

PROYECTO FINAL DE CARRERA

**"Mejora de la comunicación entre agentes y
creación de una interfaz gráfica para la
configuración de reglas difusas en el modelo
TEAKS "**

Directores: Antonio Moreno y Juan Martínez Miranda

**Laura Férrez Fortuny
ETIG**

ÍNDICE

INTRODUCCIÓN.....	3	
OBJETIVOS.....	4	
CAPÍTULO I:		
1.1.- Introducción.....	5	
1.2.- Sistema Multiagente	5	
1.3.- Estándares FIPA	6	
1.4.- El Modelo TEAKS	9	
1.4.1.- Uso de lógica difusa dentro del modelo.....	14	
1.4.2.- Arquitectura del Sistema Multiagente.....	15	
1.4.3.- Prototipo del SMA.....	16	
1.5.- Ontología.....	18	
1.6.- Conclusiones.....	19	
CAPÍTULO II: ESPECIFICACIONES		
2.1.- Introducción.....	21	
2.2.- Comunicación existente.....	21	
2.3.- Mejoras a la comunicación existente.....	23	
2.3.1.- Agregado de una Ontología.....	23	
2.3.2.- Extensión de los protocolos de comunicación.....	25	
2.4.- Proceso de obtención del comportamiento del agente.....	26	
2.5.- Conclusiones.....	28	
CAPITULO III: DISEÑO		
3.1.- Introducción.....	29	
3.2.- Diseño de la ontología.....	29	
3.3.- Protocolos de comunicación.....	33	
3.3.1.- Contract-Net protocol.....	35	
3.3.2.- Request Protocol.....	37	
3.4.- Interfaz gráfica para la configuración de las reglas difusas.....	39	
3.5.- Conclusiones.....	42	
CAPITULO IV: DESARROLLO		
4.1.- Introducción.....	44	
4.2.- Trabajo realizado para la mejora de la comunicación en el SMA.....	44	
4.3.- Trabajo realizado para el desarrollo de la interfaz para la configuración de las reglas difusas.....	57	
4.4.- Conclusiones.....	58	
CAPÍTULO V: EVALUACIÓN		
5.1.- Introducción.....	59	
5.2.- Juego de Pruebas realizado respecto a la comunicación.....	59	
5.3.- Juego de pruebas realizado para la configuración de reglas.....	71	
5.4.- Conclusiones.....	72	
CAPÍTULO VI: CONCLUSIONES.....		74
CAPÍTULO VII: REFERENCIAS Y PÁGINAS WEB.....		76
ANEXO: Código		

INTRODUCCIÓN

El trabajo realizado en el proyecto final de carrera presentado en este documento forma parte de la tesis doctoral realizada en el DEIM por Juan Martínez Miranda en el área de Inteligencia Artificial.

Esta tesis doctoral presenta un modelo basado en agentes para la simulación del comportamiento humano dentro de un equipo de trabajo. La implementación de este modelo a través de un Sistema Multi-Agentes (SMA) proporcionará una herramienta de ayuda en la toma de decisiones para la configuración de equipos de trabajo responsables de desarrollar un proyecto específico. Este modelo se implementa a través de un SMA en el cual ya existe una comunicación básica entre los agentes.

Mi trabajo en este proyecto final de carrera consiste en robustecer esta comunicación entre los agentes a través del diseño e implementación de una ontología y la adición de 2 protocolos de comunicación FIPA. El código de programación generado a partir del trabajo realizado quedará listo para integrarse en el SMA desarrollado en la mencionada tesis doctoral.

Otro aspecto que forma parte del modelo de la tesis es la creación de una serie de reglas para generar el posible comportamiento de un equipo de trabajo. Mi trabajo consiste en desarrollar una interfaz gráfica para llevar a cabo la configuración de las reglas dentro del SMA para la simulación del comportamiento de un agente sin necesidad de modificar manualmente el fichero donde están almacenadas estas reglas y sin necesidad de conocer la sintaxis del lenguaje JESS, en el cual están escritas estas reglas. Este trabajo se explicará más a detalle en el capítulo 2.

OBJETIVOS

El objetivo global de este proyecto final de carrera es el de mejorar la comunicación existente entre los agentes desarrollados en la tesis doctoral, así como el desarrollo de la interfaz para la configuración de las reglas difusas arriba mencionadas. Para conseguir este objetivo global, se deben llevar a cabo los siguientes objetivos particulares:

- Análisis y diseño de una ontología: identificar y analizar cuáles son los conceptos utilizados en el SMA y la relación entre ellos para desarrollar una ontología que permita mejorar y robustecer la comunicación y coordinación entre los agentes.
- Mejorar y modificar la comunicación en el sistema multiagente (SMA): Utilizando la ontología desarrollada, los protocolos de comunicación FIPA existentes, podrán ser mejorados significativamente.
- Desarrollo.: La ontología y la mejora en los protocolos de comunicación FIPA deben programarse en el lenguaje JAVA y dejarse listos para su integración al SMA existente.
- Interfaz gráfica para la configuración de reglas difusas: Desarrollo de una interfaz gráfica a través de la cual el usuario del SMA será capaz de configurar las reglas difusas que generan el posible comportamiento de acuerdo a las características del equipo y proyecto que se desea analizar.

CAPÍTULO I:

1.1- Introducción

Después de haber hecho una descripción de los objetivos del trabajo realizado en el proyecto final de carrera, haremos una introducción de conceptos tales como la definición de un SMA, la descripción de las especificaciones FIPA para la construcción de SMA y los conceptos básicos de una ontología. Adicionalmente se explicará un poco más a detalle el modelo de agentes, la arquitectura del SMA y el prototipo desarrollados en la tesis doctoral para tener de manera clara el contexto en el cual se ha desarrollado este proyecto final de carrera.

1.2- Sistema Multiagente (SMA)

El concepto de agente aun no ha encontrado una definición unánimemente aceptada por todo el mundo, sin embargo, una de las más aceptadas es la propuesta por Wooldridge and Jennings [*Wooldridge and Jennings, 1995*]: sistema hardware o software con las siguientes propiedades:

- Autonomía: los agentes operan sin la intervención de humanos y tienen algún tipo de control sobre sus acciones y su estado interno.
- Habilidades sociales: los agentes interaccionan con otros agentes (y posiblemente humanos) a través de algún tipo de lenguaje de comunicación de agentes.
- Reactivos: los agentes perciben el entorno (que puede ser el mundo físico, un usuario por medio de una interfaz gráfica, otros agentes, o posiblemente todos los anteriores combinados), y responde en un tiempo razonable a los cambios ocurridos.
- Proactivos: los agentes no responden simplemente al entorno, sino que pueden exhibir comportamientos basados en metas, tomando la iniciativa.

Un agente inteligente desempeña de manera continua tres funciones: percibir las condiciones dinámicas de su entorno; razonar para interpretar sus percepciones, resolver problemas, realizar inferencias y determinar acciones. [Hayes-Roth, 1995].

A partir de las definiciones anteriores, un SMA puede definirse como un grupo de agentes que trabajan conjuntamente para conseguir un conjunto de metas en común [Weiss, 1999]. La eficiencia del sistema consiste en que cada agente debe ser capaz de razonar sobre las acciones de otros agentes además de las suyas propias. Por esta razón, son necesarios mecanismos de coordinación que aseguren que los planes individuales no entren en conflicto con los planes de los demás para conseguir las metas globales.

Para llevar a cabo la programación de un SMA, la herramienta utilizada debe cumplir un conjunto de especificaciones sobre agentes y sistemas multiagente. Existe una organización que se dedica a estudiar y redactar especificaciones sobre estos sistemas, como es el caso de FIPA.

1.3- Estándares FIPA (*Foundation for Intelligent Physical Agents*)

FIPA [FIPA Agent Management Specification. Foundation for Intelligent Physical Agents, 2000] es un grupo de trabajo dentro de IEEE con el objetivo de producir especificaciones de software para la interacción de sistemas basados en agentes heterogéneos.

Para que una plataforma de desarrollo de agentes sea considerada *FIPA compliant*, debe al menos implementar las especificaciones para la Administración de Agentes y las de Lenguaje de Comunicación de Agentes (ACL).

La herramienta JADE es un entorno de desarrollo que facilita la programación de sistemas multiagente (SMA), está escrito en Java y es *FIPA Complaint*. JADE, que cumple con las especificaciones de la FIPA, proporciona una biblioteca de clases en Java para desarrollar agentes y el contenido de los mensajes para la comunicación entre los agentes está basado en FIPA-SL [Bellifemine, et al., 2003]

El modelo estándar de una plataforma de agentes, tal y como la define FIPA, se presenta en la siguiente figura [Bellifemine, et al., 2003]:

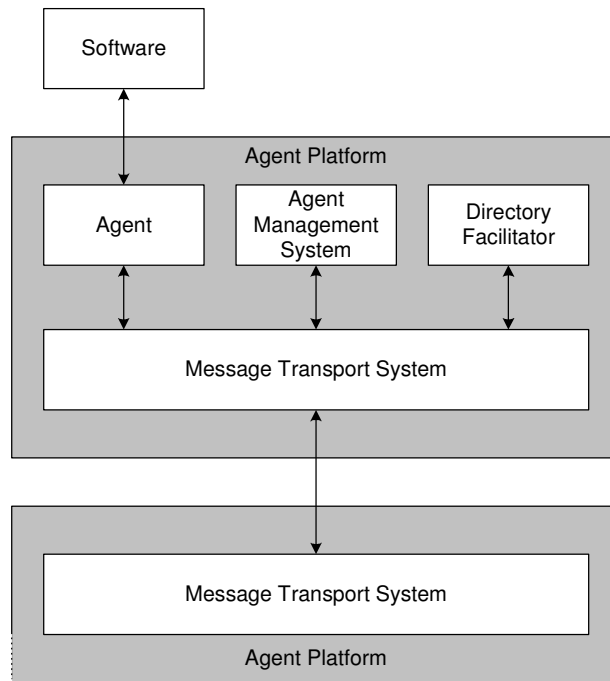


Figura 1 - Arquitectura FIPA de un SMA

Una breve descripción de cada uno de los elementos de la plataforma FIPA se presenta a continuación:

- **Agent:** programa capaz de realizar tareas de forma autónoma y comunicarse con otros agentes para resolver problemas mediante cooperación, coordinación y negociación.
- **Directory Facilitator (DF):** es un agente que proporciona un servicio de páginas amarillas mediante el cual un agente puede buscar a otros que le proporcionen el servicio que necesita para lograr su objetivo.
- **Agent Management System (AMS):** es un agente que controla el acceso y uso de la plataforma de los agentes. Ofrece los servicios de crear/eliminar agentes.
- **Message Transport System (MTS o ACC):** es el agente encargado de controlar el intercambio de mensajes con la plataforma, incluyendo mensajes para o de plataformas remotas.

Estas características de los agentes y SMA son muy útiles en diferentes tipos de problemas. Un área de aplicación en la cual estas características son especialmente útiles es el área de modelado y simulación. La organización AgentLink (www.agentlink.org) ha identificado áreas de aplicación de los SMA, las cuales divide principalmente en aquellas en las cuales la esencia del SMA es la toma de decisiones por sí mismo, y aquellas áreas en las cuales un SMA sólo provee información que ayuda en la toma de decisiones en contextos específicos [Luck et al., 2003]:

- Sistemas de decisión multiagente: donde los agentes que interactúan en el sistema deben tomar decisiones conjuntamente. Los mecanismos para la toma de esas decisiones conjuntas puede ser a partir de mecanismos económicos tales como subastas o mecanismos alternativos tales como razonamientos. Algunos ejemplos son sistemas para la asignación de recursos, control de procesos, reunión y filtros de información, comercio electrónico, administración de procesos de negocio, etc.
- Sistemas de simulación multiagente: donde el SMA se usa como modelo para simular algún dominio real. Normalmente se usa en dominios que implican diferentes componentes, interactuando en diversos y complejos caminos y donde las propiedades del nivel del sistema no infieren fácilmente en las propiedades de los componentes. Algunos ejemplos de tales dominios incluyen sociedades económicas, humanas y animales, poblaciones biológicas, sistemas de tráfico de carreteras, juegos, etc.

El trabajo desarrollado en la tesis doctoral a la cual pertenece este trabajo de PFC cae dentro de los SMA utilizados para simulación. En la siguiente sección se presentará brevemente el modelo basado en agentes desarrollado en la tesis y el problema que se pretende resolver con este modelo.

1.4- El Modelo TEAKS

En muchas empresas, cuando se ha de desarrollar un nuevo proyecto, el director de dicho proyecto es el responsable de dividirlo en diferentes tareas y seleccionar de forma correcta las personas que realizarán cada una de estas tareas.

El éxito de un proyecto la mayoría de las veces se da no solo considerando las habilidades técnicas de los posibles integrantes, sino aspectos tales como la disponibilidad de las personas, las características personales, emocionales y sociales de cada una de ellas. Es importante también que haya una buena comunicación y coordinación entre los distintos miembros de un equipo de trabajo [Biegler et al. 1997].

La tesis doctoral consiste en el desarrollo de un nuevo modelo llamado TEAKS (TEAM Knowledge-based Structuring) [Martínez-Miranda et al., 2005]. TEAKS es un modelo basado en agentes, en el cual, un agente de software representa a un integrante real del equipo de trabajo con unas determinadas características (emocionales, sociales, cognitivas y de personalidad) y a través de su implementación en una herramienta de software, es capaz de generar resultados estadísticos significativos de cuál sería el posible comportamiento tanto a nivel individual como global de un equipo de trabajo a partir de la interacción entre los candidatos para formar el equipo.

La siguiente figura representa la arquitectura general del modelo:

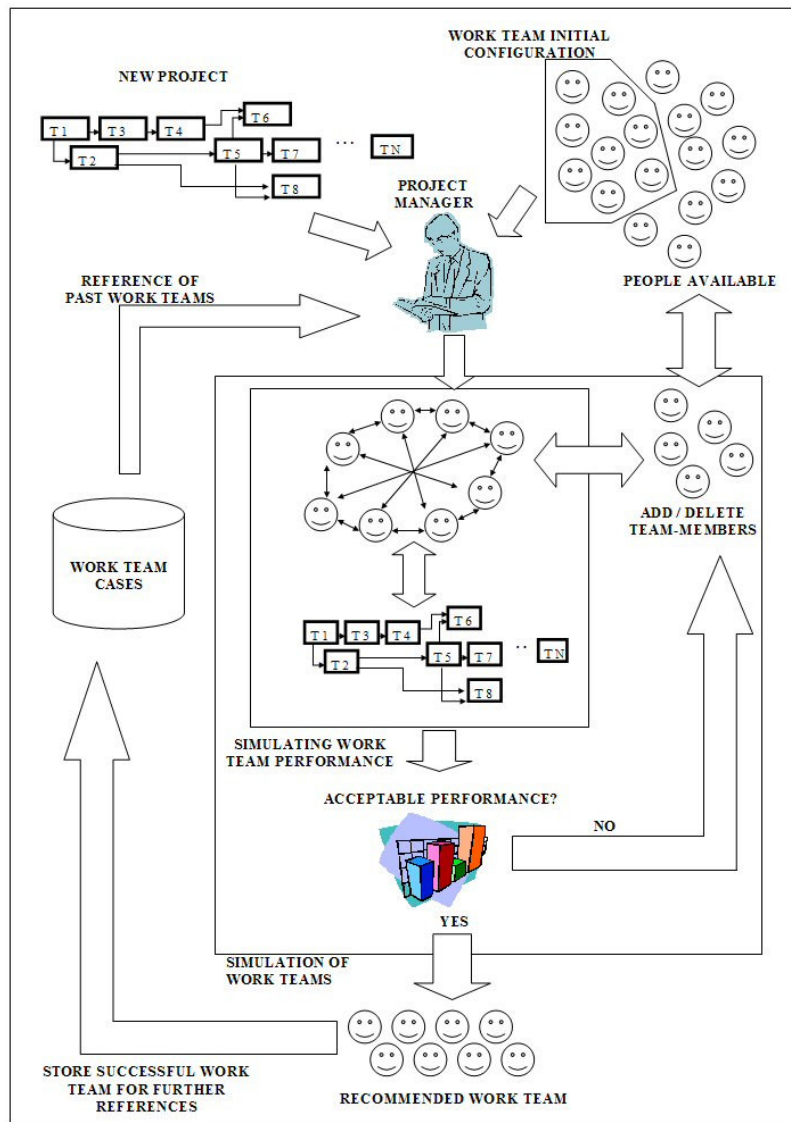


Figura 2 - Arquitectura general

La descripción del modelo TEAKS se presenta a continuación:

- **Configuración inicial del equipo de trabajo:** los agentes son configurados con las características de las personas candidatas a integrar el equipo y de la misma manera, las características de las tareas que forman el proyecto son también integradas al modelo. Una vez hecho este paso, cada tarea del proyecto es asignada a uno o más agentes. Estas decisiones las hace el Administrador del Proyecto.

- **Simulación del equipo de trabajo:** una vez configurados el equipo y el proyecto, se realiza la simulación de la interacción entre los agentes y sus tareas asignadas para obtener la información sobre el posible rendimiento del equipo de trabajo.

A partir de los resultados obtenidos, la configuración del equipo puede cambiar para analizar si esta nueva configuración mejora el rendimiento del equipo. Es decir, nuevos agentes con diferentes características pueden añadirse al equipo, otros pueden ser dados de baja del sistema y producirse una reasignación de una o más tareas del proyecto.

- **Configuración final del equipo de trabajo propuesto:** cuando los resultados sobre el desempeño de un equipo de trabajo se considera óptimo, la información de las características de cada integrante y su posible desempeño servirá al director del proyecto como ayuda para tomar la decisión final en la configuración real del equipo de trabajo.

El objetivo del modelo TEAKS es el generar información estadística sobre el posible desempeño de los trabajadores de un equipo a partir de la simulación de la interacción entre los posibles integrantes del equipo respecto a las características de sus tareas asignadas. Para ello se consideran un conjunto de características personales, emocionales, cognitivas y sociales correspondientes a los trabajadores, características sobre el rendimiento obtenido a la hora de realizar dicho proyecto y un conjunto de características de las tareas a realizar para desarrollar el proyecto.

➤ CARACTERÍSTICAS HUMANAS

Hay cuatro características humanas que en la tesis doctoral se han considerado con una importancia relevante en la influencia del comportamiento de una persona cuando realiza una tarea de trabajo y son los siguientes:

- Personalidad: donde los parámetros para definir la personalidad de un agente son:
 - ✓ Dominante (*driver*)
 - ✓ Analítico (*analytical*)

- ✓ Amable (*amiable*)
 - ✓ Expresivo (*expressive*)
- Estado emocional: donde los parámetros para definir las emociones de un agente son:
- ✓ Ansiedad (*anxiety*)
 - ✓ Disgusto (*disgust*)
 - ✓ Deseo (*desire*)
 - ✓ Interés (*interest*)
- Características sociales: para realizar tareas en grupo es vital una buena comunicación y coordinación entre los miembros del equipo. Los parámetros para definir las características sociales son:
- ✓ Introvertido (*introverted*)
 - ✓ Extrovertido (*extroverted*)
 - ✓ Trabaja solo (*work alone*)
 - ✓ Trabaja en equipo (*work team*)
- Capacidades cognitivas: para representar el conocimiento técnico de una persona con un equipo. Los parámetros para definir las capacidades cognitivas son:
- ✓ Experiencia (*experience*)
 - ✓ Creatividad (*creativity*)
 - ✓ Rol (*role*) y sus roles son los siguientes: *director, coordinador, especialista, técnico y ayudante (Project Manager, Co-ordinator, Specialist, Technician y Assistant)*

Además de estas cuatro características, un parámetro que se ha considerado importante y que es determinante en el desempeño de una persona frente a su trabajo es el **estrés**. Este parámetro no puede ser englobado en ninguna de las anteriores características pero sin embargo se ha tomado en cuenta en la obtención del posible desempeño del equipo dentro del modelo.

➤ DESEMPEÑO DE LOS AGENTES

El posible desempeño de los agentes se obtiene a partir de la interacción entre los agentes con sus tareas asignadas. Esto ayudará a integrar el equipo de trabajo real. Los parámetros para medir el desempeño de los agentes son:

- ✓ Nivel de objetivos conseguidos en su(s) tarea(s) asignada(s) (*goals achievment*)
- ✓ Tiempo en el desarrollo de una tarea (*task duration*)
- ✓ Calidad del rendimiento final obtenido en la tarea (*task quality*)
- ✓ Nivel de colaboración del equipo (*team level*)
- ✓ Nivel de contribución individual a sus tareas asignadas (*individual contribution level*)
- ✓ Nivel de supervisión requerido (*required supervisión level*)

➤ PROYECTO

Un proyecto se representa mediante la partición de tareas donde éstas están asignadas a uno o más miembros del equipo de trabajo. Los parámetros para definir una tarea son:

- ✓ Número de participantes (*num participants*)
- ✓ Duración (*duration*)
- ✓ Identificador (*id*)
- ✓ Dificultad (*difficulty*)
- ✓ Tipo (*type*)
- ✓ Plazo (*deadline*)
- ✓ Prioridad (*priority*)
- ✓ Coste (*cost*)
- ✓ Calidad (*quality*)
- ✓ Secuencia dentro del proyecto

Un ejemplo de la representación gráfica del proyecto se puede ver en la (*figura 3*):

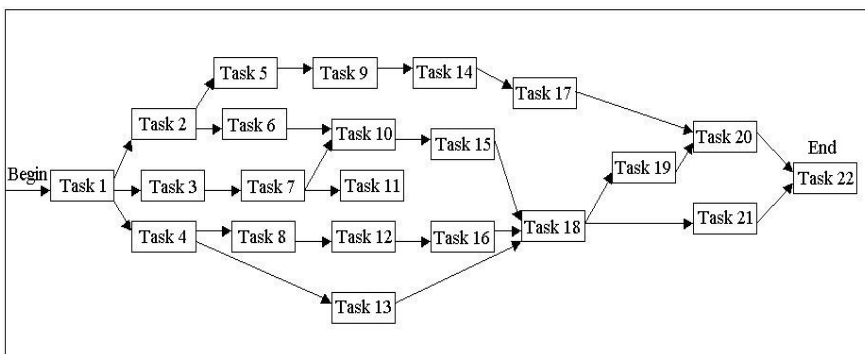


Figura 3 - Representación gráfica de un proyecto con 22 tareas

1.4.1- Uso de lógica difusa dentro del modelo

El comportamiento de un agente dentro del modelo desarrollado en la tesis es obtenido a través de reglas difusas las cuales utilizan valores difusos para representar los principales aspectos del comportamiento humano en el trabajo. Estos aspectos son las características internas del agente, los parámetros de la tarea que influyen en el comportamiento humano (*difficulty*, *duration* y *type* de tarea), los parámetros para medir el resultado del comportamiento del agente y el comportamiento generado por los agentes. Todos ellos ya han sido descritos anteriormente.

