la suma de las 85 subastas analizadas anteriormente, en el caso de la subasta de sobre cerrado de primer precio, y en el segundo caso se obtuvo un resultado más lento, lo que nos indica que a priori, no existe una diferencia significativa en el coste de realizar las subasta de forma individual, con respecto a utilizar el sistema de rondas.

Si realizamos un test de hipótesis para comprobar si existe una diferencia significativa entre los resultados obtenidos, utilizando las distintas subastas cerradas, podemos validar el correcto funcionamiento de las mismas de manera fiable.

Para ello se han realizado tres experimentos diferentes utilizando los resultados obtenidos en las pruebas anteriores, y mediante el uso de las Z-pruebas mencionadas en el apartado de metodología, incluido anteriormente en este documento, se han obtenido los siguientes resultados. Antes de realizar las pruebas, ha sido necesario extraer el valor real de la varianza de los datos para cada uno de los tipos de subastas utilizados.

Closed_card1 y Closed_card2:

Prueba z para medias de dos muestras		
	Variable 1	*Variable 2*
Media	4065.55952	4027.10714
Varianza (conocida)	799.7794	822.7745
Observaciones	84	84
Diferencia hipotética de las medias	0	
z	0.08749099	
P(Z<=z) una cola	0.46514062	
Valor crítico de z (una cola)	1.64485363	
Valor crítico de z (dos colas)	0.93028125	
Valor crítico de z (dos colas)	1.95996398	

Tabla 4 : cc1-cc2

A la vista de los resultados obtenidos, podemos afirmar con un margen de confianza del 95% que no existen diferencias significativas entre las medias de los resultados obtenidos anteriormente. Este hecho viene indicado por el valor obtenido de Z, el cual debe ser inferior a 1.96 para aceptar la hipótesis nula, la cual afirma que no existen diferencias significativas en las medias de los datos muestreados.

Closed_card1 y downward_auction

En este caso se repite el mismo procedimiento utilizado anteriormente, con los resultados de las subastas de tipo sobre cerrado de primer precio y subasta a la baja. A la vista de los resultados podemos concluir que en este caso tampoco existe una diferencia significativa en las medias de los dos tipos de subastas, dentro de un 95% de margen de error, con respecto a los resultados obtenidos en la experimentación ya que los valores de Z son inferiores a 1.96, lo cual consolida la hipótesis nula.

Prueba z para medias de dos muestras		
	Variable 1	*Variable 2*
Media	4065.55952	4022.5119
Varianza (conocida)	799.7794	822.7745
Observaciones	84	84
Diferencia hipotética de las medias	0	
z	0.13950229	
P(Z<=z) una cola	0.44452662	

Valor crítico de z (una cola)	1.64485363	
Valor crítico de z (dos colas)	0.88905325	
Valor crítico de z (dos colas)	1.95996398	

Tabla 5 : cc1-da

Closed_card2 y downward_auction:

La última de las pruebas realizadas relaciona la subasta de sobre cerrado de segundo precio con la subasta a la baja. Igual que en los casos anteriores los datos usados son los obtenidos el experimento anterior, referentes a 85 subastas diferentes. Como en los casos anteriores, comprobamos que no existe una diferencia significativa en las medias de los resultados, obteniendo un valor de Z inferior a 1.96, lo que corrobora la hipótesis nula la cual indica que no existe una diferencia significativa en las medias de los resultados obtenidos, lo que implica un comportamiento similar entre las subastas.

Prueba z para medias de dos muestras		
	Variable 1	*Variable 2*
Media	4022.5119	4027.10714
Varianza (conocida)	822.7745	868.9604
Observaciones	84	84
Diferencia hipotética de las medias	0	
z	-0.01023957	
P(Z<=z) una cola	0.49591508	
Valor crítico de z (una cola)	1.64485363	
Valor crítico de z (dos colas)	0.99183015	
Valor crítico de z (dos colas)	1.95996398	

Tabla 6 : cc2-da

Conclusiones de la validación de subastas cerradas

A la vista de los resultados obtenidos, podemos afirmar que las distintas subastas de tipo cerrado funcionan correctamente y con unos costes computacionales que no presentan diferencias significativas. Todo ello realizado en un tiempo acorde a lo esperado de manera teórica.

