PRACTICA 2: AGENTE PARTICIPANTE

GRADO EN INGENIERÍA INFORMÁTICA, 4º CURSO

INTELIGENCIA ARTIFICIAL EN LAS ORGANIZACIONES

JORGE SERRANO PEREZ 100405987@alumnos.uc3m.es

Contenido

EXPLICACIÓN XML

En el *XML* he seguido de nuevo la estructura correspondiente:

<imports> Librerías y clases necesitadas **</imports>**

En este caso no se han necesitado importar las creencias, ya que al ser variables de tipo simple no es necesario crear una clase java

<capabilities></capabilities>

<beliefs> Aquí se han incluido todas las creencias del tablero **</beliefs>**

Como decía, en este caso únicamente se les ha aportado un tipo simple de java y no una clase. A continuación, pasaré a detallar la información que aporta cada una.

- *marcador_fase*: indica la fase del juego en un momento determinado dentro de la fase política.
- *partida_iniciada*: indica si la partida está iniciada. Se utiliza para realizar el encuentro inicial entre el jugador y el tablero.
- *num_cartas_politicas*: indica el número de cartas políticas que tiene el jugador
- *cargo*: indica el cargo que le ha asignado el presidente, o si es él el presidente.
- *billetes*: indica el número de billetes que tiene el jugador en mano.
- *localizacion*: indica la localización que ha elegido el jugador. Se incluye el Exilio como localización.
- *num_cartas_asesinato*: indica el número de cartas de asesinato que tiene el jugador, si tiene.
- *excusa_activada*: indica si la ficha excusa está activada para poder dar un golpe y entrar en la Fase de Golpe.
- *billetes_en_suiza*: indica el número de billetes que tiene el jugador en su cuenta de Suiza.

<goals> Objetivos a cumplir por el agente tablero **</goals>**

Todos los *goals* implementados son del tipo *perform* excepto los necesarios para buscar y registrar un agente en el *df*. Se incluyen tan solo los que inicia el jugador, que no son muchos.

Esto se debe a que, según nuestro diseño, es casi siempre el tablero el que hace el *request* pidiendo al jugador que ejecute una acción.

- *encuentro_inicial*: tan solo se comprueba si la creencia partida_iniciada es falsa para poder iniciar el plan. Se ha incluido un *retry*=" true" y *retrydelay*="3000" para que no se compruebe a cada instante si ya existe el tablero.
- *intercambiar_cartas*: se comprueba si el jugador tiene cartas para intercambiar
- *recibe_ayuda*: se comprueba si la fase del juego se corresponde con la fase de recibir la ayuda extranjera, es decir, el dinero, y se comprueba también si el jugador tiene el cargo de presidente.
- *empezar_golpe*: se comprueba si la fase del juego es la correcta, si la ficha excusa está activada, es decir, si la situación política del juego es inestable, y si el jugador no es el presidente, ya que este no puede iniciar una rebelión contra el mismo.
- *intentar_asesinar*: se comprueba si la fase es la correcta y si el jugador es el ministro, que puede realizar un asesinato enviando a la Policía Secreta o si tiene cartas de asesinato.
- *intercambiar_billetes*: tan solo se comprueba si el jugador tiene billetes para intercambiar.

<plans> Planes a llevar a cabo por el agente tablero **</plans>**

Como disparador se ha incluido su objetivo relacionado o el mensaje que recibe del tablero para cada caso. Para los planes que he elegido, el del encuentro inicial utiliza el *goal* correspondiente. Los otros dos, sin embargo, el de elegir localización y el de exiliarse activarán la ejecución del plan correspondiente cuando reciban cada uno el request necesario desde el tablero.

<events> Mensajes **</events>**

Se han añadido por un lado los distintos mensajes que se envían con su *performative* correspondiente, como puede ser *request* o *inform*, y por otro lado todos los mensajes que reciben del tablero, con su correspondiente clase asociada además del *performative*.

<configurations> Estado inicial **</configurations>**

En las *beliefs* hemos inicializado aquellas que requieren de un hecho inicial, es decir, el valor que toman cuando se acaba de iniciar la partida (en este caso para ejecutar los planes elegidos), como por ejemplo el número de billetes en mano o la fase del juego. A su vez, se añade el encuentro inicial de los agentes como objetivo inicial del jugador.