Para medir estas características se utiliza la teoría de conjuntos difusos introducida por Lotfi A. Zadeh [Zadeh, 1965].

La lógica difusa es un conjunto de técnicas operativas que permiten llevar a cabo los procesos de la toma de decisiones en situaciones caracterizadas por la incertidumbre. Se hace posible el tratamiento tanto de lo subjetivo como de lo incierto, en un intento por representar los fenómenos tal como se muestran en la realidad y de llevar a cabo su tratamiento sin intentar deformarlos o hacerlos precisos o ciertos. Los sistemas difusos combinan la información de expertos humanos (lenguaje natural) con mediciones y modelos matemáticos basados en leyes físicas (Ej. Aplicaciones en Ingeniería). Los sistemas difusos transforman la base de conocimiento humano en una formula matemática.

Para ello se amplía la teoría de conjuntos y de lógica booleana de manera que un individuo pueda pertenecer parcialmente a un conjunto y que las operaciones lógicas

además de unos y ceros, puedan ser 0,01 o 0,75. Por ejemplo, en el caso de las emociones, no se dice que la ansiedad es de 75 o 80 sino que es baja, media o alta. De esa forma, se intenta aplicar la forma de pensar humana a la programación de los computadores.

Una vez se conoce el modelo que se quiere desarrollar en el SMA, se hará una breve descripción del comportamiento de los agentes que interactúan en el sistema (SMA).

1.4.2- Arquitectura del Sistema Multiagente

A partir del modelo presentado en la sección anterior, se ha desarrollado el SMA que permitirá desarrollar la interacción descrita en el modelo [Martínez-Miranda et al., 2003]. En este SMA, hay dos clases de agentes: los agentes del equipo de trabajo (representan las personas reales que forman el equipo) y los agentes del sistema (utilizados para ejecutar acciones relacionadas con la funcionalidad del sistema).

Los agentes del sistema son:

Agente Interfaz (*Gui Agent*): agente de la interfaz gráfica que hace de intermediario entre el sistema y el usuario. A través de este agente, el usuario configura el equipo de trabajo, construyendo el proyecto y asignando las tareas a cada agente.

Agente Simulador (*Simulator Agent*): tiene el control sobre el número de simulaciones y los factores externos que influyen en cada simulación. Estos factores externos se dividen en dos grupos: los factores que afectan al proyecto (por ejemplo debido a modificaciones propuestas por la empresa o por el cliente) que afectan al comportamiento del equipo y el otro factor es el que afecta a los agentes, los cuales tienen relación con la vida real (por ejemplo, problemas familiares, algunos accidentes físicos entre otros).

Agente Asignador (*Assigner Agent*): asigna las correspondientes tareas a los miembros del equipo de trabajo.

Agente Resultado (*Resulter Agent*): recibe los resultados del rendimiento de los miembros del equipo de trabajo y realiza la evaluación del trabajo en equipo en cada simulación.

A partir de la siguiente figura se explica el comportamiento de los diferentes agentes en la arquitectura del SMA.

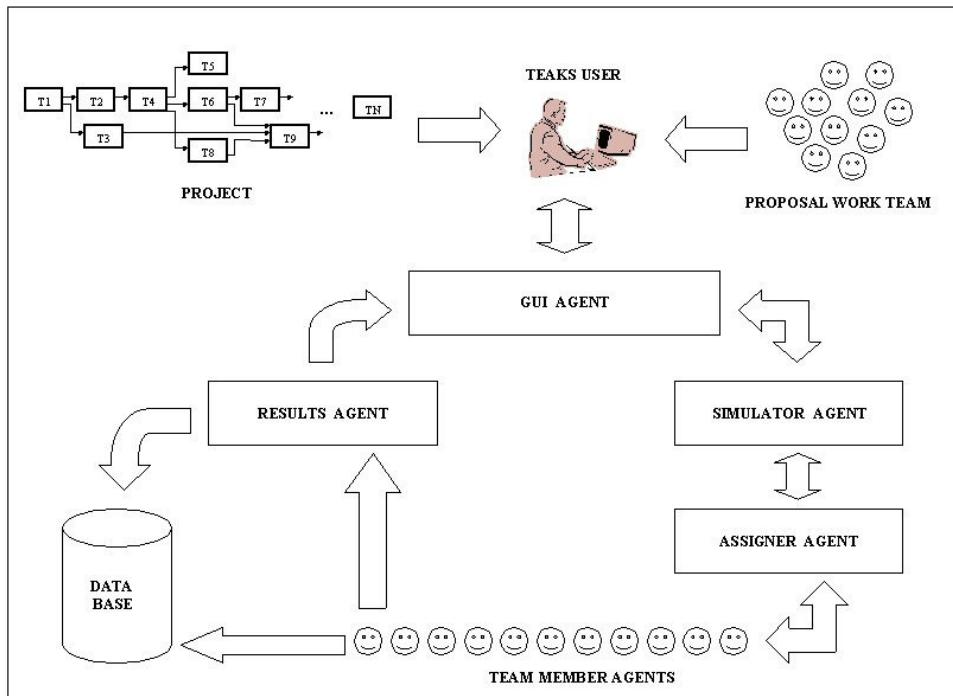


Figura 4 - Arquitectura del SMA

A través del agente Interfaz, se selecciona una primera propuesta de equipo de trabajo a partir de las características reales de las personas y de las tareas que forman un proyecto. Cuando la simulación comienza, el agente Simulador toma el control del número de simulaciones y en cada simulación el agente Asignador informa a cada miembro del equipo qué tareas del proyecto debe realizar. Los resultados del rendimiento de cada miembro del equipo se informan al agente Resultado. Esa información se muestra al usuario a través del agente Interfaz y posteriormente puede ser guardada en el disco.

1.4.3- Prototipo del SMA

El prototipo del SMA desarrollado se divide en tres módulos: el de configuración, simulación y el de resultados.

En el modelo de configuración es donde el usuario configura las características internas de los miembros del equipo y selecciona el rol del agente que quiere crear. Después, asigna los valores de cada una de las características internas a partir de los valores reales. Una vez el equipo está configurado, se hace lo mismo con el proyecto (*Figura 5*).

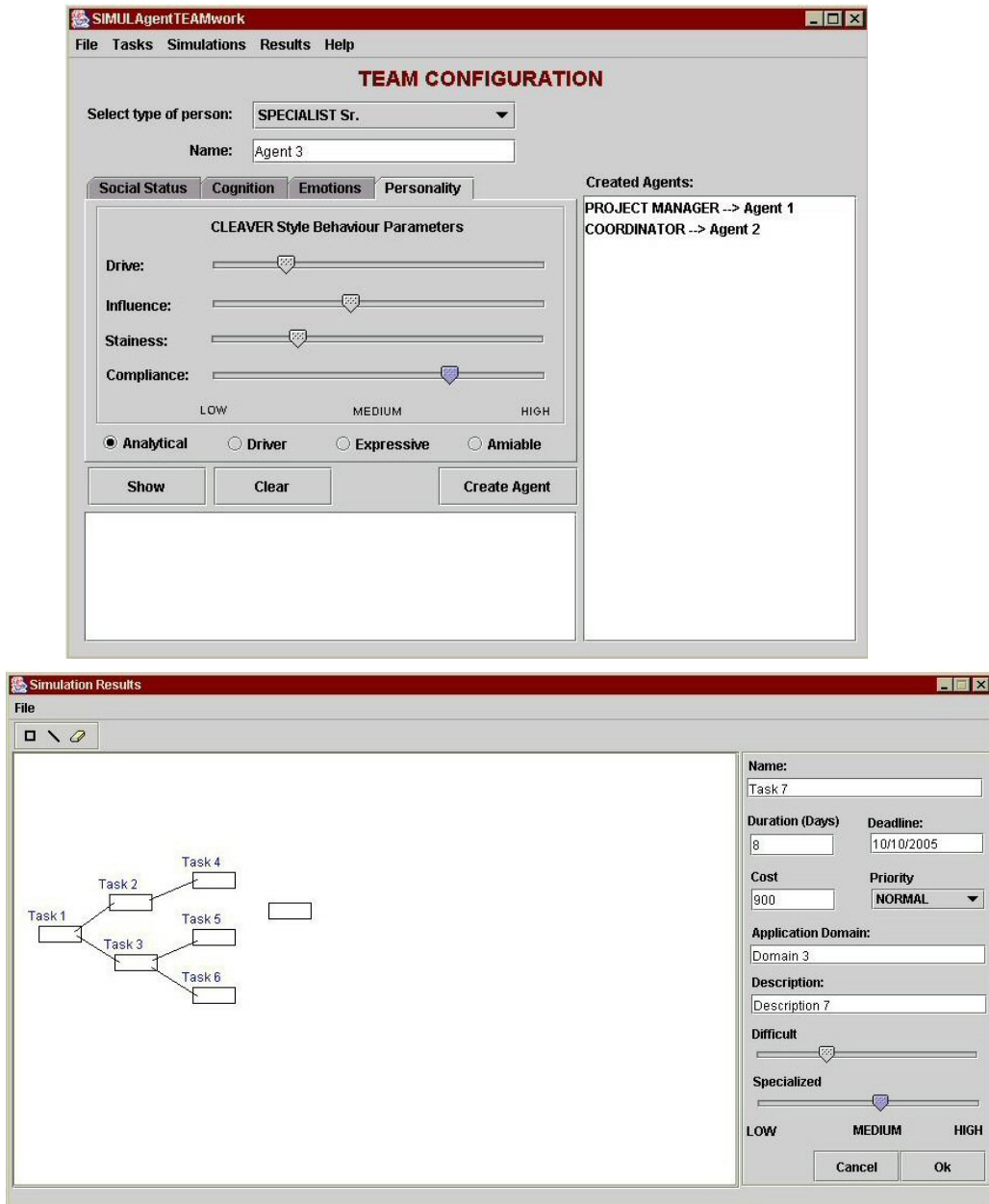


Figura 5 - Configuración del equipo de trabajo y del proyecto

En el módulo de simulación, se asignan las tareas del proyecto a cada uno de los miembros del equipo (a uno o más agentes) y el agente Asignador durante la ejecución distribuye cada tarea a su correspondiente agente. Después el usuario puede empezar el proceso de simulación estableciendo el número de simulaciones.

Una vez se han acabado todas las simulaciones, en el módulo de resultados, se puede observar los resultados estadísticos del equipo de trabajo.

El usuario puede analizar la información generada y también puede volver al módulo de configuración para añadir o eliminar un miembro del equipo, reasignando las tareas o cambiando la configuración del proyecto y ejecutando nuevas simulaciones para observar las diferencias. Estos análisis ayudan al usuario a decidir qué configuración es la más adecuada para un proyecto específico.

Una parte importante del SMA descrito, es la comunicación que debe haber entre los diferentes tipos de agentes (agentes del equipo de trabajo y los agentes del sistema) para poder realizar la simulación de manera correcta. Con tal de mejorar la comunicación existente, se propone la creación de una ontología construida a partir de los conceptos utilizados por los agentes del sistema. Antes de explicar la comunicación existente entre los agentes y justificar la necesidad de agregar una ontología, en la siguiente sección se explicará brevemente los conceptos básicos de una ontología.

1.5- Ontología

Podemos definir una ontología como el resultado de seleccionar un dominio o contexto (del mundo real que se quiere representar) y aplicar sobre él un método para obtener una representación formal de los conceptos que contiene y las relaciones que existen entre los mismos. Lo que incluye la Ontología es un vocabulario de términos para referirse al dominio. [Sanchez-Alonso, 2003]

Por tanto, una Ontología clasifica toda la información de un dominio como lo haríamos si utilizásemos una metodología orientada a objetos. Es decir, definiendo una serie de conceptos que se representan en clases con sus correspondientes atributos [Noy and McGuinness, 2001].

Una Ontología juega un papel importante en la comunicación entre agentes que componen un SMA, puesto que se necesita una conceptualización de un dominio para que sirva de estándar de comunicación entre los agentes y así cada agente pueda entender lo que los otros agentes del sistema le están diciendo.

Para poder llevar a cabo esta comunicación es necesario definir el significado de los elementos que se utilizarán en la sintaxis del contenido de los mensajes. Los elementos de una ontología son los siguientes:

Concepto: expresiones que representan una estructura compleja definidas en términos de slots.

Predicado: sirve para hacer una consulta sobre la existencia de objetos que cumplen unas características determinadas.

Acción: especifica las acciones que pueden realizar los agentes. [Caire, 2002].

Los requisitos de una ontología son:

- 1- Claridad: una ontología debe poder comunicar de manera efectiva el significado de sus términos. Las definiciones deben ser objetivas y comentadas en lenguaje natural.
- 2- Coherencia: una ontología debe de expresar un significado que sea consistente con las definiciones de los conceptos, y si a una ontología se le realiza algún tipo de razonamiento o inferencia éste no debe trastocar el significado de los conceptos que inicialmente se tenía.
- 3- Extensible: una ontología debe anticipar uso puesto que una ontología se puede reutilizar fácilmente y por tanto debe estar abierta a la inserción de nuevo conocimiento en ella.
- 4- Sesgo de codificación mínimo: debe especificar al nivel de conocimiento, sin depender de una codificación particular a nivel de símbolo, por tanto la ontología debe ser independiente del lenguaje representado y la base de conocimiento debe ser igual para cualquiera de los lenguajes.

1.6- Conclusiones

En este capítulo se ha presentado la definición de agente y SMA, la descripción de las especificaciones FIPA y el modelo TEAKS creado para simular el comportamiento humano en un equipo de trabajo al realizar un proyecto determinado, donde se han destacado las características humanas, las medidas de desempeño de los componentes del equipo y el modelo del proyecto a realizar. Otro aspecto importante que se ha descrito ha sido la arquitectura del modelo donde los agentes interactuarán así como una

breve descripción de la interfaz gráfica del modelo. También se han presentado los conceptos más importantes para la comprensión del modelo, así como la definición de una ontología para poder llevar a cabo mi trabajo realizado en este proyecto final de carrera.

En el siguiente capítulo se especificará el trabajo realizado para mejorar la comunicación existente entre los agentes del sistema mencionado en este capítulo, así como la utilización de una ontología para la mejora de la comunicación.

CAPÍTULO II: ESPECIFICACIONES

2.1- Introducción

Después de haber hecho una breve descripción del trabajo realizado en la tesis doctoral y presentados algunos conceptos importantes, nos centraremos en el SMA y específicamente analizaremos la comunicación existente entre los agentes del sistema. De esta manera se presentarán las limitaciones en esta comunicación y se justificará la necesidad de crear una ontología para mejorar esta comunicación inicial.

En este capítulo también se hará una breve explicación de cómo a través de las reglas difusas se obtiene el posible comportamiento del equipo a partir de la combinación de las características personales de los integrantes del equipo y de las características de las tareas que les han sido asignadas.

2.2- Comunicación existente

Inicialmente, en el sistema había implementado un único protocolo de comunicación: el FIPA Request Protocol [*FIPA Request Interaction Protocol Specification*]. Este protocolo era utilizado para comunicar desde el número de simulaciones elegidas por el usuario, hasta la asignación de tareas por parte del Agente Asignador a cada Agente del Equipo de Trabajo.

Una parte a destacar es que en esta comunicación inicial no existía una manera de comunicar los conceptos utilizados en el modelo entre los agentes del sistema. Para explicar esto, a continuación se presenta el diagrama del protocolo de comunicación y el proceso que llevan a cabo cada uno de los agentes involucrados:

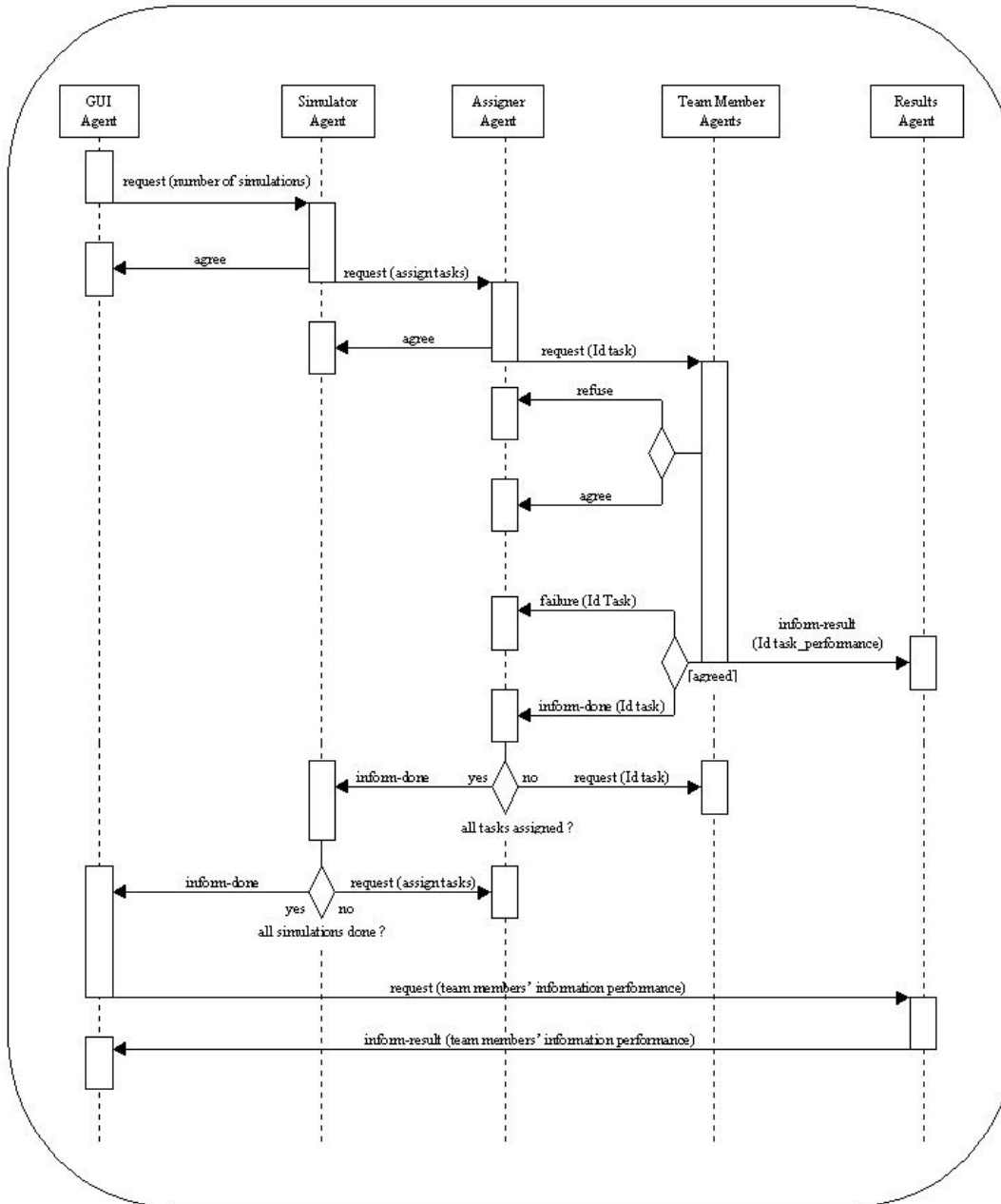


Figura 6 - Protocolo de comunicación *FIPA Request Protocol*

El protocolo de comunicación comienza cuando el agente Interfaz envía un mensaje del tipo *request* al agente Simulador con el número de simulaciones que el usuario ha solicitado. A continuación, el agente Simulador envía un mensaje del tipo *request* al agente Asignador solicitando que se asignen las tareas a los Agentes del Equipo de Trabajo (Team Member Agents). Entonces, el agente Asignador debe encargarse de asignar las tareas a los trabajadores. Para llevarlo a cabo, existe un vector global creado

en el momento en que el usuario crea el proyecto a partir de los parámetros de cada tarea. Este vector contiene todas las tareas que se han de realizar, los punteros a las tareas siguientes y anteriores (para saber y llevar un control sobre las tareas que se han de ir desarrollando en cada momento) y el nombre de los agentes que van a realizar la tarea que les fue asignada. El agente Asignador busca en el vector la tarea que se quiere desarrollar y una vez sabe cual es la tarea, envía un mensaje del tipo *request* con el identificador de la tarea a los correspondientes candidatos del equipo de trabajo (Team Member Agents). Estos se encargarán de acceder al vector para buscar las características de las tareas que han de realizar y las características de los agentes con los que han de trabajar. Es necesario saber con quienes van a trabajar ya que cuando a un posible trabajador (Worker Agent) se le asigna una tarea, su comportamiento no solo está determinado por las características de la tarea asignada sino también por las características de los compañeros con los que trabaja en esa tarea, que también influyen en su desempeño.

Cuando los miembros del equipo de trabajo han acabado las tareas que le había asignado el agente Asignador, envían un mensaje del tipo *inform-done* con el identificador de la tarea al agente Asignador conforme han acabado de realizar las tareas que les fueron asignadas y a continuación, cada trabajador envía un mensaje del tipo *inform-results* al agente Resultado con el identificador de la tarea. A través de en un fichero compartido por todos los miembros del equipo de trabajo (Team Member Agents), los agentes trabajadores escribirán los resultados de su rendimiento en cada una de las tareas que le fueron asignadas.