8.2 – Subastas abiertas

Para la validación de las subastas de tipo abierto con varias rondas, se utilizará un ejemplo ilustrativo del proceso asociado al uso de una de ellas dentro del entorno JGOMAS, en concreto

la subasta japonesa, donde los participantes son filtrados en cada ronda sino son capaces de alcanzar el valor asociado a ella.

Para ello, se utilizará una situación habitual en las partidas de JGOMAS, la petición de botiquines a uno de los médicos del equipo, y se verá cómo puede el sistema de subastas mejorar está funcionalidad, obteniendo así mejores resultados dentro de la partida.

Esta situación se produce cuando uno de los agentes participantes llega a un umbral de vida establecido, y envía una petición a los médicos de su equipo para recibir ayuda. Por defecto, esta petición se envía a todos los médicos del equipo que aún están vivos, los cuales, van a ayudar sin tomar ninguna decisión respecto a cuál de ellos es el más indicado para atenderla.

Para ello se lanzará una subasta japonesa, utilizando el parámetro de vida restante del agente, dentro del plan correspondiente a la respuesta de los médicos a la petición. Esto desencadena el proceso de subastas, el cual siempre comienza con una selección de líder. En la Figura 19 se puede observar un ejemplo gráfico del funcionamiento de este proceso.

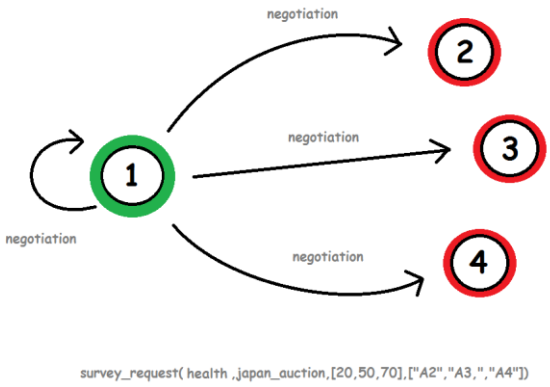


Fig. 19: Proceso Negociación

Teniendo en cuenta que en la partida solo quedasen los 4 agentes numerados, y el solicitante de la subasta fuera el agente 1, se enviaría un mensaje de negociación a todos ellos, incluyendo al propio solicitante. Dada la implementación actual, el agente solicitante sería seleccionado como subastador para esta encuesta, y se encargaría de administrar el proceso completo.

Es importante remarcar que, aunque los participantes que se han seleccionado para la subasta son los agentes 2, 3 y 4, todos los agentes vivos del equipo participan en el proceso de selección de subastador. Una vez seleccionado el subastador, comienza el proceso de encuesta, donde se realiza la ronda de apuestas.

En este proceso es donde intervienen los perfiles de apostador de cada agente. Para este ejemplo, cada médico tendrá un perfil diferente, desde el más arriesgado, que apuesta el valor total del parámetro solicitado, pasando por uno seguro que solo apuesta la mitad y otro cobarde, el cual apuesta un valor muy por debajo del real.