También hemos registrado el agente jugador en el *df*, para luego buscarlo desde los planes si fuera necesario.

ESTRATEGIA DE LOS PLANES

En cuanto a los planes, se debían implementar los asociados a tres protocolos del participante incluidos en el documento de diseño. Uno de ellos debía ser el protocolo correspondiente al encuentro inicial en el que se realiza una búsqueda en el *df* para poner en contacto a la organización (tablero) con el participante (jugador). El resto debían implementar decisiones no triviales y con cierta inteligencia haciendo uso de información pasada o previendo acciones futuras. En mi caso, he elegido el protocolo en el que el jugador debe elegir la localización entre las 5 posibles, y también el protocolo que usaría el jugador para exiliarse. Ambos cuentan solo con dos mensajes, pero a la hora de elegir es donde se inserta la inteligencia, por lo que se trata de decisiones no triviales.

La estrategia a seguir es sencilla. En el primer plan mencionado, las localizaciones posibles son el Banco, el Cuartel General, la Residencia, el Burdel y el Club Nocturno. De estas cinco, la que más tarde te permitirá realizar acciones será tan solo el Banco (ya que el Cuartel General forma parte de la Fase del Golpe, que no era necesario contemplar en la implementación). Por lo tanto, la estrategia será elegir el Banco como localización en algunos casos, y cualquiera de las otras localizaciones en el otro. ¿Qué estrategia? Pues personalmente he elegido tener en cuenta el número de billetes en mano. Si se tienen más de 5 billetes, eso ya opino que es cantidad suficiente para guardarlo en la cuenta de Suiza y que así no me lo puedan quitar. Si se tienen menos, se elegirá otra localización. El escoger una u otra es irrelevante, ya que ninguna aporta mayor o menor beneficio a futuro. Sin embargo, se podrían realizar otras estrategias, como tener en cuenta qué localizaciones se

han elegido anteriormente para que el resto de jugadores, si te quieren asesinar, que no lo adivinen tan fácilmente.

En la parte de la implementación, se ha procedido de la siguiente manera:

- En primer lugar, según nuestro diseño es el tablero el que solicita al jugador que realice la elección de la localización. Por lo tanto, el tablero tendrá un *goal* que se activará cuando comience la *Fase_Eleccion_Localizaciones* y que dará comienzo al plan. En este fichero java, el tablero comienza buscando en el *df* al jugador (con nombre jugador1, aunque en este caso no es necesario incluir más jugadores para completar el plan) y enviando el *request* con la acción *Elige_localizacion*.
- Por la parte del jugador, este activará su plan correspondiente solo cuando reciba el mensaje *request* concreto mencionado anteriormente. En el fichero java, se leerá el mensaje y se obtendrá el contenido y el *sender*, y a continuación se tendrá en cuenta la estrategia.
- Obtendremos la creencia de la base de creencias del jugador que nos dice el número de billetes que posee, y con un *if* comprobaremos si este es mayor o menor que 6. Si es mayor, se elegirá el Banco como localización para posteriormente guardar el dinero en la cuenta de suiza, y si es menor, se generará un número aleatorio para elegir cualquiera de las otras localizaciones.
- En todos los casos, primero se guardará la localización elegida en la base de hechos, luego se añadirá la localización al predicado, y luego se incluirá ese predicado en el contenido del mensaje.
- Por último, se especificará que el tablero que antes era el *sender*, ahora es el *receiver* (el que recibe el mensaje) y se enviará el mensaje *inform*. Se han incluido también algunos *prints* para seguir el curso del plan en la consola.
- Ese mensaje activará un nuevo plan en el tablero. En el mismo, primero obtendrá el contenido del mensaje y, a continuación, actualizará sus creencias en base al mismo.
- Para ello, cuenta con una clase *Localizaciones_jugador* que contiene un *HashMap* en el que relaciona cada jugador (*AgentIdentifier*) con su localización elegida (*String*).