Cuando el agente Asignador recibe el mensaje del tipo *inform-done*, busca en el vector la siguiente tarea que se ha de realizar y por consiguiente, los miembros del equipo a los cuales les ha de asignar esa tarea. Después, vuelve a enviar otra vez el mensaje del tipo *request* asignando la tarea a todos los agentes que han de realizarla. Este proceso se repetirá hasta que todas las tareas hayan acabado.

Una vez las tareas han acabado, el agente Asignador envía un mensaje del tipo *inform-done* al agente Simulador informándole que la tarea se ha realizado. Este proceso se repetirá tantas veces como el número de simulaciones que el agente Interfaz ha solicitado. Cuando el agente Simulador observe que ya se han hecho todas las

simulaciones que se habían pedido, envía un mensaje del tipo *inform-done* al agente Interfaz.

Una vez hecho esto, el agente Interfaz envía un mensaje del tipo *request* al agente Resultado pidiéndole todos los resultados de los miembros del equipo de trabajo para que después el usuario pueda mostrarlos gráficamente. Para obtener los resultados, el agente Resultado deberá acceder al fichero compartido.

Esta era en líneas generales la comunicación llevada a cabo por los agentes hasta antes de incorporar el trabajo desarrollado en este PFC. En la siguiente sección se describirán las mejoras que se han producido al crear la ontología y añadir sus conceptos dentro de los mensajes de los agentes, así como la incorporación de nuevos mensajes utilizando el FIPA Contract Net Protocol.

2.3- Mejoras a la comunicación existente

Las mejoras que se han producido en la comunicación existente a través de la ontología y la incorporación del FIPA *Contract-Net protocol* se explican a continuación.

2.3.1- Agregado de una Ontología

En un SMA es fundamental que cada agente pueda entender lo que le dicen los otros agentes del sistema. Para ello, lo que se quiere es que los agentes puedan acceder a la información que necesitan, a través del paso de mensajes, sin necesidad de tener que ir a buscarla en un vector global o en un archivo compartido que es como se venía haciendo hasta el momento.

Con el desarrollo de la ontología, ya no será necesario que el trabajador (Worker Agent) una vez reciba el identificador de la tarea vaya a buscar las características de su tarea asignada en el vector. Ahora, simplemente pasándole la tarea a través de la ontología, el trabajador ya sabrá todos los parámetros de la tarea, sin la necesidad de acceder al vector. Con esto, ya no será necesario que todos los agentes trabajadores creados en el sistema accedan al vector, ya que el único que debe acceder a la información del vector

es el agente Asignador, quien buscará la tarea a realizar y utilizando los conceptos definidos en la ontología se la pasará al candidato asignado para que la haga.

Otra de las mejoras que se han hecho es respecto a la utilización de un fichero compartido. Es decir, cuando el trabajador (Worker Agent) acababa una tarea, éste accedía a un fichero compartido por todos los candidatos del equipo (Team Member Agents) y escribía en él los resultados obtenidos por el agente en una determinada tarea, con los respectivos controles para acceder un recurso compartido evitando que un agente pudiera escribir en el fichero hasta que no hubiera acabado el agente que estaba escribiendo en ese momento. Ahora, ya no tendrán la necesidad de escribir los resultados en el fichero, simplemente a través de la ontología, el trabajador enviará al agente Resultado sus resultados generados en esa tarea. Y por último, el agente resultado los guardará en una lista esperando que el proceso completo de simulación termine para mostrarlos a través del Agente Interfaz.

Con esta última mejora también se conseguirá que el agente Interfaz no tenga que leer del fichero global, en el que todos los miembros del equipo escribían sus resultados. Ahora, el agente Resultado le pasará esa lista utilizando la ontología.

2.3.2- Extensión de los protocolos de comunicación

Otra mejora hecha al SMA ha sido la manera de saber los valores del estado interno los compañeros con los que un agente trabaja en una tarea determinada.

En la implementación inicial, cada agente del equipo tenía acceso a las características internas de sus compañeros de tarea mediante el nombre del agente que estaba guardado en el vector global. El riesgo de este acceder a este vector, es que cualquier agente del equipo podía tener acceso a las características internas de cualquier otro agente, aún no siendo compañero de la misma tarea. Con la implementación del nuevo protocolo *Request Protocol* ya no se tendrá acceso a las características de todos los agentes. Ahora será el propio trabajador (Worker Agent) quien a través de este protocolo y la ontología creada obtendrá las características internas de los compañeros con los que trabaja en una tarea determinada tal y como se explicará en el capítulo 3.

Otra mejor realizada en este PFC al modelo de comunicación existente, es a través de la implementación de un nuevo protocolo de comunicación: el *Contract-Net Protocol*.

Este protocolo se ha añadido para mejorar el modelado de los factores externos que afectan al desempeño normal del equipo.

Existen dos tipos de factores externos que afectan al desempeño de un agente:

Factores externos que afectan al proyecto: Algunas decisiones tomadas por la empresa donde el equipo trabaja o las decisiones tomadas por el cliente sobre el producto o servicio final normalmente afectan el comportamiento del equipo. Muy a menudo el motivo de esas decisiones es debido a la modificación total o parcial de algunas tareas, incluso cuando esas tareas fueron realizadas correctamente. Para modelar esas modificaciones de la tarea en una simulación se utiliza el parámetro de la duración de una tarea.

Factores externos que afectan al agente: todos los factores externos que afectan al comportamiento individual de las personas mientras están haciendo algunas tareas (temporalmente o durante todo el proyecto). Las causas de esos factores externos tienen que ver con la vida real (tales como problemas familiares, problemas financieros y algunos accidentes físicos entre muchos otros).

Estas dos características afectan al modelo de comunicación existente pues cuando una tarea no es terminada por el agente al que fue asignada, el agente Asignador necesita reasignarla a cualquier otro agente disponible del equipo. En esta nueva asignación es donde se implementa el nuevo Contract-Net Protocol para solventar este problema.

2.4- Proceso de obtención del comportamiento del agente

El posible comportamiento de un agente dentro del modelo desarrollado en la tesis se obtiene mediante reglas difusas, las cuales utilizan valores difusos para representar los principales aspectos del comportamiento humano en el trabajo. Estos aspectos son las características internas del agente (personalidad, estado emocional, características sociales, características cognitivas, y por último el estrés), los parámetros de la tarea que influyen en el comportamiento humano (dificultad, tipo y duración de tarea), y los parámetros para medir el resultado del comportamiento del agente (Nivel de objetivos conseguidos en su(s) tarea(s) asignada(s), tiempo en el desarrollo de una tarea, calidad

del rendimiento final obtenido en la tarea, nivel de colaboración del equipo, nivel de contribución individual a sus tareas asignadas y nivel de supervisión requerido).

El proceso para la obtención del posible comportamiento de los integrantes del equipo en el modelo TEAKS es a través de la ejecución de dos diferentes conjuntos de reglas difusas:

El primer conjunto de reglas difusas utiliza como premisas los valores de las características sociales, cognitivas, y de personalidad de los agentes combinados con los valores de la tarea que tiene asignada. La consecuencia de la ejecución de estas reglas es la modificación de los valores emocionales y en el estrés en cada uno de los agentes. Un ejemplo de estas reglas se muestra a continuación:

SI tenemos una tarea que presenta una dificultad *baja* y el agente al que se le asigna dicha tarea tiene el rol de *especialista* con un nivel de *experiencia alto* ENTONCES

El deseo por hacer esa tarea será *bajo*

El interés por hacer esa tarea será *bajo*

El disgusto que le producirá hacer esa tarea será *alto*

La ansiedad que le producirá hacer esa tarea será *baja*

El estrés que le causará hacer esa tarea será *bajo*

El segundo conjunto de reglas difusas son las que generan los valores para cada uno de los seis parámetros que miden el comportamiento de un agente sobre su tarea asignada. Los parámetros utilizados en las premisas de este segundo conjunto de reglas son los valores en las emociones del agente, el estrés y su tipo de personalidad. Un ejemplo de estas reglas se muestra a continuación:

SI tenemos un agente con un nivel de *interés alto* sobre una tarea concreta, una experiencia *alta* y un estrés *bajo* ENTONCES

El nivel de objetivos conseguidos en dicha tarea será *medio-alto*

El tiempo en el desarrollo de la tarea será *alto porcentaje de avance*

La calidad del rendimiento final obtenido en la tarea será *media-alta*

El nivel de colaboración del equipo será *normal*

El nivel de contribución individual al equipo en esa tarea será *medio*

El nivel de supervisión requerido en esa tarea será *medio-bajo*

Actualmente para validar el modelo TEAKS se ha utilizado un caso de estudio tomado realizado en el Instituto Mexicano del Petróleo [Martínez Miranda et al., 2005b]. Para realizar esta validación se ha obtenido información sobre las características de un equipo de trabajo encargado de realizar un proyecto para el diseño, desarrollo e implementación de un Sistema de Información Geográfica. Para este caso de estudio se tienen implementadas alrededor de 400 reglas difusas que generan el posible comportamiento de este equipo de trabajo.

Estas reglas difusas están escritas en el lenguaje JESS dentro de un fichero de texto y para hacer cualquier modificación sobre ellas se debe modificar este fichero y conocer la sintaxis de JESS. En este PFC se presenta también el trabajo desarrollado para crear una interfaz gráfica que permite modificar estas reglas difusas desde el mismo SMA y sin necesidad de conocer la sintaxis de JESS, para que sea el usuario mismo del sistema quien sea capaz de configurar las reglas que generan el posible comportamiento de acuerdo a las características del equipo y proyecto que quiera analizar.

2.5- Conclusiones

En este capítulo se ha explicado la comunicación existente entre los agentes del sistema, así como sus carencias, la necesidad de la creación de una ontología, y la extensión de los protocolos de comunicación con la incorporación del *FIPA Contract-Net protocol* y el *Request Protocol* para mejorar la comunicación inicial. Con estas nuevas características se llegará a una mejora de la comunicación existente.

En este capítulo también se ha presentado una breve explicación del proceso utilizado en el modelo TEAKS para simular el comportamiento de un agente a través del uso de reglas difusas.

El diseño de estos cambios realizados en este PFC al modelo de comunicación existente, así como el diseño para la construcción de una interfaz gráfica para la configuración de las reglas difusas dentro del SMA es explicado en el siguiente capítulo.

CAPITULO III: DISEÑO

3.1- Introducción

A continuación se van a describir los elementos que se han identificado para construir la ontología, los cambios que esta ontología ha aportado al sistema y la extensión de los nuevos protocolos de comunicación que se han añadido. Este capítulo también describirá la necesidad del usuario para poder configurar las reglas dentro del SMA y cuál ha sido mi trabajo realizado para cumplir este objetivo.

3.2- Diseño de la ontología

De acuerdo al modelo implementado en la tesis, la ontología desarrollada en este PFC para mejorar la comunicación entre los agentes es la que se presenta en la siguiente figura:

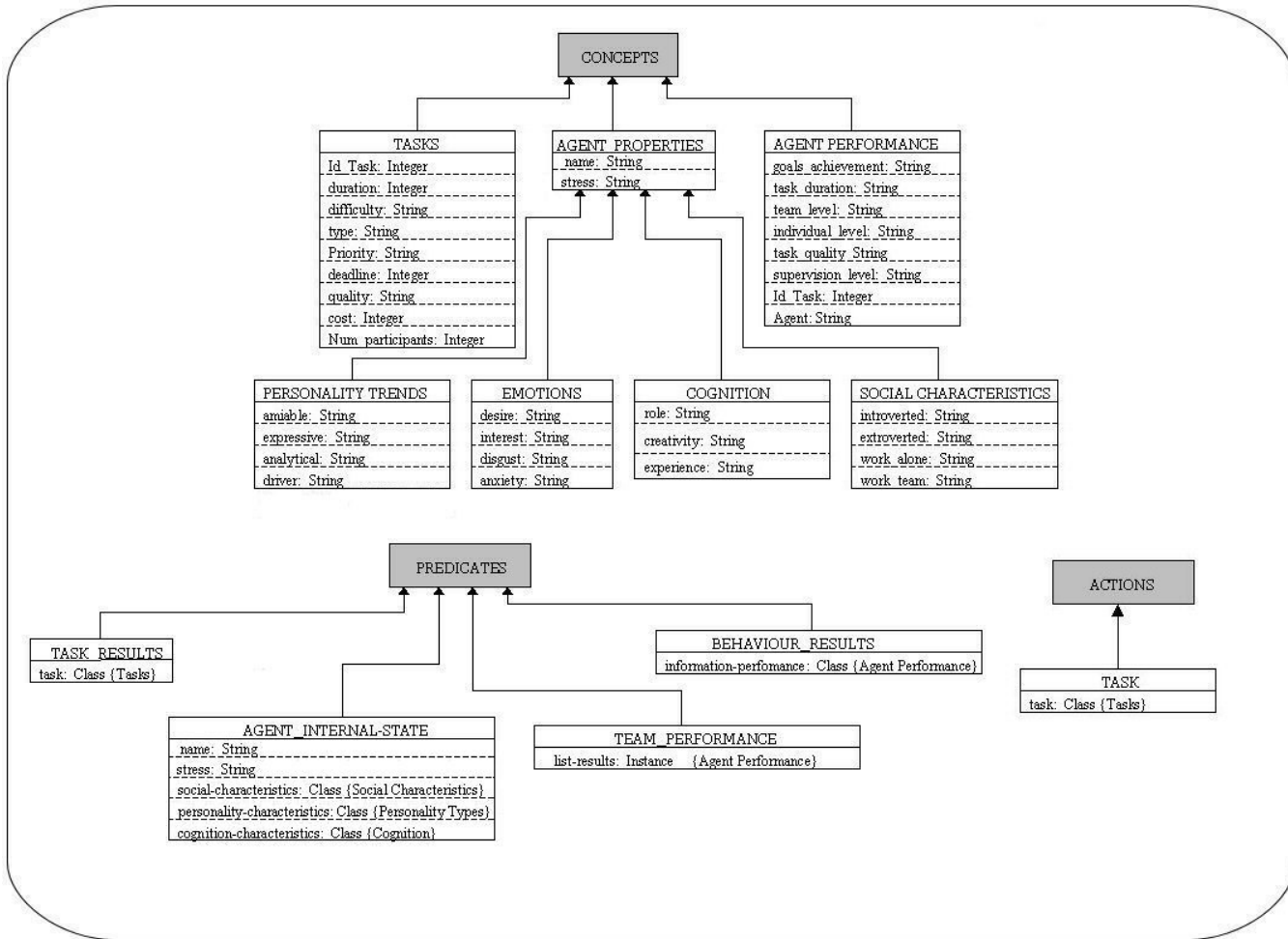


Figura 7 - Diseño de la Ontología

La herramienta de software utilizada para implementar esta ontología ha sido mediante el editor de ontologías Protégé [Protege, 2000], el cual permite la creación de las ontologías de manera gráfica e independientemente del lenguaje que se utiliza.

Este editor dispone de un plug-in (BeanGenerator) que genera el código correspondiente para una ontología de JADE.

La descripción de cada uno de los elementos de la ontología mostrados en la *figura 7* son los siguientes:

- Conceptos:

Tasks: contiene todas las características de las tareas de acuerdo al modelo de *tarea* presentado en la tesis. A todas estas propiedades se les llaman slots. Por ejemplo, el tipo slot *difficulty* está definido como *String* ya que para este tipo de información no se puede medir como un valor exacto sino que se ha de medir según si es alto, medio o bajo, por eso el uso de la lógica difusa. Para el resto de slots (*Type*, *priority* y *quality*) del tipo *String* ocurre exactamente lo mismo.

El otro tipo definido es el tipo *Integer*, como por ejemplo el slot *id_task* ya que como identificador de la tarea ha de ser un valor numérico y diferente para cada una de las tareas.

Agent_Performance: contiene los resultados obtenidos por los agentes al realizar una tarea determinada. Por ejemplo, todos los slots de este concepto (*goals_achievement*, *task_duration*, *team_level*, *individual_level*, *task_quality*, *supervisión_level* y *agent*) están definidos como *String* porque los resultados obtenidos se han de medir según el parámetro difuso alto, medio o bajo.

Agent_Properties: es una superclase que contiene las características internas del agente. Por ejemplo, el tipo de slot *name* definido es un *String*, ya que contiene el nombre del agente y el slot *stress*, también definido como *String*, parámetro que se ha tenido en cuenta en la obtención del posible desempeño del equipo. Esta superclase se divide en cuatro subclases según los siguientes aspectos:

En el caso de la personalidad está definida la subclase *personality_trends*, donde el tipo de los slots definidos es *String* (*amiable*, *expressive*, *analytical*, y *drive*) porque son características que se han de medir según si son altas, medias o bajas.

En el caso de las emociones está definida la subclase *emotions*, donde el tipo de los slots definidos también es del tipo *String* (*desire*, *interest*, *disgust* y *anxiety*) por los mismos motivos que en el caso de la personalidad.

En el caso de las características sociales está definida la subclase *social_characteristics*, donde el tipo de los slots definidos también es del tipo *String*

(*introverted*, *extroverted*, *work_alone* y *work_team*) por los mismos motivos que en el caso de la personalidad y el de las emociones.

En el último caso, el del aspecto cognitivo, está definida la subclase *cognition*, donde el tipo de los slots definidos también es del tipo *String* (*creativity* y *experience*) por los mismos motivos que en el caso de la personalidad, de las emociones y características sociales. El slot *role* es del tipo *String* pero contiene los diferentes roles que puede tener un agente. Los diferentes roles que pueden ser asignados a los agentes son coordinador, director, ayudante, técnico y especialista.

- Predicados:

Tasks_Results: contiene el slot *task* que es una referencia a la clase *Task* y se utiliza en el mensaje *failure* (figura 8) que los miembros del equipo de trabajo envían al agente Asignador cuando éstos no han llevado a cabo las tareas que le habían asignado. Y así el agente Asignador es capaz de consultar los parámetros tanto de la tarea realizada como la no realizada. También se utiliza en el protocolo que se ha añadido a la implementación del modelo, el *Contract-Net Protocol*, en los casos en que un trabajador (Worker Agent) se propone para realizar la tarea que tenía asignada otro agente y que por un factor externo no la ha llevado a cabo, así como cuando el agente Asignador acepta la propuesta (figura 8.1).

Agent_Internal_State: contiene tres slots con referencia a las subclases *social_characteristics*, *cognition_characteristics* y *personality_characteristics* respectivamente. Además de dos slots del tipo *String*, *name* y *stress*. El primero hace referencia al nombre del agente y el segundo es del tipo *String* ya que para este tipo de información no se puede medir como un valor exacto. Se utiliza en el *Request Protocol* que se ha añadido en la implementación del modelo, donde a través de un *inform* (figura 8.2), un trabajador (Worker Agent) le puede interesar en un cierto momento consultar el estado interno y el nombre de los otros compañeros de trabajo que desarrollan la misma tarea que él.

Team_Performance: contiene el slot *list_results* que hace referencia a la clase *Agent_Performance*, que es el rango de ese slot. Es decir, el slot contiene una lista

con los resultados del rendimiento generado por todos los agentes en las tareas asignadas. La cardinalidad del slot es múltiple, permitiendo así más de un resultado generado por el comportamiento del agente en una tarea determinada.

Se utiliza a través de un *inform* (figura 8) para que una vez el agente Interfaz haya pedido la lista con los resultados globales de todos los agentes, el agente Resultado le envíe la lista y el agente Interfaz pueda consultar los resultados generados a través de esa lista.

Behaviour_Results: tiene el slot *information_performance* y hace referencia a la clase *Agent_Performance*. Se utiliza a través de un *inform* (figura 8) para que cuando los miembros del equipo de trabajo hayan acabado la tarea asignada, avisen al agente Resultado de los resultados generados de un agente en una tarea determinada. Así el agente Resultado podrá consultar esos resultados y añadirlos a una lista para después poderse la enviar al agente usuario.

- Acción: utilizada para especificar el servicio que un agente puede ofrecer.

Task: contiene el slot *task* que hace referencia a la clase *Tasks*. Se utiliza cuando el agente le pide a un miembro del equipo que haga una determinada tarea a través de un *request* (figura 8). Así el trabajador (Worker Agent) recibe la tarea que le ha ofrecido el agente Asignador. También se utiliza en el *Contract-Net Protocol* cuando el agente Asignador hace un *cfp* pidiendo que hagan la tarea que no ha podido realizarse a los trabajadores que comparten la tarea (figura 8.1).

3.3- Protocolos de comunicación

La utilización del protocolo *Request* es utilizado cuando un agente le asigna una serie de tareas a los integrantes del equipo de trabajo.