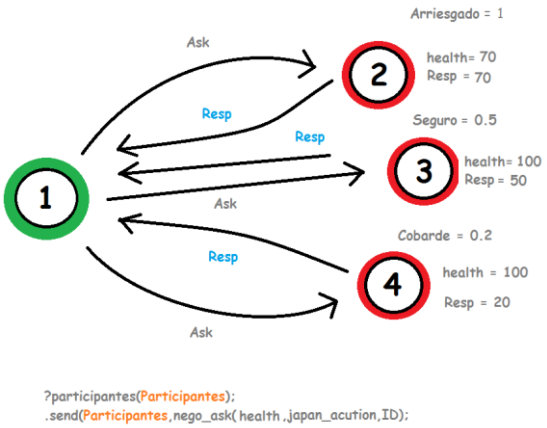


Fig. 20: Proceso Encuesta

Se puede observar en la Figura 20 que en este ejemplo los más indicados para asistir al agente solicitante son los médicos 3 y 4, pero debido a sus perfiles, apuestan con un valor inferior al que realmente tienen. Este sistema de perfiles podría relacionarse fácilmente con un sistema de emociones, dando una mayor variabilidad y realismo al juego.

Es necesario remarcar que, debido a que la modalidad de subasta seleccionada es de tipo abierta, cada vez que el subastador recibe una respuesta con el valor apostado, la retransmite al resto de participantes. De esta manera, todos los agentes conocen el valor apostado por el resto de participantes en cada ronda.

Llegados a ese punto el agente 1 se encarga del filtrado de datos en función de la subasta seleccionada. Como se trata de una subasta japonesa, se utiliza el primer valor de la lista de filtros indicada por el usuario para comprobar que agentes alcanzan o superan el valor establecido, y por tanto, participar en la siguiente ronda. En este caso, el primer valor de filtro es 20, por lo tanto todos los participantes pasan a la siguiente ronda. En este momento se almacenan los resultados de la primera ronda, y se da paso a la siguiente.

En la siguiente ronda el filtro establecido es 50, por lo tanto repitiendo el proceso anterior, tras la segunda ronda de subastas, el agente numero 4 no alcanzará el valor y establecido, y no podrá participar en la última ronda. Esta situación puede observar en la Figura 21 y se da al haber seleccionado la subasta japonesa, diferenciándola de la inglesa, la que si se hubiera seleccionado, no habría excluido ningún agente en las rondas de negociación.

Tras la última ronda, el único participante capaz de apostar un valor suficientemente alto para superar el filtro es el agente número 2, siendo finalmente seleccionado como ganador de la subasta. En el caso de que ningún agente superara alguno de los filtros establecidos, por defecto en la implementación actual se devuelve 1 de los últimos agentes que ha superado el último filtro.

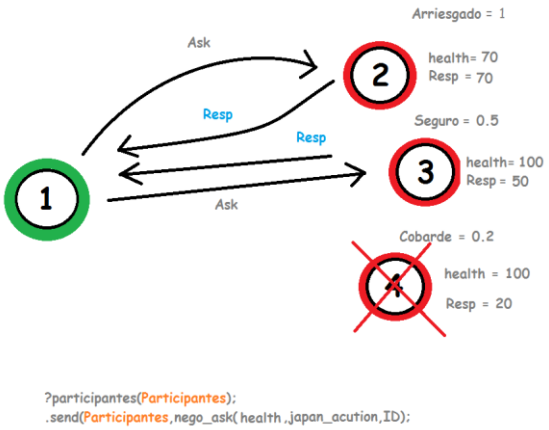


Fig. 21: Proceso Filtrado

Finalmente el agente subastador se encarga de enviar al agente solicitante de la subasta el identificador del ganador, y su valor asociado de apuesta. En el ejemplo utilizado, ya que coindicen subastador y solicitante, se informaría a si mismo del resultado. Con este resultado se puede informar a todos los médicos participantes y en base a ello decidir cuál es el encargado de acudir a la petición de ayuda, refinando así el proceso de selección de médico.

Todo este proceso de subastas implica un coste computacional y un envío de mensajes que no es despreciable. En concreto en este ejemplo, al ser una subasta japonesa, el número de mensajes enviados está relacionado con el número de participantes que quedan en cada ronda. En total, se envían 4 mensajes para empezar la negociación, en la primera ronda se envían 9 mensajes para el proceso de encuesta, al tener que informar a todos los agentes de las apuestas realizadas además de los mensajes de pregunta y respuesta habituales. La segunda ronda es idéntica a la primera, con un total de 9 mensajes. Finalmente en la última ronda solo se envían 6 mensajes, al haber filtrado a 1 participante. El proceso termina con 1 mensaje adicional para informar al solicitante de la encuesta.