Pasamos ahora al segundo plan, el del exilio. Este es sustituto del anterior, es decir, o eliges uno o eliges el otro, pero no puedes realizar los dos. Por ello, el *goal* que activa el plan es el mismo que en el de elegir localización: la fase tiene que ser la correcta

(*Fase_Eleccion_Localizaciones*). Además, por el mismo motivo, será necesario crear dos configuraciones en el *XML*, una para cada plan. En este caso, la inteligencia implementada es muy similar a la anterior. Se trata de una estrategia conservadora, en la que se comprueba el número de billetes que tiene el jugador en la cuenta de Suiza. Si tiene más de cierta cantidad es cuando decide exiliarse, ya que de esta manera no puede ser asesinado y no pierde su dinero.

En la parte de la implementación:

- Como en el caso anterior, el tablero inicia el plan de la misma manera.
- El jugador activa su plan cuando le llega el mensaje *request* correspondiente. En el fichero java, de nuevo obtenemos obtendrá el contenido y el *sender*, y a continuación se tendrá en cuenta la estrategia.
- Sacamos la creencia que nos indica el número de billetes que tiene el jugador en Suiza, y si el número es mayor que 15, elegiremos exiliarnos.
- De nuevo, primero se guardará la localización elegida, en este caso el exilio, en la base de hechos, luego se añadirá la localización al predicado, y luego se incluirá ese predicado en el contenido del mensaje. En este caso, el predicado será *Exiliado* o *No_exiliado* en base a la elección que se haya realizado.
- De la misma manera que antes, se preparará el mensaje *inform* con su *sender* y su contenido, y se enviará al tablero.
- A continuación, destacar que, en este caso, al haber dos mensajes diferentes con predicados diferentes (y por lo tanto clases diferentes), será necesario crear también dos planes diferentes que se activarán cada uno con su mensaje.
- Ambos realizan en rasgos generales lo mismo que en el plan del protocolo de elegir localización: obtener el contenido del mensaje y actualizar la base de creencias del tablero para indicar que el jugador está en el exilio o no.

Por último, para la implementación del plan del encuentro inicial entre el tablero y el jugador, se activa en el *XML* con un *goal* que identifica si la partida se ha iniciado o no (como ya se ha mencionado en la parte de la memoria correspondiente). En el fichero java, tan solo se realiza la búsqueda estableciendo el type a *tablero*. Si se encuentra un resultado se saca por pantalla y se indica en la base de creencias que la partida se ha iniciado, y si no se encuentra, se indica también por pantalla con un *print*.

Sin embargo, en mi caso este plan no era necesario, ya que es siempre el tablero el que envía los mensajes y, por lo tanto, el que realiza la búsqueda en el *df* para localizar y mandárselos al jugador.

CONCLUSIONES

En primer lugar, creo que el uso de agentes es muy desconocido, o al menos lo era en mi caso, y sin embargo me ha resultado muy interesante. Tanto esta práctica como los casos prácticos de las clases de teoría me han parecido enormemente útiles para la correcta comprensión de la asignatura y para asimilar los conceptos correctamente.

En cuanto a la práctica, más concretamente, nos enfrentamos primero a trabajar en grupos bastante grandes, que ya en último curso es más complicado para organizar debido a las diferencias de horarios y todo ese tema. Otro problema que tuvimos fue la instalación de JADEX y del entorno, que no encontramos mucha documentación para realizarlo correctamente. Sin embargo, un punto a favor fueron los ejemplos que teníamos a nuestra disposición de otras prácticas pasadas, que nos sirvieron enormemente para completar la nuestra.

Como crítica constructiva, creo que quizá se debería indicar en las primeras prácticas la utilidad de las mismas, ya que siempre se decía que “luego entenderás todo”, y aunque es verdad, creo que genera cierta frustración el tener que cambiar el contenido en todas y cada una de las entregas pasadas porque lo que habías hecho no tenía sentido. Además, encontramos ciertas discrepancias entre la teoría vista en clase y subida a aula global y la práctica, como por ejemplo la clase *content* del mensaje, que en realidad era *content-class*.

En definitiva, una práctica muy útil e incluso gratificante de hacer una vez comprendes todo, pero que cuesta llegar a ese punto en el que sabes arreglar cualquier error.