Después de añadir la ontología, la explicación de la comunicación (*Request Protocol*) queda como sigue a continuación:

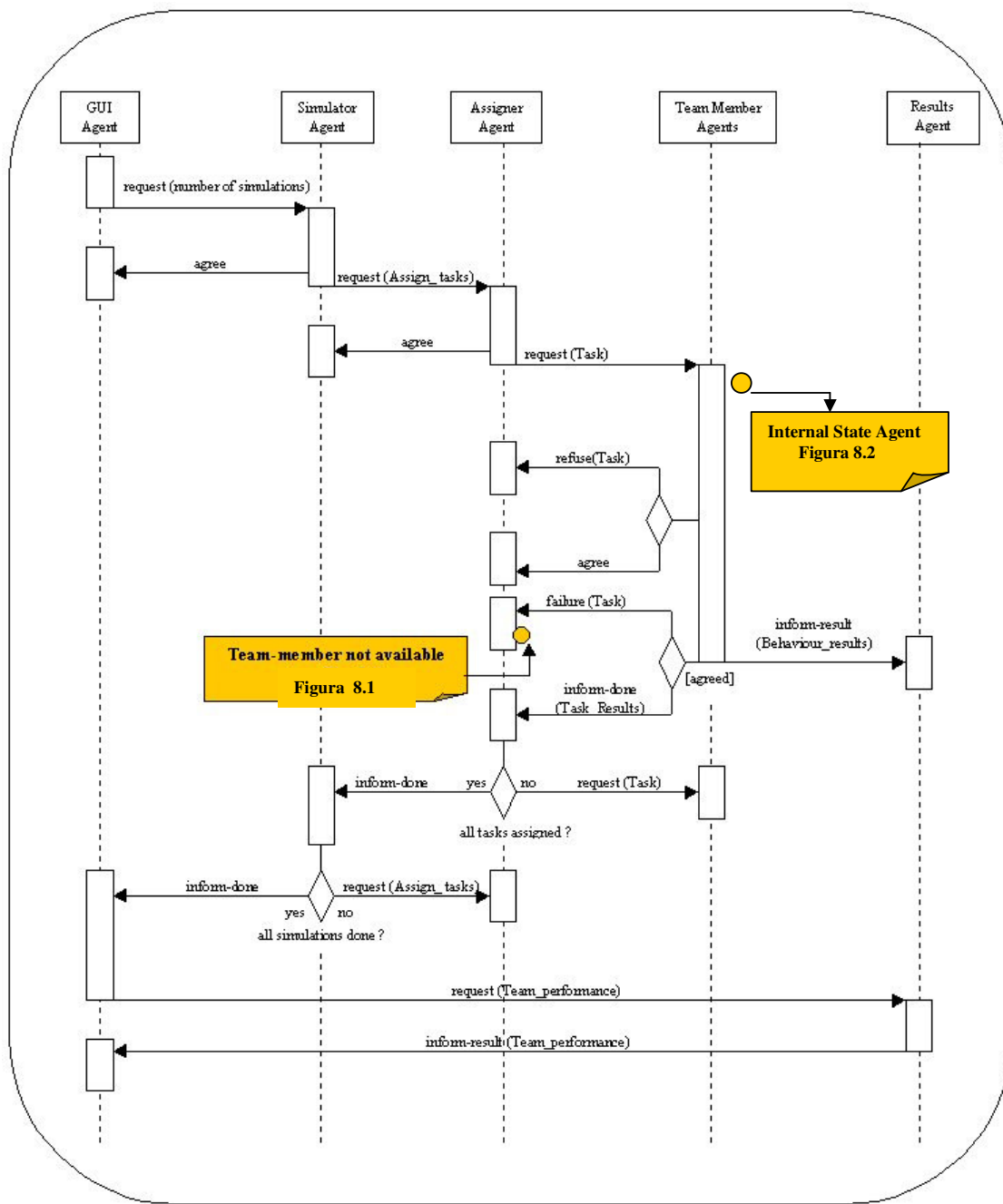


Figura 8 - Protocolo de comunicación *FIPA Request Protocol*

El protocolo de comunicación comienza cuando el agente Interfaz envía el tipo de mensaje *request* al agente Simulador donde el contenido del mensaje es el número de simulaciones que el usuario ha solicitado. A continuación, el agente Simulador envía un mensaje tipo *request* al agente Asignador solicitando que se asignen las tareas a los agentes del equipo de trabajo (Team Member Agents). Entonces, el agente Asignador

envía un mensaje del tipo *request* con la tarea a asignar, a los correspondientes miembros del equipo de trabajo. Gracias a la ontología, los miembros del equipo podrán acceder a los parámetros de la tarea asignada por el contenido del mensaje.

Cuando los miembros del equipo de trabajo han acabado las tareas que le había asignado el agente Asignador, entonces cada trabajador (Worker Agent), envía un mensaje del tipo *inform-done* al agente Asignador, con el contenido de la tarea, conforme han acabado de realizar las tareas que les fueron asignadas y a continuación, cada trabajador envía un mensaje del tipo *inform-results* al agente Resultado con los resultados obtenidos en tarea que le fue asignada.

Cuando el agente Asignador recibe el mensaje tipo *inform-done*, busca la siguiente tarea que se ha de realizar y por consiguiente, los miembros del equipo a los cuales les ha de asignar esa tarea. Entonces vuelve a enviar otra vez el mensaje del tipo *request*, asignando la tarea a todos los agentes que han de realizarla. Este proceso se repetirá hasta que todas las tareas hayan sido asignadas.

Una vez las tareas han terminado, el agente Asignador envía un mensaje del tipo *inform-done* al agente Simulador informándole que la tarea se ha realizado. Este proceso se repetirá tantas veces como el número de simulaciones que el agente Interfaz ha solicitado. Cuando el agente Simulador observe que ya se han hecho todas las simulaciones que se habían pedido, envía un mensaje del tipo *inform-done* al agente Interfaz.

Después de esto, el agente Interfaz envía un mensaje del tipo *request* al agente Resultado pidiéndole todos los resultados de los miembros del equipo de trabajo para mostrarlos gráficamente. Para ello, el agente Resultado envía la lista con un mensaje del tipo *inform* con los resultados globales generados.

3.3.1- Contract-Net protocol

Este nuevo protocolo se ha implementado para poder reasignar las tareas que un agente no ha podido finalizar en un determinado momento, (esto es el modelado de factores externos hecho en la tesis doctoral, por ejemplo por una baja laboral), y así conseguir

nuevos posibles comportamientos para la simulación del comportamiento de los equipos de trabajo. A continuación, se muestra el *Contract-Net protocol* que se utiliza para la negociación entre diferentes agentes, en este caso la negociación es para obtener el agente que realizará la tarea que ha quedado pendiente de ser acabada.

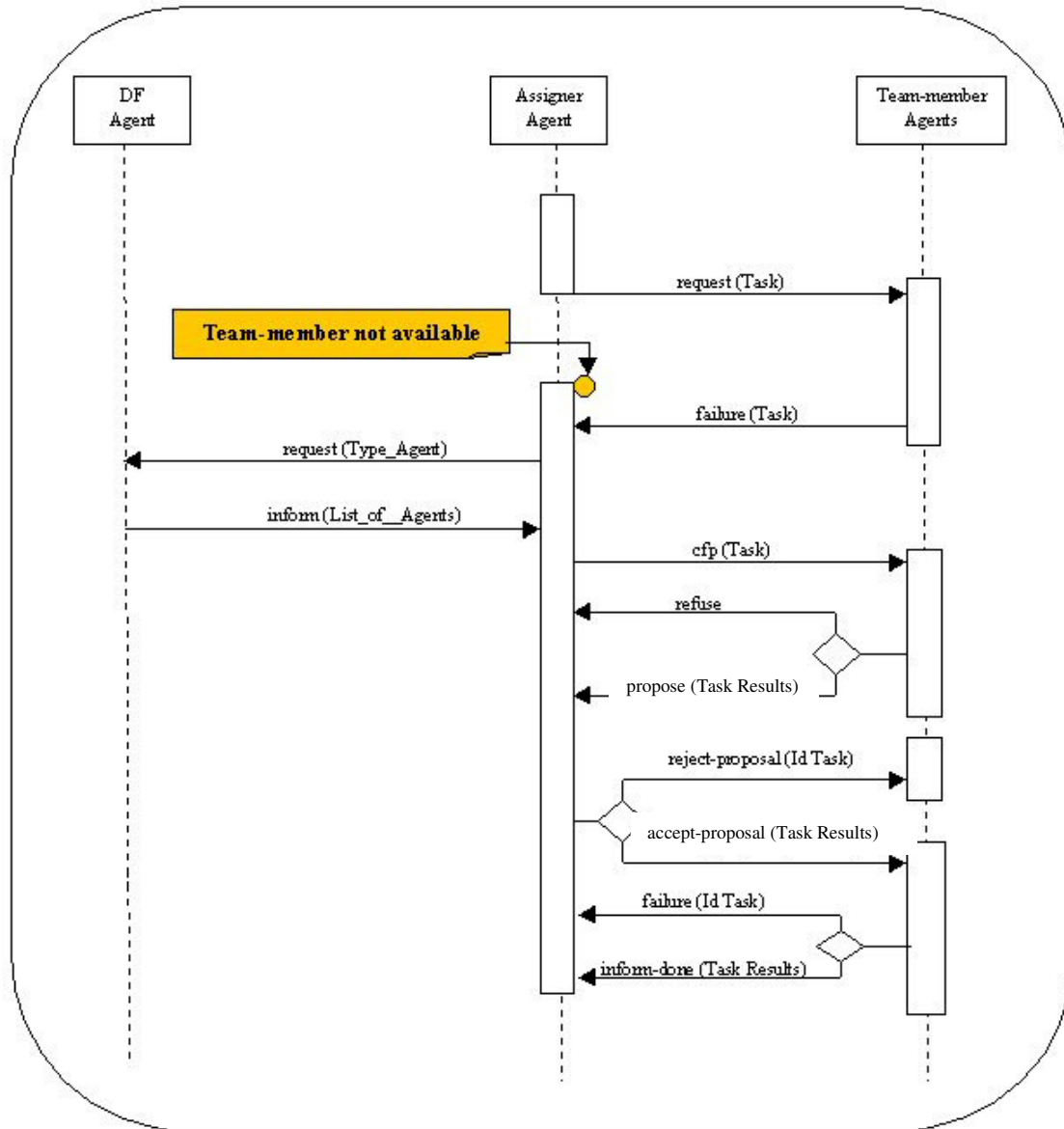


Figura 8.1 - Protocolo de comunicación *FIPA Contract-Net protocol*

El protocolo de comunicación inicia cuando un miembro del equipo de trabajo no ha podido realizar una determinada tarea, entonces, envía un mensaje del tipo *failure* con el contenido de la tarea al agente Asignador para que sepa que tarea no ha sido finalizada. Entonces, el agente Asignador, a partir de un criterio (por ejemplo aquellos agentes

cuyo rol dentro del equipo sea el de Técnico. En la tesis dependerá de las características de cada tarea), pide al DF un servicio a través de un mensaje del tipo *request*, para que le mande una lista con los agentes de un determinado rol (e.g. Técnico). El DF le informa a través de un *inform* de la lista de trabajadores que cumplen con el rol requerido para que con estos agentes comience la negociación (*Contract-Net Protocol*).

El agente Asignador manda un *cfp* a todos los agentes que cumplen con el rol solicitado. Según la implementación desarrollada en el proyecto final de carrera el rol solicitado es el de Técnico (una vez la parte desarrollada en el proyecto se incorpore a la tesis, la selección de los trabajadores a realizar la tarea a reasignar dependerá de las características de la tarea. Por ejemplo, si la tarea es de dificultad alta y nivel de especialización alto entonces el tipo de agente que se escogerá será Coordinador, según las reglas difusas implementadas en el modelo). Los agentes trabajadores que reciban el *cfp* podrán proponerse para hacer la tarea o rechazarla, según la implementación desarrollada en el proyecto final de carrera a partir de unos valores aleatorios. Una vez el agente Asignador reciba el *propose* con la tarea a realizar, este podrá hacer dos cosas, o aceptar la propuesta o rechazar la propuesta.

El criterio, según la implementación del proyecto final de carrera para aceptar la propuesta es que siempre aceptará al primero que se proponga, rechazando así al resto de los agentes que en su momento se propusieron para hacer la tarea.

Si el agente trabajador recibe un *acceptal_proposal* con la tarea a realizar podrán pasar otras dos cosas, o bien que el agente vuelva a fallar o que acabe la tarea que le fue asignada y envíe entonces un *inform* de la tarea acabada y continuar con el resto del proceso de la simulación.

3.3.2- Request Protocol

Este nuevo protocolo de comunicación se ha agregado para que un agente sepa las características de los agentes con quienes van a trabajar en una determinada tarea, ya que cuando a un trabajador (Worker Agent) se le asigna una tarea, su comportamiento no solo está determinado por las características de la tarea asignada sino también por las características de los compañeros con los que trabaja en esa tarea, que también influyen

en su desempeño. Este nuevo protocolo (*Request Protocol*) queda como sigue a continuación:

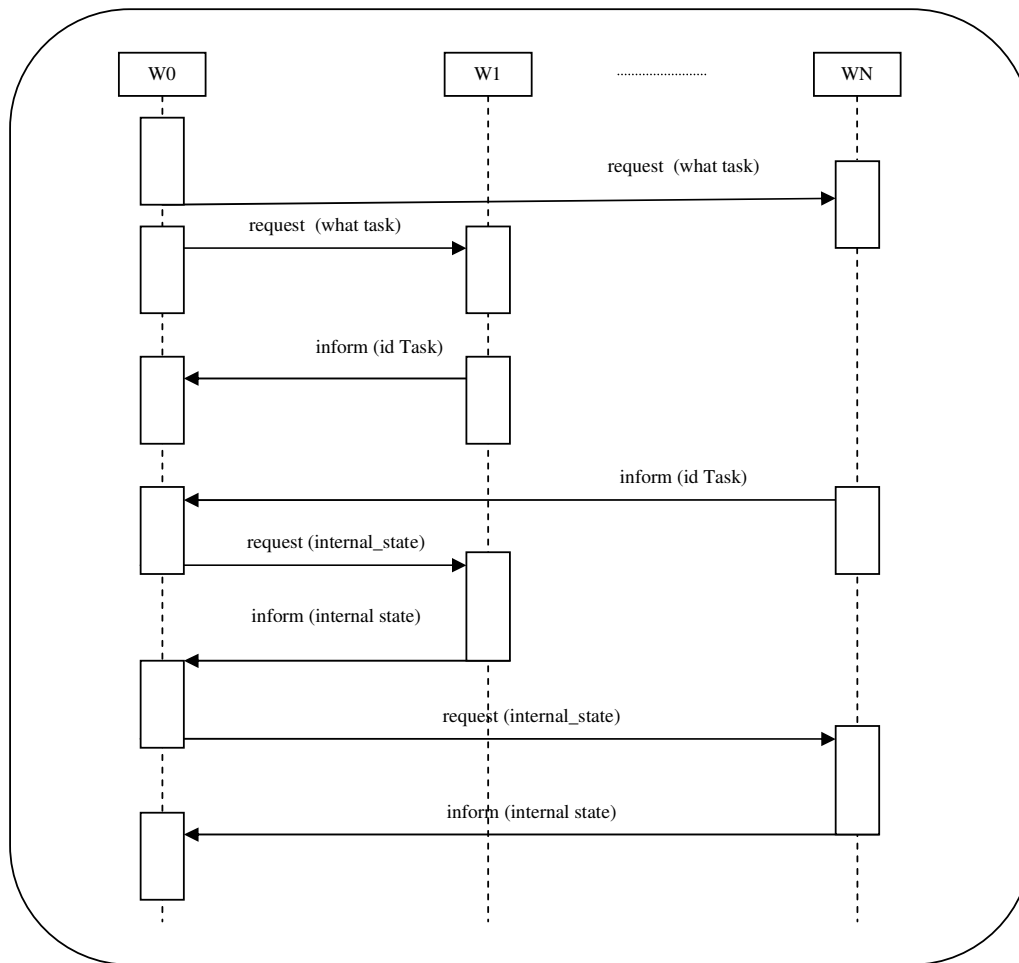


Figura 8.2 - Protocolo de comunicación *FIPA Request Protocol*

La comunicación inicia en el momento en el que el agente Asignador le manda a los miembros del equipo las tareas asignadas. Es entonces cuando a través de un mensaje del tipo *request*, un agente trabajador determinado busca cuales son los trabajadores que están desarrollando la misma tarea que él. Para llevar a cabo esto, envía un *request* a todos los miembros del equipo de trabajo, hagan o no la misma tarea, para así averiguar cuales de ellos trabajan con él. Entonces, todos los agentes le contestarán con un mensaje del tipo *inform* con el contenido del identificador de la tarea para que el agente pueda consultar y seleccionar los agentes que tienen la misma tarea que él, guardándolos en un vector. Una vez ya sabe quienes son, envía un *request* pidiéndoles

su estado interno. Y cada uno de ellos le contestará con un *inform* donde el contenido del mensaje será el estado interno para que el agente que había pedido el estado interno pueda consultar los parámetros que se pasan a través de la ontología.

Esto se hará para todos los agentes que tengan la misma tarea que el agente que hizo el *request*, todos aquellos que no tengan esa tarea no harán esta comunicación.

3.4- Interfaz gráfica para la configuración de las reglas difusas

Las reglas difusas creadas en la tesis para la simulación del comportamiento de los agentes están definidas para un proyecto y equipo concretos pero interesaría poder aplicarlo a cualquier tipo de caso de estudio. Estas reglas difusas están escritas en el lenguaje JESS dentro de un fichero de texto y para hacer cualquier modificación sobre ellas se debe modificar este fichero y conocer la sintaxis de JESS. Por eso mi trabajo se ha ampliado con la incorporación de una interfaz gráfica, para dar la posibilidad al usuario de configurar las reglas según el tipo de proyecto y equipo de trabajo que quiera analizar sin necesidad de conocer la sintaxis de JESS. Antes de comenzar con la explicación de esta interfaz, se presenta a continuación una ventana en la que se da la opción de poder seleccionar el tipo de configuración que se quiere realizar (figura 9). Esta ventana solo es útil para facilitar al usuario el acceso al tipo de configuración deseado para mi PFC, pero no se llegará a integrar dentro del SMA puesto que ya existe un menú de opciones en la cual se integrarán directamente.



Figura 9- Menú para la selección del tipo de reglas a configurar

Como se puede ver en la figura 9 aparece la posibilidad de poder seleccionar entre dos tipos de opciones:

Behaviour Rules que consiste en otra ventana y sirve para poder cambiar el estado interno del agente (el nivel de las emociones y el estrés) según sean las características de las tareas asignadas, el tipo de agente y su tipo de personalidad (Figura 10).

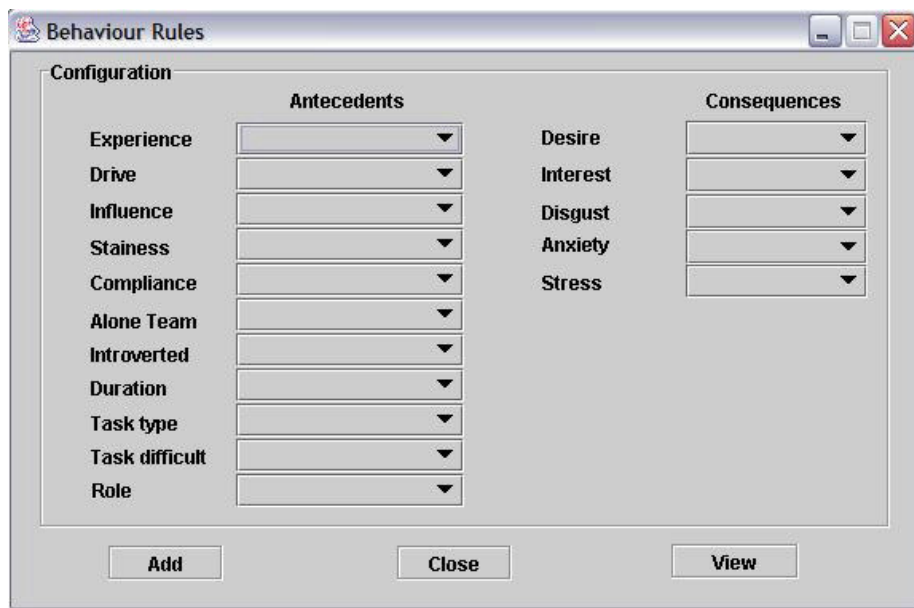


Figura 10 - Ventana para la configuración del estado interno del agente a partir de la tarea asignada, el rol y la personalidad

La segunda ventana *Task Rules* sirve para definir las reglas que generan los valores en los parámetros que miden el desempeño del agente en la tarea que tiene asignada a partir de su estado emocional, su tipo de personalidad, aspectos cognitivos y de las características de la tarea que simula realizar. (Figura 11).

Configuration	
Antecedents	Consequences
Creativity	Change Task Days
Experience	Change Task Quality
Drive	Change Goals Achievement
Influence	Change Team Collaboration
Stainess	Change Individual Contribution
Compliance	Change Required Supervision
Desire	
Interest	
Disgust	
Anxiety	
Stress	
Task type	
Task difficult	

Buttons: Add, Close, View

Figura 11 - Ventana para la configuración de la calidad y duración de la tarea a partir del estado interno del agente

Ambas ventanas disponen de la posibilidad de poder mostrar las reglas creadas de una manera entendible para el usuario a partir del botón *View* como se puede ver en la figura 12.

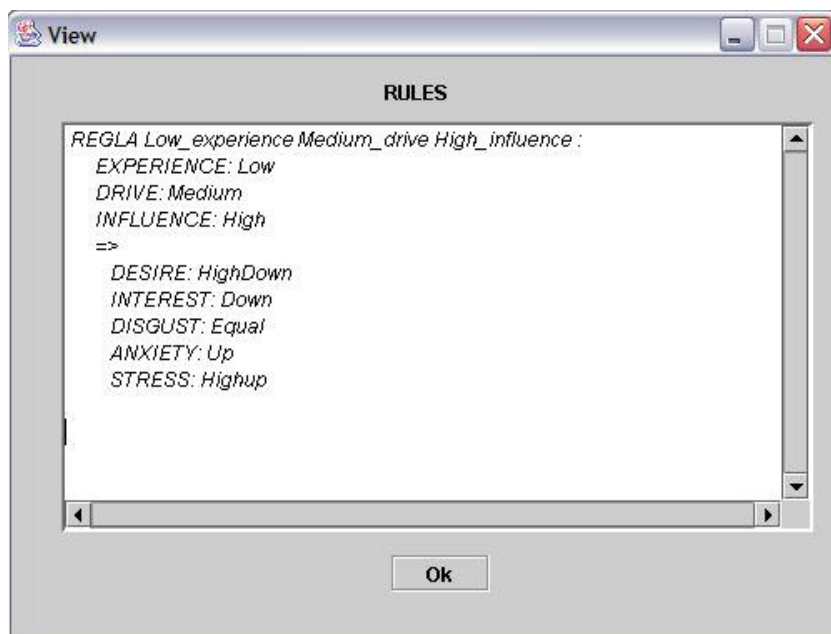


Figura 12 – Visualización de las reglas creadas durante la ejecución

Para poder añadir una nueva regla, primero se comprobará si al menos se ha seleccionado alguno de los campos de las premisas (*antecedents*) así como todos los campos de las consecuencias (*consequences*) para poder crear una nueva regla. Tampoco se deja añadir información que no corresponde a la que aparecen en los campos desplegables.