Conclusiones de la validación de subastas abiertas

En total se envían 29 mensajes para realizar todo el proceso de subasta japonesa, con un tiempo aproximado de 1.5 segundos por ronda, en total 4.5 segundos aproximadamente. Ya que el proceso de selección de un médico para asistir una petición de ayuda debe ser lo más rápido posible, podría utilizarse una subasta de sobre cerrado de primer precio para realizar la selección, lo que conllevaría un tiempo de 1.5 segundos aproximadamente, y un total de 11 mensajes.

9. Conclusiones

En lo referente a la implementación del módulo de subastas se alcanzaron todos los objetivos iniciales planteados, junto a los sub-objetivos específicos del proyecto. El módulo ha sido testeado en distintas situaciones, con variaciones, tanto en los parámetros de la partida, como el número y tipo de participantes, utilizando las diferentes subastas implementadas, presentando un funcionamiento sin errores.

Aunque dentro del entorno JGOMAS, la utilización de la herramienta se ve limitada por la necesidad de implementar un sistema adicional, donde las apuestas tengan un peso relevante a la hora de coordinar las distintas estrategias para conseguir la victoria, el módulo ofrece una base sólida para implementar futuras mejoras, abriendo el camino a la introducción de estrategias para la toma de decisiones respecto a los valores apostados por los agentes.

Por otra parte, es posible relacionar fácilmente este sistema de perfiles de apostador con sistemas de simulación de emociones, donde las apuestas realizadas por los mismo se hagan en función de su perfil emocional en el momento actual de la partida, el cual puede variar a lo largo de la misma, dependiendo de las acciones realizadas o los resultados obtenidos por el agente.

En lo referente a la flexibilidad y funcionalidad del módulo, es relevante señalar que permite una reutilización relativamente sencilla del sistema de subastas, permitiendo la implementación de nuevas estrategias personalizadas, la creación de nuevos tipos de subastas, utilizando los existentes o por el contrario implementando nuevos, además de facilitar la adición de nuevos parámetros que seleccionar a la hora de realizar la subasta.

Teniendo en cuenta el uso del módulo fuera del entorno JGOMAS, es posible la extrapolación a cualquier otro sistema MAS basado en JADE y Jason, permitiendo así la utilización de un módulo adicional con toda la infraestructura necesaria para la realización de subastas y capaz de generar registros de datos para su uso en cualquier ámbito en que se utilicen este tipo de sistemas. De igual manera, sería posible añadir sistemas propios para los perfiles de apostador, permitiendo incluso utilizar herramientas de aprendizaje automático, para desarrollar las capacidades de los agentes a la hora de realizar las apuestas. Este tipo de sistemas está siendo muy utilizado actualmente en la creación de corporaciones virtuales, siendo recurrente el uso de este tipo de subastas para la asignación de procesos a realizar por las empresas participantes en ellas.

En la referente a la aplicación para el análisis de datos, aunque con un diseño simple, se ha obtenido una herramienta fácil de utilizar y capaz de presentar toda la información relevante en el proceso de negociación de los agentes de una manera clara y ordenada, pretendiendo facilitar así la tarea didáctica de las asignaturas impartidas en el centro relacionadas con técnicas de inteligencia artificial que utilizan los procesos de subastas como base de coordinación o comunicación. En el anexo se incluye un posible uso del módulo, junto a la aplicación para el análisis de datos, en forma de práctica realizable por los alumnos que cursen la asignatura de agentes inteligentes del grado de Ingeniería Informática en la Universitat Politécnica de València.