Esta nueva funcionalidad incorporada en el pfc permite que cualquier usuario del sistema TEAKS sea capaz de agregar y/o modificar nuevas reglas para generar el posible comportamiento dependiendo de las características de distintos casos de estudio.

3.5- Conclusiones

En este capítulo se ha explicado el desarrollo de la ontología utilizada para mejorar la comunicación existente y la extensión de los protocolos así como la modificación del contenido de los mensajes para la incorporación de la ontología. Además se ha presentado el desarrollo para llevar a cabo la configuración de las reglas dentro del SMA para la simulación del comportamiento de un agente sin necesidad de modificar manualmente el fichero donde están almacenadas estas reglas y sin necesidad de conocer la sintaxis del lenguaje JESS, en el cual están escritas estas reglas.

En el siguiente capítulo se muestra el desarrollo de cada uno de los conceptos explicados en este capítulo para la mejora de la comunicación, así como los ficheros creados a partir del editor utilizado para la creación de la ontología además de los ficheros necesarios para la configuración de las reglas.

CAPITULO IV: DESARROLLO

4.1- Introducción

En este capítulo se presenta por una parte una breve descripción de los ficheros y fundamentos necesarios para el desarrollo de la ontología creada, los métodos necesarios para el paso de mensajes en la comunicación utilizada, así como parte del código para ejemplificar lo mencionado. Y por otra parte, la descripción de los ficheros y fundamentos necesarios para el diseño de la interfaz para la configuración de las reglas difusas que generan el posible comportamiento de los agentes.

4.2- Trabajo realizado para la mejora de la comunicación en el SMA

En el proyecto final de carrera se han creado una serie de ficheros para la incorporación de la ontología en el SMA.

Con el editor de protégé se ha creado el Proyecto.prj para representar de manera gráfica las clases que participan en la ontología y después a partir del plug-in BeanGenerator obtener los ficheros en Java. A continuación, se describen brevemente:

ProyectoOntology.java: Ontología de los resultados globales obtenidos a través del comportamiento de los agentes al realizar una determinada tarea.

Tasks.java: Clase con las características de la tarea a realizar.

Agent_Performance.java: Clase con los parámetros de los resultados globales obtenidos por un agente al realizar una tarea específica.

Agent_Properties.java: Clase con las características internas de un agente.

Emotion.java: Subclase con las características emocionales de un agente.

Personality.java: Subclase con las características de la personalidad de un agente.

Cognition.java: Subclase con las características de cognición de un agente.

Social.java: Subclase con las características sociales de un agente.

TaskResults.java: Clase predicado con las características de la tarea.

TeamResults.java: Clase predicado con los resultados globales obtenidos por todos los agentes al realizar sus tareas.

BehaviourResults.java: Clase predicado con los resultados obtenidos por un agente al realizar una tarea concreta.

AgentCharacteristics.java: Clase predicado con las características internas de un agente.

AssignsTask.java: Clase acción de asignación de tarea.

Los paquetes a importar para el desarrollo de la ontología son los siguientes:

*jade.content.onto.**

*jade.content.schema.**

Inicialización de la ontología:

```
public static final String ONTOLOGY_NAME = "Proyecto";
private static Ontology theInstance = new ProyectoOntology();
public static Ontology getInstance() {
    return theInstance;
}
private ProyectoOntology(){
    super(ONTOLOGY_NAME, BasicOntology.getInstance());
    .... }
```

Es necesario que la ontología generada extienda la interfaz correspondiente según el tipo de objeto (acción, predicado o concepto):

```
public class Agent_Performance implements Concept {...}
public class Tasks implements Concept {...}
public class Agent_Properties implements Concept {...}
public class Emotion extends Agent_Properties{...}
public class Personality extends Agent_Properties{...}
public class Cognition extends Agent_Properties{...}
public class Social extends Agent_Properties{...}

public class TaskResults implements Predicate {...}
public class TeamResults implements Predicate {...}
public class BehaviourResults implements Predicate {...}
public class AgentCharacteristics implements Predicate {...}

public class AssignsTask implements AgentAction {...}
```

La especificación del concepto *tarea*, la definición de sus slots y sus respectivos tipos:

```

ConceptSchema tasksSchema = new ConceptSchema(TASKS);
add(tasksSchema, Tasks.class);
tasksSchema.add(TASKS_TYPE, (PrimitiveSchema)getSchema(BasicOntology.STRING));
tasksSchema.add(TASKS_DEADLINE, (PrimitiveSchema)getSchema(BasicOntology.INTEGER));
tasksSchema.add(TASKS_ID, (PrimitiveSchema)getSchema(BasicOntology.INTEGER));
tasksSchema.add(TASKS_COST, (PrimitiveSchema)getSchema(BasicOntology.INTEGER));
tasksSchema.add(TASKS_DURATION, (PrimitiveSchema)getSchema(BasicOntology.INTEGER));
tasksSchema.add(TASKS_QUALITY, (PrimitiveSchema)getSchema(BasicOntology.STRING));
tasksSchema.add(TASKS_PRIORITY, (PrimitiveSchema)getSchema(BasicOntology.STRING));
tasksSchema.add(TASKS_NUM_PARTICIPANTS, (PrimitiveSchema)getSchema(BasicOntology.INTEGER));
tasksSchema.add(TASKS_DIFFICULTY, (PrimitiveSchema)getSchema(BasicOntology.STRING));

```

Para el resto de los conceptos se hará exactamente igual. Pero en el caso particular de las subclases *personalidad*, *social*, *emocional* y *cognición* se ha de añadir lo siguiente:

```

personalitySchema.addSuperSchema(agent_PropertiesSchema);
socialSchema.addSuperSchema(agent_PropertiesSchema);
emotionSchema.addSuperSchema(agent_PropertiesSchema);
cognitionSchema.addSuperSchema(agent_PropertiesSchema);

```

La especificación de la acción *asignar tarea* y de su slot:

```

AgentActionSchema assignsTaskSchema = new AgentActionSchema(ASSIGNTASK);
add(assignedTaskSchema, AssignsTask.class);
assignedTaskSchema.add(ASSIGNTASK_TASK, tasksSchema);

```

La especificación del predicado *resultados del equipo*. En este caso el slot es del tipo agregado. Para el resto de los predicados se hará exactamente igual:

```

PredicateSchema teamResultsSchema = new PredicateSchema(TeamResults);
add(teamResultsSchema, TeamResults.class);
teamResultsSchema.add(TeamResults_LISTRESULTS, agent_PerformanceSchema, 1,
ObjectSchema.UNLIMITED);

```

Los métodos de acceso a los slots según la ontología desarrollada son del tipo discreto y agregado.

En el caso de un tipo discreto se muestran los métodos relacionados con el tipo de la tarea desarrollada:

```

private String type;
public void setType(String value) {
    this.type=value; }
public String getType() {
    return this.type; }

```

En el caso de un tipo agregado se muestran los métodos relacionados con los resultados globales obtenidos por todos los agentes en sus tareas:

```

public void addListresults(Agent_Performance elem) {
    listresults.add(elem);
}
public List getListresults() {return listresults; }

```

Para poder crear mensajes, el agente ha de decir a JADE que ontología se usará, para ello debe registrar la ontología como sigue:

```

private Ontology ontology=ProyectoOntology.getInstance();
private ContentManager manager= (ContentManager)getContentManager();
manager.registerOntology(ontology,ontology.getName());

```

También será necesario indicar el lenguaje que codificará los mensajes:

```

private Codec codec= new SLCodec();
manager.registerLanguage(codec);

```

Utilizar los métodos de las características internas de un agente para poder crear el contenido del mensaje del predicado :

```

AgentCharacteristics agch=new AgentCharacteristics();
agch.setName_agent(getLocalName());
agch.setStress("MEDIUM");
Personality p3=new Personality();
p3.setAmiable("HIGH");
p3.setAnalytical("HIGH");
p3.setDriver("LOW");
p3.setExpressive("LOW");
agch.setPersonalitycharacteristics(p3);

Social s=new Social();
s.setIntroverted("LOW");

```

```

s.setWork_alone("LOW");
s.setWork_team("HIGH");
s.setExtroverted("HIGH");
agch.setSocialcharacteristics(s);

Cognition c=new Cognition();
c.setCreativity("MEDIUM");
c.setExperience("HIGH");
c.setRole("COORDINATOR");
agch.setCognitioncharacteristics(c);

```

Utilizar los métodos de la tarea para poder crear el contenido de los mensajes de la acción asignar tarea:

```

Tasks t=new Tasks();
t.setId(22);
t.setType("HIGH");
t.setDeadline(3);
t.setCost(2);
t.setDuration(22);
t.setQuality("HIGH");
t.setPriority("HIGH");
t.setNum_participants(2);
t.setDifficulty("HIGH");

AssignsTask at=new AssignsTask();
at.setTask(t);

```

Para generar/leer el contenido de los mensajes se utilizan los siguientes métodos a partir de los dos ejemplos anteriores:

En el caso del predicado en el que se quiere informa de las características internas de un agente:

```

manager.fillContent(msg,agch);

ContentElement ce= manager.extractContent(msg);
AgentCharacteristics agc= (AgentCharacteristics) ce;

```

En el caso de la acción en la que se quiere asignar la tarea a un agente:

```

Action ac=new Action(getAID(),at);

```

```
manager.fillContent(msg,ac);
```

```
ContentElement ce= manager.extractContent(msg);
```

```
Concept action=((Action)ce).getAction();
```

```
AssignsTask at= (AssignsTask) action;
```

A continuación se explica la parte del código donde se añade la ontología al protocolo *Request Protocol*:

- Caso en que el agente Asignador pide a los trabajadores (Worker Agent) que hagan una tarea. En este proyecto final de carrera las tareas y los nombres de los trabajadores están definidos como valores estáticos, pero en la tesis se hará a través del usuario:

```
for (int c=0;c<numAgents;c++) {  
    // Prepara el mensaje del tipo REQUEST  
    ACLMessage msg3= new ACLMessage(ACLMessage.REQUEST);  
    msg3.setSender(getAID());  
    msg3.setLanguage(codec.getName());  
    msg3.setOntology(ProyectoOntology.ontology_NAME);  
    msg3.addReceiver(new AID("WORKER"+c,AID.ISLOCALNAME));  
    Tasks t=new Tasks();  
    AssignsTask at=new AssignsTask();  
    t.setType("HIGH");  
    t.setDeadline(3);  
    t.setCost(2);  
    t.setDuration(22);  
    t.setQuality("HIGH");  
    t.setPriority("HIGH");  
    t.setNum_participants(2);  
    t.setDifficulty("HIGH");  
  
    if(("WORKER"+c).compareTo("WORKER0")==0)    t.setId(22);  
    else if(("WORKER"+c).compareTo("WORKER3")==0)    t.setId(10);  
    else if(("WORKER"+c).compareTo("WORKER4")==0)    t.setId(5);  
    else    t.setId(15);  
    at.setTask(t);  
    Action ac=new Action(getAID(),at);  
    manager.fillContent(msg3,ac);  
    send(msg3);  
}
```

```
}
```

- Caso en que el trabajador informa al agente Asignador que falló en la tarea que le fue asignada. En este proyecto final de carrera, a partir de un valor aleatorio ($i=[0,1]$) se simula el comportamiento del agente pero en la tesis será sustituido por las reglas del comportamiento:

```
if (i==0){  
  // Prepare the INFORM message  
  ACLMessage msgInfD2 = new ACLMessage(ACLMessage.INFORM);  
  msgInfD2.setSender(getAID());  
  msgInfD2.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));  
  msgInfD2.setContent(new Integer(t5.getId()).toString());  
  send(msgInfD);  
}  
else{  
  // Prepara el mensaje del tipo FAILURE  
  ACLMessage msgF2 = new ACLMessage(ACLMessage.FAILURE);  
  msgF2.setSender(getAID());  
  msgF2.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));  
  msgF2.setLanguage(codec.getName());  
  msgF2.setOntology(ProyectoOntology.ontology_NAME);  
  TaskResults tr=new TaskResults();  
  
  tr.setTask(task);  
  manager.fillContent(msgF2,tr);  
  send(msgF2);  
}
```

Para seguir con la comunicación especificada, se hace un control para simular cuando se han acabado las tareas:

```
if (Tasks_Complets == numAgents){  
  ACLMessage msg4= new ACLMessage(ACLMessage.INFORM);  
  msg4.setSender(getAID());  
  msg4.addReceiver(new AID("COUNTER", AID.ISLOCALNAME));  
  msg4.setContent("INFORM-DONE");  
  send(msg4);  
  Tasks_Complets=0;  
}
```

- Caso en que el trabajador informa al agente Resultado que ha acabado la tarea que le fue asignada. En el caso que el trabajador no haya podido acabar la tarea, el desempeño tendrá los peores resultados. Para este proyecto final de carrera los resultados para cada uno de los parámetros que miden el desempeño de cada agente están definidos como valores estáticos, pero en la tesis estos valores serán generados por las reglas del comportamiento:

```
// Prepara el mensaje del tipo INFORM
ACLMessage msgInfR = new ACLMessage(ACLMessage.INFORM);
msgInfR.setSender(getAID());
msgInfR.addReceiver(new AID("RESULTER", AID.ISLOCALNAME));
msgInfR.setLanguage(codec.getName());
msgInfR.setOntology(ProyectoOntology.ONTOLOGY_NAME);

Agent_Performance ap=new Agent_Performance();
ap.setIndividual_level("MEDIUM");
ap.setGoals_achievement("HIGH");
ap.setSupervision_level("MEDIUM");
ap.setTask_duration("LOW");
ap.setTask_quality("HIGH");
ap.setTeam_level("MEDIUM");
ap.setId_task(task.getId());
ap.setAgent(getLocalName());
BehaviourResults br=new BehaviourResults();
br.setInformation_performance(ap);
manager.fillContent(msgInfR,br);
send(msgInfR);
```

El agente resultado extrae el contenido del mensaje y guarda los resultados obtenidos por los trabajadores en una lista:

```
ContentElement ce= manager.extractContent(msg);
BehaviourResults br=(BehaviourResults) ce;
Agent_Performance ap=br.getInformation_performance();
tr.addListresults(ap);
```

El agente resultado informa al agente Interfaz, a través de la lista, de los resultados globales de los agentes en sus tareas:

```
// Prepare el mensaje del tipo INFORM
ACLMessage msgInf = new ACLMessage(ACLMessage.INFORM);
msgInf.setSender(getAID());
```

```

msgInf.addReceiver(new AID("GUI", AID.ISLOCALNAME));
msgInf.setLanguage(codec.getName());
msgInf.setOntology(ProyectoOntology.ONTOLOGY_NAME);
manager.fillContent(msgInf,tr);
send(msgInf);

```

El agente Interfaz extrae la lista del mensaje:

```

ContentElement ce= manager.extractContent(msg);
if (ce instanceof TeamResults){
TeamResults tr=(TeamResults) ce;
Collection listteam = (Collection)tr.getListresults();
Agent_Performance ap=new Agent_Performance();
for (Iterator it=listteam.iterator(); it.hasNext();) {
    ap=(Agent_Performance)it.next();
    System.out.println("Individual level: "+ap.getIndividual_level()+"\n");
    System.out.println("Task duration: "+ap.getTask_duration()+"\n");
    System.out.println("Goals achievement: "+ap.getGoals_achievement()+"\n");
    System.out.println("Supervision level: "+ap.getSupervision_level()+"\n");
    System.out.println("Team level: "+ap.getTeam_level()+"\n");
    System.out.println("Team level: "+ap.getTask_quality()+"\n");
    System.out.println("Id task: "+ap.getId_task()+"\n");
    System.out.println("Agent: "+ap.getAgent()+"\n");
}

```

A continuación se explica la parte del código donde se añade la ontología al protocolo *Contract-Net Protocol*:

- Caso en que el agente Asignador le pide al DF una lista con todos los agentes de un determinado tipo. En este proyecto final de carrera el tipo que se pide es estático, pero en la tesis dependerá de un conjunto de reglas dependientes de las características de la tarea:

```

private AID[] searchAgents(){
AID[] listAgents=null;
DFAgentDescription dfd=new DFAgentDescription();
ServiceDescription sd=new ServiceDescription();
sd.setType("TECHNICIAN");
dfd.addServices(sd);
try{
    DFAgentDescription[] result=DFService.search(this,dfd);
    listAgents=new AID[result.length];
}

```

```

for (int i=0;i<result.length;i++)
    listAgents[i]=result[i].getName();
}catch (FIPAException fe){
    fe.printStackTrace();
}
return listAgents;
}

```

Una tarea no ha sido realizada por el agente al cual fue asignada. El agente Asignador pide a los trabajadores que son del tipo Técnico si pueden hacer esa tarea. Después, se guarda en una lista los identificadores de las tareas de los agentes propuestos para poder aceptar o rechazar las propuestas:

```

TaskResults tr2= (TaskResults) ce2;
Tasks t3=tr2.getTask();
AID[] listAgents=searchAgents();
if ((listAgents.length>1) || (listAgents.length==1)
&& !listAgents[0].getLocalName().equals(msg.getSender().getLocalName())) ) {
    // Prepara el mensaje del tipo CFP
    ACLMessage cfp= new ACLMessage(ACLMessage.CFP);
    // envia a todos los agentes de un determinado tipo menos a él mismo
    int i=0;
    while(i<listAgents.length){
        if (!listAgents[i].getLocalName().equals(msg.getSender().getLocalName())){
            cfp.addReceiver(listAgents[i]);
        }
        i++;
    }
    cfp.setLanguage(codec.getName());
    cfp.setOntology(ProyectoOntology.ontologyName);
    cfp.setSender(getAID());
    cfp.setReplyWith("cfp"+System.currentTimeMillis());
    AssignsTask at2=new AssignsTask();
    at2.setTask(t3);
    Action ac2=new Action(getAID(),at2);
    manager.fillContent(cfp,ac2);
    send(cfp);
    //añadir la tarea a la lista
    listIdTasks.add(new Integer(t3.getId()));
}

```

Un trabajador se propone o rechaza para realizar la tarea. En este proyecto final de carrera, la elección de un agente para proponerse o no a realizar la tarea que se oferta se da a través de un valor aleatorio ($i=[0,1]$), sin embargo cuando esta parte del código esté integrada en la tesis, esta decisión aleatoria será sustituida por el resultado de ejecutar las reglas del comportamiento de cada agente:

```

ContentElement ce2= manager.extractContent(msg);
Concept action2=((Action)ce2).getAction();
AssignsTask at2= (AssignsTask) action2;
Tasks t4=at2.getTask();
ACLMessage replyCFP=msg.createReply();
replyCFP.setInReplyTo(msg.getReplyWith());
int i=r.nextInt(2);
if (i==0) {
    // Prepara el mensaje del tipo PROPOSE
    replyCFP.setPerformative(ACLMessage.PROPOSE);
    TaskResults tr3=new TaskResults();
    tr3.setTask(t4);
    manager.fillContent(replyCFP,tr3);
}
else{
    // Prepara el mensaje del tipo REFUSE
    replyCFP.setPerformative(ACLMessage.REFUSE);
    replyCFP.setContent("REFUSE "+String.valueOf(t4.getId()));
}
send(replyCFP);
}

```

El agente Asignador recibe las propuestas. En este proyecto final de carrera, se asignará la tarea al primer agente que se proponga, al resto los rechazará. Para asignar la tarea al primero se hace a partir de la lista que contiene los identificadores. Si la tarea que ha fallado está en la lista, entonces se le asignará la tarea y se eliminará de la lista para que cuando otro trabajador se proponga para la misma tarea, la rechace:

```

ContentElement ce3= manager.extractContent(msg);
TaskResults tr3= (TaskResults) ce3;
Tasks t4=tr3.getTask();
int i=listIdTasks.indexOf(new Integer(t4.getId()));
if(i!=-1){

```

```

        ACLMessage msgPr=new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
        msgPr.setSender(getAID());
        msgPr.addReceiver(msg.getSender());
        msgPr.setLanguage(codec.getName());
        msgPr.setOntology(ProyectoOntology.ONTOLOGY_NAME);
        manager.fillContent(msgPr,tr3);
        listIdTasks.remove(i);
        send(msgPr);
    }
    else {
        ACLMessage msgRe=new ACLMessage(ACLMessage.REJECT_PROPOSAL);
        msgRe.setSender(getAID());
        msgRe.addReceiver(msg.getSender());
        msgRe.setContent(String.valueOf(t4.getId()));
        send(msgRe);
    }
}

```

A continuación se explica la parte del código donde se añade la ontología al protocolo *Request Protocol* (Los candidatos a formar el equipo):

Cuando el trabajador recibe la tarea que le ha sido asignada, la guarda en una variable global (*task*) para saber qué tarea está realizando. Después se guarda en una lista todos los nombres de los trabajadores que comparten la tarea para luego pedirles su estado interno:

```

if (cad2.equals(msg.getContent())) {
    if(task.getId()==id_t)// si el Agent Worker trabaja en la misma tarea
        listAgents.add(msg.getSender().getLocalName());
    numReq++;
    if(numReq==numAgents-1){ // si todos los Agent Worker han contestado
        if (listAgents.size(>0) { // Si el Agent Worker comparte la tarea con otro agente
            // Prepara el mensaje del tipo REQUEST
            ACLMessage msgReq = new ACLMessage(ACLMessage.REQUEST);
            msgReq.setSender(getAID());
            msgReq.setContent("INTERNAL-STATE");
            msgReq.setReplyWith(getLocalName()+"Internal-State");

            // Agent Worker envía el REQUEST a todos los compañeros de tarea
            for (int c=0;c<listAgents.size();c++)

```

```

        msgReq.addReceiver(new AID(listAgents.get(c).toString(),AID.ISLOCALNAME));
    send(msgReq);
}
// Si el Agent Worker hace la tarea solo
if(listAgents.size()==0) finished=true;
}
else{
    num++;
    if(num==listAgents.size())
        // Si el Agent Worker recibe el estado interno del agente
        finished=true;
    }
}

```

El booleano *finished* se utiliza cuando los agentes han acabado de enviar su estado interno o cuando la tarea la hace solo y así poder continuar con el resto de la comunicación.

- Los trabajadores que comparten la tarea le informan de su estado interno. En este proyecto final de carrera el estado interno está definido con valores estáticos, pero en la tesis se sustituirán por aquellos valores generados por las reglas del comportamiento:

```

// Prepara el mensaje del tipo INFORM
AgentCharacteristics agch=new AgentCharacteristics();
Personality p3=new Personality();
p3.setAmiable("HIGH");
p3.setAnalytical("HIGH");
p3.setDriver("LOW");
p3.setExpressive("LOW");
agch.setPersonalitycharacteristics(p3);
Social s=new Social();
s.setIntroverted("LOW");
s.setWork_alone("LOW");
s.setWork_team("HIGH");
s.setExtroverted("HIGH");
agch.setSocialcharacteristics(s);
Cognition c=new Cognition();
c.setCreativity("MEDIUM");
if (getLocalName().equals("WORKER1")) {
    c.setExperience("HIGH");
}

```

```

    c.setRole("COORDINATOR");
}
else if (getLocalName().equals("WORKER3")) {
    c.setExperience("MEDIUM");
    c.setRole("SPECIALIST") }
else {
    c.setExperience("LOW");
    c.setRole("TECHNICIAN");
}
agch.setCognitioncharacteristics(c);
agch.setName_agent(getLocalName());
agch.setStress("MEDIUM");
ACLMessage reply = msg.createReply();
reply.setPerformative(ACLMessage.INFORM);
reply.setLanguage(codec.getName());
reply.setOntology(ProyectoOntology.ONTOLOGY_NAME);
reply.setInReplyTo(msg.getReplyWith());
manager.fillContent(reply,agch);
send(reply);

```

4.3- Trabajo realizado para el desarrollo de la interfaz para la configuración de las reglas difusas

En el proyecto final de carrera se han creado una serie de ficheros para la incorporación de la interfaz en el SMA:

ConfigurationBehaviourRule.java: Fichero que se encarga de la creación de la ventana de configuración de las características internas de un agente al asignarle una tarea.

ConfigurationTaskRule.java: Fichero que se encarga de la creación de la ventana de configuración del desempeño de un agente ante la asignación de una tarea.

menu.java: Fichero que se encarga de la creación de la ventana para la selección del tipo de configuración de las reglas deseadas.

panel.java: Fichero que se encarga de la creación de la ventana para visualizar las reglas generadas en un determinado momento en un lenguaje entendible para el usuario.

Se han creado los siguientes métodos para comprobar que se selecciona alguno de los campos (distinto del campo en blanco, utilizado para la inicialización de los campos

antes de generar una regla) que aparecen en la lista de desplegables: `selectedComboBoxAntec(JComboBox combo)` y `selectedComboBoxCons(JComboBox combo)`. Los métodos `selectedAntecedents()` y `selectedConsequences()` se utilizan para comprobar que se ha seleccionado alguno de los campos que aparecen en la parte de premisas y todos los campos que aparecen en el apartado de consecuencias respectivamente.

Las reglas generadas están implementadas en el lenguaje JESS y se guardarán en dos ficheros, *BehaviourRules.txt* y *TasksRules.txt*. Existirán otros dos ficheros *DefinitionBehaviourRules.txt* y *DefinitionTasksRules.txt* con las definiciones de las variables que se utilizarán para la configuración de las reglas.

Antes de agregar las reglas que el usuario quiere configurar, se leerán los ficheros *DefinitionBehaviourRules.txt* y *DefinitionTasksRules.txt* y se copiarán en unos ficheros nuevos *BehaviourRules.txt* y *TasksRules.txt* respectivamente, según si se quiere configurar el estado interno del agente o el desempeño realizado.

4.4- Conclusiones

En este capítulo se ha explicado el desarrollo del trabajo en el PFC describiendo los conceptos de la ontología creada a partir de las partes del código extraídas (especificación de las clases, las acciones, los predicados, el registro de la ontología...), así como la creación y paso de mensajes en la comunicación implementada también a través de las partes del código extraídas.

En este capítulo también se han explicado brevemente los ficheros creados para el desarrollo de la interfaz para la configuración de las reglas, los controles necesarios para su funcionamiento y cómo se guardan esas reglas generadas en los respectivos ficheros para la simulación del comportamiento de un agente sin necesidad de modificar manualmente el fichero donde están almacenadas estas reglas y sin necesidad de conocer la sintaxis del lenguaje JESS, en el cual están escritas estas reglas.

En el anexo del documento, se puede ver la implementación completa del trabajo desarrollado en el PFC.

CAPÍTULO V: EVALUACIÓN

5.1- Introducción

En este capítulo se hace una explicación sobre algunos aspectos importantes que se han tenido en cuenta a la hora de implementar la parte del código, así como el juego de pruebas realizado para la comprobación del funcionamiento del trabajo desempeñado. Este se divide en dos sub-secciones: la primera respecto a la mejora y extensión de la comunicación existente y el segundo respecto a la configuración de las reglas para la simulación del posible comportamiento del agente.

5.2- Juego de Pruebas realizado respecto a la comunicación

Antes de comenzar con la explicación de las pruebas realizadas para la comprobación del correcto funcionamiento, se han tenido en cuenta una serie de consideraciones:

Los resultados obtenidos en el desempeño de una tarea están definidos como valores estáticos. Si los trabajadores han acabado la tarea, sus valores serán aleatorios pero si no la han acabado entonces sus valores serán los peores.

La elección para que un trabajador rechace o acepte, informe o falle, rechace o se proponga en una tarea se realiza de manera aleatoria. Estos valores aleatorios serán sustituidos en la tesis por las reglas difusas que generan el comportamiento de cada agente. Por tanto, se podrá dar el caso en que un trabajador, al cual le fue asignado un tipo de tarea, durante el desempeño de la tarea deba informar al agente Asignador cuando haya acabado su trabajo o avisarle que no la ha podido acabar por factores externos (por ejemplo baja laboral). Esto se contemplará generando un aleatorio que oscilará entre 0 y 1 reflejando que el trabajador ha acabado la tarea o que no la ha podido finalizar. Otro caso que se puede llegar a producir consistirá en el momento en que un trabajador no ha podido finalizar la tarea que le fue asignada, donde el agente Asignador pedirá una lista de los posibles candidatos para terminar la tarea. Pudiendo cada uno de los trabajadores rechazar o proponerse para dicha tarea. Esto se contemplará generando un aleatorio que oscilará también entre 0 y 1, reflejando que un

trabajador rechace o se proponga a una tarea debido a un factor externo o no esté realizando ninguna tarea más y pueda ser capaz de desempeñar otra tarea además de la que ya tenía asignada. También estará el caso que un trabajador rechace la tarea que le fue asignada o al contrario, esté de acuerdo en hacerla. Es por eso que se podrán dar situaciones irreales en las que un trabajador rechace una tarea y a continuación informe al agente Asignador que ha finalizado la tarea cuando la había rechazado, provocado por los aleatorios anteriormente mencionados.

El DF se encarga de informar al agente Asignador de todos los trabajadores que cumplen que son de un tipo concreto, pero esta condición será sustituida en la tesis por unas reglas dependiendo de las características de las tareas.

Dado que en JADE cada agente es un hilo de ejecución, puede ocurrir que en la ventana gráfica de JADE donde se muestran los mensajes de cada agente, estos mensajes no aparezcan de manera secuencial, sin embargo si se respeta la secuencia definida en los protocolos de comunicación.

Una vez explicadas las consideraciones tomadas para el desarrollo del trabajo, pasaremos a la explicación del juego de pruebas realizado a partir de la herramienta gráfica de Jade, donde se mostrará la comunicación entre los agentes del sistema.

Primero se crearán los agentes que aparecerán en el sistema: el agente Asignador (ASSIGNER), el agente Resultado (RESULTER), el agente Simulador (COUNTER), el agente Interfaz (GUI), los trabajadores (WORKERS) a formar el equipo.

Caso 1: Se ha hecho una simulación con cinco trabajadores a formar el equipo de trabajo (Worker Agent). El rol de los trabajadores está definido como valor estático y a continuación puede verse cuales son dichos roles:

WORKER0, WORKER2, WORKER4 : <i>TECHNICIAN</i> WORKER1 : <i>COORDINATOR</i> WORKER2 : <i>SPECIALIST</i>
--

WORKER0 tiene asignada la tarea 22
WORKER3 tiene asignada la tarea 10
WORKER4 tiene asignada la tarea 5
WORKER1, WORKER2: tienen asignada la tarea 15



Como se puede ver en el gráfico, el agente Interfaz pide al Simulador el número de simulaciones que se quieren hacer para la formación de equipos de trabajo.

Una vez el agente Simulador recibe el mensaje, le pide al agente Asignador que vaya asignando las tareas a los trabajadores y este último acepta la petición del agente Simulador. A continuación, el agente Simulador acepta la petición del agente Interfaz

Después, el agente Asignador pide que asigne a los cinco trabajadores con un request, las tareas 22, 15, 15, 10 y 5. Los trabajadores que reciban la misma tarea, deberán además informarse entre ellos de su estado interno.

Una vez asignadas todas las tareas a los trabajadores, cada uno de ellos pregunta al resto mediante un *request* la tarea que están realizando, para saber así con que compañeros comparten la tarea, si es que la comparten.

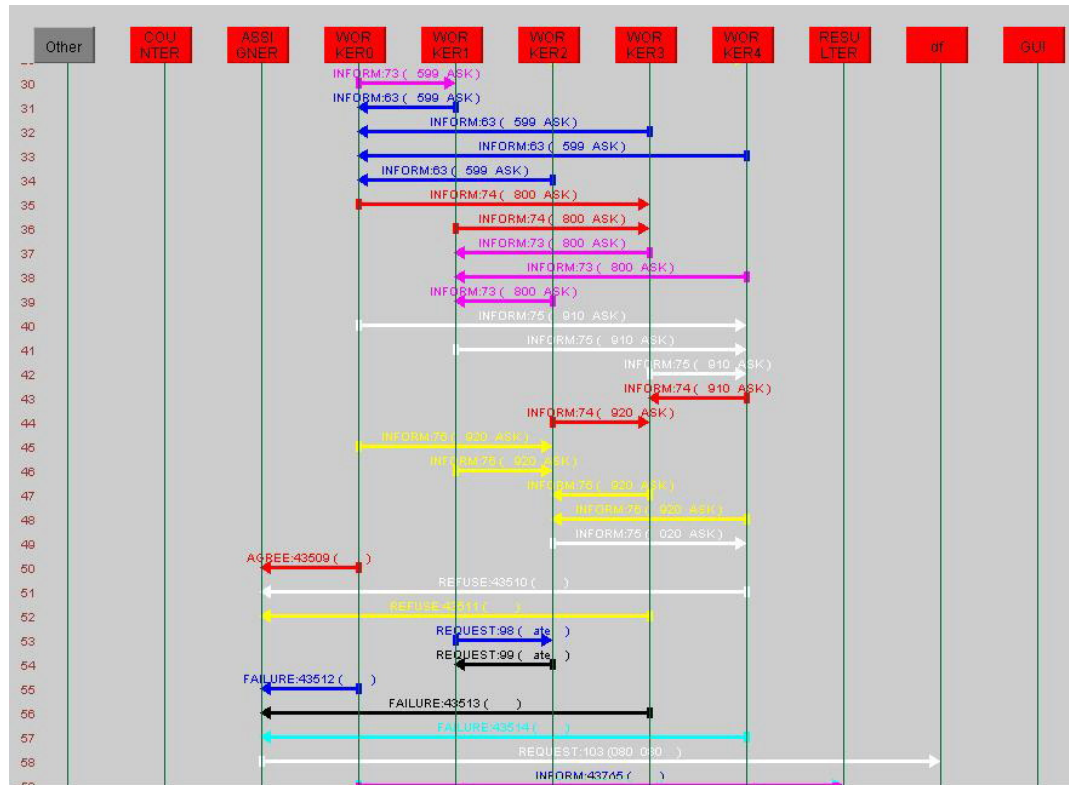


Figura 14 – Consulta sobre compartición de las tareas y proceso *Request Protocol*

Una vez los trabajadores han mandado el *request*, cada uno de los trabajadores va enviando mediante un *inform* las tareas que les han sido asignadas. Por ejemplo, WORKER0 informará al resto de los agentes que tiene la tarea 22 asignada. En el caso del WORKER1 y WORKER2 sus tareas asignadas son la 15, el WORKER3 la tarea 10 y el WORKER4 la tarea 5.

Como el WORKER4 y WORKER3 no comparten tarea, continúan con la ejecución, donde ambos rechazan y fallan. En el caso del WORKER1 y WORKER2 como comparten la tarea 15 primero han de conocer el estado interno de su compañero de tarea ya que influye en su desempeño.

WORKER0 que no comparte tarea, acepta y falla en la tarea que estaba realizando.

WORKER0 ya puede mandar los resultados obtenidos por su comportamiento en la tarea 22 al agente Resultado.

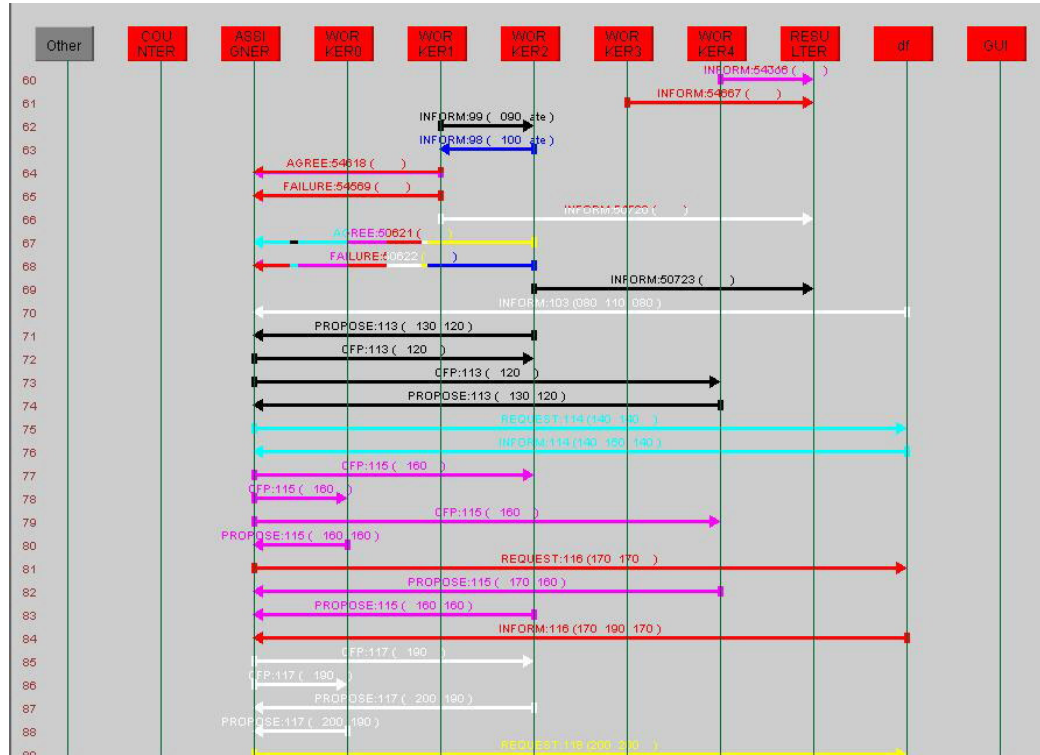


Figura 15 – Proceso Contract-Net Protocol

Lo mismo pasa en el caso del WORKER4 y WORKER3. Después WORKER1 y WORKER2 se informan de su estado interno. Ambos avisan al agente Asignador que han aceptado y fallado en las tareas que les fueron asignadas. Seguidamente informan al agente Resultado de los resultados obtenidos por su comportamiento en la tarea 15.

3.- Proceso Contract-Net Protocol

WORKER0 falló al realizar la tarea 22 y el agente Asignador le pidió al DF que le enviase la lista con los trabajadores que fueran Técnicos para hacer la reasignación de la tarea 22.

El DF envía la lista con los trabajadores que cumplen el criterio, en este caso son el WORKER0, WORKER2 y WORKER4. A continuación se pasa a la negociación de la tarea a reasignar a través del mensaje *cfp*, donde se pide a los trabajadores anteriores (en este caso, como el que ha fallado ha sido el WORKER0, tan solo se pide al WORKER2

En el caso que la tarea que había fallado era la 10, el agente Asignador reasigna la tarea 5 al WORKER2 y rechaza al WORKER0. WORKER2 no puede terminar la tarea que le fue reasignada y envía los resultados obtenidos en la tarea reasignada al agente Resultado.

En el caso que la tarea que había fallado era la 15 (que estaba haciendo el WORKER1), el agente Asignador reasigna la tarea 15 al WORKER2 y rechaza al WORKER0 y al WORKER4. WORKER2 informa que ha acabado y envía sus resultados.

El último caso, que la tarea que había fallado era la 15 (que estaba haciendo el WORKER2), el agente Asignador reasigna la tarea 15 al WORKER0 y rechaza al WORKER4. WORKER0 falla en su tarea reasignada y envía sus resultados.

4.- Proceso de finalización

El agente Asignador avisa al agente Simulador que todas las tareas han sido asignadas y reasignadas.

El agente Simulador informará que ya se han acabado todas las simulaciones y por lo tanto, el agente Interfaz pedirá la lista de todos los resultados obtenidos en todas las tareas.

El agente Resultado envía la lista con todos los resultados al agente.

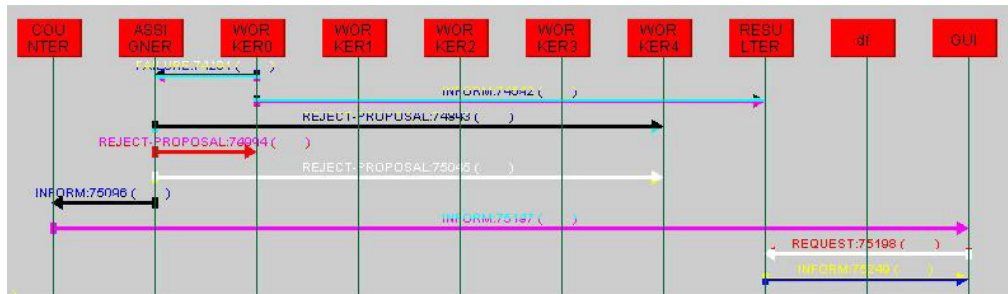


Figura 17 – Finalización simulación

Caso 2: Se ha hecho una simulación con tres trabajadores a formar el equipo de trabajo (Worker Agent). El rol de los trabajadores está definido como valores estáticos.

WORKER0 : <i>TECHNICIAN</i> WORKER1 : <i>TECHNICIAN</i> WORKER2 : <i>TECHNICIAN</i>
--

Las tareas están definidas como valores estáticos. Los identificadores de las tareas de cada uno de los trabajadores son las siguientes:

WORKER0 tiene asignada la tarea 22
 WORKER1, WORKER2: tienen asignada la tarea 15

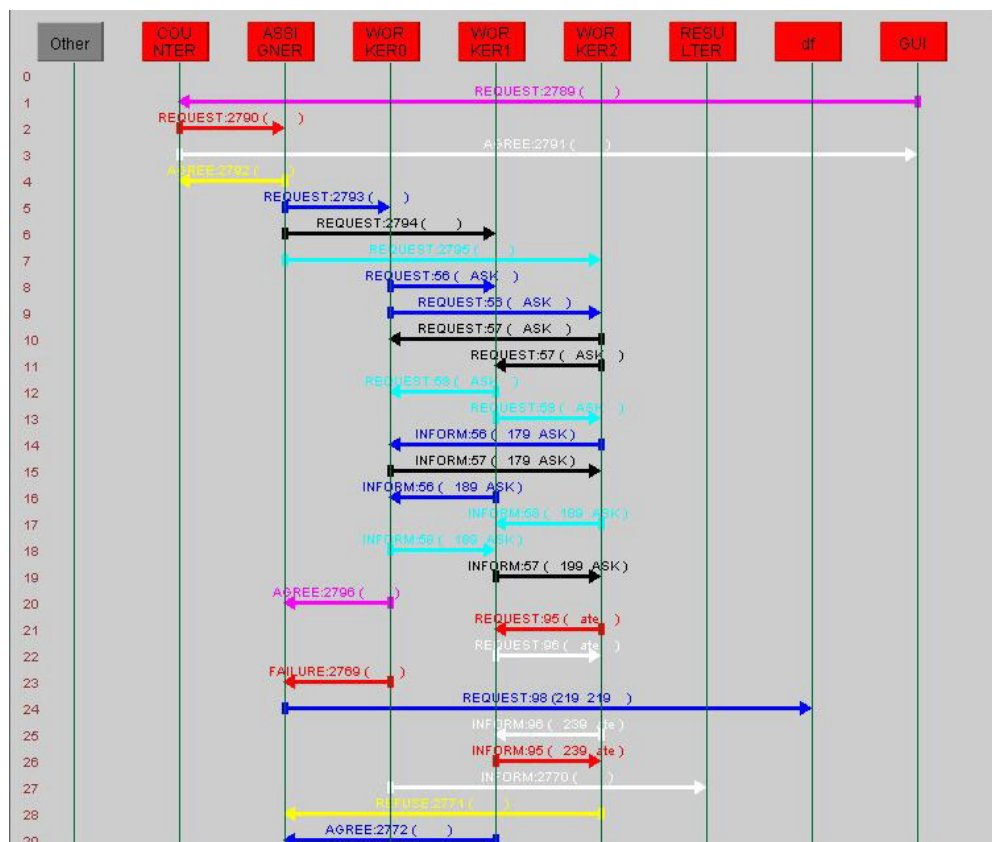


Figura 18 – Asignación de las tareas a los trabajadores y consulta sobre compartición de las tareas

1.- Inicio del proceso

Como se puede ver en el gráfico, el agente Interfaz pide al Simulador el número de simulaciones que se quieren hacer para la formación de equipos de trabajo.

2.- Proceso del Request Protocol

Una vez el agente Simulador recibe el mensaje, le pide al agente Asignador que vaya asignando las tareas a los trabajadores. A continuación, el agente Simulador acepta la petición del agente Interfaz y el agente Asignador acepta la petición del agente Simulador. A continuación, el agente Asignador pide que asigne a los tres trabajadores

con un *request*, las tareas 22, 15 y 15. Los trabajadores que reciban la misma tarea además deberán informarse entre ellos de su estado interno.