10. Bibliografía

1. A. Mowshowitz. Social Dimensions of Office Automation. (1986). Advances in Computers, 25:335-404.

2. Franklin, S. y Graesser, A. Is It an agent, or just a program?: A taxonomy for autonomous agents. *Intelligent Agents III Agent Theories, Architectures, and Languages.* Heidelberger : Springer Berlin Heidelberg, 1996.

3. Guerra-Hernández, A., Fallah-Seghrouchni, A. El y Soldano, H. *Learning in BDI Multi-agent Systems.* Heidelberger : Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-24010-5.

4. Huhns, M.N. & Singh, M.P. (Eds.) (1997) Readings in Agents. San Francisco, CA: Morgan Kaufmann.

5. J. A. Byrne, R. Brandt, and O.Bort. (1993) The Virtual Corporation. Business Week, February 8:36-40.

6. Jason. [En línea] http://jason.sourceforge.net/wp/description/.

7. J.K. Murnighan. (1991). The dynamics of bargaining Games. Englewood Cliffs, NJ: Prentice Hall.

8. linamce. monografias. [En línea] 18 de Octubre de 2004. http://www.monografias.com/trabajos16/la-inteligencia-artificial/la-inteligencia-artificial.shtml.

9. Klaus Fischer , Jörg P. Müller , Ingo Heimig , August-Wilhelm Scheer (1996). Intelligent agents in virtual enterprises.Geman AI Research Centre – DFKI GmbH.

10. Rafael H. Bordini, Jomi F. Hübner. Jason Documents. [En línea] 0.9.5, Febrero de 2007. http://jason.sourceforge.net/Jason.pdf.

11. Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. Computer Science Department, Carnegie Mellon University, Pittsburgh, USA.

12. Schut, M.C. (2007). Scientific handbook of simulation of collective intelligence. Publicado bajo licencia Creative Commons licence. Disponible en <http://www.mpcollab.org/MPbeta1/node/143>.

13. V. Julián, V. Botti. *Agentes Inteligentes: el siguiente paso en la inteligencia artificial.* Especial 25 aniversario, Barcelona : ATI, 2000, NOVATICA.

14. Wooldridge, M. & Jennings, N. R. (1995) “Intelligent agents: theory and practice”. Knowledge Engineering Review 10(2):115-152.

11. Anexo

11.1 – Práctica selección de líder en JGOMAS

Para la realización de esta práctica es necesaria la inclusión del archivo negotiation.asl en el directorio raíz del proyecto JGOMAS, además de añadir al fichero original Jgomas.asl la orden para incluir el módulo de negociación:

```
{ include("negotiation .asl") }
```

Una vez realizado esto, los agentes participantes en la partida serán capaces de realizar procesos de negociación, utilizando un sistema de subastas clásico. Para ilustrar este método, se realizara una selección de líder de equipo, con las distintas subastas que se ofrecen en el módulo.

Para ello, se utilizará la funcionalidad proporcionada por el interfaz incluido en este documento y la herramienta NegoLog.java para el análisis de los registros. Es necesario familiarizarse previamente con ella para responder correctamente a las preguntas.

Pregunta 1- ¿Qué plan sería conveniente sobre-escribir, en los archivos específicos para cada tipo de agente, para que los agentes ejecuten una sola vez la negociación?

Respuesta 1 – El plan !init se ejecuta una sola vez al inicio de la partida, si queremos utilizar el plan de petición de subastas solamente una vez, sería la mejor opción. Otra manera de hacerlo sería creando un especifico que lanzase el proceso de negociación como objetivo inicial de los agentes.

Pregunta 2- ¿De los diferentes tipos de subasta que ofrece el módulo, cuál de ellos debería seleccionarse si se quiere realizar una selección de líder rápida?

Respuesta 2 – Cualquier tipo de subasta cerrada sería recomendable para este tipo de selección, ya que solamente se realiza una ronda de negociación, y por lo tanto son las más rápidas.

Pregunta 3 - ¿Qué parámetro sería el más indicado utilizar para realizar la selección de líder?