Una vez asignadas todas las tareas, cada uno de ellos pregunta al resto mediante un *request* la tarea que están realizando, para saber así con que compañeros comparten la tarea, si es que la comparten.

Una vez los trabajadores han mandado el *request*, cada uno de los trabajadores va enviando mediante un *inform* de las tareas que les han sido asignadas. Por ejemplo, WORKER0 informará al resto de los agentes que tiene la tarea 22 asignada. En el caso del WORKER1 y WORKER2 sus tareas asignadas son la 15.

En el caso del WORKER1 y WORKER2 como comparten la tarea 15 primero han de conocer el estado interno de su compañero de tarea ya que influye en su desempeño. Por tanto, ambos se piden y se contestan sobre el estado interno. Una vez han recibido el estado interno de su compañero, ya puede continuar con la ejecución, donde acepta y informa en el caso del WORKER1 y rechaza y falla en el caso del WORKER2.

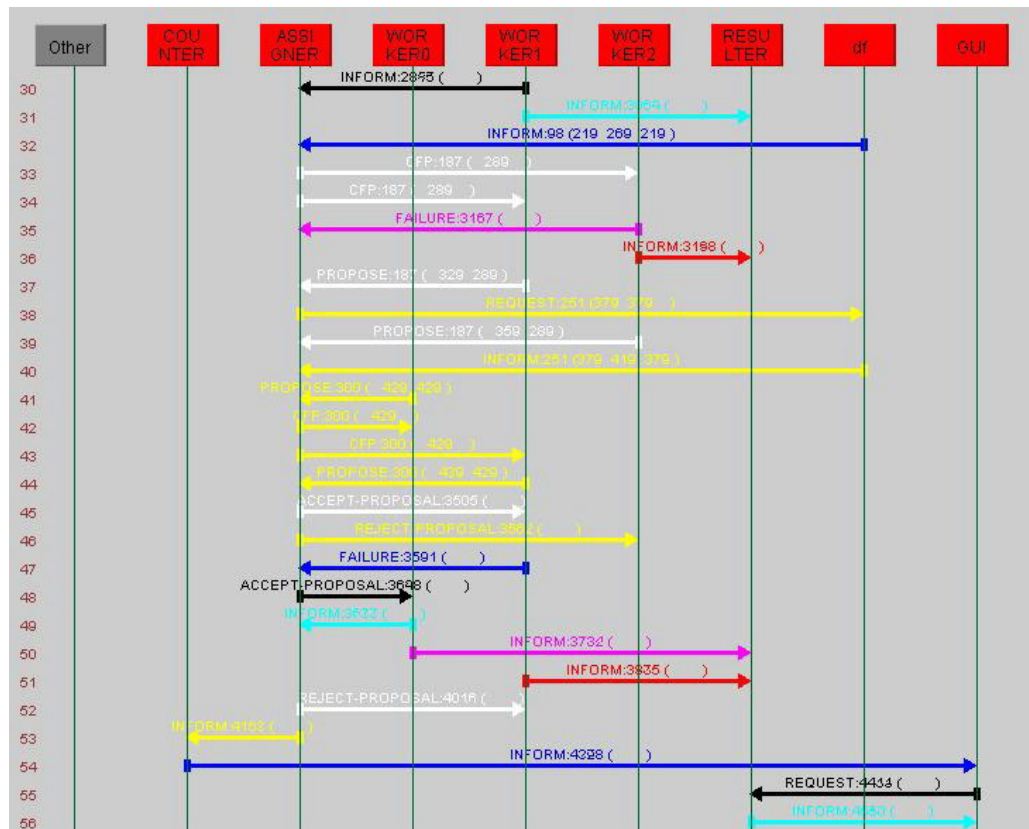


Figura 19 - Proceso Contract-Net Protocol y finalización simulación

WORKER1 envía al agente Resultado los resultados obtenidos en esa tarea.

3.- Proceso Contract-Net Protocol

El DF envía la lista con los 3 trabajadores al agente Asignador y empieza la negociación. Como han fallado dos tareas, la 22 y la 15 enviarán cuatro *cfp*. Y el agente Asignador pedirá al DF que le envíe la lista con los trabajadores una vez más.

En el caso de la tarea 15 fallada por el WORKER2, se proponen tanto WORKER0 como WORKER1 y se reasigna al WORKER0. WORKER0 acaba su tarea reasignada y envía sus resultados. En el caso de la tarea 22 fallada, se proponen tanto WORKER1 como WORKER2 y se reasigna al WORKER1, rechazando al WORKER2. WORKER1 falla en su tarea reasignada y envía sus resultados.

4.- Proceso de finalización

El agente Simulador avisa al agente Interfaz que ya se han hecho todas las simulaciones y por lo tanto, este último pide al agente Resultado que le envíe la lista con los resultados globales.

Caso 3: Se ha hecho una simulación con tres trabajadores a formar el equipo de trabajo (Worker Agent). El rol de los trabajadores está definido como valore estático.

WORKER0 : <i>TECHNICIAN</i>
WORKER1 : <i>TECHNICIAN</i>
WORKER2 : <i>TECHNICIAN</i>

Las tareas están definidas como valores estáticos. Los identificadores de las tareas de cada uno de los trabajadores son las siguientes:

WORKER0 tiene asignada la tarea 22
WORKER1, WORKER2: tienen asignada la tarea 15

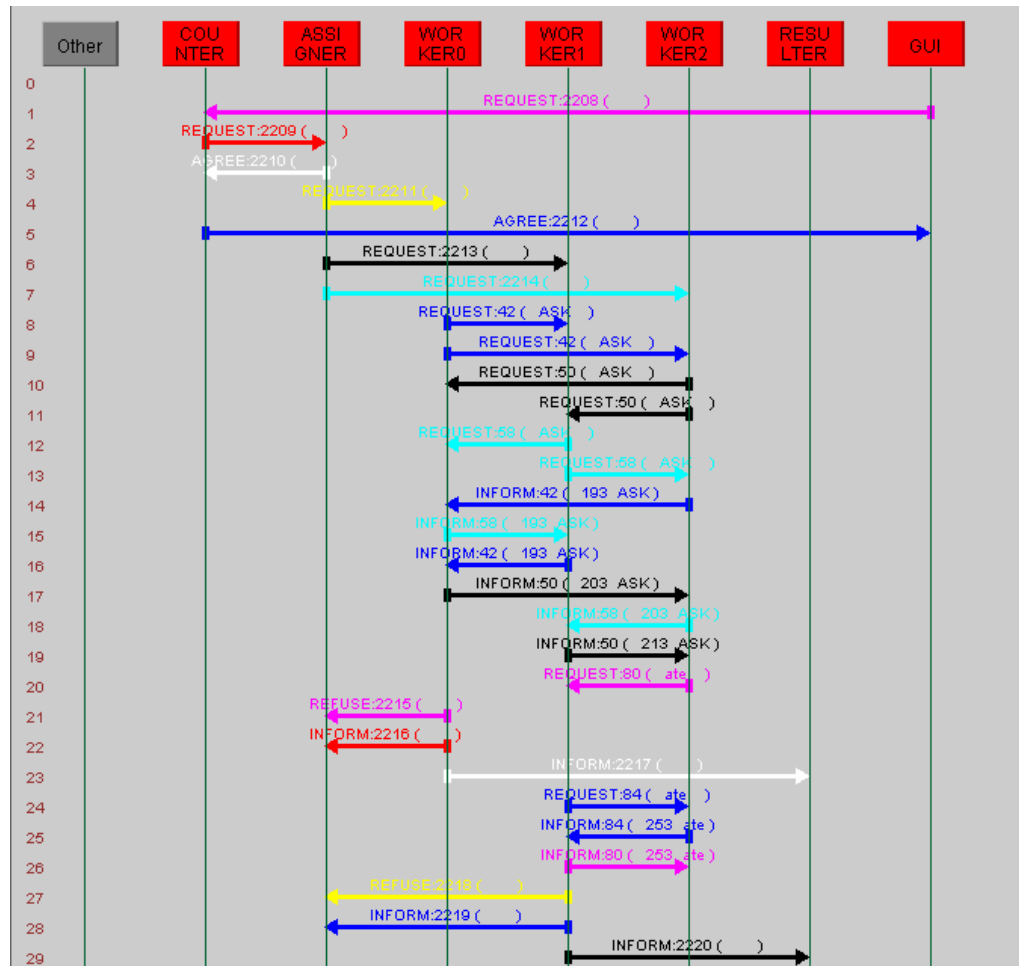


Figura 20 - Asignación de las tareas a los trabajadores y consulta sobre compartición de las tareas

1.- Inicio del proceso

Como se puede ver en el gráfico, el agente Interfaz pide al Simulador el número de simulaciones que se quieren hacer para la formación de equipos de trabajo.

2.- Proceso del Request Protocol

Una vez el agente Simulador recibe el mensaje, le pide al agente Asignador que vaya asignando las tareas a los trabajadores. A continuación, el agente Simulador acepta la petición del agente Interfaz y el agente Asignador acepta la petición del agente Simulador.

Después el agente Asignador pide que asigne a los tres trabajadores con un *request*, las tareas 22, 15 y 15.

Los trabajadores que reciban la misma tarea, deberán además informarse entre ellos de su estado interno.

Una vez asignadas todas las tareas a los trabajadores, cada uno de ellos pregunta al resto mediante un *request* la tarea que están realizando, para saber así con que compañeros comparten la tarea, si es que la comparten.

Una vez los trabajadores han mandado el *request*, cada uno de los trabajadores va enviando mediante un *inform* de las tareas que les han sido asignadas. Por ejemplo, WORKER0 informará al resto de los agentes que tiene la tarea 22 asignada. En el caso del WORKER1 y WORKER2 sus tareas asignadas son la 15.

En el caso del WORKER1 y WORKER2 como comparten la tarea 15 primero han de conocer el estado interno de su compañero de tarea ya que influye en su desempeño. Por tanto, se informa del estado interno y después, WORKER1 rechaza e informa su tarea al Asignador y a continuación informa al agente Resultado de los resultados obtenidos en la tarea 15. En el caso del WORKER0, rechaza y informa la tarea 22 y envía al agente Resultado los resultados obtenidos en esa tarea.

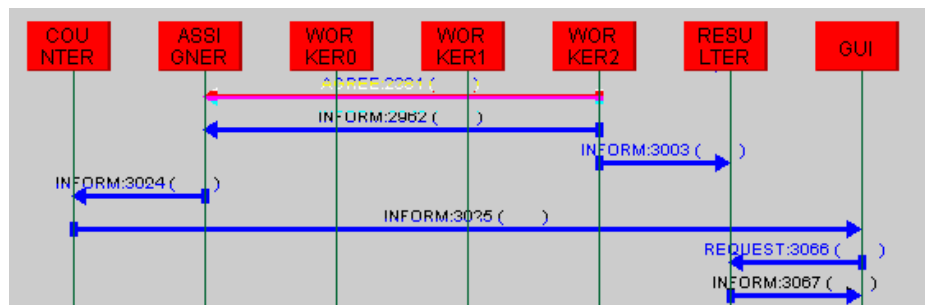


Figura 21 - Proceso *Request Protocol* y finalización simulación

3.- Proceso de finalización

WORKER2 acepta e informa y después informa también al agente Resultado de los resultados obtenidos en la tarea 15. Después, el agente Asignador informa al agente Simulador que ya se han asignado las tareas, este último le pide al agente Resultado que le informe de la los resultados globales obtenidos por los trabajadores en sus tareas y le informa con una lista.

En el caso 2 y 3, a pesar de tener los mismos parámetros de entrada, es decir, 3 agentes con el mismo rol y la tarea 15 asignada a dos agentes, existe una diferencia respecto a la ejecución. En el caso 2, existe alguna tarea que ha fallado y por eso está el paso 3

protocolo Contract-Net pero en el caso 3, no hay ninguna tarea que falle y por tanto del paso 2 *Request Protocol* ya se pasa al proceso de finalización donde el agente Interfaz pide la lista de los resultados globales obtenidos por los agentes.

5.3- Juego de pruebas realizado para la configuración de reglas

A continuación se pueden ver algunos ejemplos de posibles reglas construidas según los siguientes casos:

Caso en el que un agente es introvertido y el jefe de proyecto le ha puesto a trabajar con otros compañeros para realizar la tarea.

RULE High_aloneteam High_introextro :

ALONE TEAM: High

INTROVERTED: High

=>

DESIRE: HighDown

INTEREST: Down

DISGUST: Highup

ANXIETY: Up

STRESS: Up

Con esta regla se pretende simular el estado interno del agente (*figura 11*) y por lo tanto se agregará en el fichero *BehaviourRules.txt*.

Como consecuencia, el deseo y el interés para desempeñar esa tarea será bajo. Por otro lado, tendrá un alto nivel de estrés, ansiedad y disgusto por la tarea que deberá realizar.

Caso en el que un agente tiene poca experiencia en la tarea que se le ha asignado, y el nivel de dificultad de dicha tarea es alto.

RULE Low_experience High_taskdifficult :

EXPERIENCE: Low

TASK DIFFICULT: High

=>

CHANGE_TASK_DAYS: High_Delay

CHANGE_TASK_QUALITY: Medium_Decrease

CHANGE_GOALS_ACHIEVEMENT: Medium_Decrease

CHANGE_TEAM_COLLABORATION: Medium_Decrease

CHANGE_INDIVIDUAL_CONTRIBUTION: Normal

CHANGE_REQUIRED_SUPERVISION: Medium_Increase

Con esta regla se pretende simular el desempeño del agente (*figura 11*) y por lo tanto se agregará en el fichero *TasksRules.txt*.

Como consecuencia, el cumplimiento de las metas tendrá un alto retraso debido a la dificultad que le ha implicado. El nivel de supervisión se habrá visto incrementado, la calidad del trabajo realizado en cambio habrá disminuido así como el resto de parámetros.

Caso en el que un agente ya ha acabado la tarea y lo ha hecho con retraso.

RULE HighDelay_duration :

DURATION: HighDelay

=>

DESIRE: Down

INTEREST: Down

DISGUST: HighUp

ANXIETY: Up

STRESS: Up

Con esta regla se pretende simular el estado interno del agente (*figura 10*) y por lo tanto se agregará en el fichero *BehaviourRules.txt*.

Como consecuencia, su nivel de disgusto será alto, así como la ansiedad y el estrés que le habrá causado realizar esa tarea. El interés y el deseo por otro lado habrá bajado debido a que no ha podido acabar el trabajo a tiempo.

Los dos ficheros generados por la interfaz gráfica (*BehaviourRules.txt* y *TasksRules.txt*) han sido probados dentro del SMA TEAKS y han funcionado correctamente.

5.4- Conclusiones

En este capítulo se ha presentado el juego de pruebas con tres tipos de casos a través de la herramienta de Jade donde se han podido observar los gráficos generados junto con un seguimiento del funcionamiento del SMA. El caso 1 ha consistido en la simulación de cinco trabajadores a formar el equipo de trabajo, tres de ellos con el rol de técnico y los otros dos uno como coordinador y el otro como especialista. A cada uno se les ha asignado una tarea a desarrollar donde una de ellas está compartida entre dos trabajadores. Durante la ejecución se ha podido ver como se han llevado a cabo los tres protocolos de comunicación: *Request Protocol* de la comunicación inicial, *Contract-Net Protocol* y *Request Protocol* para la obtención del estado interno entre trabajadores que

comparten tarea. En el caso 2 se ha hecho la simulación de tres trabajadores a formar el equipo de trabajo, todos ellos con el rol de técnico. A cada uno se les ha asignado una tarea a desarrollar donde una de ellas está compartida entre dos trabajadores. Durante la ejecución se ha podido ver como se han llevado a cabo los tres protocolos de comunicación: *Request Protocol* de la comunicación inicial, *Contract-Net Protocol* y *Request Protocol* para la obtención del estado interno entre trabajadores que comparten tarea. El último caso ha sido el caso 3 donde se ha hecho la simulación de tres trabajadores a formar el equipo de trabajo, todos ellos con el rol de técnico. A cada uno se les ha asignado una tarea a desarrollar donde una de ellas está compartida entre dos trabajadores. Durante la ejecución se ha podido ver como se han llevado a cabo dos de los tres protocolos de comunicación: *Request Protocol* de la comunicación inicial y para la obtención del estado interno entre trabajadores que comparten tarea.

En este capítulo también se ha presentado tres posibles casos relacionados con la configuración de reglas. El primer caso se pretende simular el comportamiento de un agente introvertido que se le ha asignado una tarea con otros compañeros. Con esa regla lo que se pretende es simular su estado interno en el momento en el que se le ha asignado la tarea y los compañeros de equipo con los que trabajará. El segundo caso, se pretende simular el desempeño de un agente que tiene poca experiencia en la tarea que se le ha asignado, teniendo un nivel de dificultad alto. Con esa regla lo que se pretende es simular su desempeño en la tarea asignada. El tercer y último caso, se pretende simular el comportamiento de un agente que ha acabado su tarea con retraso. Con esa regla lo que se pretende es simular su estado interno una vez acabada la tarea asignada.

CAPÍTULO VI: CONCLUSIONES

Mi trabajo realizado ha sido la creación de una ontología para mejorar la comunicación inicial de los agentes en la tesis doctoral realizada en el DEIM por Juan Martínez Miranda y con ello la extensión de dos protocolos, uno de ellos basado en la negociación entre los agentes de una determinada tarea y el otro para conocer las características internas de los compañeros que comparten tarea. Adicionalmente se ha desarrollado una interfaz gráfica que facilita la configuración dinámica de las reglas difusas que generan el posible comportamiento del equipo frente a un proyecto específico.

Las aportaciones de mi trabajo han contribuido a la mejora en la comunicación de los agentes, facilitando mediante la creación de la ontología la comprensión entre ellos, pasando a través de los mensajes la información completa (como es el caso de la tarea), evitando así el tener que acceder a vectores compartidos adicionales para poder obtener la información deseada. Además de la extensión de dos protocolos de comunicación que han mejorado la comunicación inicial, uno de ellos para la obtención de las características internas de un agente en el caso que compartan la tarea y el otro para la negociación de una tarea en el caso que el agente no esté disponible para realizarla.

La parte de la interfaz gráfica desempeñada ha contribuido a la mejora de la utilización del modelo TEAKS en cualquier tipo de caso de estudio. De esta manera, es el propio usuario quien decide la configuración de reglas más adecuada para el tipo de equipo y proyecto a analizar.

A lo largo de la documentación se han explicado una serie de controles para poder simular el comportamiento del agente. Estos serán sustituidos en la tesis por sus reglas del comportamiento.

Para llevar a cabo ese trabajo serán necesarios todos los ficheros generados por el editor Protégé obtenidos tras la creación de la ontología, la modificación de la implementación desarrollada en la tesis como por ejemplo el vector que contiene las tareas asignadas para los trabajadores no será necesario que continúe siendo una variable global, ya que lo correcto será que esa información solo la conozca el agente Asignador, encargado de distribuir las tareas a los miembros del equipo.

También será necesaria la modificación del paso de mensajes en la comunicación de los agentes para poder incorporar la ontología así como la adición de los dos protocolos considerados en mi parte del trabajo.

La última parte desarrollada ha sido una interfaz gráfica para configurar las reglas dentro del SMA. Estas reglas simulan el comportamiento de un agente, y con esta interfaz se ha mejorado su configuración. Se ha creado el generador de reglas para evitar modificar manualmente el fichero donde están almacenadas, sin la necesidad de conocer la sintaxis del lenguaje JESS, en el cual están escritas y conseguir así generar el posible comportamiento de acuerdo a las características del equipo y proyecto que se quiera analizar. Para ello se han creado los ficheros (ConfigurationBehaviourRule.java, ConfigurationTaskRule.java, panel.java y menu.java).

Este proyecto me ha ayudado a ampliar mis conocimientos en el área de Inteligencia Artificial. A entender la importancia de un sistema basado en agentes, a conocer una herramienta para el desarrollo del SMA, conocer algunos tipos de comunicación y ser capaz de desarrollar una ontología para que los agentes del sistema sean capaces de entenderse.

CAPÍTULO VII: REFERENCIAS Y PÁGINAS WEB

[Bellifemine et al., 2005] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco (TILAB, formerly CSELT), Giovanni Rimaza (University of Parma).” *JADE PROGRAMMER’S GUIDE*”. Pág.4.

[Biegler et al, 1997] L. T. Biegler, I. E. Grossmann and A. Westerberg, *Systematic Methods of Chemical Process Design*, Prentice Hall International Series. In The Physical and Chemical Engineering Sciences. Chapt. I pp 1-21, 1997.

[Caire, 2002] Giovanni Caire(TILAB, formerly CSELT). “*Jade Tutorial. Application-defined content languages and ontologies*”.

[FIPA Contract Net Interaction Protocol Specification, 2002]

<http://www.fipa.org/specs/fipa00029/>

[FIPA Request Interaction Protocol Specification, 2002]

<http://www.fipa.org/specs/fipa00026/>

[FIPA Agent Management Specification. Foundation for Intelligent Physical Agents, 2000] <http://www.fipa.org>

[Hayes-Roth, 1995] B. Hayes-Roth. “An Architecture for Adaptive Intelligent Systems”, *Artificial Intelligence*, 1995

[Luck et al., 2003] Luck, M., McBurney, P. and Preist, C. “Agent Technology: Enabling Next Generation Computing: A Roadmap for Agent Based Computing”. AgentLink II, 2003.

[Martínez-Miranda et al., 2003] Martínez-Miranda Juan, Aldea Arantza. and Bañares-Alcántara René. “Simulations of Work Teams using a Multi-Agent System” in Proceedings of the 2nd International Joint Conference on Autonomous Agents & Multi Agent Systems, AAMAS 2003, pp. 1064-1065 (2003). ISBN: 1-58113-683-8.