Respuesta 3 - Dado que al inicio de la partida todos los agentes empiezan con los mismos valores de munición y vida por defecto, lo más recomendable sería realizar una selección de líder utilizando la distancia al objetivo de cada agente, mediante una subasta cerrada a la baja, lo que devolverá la posición del agente más cercano a la bandera .

Pregunta 4 - ¿Utilizando la funcionalidad que ofrece el modulo, como se realizaría una selección de líder en la que solamente participasen los soldados básicos, y no los médicos ni los soldados de apoyo?

Respuesta 4 - El módulo de subastas permite seleccionar qué agentes participan en el proceso de negociación, por lo tanto, mediante la interfaz que proporciona JGOMAS, es posible obtener la lista de los soldados básicos de nuestro equipo, y utilizarla como parámetro del módulo de subastas.

Pregunta 5 - Teniendo en cuenta todo lo anterior, realiza una selección de líder al inicio de la partida, utilizando una subasta a la baja de sobre cerrado con el parámetro distancia al objetivo, en la que solo participen los agentes de tipo médico y analízala utilizando la aplicación NegoLog.java.

- ¿Qué agentes han participado en el proceso de negociación, y cuales han intervenido sólo en la subasta?
- ¿Cuántas subastas se han realizado?
- ¿Qué agente ha sido seleccionado como subastador de las subastas realizadas?
- ¿Qué agente ha sido el ganador de las subastas?
- ¿El subastador es el mismo en todos los casos?
- ¿El ganador es el mismo en todos los casos?
- ¿Cuánto ha tardado el proceso de negociación completo de cada subasta?
- ¿Cómo utilizarías el identificador devuelto a cada agente con el ganador de la subasta para que todo el equipo sepa que es el líder?

ID Subasta	Participantes	Subastador	Ganador	Valor	Solicitante	Tiempo

Respuesta 5 – Cualquier tipo de agente puede iniciar el proceso de negociación, pero hay que tener en cuenta que todos los agentes del mismo tipo lanzaran de manera simultánea el proceso de subastas, lo que implicará realizar tantas subastas como agentes la lancen, y el subastador tendrá que procesarlas de manera concurrente.

El subastador será siempre el mismo en todos los caso a menos que muera en mitad del proceso de negociación. Todos los agentes solicitantes recibirán el mismo identificador de ganador al obtenerse resultados casi idénticos.

El ganador en todos los casos será el agente más próximo a la bandera inicialmente.

Se podrá enviar el identificador a todos los agentes, añadiéndolo como una creencia adicional a la base de hechos y siendo utilizada posteriormente para identificar al líder del equipo.

Pregunta 6 – Ahora repite el proceso anterior utilizando una subasta inglesa con una lista de filtros ascendente y el parámetro health, pero que se lance cuando el líder actual tenga vida 0.

- ¿Qué agentes han participado en el proceso de negociación, y cuales han intervenido sólo en la subasta?
- ¿Cuántas subastas se han realizado?
- ¿Qué agente ha sido seleccionado como subastador de las subastas realizadas?
- ¿Qué agente ha sido el ganador de las subastas?
- ¿Cuánto ha tardado el proceso de negociación completo de la subasta?
- ¿Cuántas rondas se han realizado?
- ¿Cómo utilizarías el identificador devuelto a cada agente con el ganador de la subasta para que todo el equipo sepa que es el nuevo líder?

ID Subasta	Participantes	Subastador	Ganador	Valor	Solicitante	Tiempo

Respuesta 6 – En este caso solamente se realizará una subasta, en el momento que el líder tenga vida cero y debería implementarse en el plan asociado a recibir daño.

Al ser una subasta inglesa, se realizaran tantas rondas como valores se han incluido en el filtro, y su tiempo estará en función del número de rondas realizadas.

El ganador de la subasta será el agente participante que tenga más vida en el momento. Esta subasta es más lenta que una subasta de sobre cerrado y en el caso de una selección de líder, sería menos conveniente su uso.