[Martínez-Miranda et al., 2005] Martínez-Miranda J., Alvarado M., Aldea A. and Bañares-Alcántara R. “Modelling Human Behaviour at Work using Fuzzy Logic: The Challenge of Work Teams Configuration” Submitted to Journal of Intelligent & Fuzzy Systems. Reza Langari (Ed). ISSN: 1064-1246.

[Martínez-Miranda et al., 2005b] J. Martínez-Miranda, M. Alvarado, A. Aldea and R. Bañares-Alcántara. “Industrial Validation of a System to Support the

Configuration of Teams” To appear in Proceedings of the 7th World Congress of Chemical Engineering, Glasgow 2005.

[**Noy and McGuiness, 2001**] Natalia F. Noyd and Deborah L. Macguiness. “Ontology Development 101: A Guide to Creating your First Ontology”.

[**Sanchez-Alonso, 2003**] “Investigación y Análisis de Ontologías para el manejo de información en el World Wide Web”. Javier Sánchez Alonso, Proyecto Final de Carrera curso 02-03

[**Protégé, 2000**] The Protégé Project. <http://protege.stanford.edu>

[**Weiss, 1999**] G. Weiss “*Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*”. In *Proceedings of PAAM’99*, pages 97-108. 1999. The MIT Press, 1999.

[**Wooldridge and Jennings, 1995**] Wooldridge M. and Jennings N. R. “Intelligent Agents: Theory and Practice” *The Knowledge Engineering Review*, Vol. 10, No. 2, pp 115-152.

[**Zadeh, 1965**] L. A. Zadeh, “*Fuzzy Sets*”. *Information and Control*, 8. pp. 338-359, 1965.

<http://java.sun.com/j2se/1.5.0/docs/api/>

Documentación sobre Java

<http://agentcities.cs.bath.ac.uk/docs/jade/>

Documentación de Jade

<http://protege.stanford.edu/doc>

Documentación Ontologías

<http://www.fipa.org>

Especificaciones FIPA

<http://protege.stanford.edu/>

Editor Protégé y plug-in Bean Generator

<http://jade.tilab.com/>

Jade (software implementado en Java)

ANEXO:

Agent Gui.java:

```
import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.*;
import jade.content.lang.sl.SLCodec;
import jade.content.lang.Codec;
import jade.content.onto.Ontology;
import jade.content.ContentManager;
import jade.content.ContentElement;
import jade.util.leap.*;

public class AgentGui extends Agent{

    private ContentManager manager= (ContentManager)getContentManager();
    private Codec codec= new SLCodec();
    private Ontology ontology=ProyectoOntology.getInstance();
    private boolean finished;
    protected void setup(){
        //Registering content language and ontology to an agent
        manager.registerLanguage(codec);
        manager.registerOntology(ontology,ontology.getName());
        System.out.println("My name is "+this.getLocalName());
        addBehaviour(new SimpleBehaviour(this){
            public void action(){
                try{
                    // Prepare the REQUEST message
                    ACLMessage msg1= new ACLMessage(ACLMessage.REQUEST);
                    msg1.setSender(getAID());
                    msg1.addReceiver(new AID("COUNTER", AID.ISLOCALNAME));
                    msg1.setContent("1"); // Number of simulations
                }
                send(msg1);
                finished=true;
            }
        });
    }
}
```

```

System.out.println(getLocalName()+": envia peticion\n"+"("+msg1.getContent()+")");
} catch (Exception e){
    e.printStackTrace();
}
}

public boolean done(){
    return finished;
}

});
addBehaviour(new CyclicBehaviour(this){
public void action(){
    try{
        ACLMessage msg=receive();
        if (msg!=null){
            switch(msg.getPerformative()) {
                case ACLMessage.AGREE:
                    System.out.println(getLocalName()+": ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");
                    break;
                case ACLMessage.INFORM:
                    if ("INFORM-DONE".equals(msg.getContent())) //se han acabado todas
las simulaciones
                    {
                        System.out.println(getLocalName()+": recibe mensaje con el
contenido\n"+"("+msg.getContent()+ ")\n");

                        // Prepare the REQUEST message
                        ACLMessage msg2= new ACLMessage(ACLMessage.REQUEST);
                        msg2.setSender(getAID());
                        msg2.addReceiver(new AID("RESULTER", AID.ISLOCALNAME));
                        msg2.setContent("informacion resultados");
                        send(msg2);

                        System.out.println(getLocalName()+": envia
peticion\n"+"("+msg2.getContent()+")");

```

```

    }
    else
    {
        ContentElement ce= manager.extractContent(msg);
        if (ce instanceof TeamResults){
            TeamResults tr=(TeamResults) ce;
            Collection listteam = (Collection)tr.getListresults();
            Agent_Performance ap=new Agent_Performance();
            System.out.println(getLocalName()+ " recibe los resultados obtenidos
de la lista: \n");
            for (Iterator it=listteam.iterator(); it.hasNext();)
            {
                ap=(Agent_Performance)it.next();
                System.out.print("Individual level: "+ap.getIndividual_level()+"\n");
                System.out.print("Goals achievement:
"+ap.getGoals_achievement()+"\n");
                System.out.print("Supervision level: "+ap.getSupervision_level()+"\n");
                System.out.print("Task duration: "+ap.getTask_duration()+"\n");
                System.out.print("Task quality: "+ap.getTask_quality()+"\n");
                System.out.print("Team level: "+ap.getTeam_level()+"\n");
                System.out.print("Id task: "+ap.getId_task()+"\n");
                System.out.println("Agent: "+ap.getAgent()+"\n");
            }
        }
        else System.out.println("Mensaje desconocido");
    }
    break;
}

}
else
{
    block();
    System.out.println(getLocalName()+ " esta bloqueado");
}

```

```

    }
    }catch (Exception e){
        e.printStackTrace();
    }
}

});
}
}

```

Agent Counter.java:

```

import jade.core.*;
import jade.core.behaviours.*;
import jade.lang.acl.*;
import jade.content.ContentManager;
import jade.content.lang.sl.SLCodec;
import jade.content.lang.Codec;
import jade.content.onto.Ontology;

public class AgentCounter extends Agent{
    private ContentManager manager= (ContentManager)getContentManager();
    private Codec codec= new SLCodec();
    private Ontology ontology=ProyectoOntology.getInstance();
    private int c=0;
    private int numsimula;
    private boolean end=false;
    protected void setup(){
        System.out.println("My name is "+this.getLocalName());
        //Registering content language and ontology to an agent
        manager.registerLanguage(codec);
        manager.registerOntology(ontology);
        addBehaviour(new CyclicBehaviour(this){
            public void action() {
                try{

```

```

ACLMessage msg=receive();
if (msg!=null)
{
switch(msg.getPerformative()) {
    case ACLMessage.REQUEST:
        numsimula = Integer.parseInt(msg.getContent());
        System.out.println(getLocalName()+": ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");
        end=true;
        break;
    case ACLMessage.AGREE:
        System.out.println(getLocalName()+": ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");
        break;
    case ACLMessage.INFORM:
        System.out.println(getLocalName()+": ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");
        c++;
        end=true;
        if (c==numsimula)
        {
            // Prepare the INFORM message
            ACLMessage msg4= new ACLMessage(ACLMessage.INFORM);
            msg4.setSender(getAID());
            msg4.addReceiver(new AID("GUI", AID.ISLOCALNAME));
            msg4.setContent("INFORM-DONE");
            send(msg4);
            System.out.println(getLocalName()+": envia mensaje con el
contenido\n"+"("+msg4.getContent()+ ")\n");
            end=false;
        }

        break;
}
}

```

```

if (c<numsimula && end)
{
// Prepare the REQUEST message
    ACLMessage msg2= new ACLMessage(ACLMessage.REQUEST);
    msg2.setSender(getAID());
    msg2.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));
    msg2.setContent("Asignar tareas");
    send(msg2);
    System.out.println(getLocalName()+": envia
peticion\n"+"("+msg2.getContent()+")");
    // Prepare the AGREE message
    ACLMessage msg3= new ACLMessage(ACLMessage.AGREE);
    msg3.setSender(getAID());
    msg3.addReceiver(new AID("GUI", AID.ISLOCALNAME));
    msg3.setContent("AGREE");
    send(msg3);
    System.out.println(getLocalName()+": envia mensaje con el
contenido\n"+"("+msg3.getContent()+")\n");
    end=false;
}
}

else
{
    block();
    System.out.println(getLocalName()+ " esta bloqueado");
}
} catch (Exception e){
    e.printStackTrace();
}

}
});
}
}

```

Agent Assigner.java:

```
import jade.core.*;
import jade.lang.acl.*;
import jade.core.behaviours.*;
import jade.content.lang.sl.SLCodec;
import jade.content.lang.Codec;
import jade.content.onto.Ontology;
import jade.content.onto.basic.Action;
import jade.content.ContentManager;
import jade.content.ContentElement;
import jade.domain.FIPAAException;
import jade.domain.FIPAAgentManagement.ServiceDescription;
import jade.domain.FIPAAgentManagement.DFAgentDescription;
import jade.domain.DFService;
import java.util.*;
```

```
public class AgentAssigner extends Agent{
```

```
    private ContentManager manager= (ContentManager)getContentManager();
    private Codec codec= new SLCodec();
    private Ontology ontology=ProyectoOntology.getInstance();
    private int Tasks_Complets = 0;
    private List listIdTasks=new ArrayList();
    private int numAgents=5;
```

```
    private AID[] searchAgents(){
        AID[] listAgents=null;
        DFAgentDescription dfd=new DFAgentDescription();
        ServiceDescription sd=new ServiceDescription();
        sd.setType("TECHNICIAN");
        dfd.addServices(sd);
```

```

try{
    DFAgentDescription[] result=DFService.search(this,dfd);
    listAgents=new AID[result.length];

    for (int i=0;i<result.length;i++)
    {
        listAgents[i]=result[i].getName();
    }

} catch (FIPAException fe){
    fe.printStackTrace();
}

return listAgents;
}

```

```

protected void setup(){

    System.out.println("My name is "+this.getLocalName());
    //Registering content language and ontology to an agent
    manager.registerLanguage(codec);
    manager.registerOntology(ontology,ontology.getName());

```

```

    addBehaviour(new CyclicBehaviour(this){

```

```

        public void action() {

```

```

try{

    ACLMessage msg=receive();
    if (msg!=null){
        switch(msg.getPerformative()) {

```

```

        case ACLMessage.REQUEST:

            System.out.println(getLocalName()+" : ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");

            // Prepare the AGREE message
            ACLMessage msg2= new ACLMessage(ACLMessage.AGREE);
            msg2.setSender(getAID());
            msg2.addReceiver(new AID("COUNTER", AID.ISLOCALNAME));
            msg2.setContent("AGREE");
            send(msg2);

            System.out.println(getLocalName()+" : envia mensaje con el
contenido\n"+"("+msg2.getContent()+ ")\n");

        for (int c=0;c<numAgents;c++)
        {
            // Prepare the REQUEST message
            ACLMessage msg3= new ACLMessage(ACLMessage.REQUEST);
            msg3.setSender(getAID());
            msg3.setLanguage(codec.getName());
            msg3.setOntology(ProyectoOntology.ONTOLOGY_NAME);
            msg3.addReceiver(new AID("WORKER"+c,AID.ISLOCALNAME));
            Tasks t=new Tasks();
            AssignsTask at=new AssignsTask();
            t.setType("HIGH");
            t.setDeadline(3);
            t.setCost(2);
            t.setDuration(22);
            t.setQuality("HIGH");
            t.setPriority("HIGH");
            t.setNum_participants(2);
            t.setDifficulty("HIGH");
            if(("WORKER"+c).compareTo("WORKER0")==0)
                t.setId(22);
            else if(("WORKER"+c).compareTo("WORKER3")==0)

```

```

        t.setId(10);
    else if(("WORKER"+c).compareTo("WORKER4")==0)
        t.setId(5);
    else
        t.setId(15);
    at.setTask(t);
    Action ac=new Action(getAID(),at);
    manager.fillContent(msg3,ac);
    send(msg3);

    System.out.println(getLocalName()+" envia mensaje con el contenido
(request)\n"+"("+t.getId()+ ")\n");
    }
    break;
    case ACLMessage.REFUSE:
        System.out.println(getLocalName()+" ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");
        break;
    case ACLMessage.AGREE:
        System.out.println(getLocalName()+" ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");
        break;
    case ACLMessage.FAILURE:

        StringTokenizer st2 = new StringTokenizer(msg.getContent());
        String c= st2.nextToken();
        c= st2.nextToken();
        String cad2="NOT-AVAILABLE "+c;

        if (cad2.equals(msg.getContent()))
        {
            System.out.println(getLocalName()+" recibe
FAILURE\n"+"("+msg.getContent()+")\n");
            Tasks_Complets++;

```

```

    }
else
{

    ContentElement ce2= manager.extractContent(msg);
    if (ce2 instanceof TaskResults){
        TaskResults tr2= (TaskResults) ce2;
        Tasks t3=tr2.getTask();
        System.out.println(getLocalName()+" : ha recibido mensaje con el
contenido (failure)\n"+"("+msg.getContent()+")\n");

        AID[] listAgents=searchAgents();

        if ((listAgents.length>1) ||
(listAgents.length==1 && !listAgents[0].getLocalName().equals(msg.getSender().getLocalName()))) )
        {
            // Prepare the CFP message
            ACLMessage cfp= new ACLMessage(ACLMessage.CFP);

            //enviar a todos los agentes de un determinado tipo menos a él mismo
            int i=0;
            while(i<listAgents.length)
            {
                if
(!listAgents[i].getLocalName().equals(msg.getSender().getLocalName()))
                {
                    cfp.addReceiver(listAgents[i]);
                }

                i++;
            }

            cfp.setLanguage(codec.getName());

```

```

cfp.setOntology(ProyectoOntology.ONTOLOGY_NAME);
cfp.setSender(getAID());
cfp.setReplyWith("cfp"+System.currentTimeMillis());
    AssignsTask at2=new AssignsTask();
    at2.setTask(t3);
    Action ac2=new Action(getAID(),at2);
    manager.fillContent(cfp,ac2);
    send(cfp);
    listIdTasks.add(new Integer(t3.getId()));
}

    else System.out.println("No hay agentes disponibles");
}
}
break;
case ACLMessage.INFORM:

    ContentElement ce= manager.extractContent(msg);
    if (ce instanceof TaskResults)
    {
        TaskResults tr= (TaskResults) ce;
        Tasks t2=tr.getTask();

        System.out.println(getLocalName()+" ha recibido mensaje con el
contenido (inform-done)\n"+"("+msg.getContent()+")\n");
    }
    else System.out.println("Mensaje desconocido");

    Tasks_Complets ++;
    break;
case ACLMessage.PROPOSE:

    ContentElement ce3= manager.extractContent(msg);

```

```

        if (ce3 instanceof TaskResults){
            TaskResults tr3= (TaskResults) ce3;
            Tasks t4=tr3.getTask();
            int i=listIdTasks.indexOf(new Integer(t4.getId()));
            if(i!=-1)
            {
                ACLMessage msgPr=new
ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
                msgPr.setSender(getAID());
                msgPr.addReceiver(msg.getSender());
                msgPr.setLanguage(codec.getName());
                msgPr.setOntology(ProyectoOntology.ONTOLOGY_NAME);

                manager.fillContent(msgPr,tr3);
                listIdTasks.remove(i);
                send(msgPr);

                System.out.println(getLocalName()+" envia mensaje (Acepta
Propose) con el contenido\n"+"("+msgPr.getContent());
            }
            else
            {
                ACLMessage msgRe=new
ACLMessage(ACLMessage.REJECT_PROPOSAL);
                msgRe.setSender(getAID());
                msgRe.addReceiver(msg.getSender());
                msgRe.setContent(new Integer(t4.getId()).toString());
                send(msgRe);
                System.out.println(getLocalName()+" envia mensaje (Reject propose)
con el contenido\n"+"("+msgRe.getContent());
            }
        }
        else System.out.println("Mensaje desconocido");
        break;

```

```

    }
    if (Tasks_Complets==numAgents)
    {
        ACLMessage msg4= new ACLMessage(ACLMessage.INFORM);
        msg4.setSender(getAID());
        msg4.addReceiver(new AID("COUNTER", AID.ISLOCALNAME));
        msg4.setContent("INFORM-DONE");
        send(msg4);
        System.out.println(getLocalName()+" envia mensaje con el
contenido\n"+"("+msg4.getContent()+ ")\n");
        Tasks_Complets=0;
    }
}

else{
    block();
    System.out.println(getLocalName()+ " esta bloqueado");
}

} catch (Exception e){
    e.printStackTrace();
}

}
});
}
}

```

Agent Resulter:

```

import jade.core.*;
import jade.lang.acl.*;
import jade.core.behaviours.*;
import jade.content.lang.sl.SLCodec;
import jade.content.lang.Codec;
import jade.content.onto.Ontology;

```

```

import jade.content.ContentManager;
import jade.content.ContentElement;
import jade.content.onto.basic.Action;

public class AgentResulter extends Agent{
private ContentManager manager= (ContentManager)getContentManager();
private Codec codec= new SLCodec();
private Ontology ontology=ProyectoOntology.getInstance();
private TeamResults tr=new TeamResults();
protected void setup(){
    System.out.println("My name is "+this.getLocalName());

    //Registering content language and ontology to an agent
    manager.registerLanguage(codec);
    manager.registerOntology(ontology,ontology.getName());

    addBehaviour(new CyclicBehaviour(this){
        public void action() {
            try{
                ACLMessage msg=receive();

                if (msg!=null){
                    switch(msg.getPerformative()) {
                        case ACLMessage.INFORM:
                            ContentElement ce= manager.extractContent(msg);
                            if (ce instanceof BehaviourResults){
                                BehaviourResults br=(BehaviourResults) ce;
                                Agent_Performance ap=br.getInformation_performance();
                                System.out.print(getLocalName()+ ": recibe mensaje
(informe)" +ap.getIndividual_level()+ " Individual level");
                                System.out.print(ap.getGoals_achievement());
                                System.out.print(ap.getSupervision_level());
                                System.out.print(ap.getTask_duration());
                                System.out.print(ap.getTask_quality());

```

```

        System.out.print(ap.getTeam_level());
        System.out.print(ap.getId_task());
        System.out.println(ap.getAgent());
        tr.addListresults(ap);
    }
    else System.out.println("Mensaje desconocido");
    break;

    case ACLMessage.REQUEST:
        System.out.println(getLocalName()+": ha recibido mensaje con el
contenido (request)\n"+"("+msg.getContent()+")\n");

        // Prepare the INFORM message
        ACLMessage msgInf = new ACLMessage(ACLMessage.INFORM);
        msgInf.setSender(getAID());
        msgInf.addReceiver(new AID("GUI", AID.ISLOCALNAME));
        msgInf.setLanguage(codec.getName());
        msgInf.setOntology(ProyectoOntology.ONTOLOGY_NAME);
        manager.fillContent(msgInf,tr);
        send(msgInf);
        System.out.println(getLocalName()+": ha enviado la lista");

        break;
    }
}
else
{
    block();
    System.out.println(getLocalName()+ " esta bloqueado");
}
} catch (Exception e){
    e.printStackTrace();
}
}
});

```

```
}  
}
```

Agent Worker.java:

```
import jade.core.*;  
import jade.lang.acl.*;  
import jade.core.behaviours.*;  
import jade.content.onto.basic.Action;  
import jade.content.lang.sl.SLCodec;  
import jade.content.lang.Codec;  
import jade.content.onto.Ontology;  
import jade.content.ContentManager;  
import jade.content.ContentElement;  
import jade.content.Concept;  
import jade.domain.FIPAAgentManagement.ServiceDescription;  
import jade.domain.FIPAAgentManagement.DFAgentDescription;  
import jade.domain.DFService;  
import jade.domain.FIPAException;  
import jade.util.leap.*;  
import java.util.Random;  
import java.util.StringTokenizer;  
  
public class AgentWorker extends Agent{  
    private ContentManager manager= (ContentManager)getContentManager();  
    private Codec codec= new SLCodec();  
    private Ontology ontology=ProyectoOntology.getInstance();  
    private Random r=new Random();  
    private List listAgents=new ArrayList();  
    private Tasks task=new Tasks();  
    private int numAgents=5;  
    private int numReq=0;  
    private int num=0;  
    private boolean finished=false;
```

```

void registerDF() {

    //Preparación para dar de alta en el DF
    DFAgentDescription dfd = new DFAgentDescription();
    dfd.setName(getAID());

    ServiceDescription sd = new ServiceDescription();
    if (getLocalName().equals("WORKER1") )
        sd.setType("COORDINATOR");
    else
        if (getLocalName().equals("WORKER3") )
            sd.setType("SPECIALIST");
        else
            sd.setType("TECHNICIAN");

    for (int c=0;c<numAgents;c++)
    {
        sd.setName("WORKER"+c);
    }
    dfd.addServices(sd);
    try{
        DFService.register(this,dfd);

    }catch (FIPAException fe){
        fe.printStackTrace();
    }

}

protected void setup(){

    System.out.println("My name is "+this.getLocalName());
    //Registering content language and ontology to an agent
    manager.registerLanguage(codec);
}

```

```

        manager.registerOntology(ontology,ontology.getName());
registerDF();
addBehaviour(new CyclicBehaviour(this){

public void action() {

    try{

        ACLMessage msg=receive();

        if (msg!=null){

            switch(msg.getPerformative()) {

                case ACLMessage.REQUEST:

                    if ("INTERNAL-STATE".equals(msg.getContent()))
                        {

                            System.out.println(getLocalName()+" : ha recibido mensaje con el
contenido\n"+"("+msg.getContent()+")\n");

                            // Prepare the INFORM message

                            AgentCharacteristics agch=new AgentCharacteristics();

                            agch.setName_agent(getLocalName());

                            agch.setStress("MEDIUM");

                            Personality p3=new Personality();
                            p3.setAmiable("HIGH");
                            p3.setAnalytical("HIGH");
                            p3.setDriver("LOW");
                            p3.setExpressive("LOW");

```

```

agch.setPersonalitycharacteristics(p3);

    Social s=new Social();
    s.setIntroverted("LOW");
    s.setWork_alone("LOW");
    s.setWork_team("HIGH");
    s.setExtroverted("HIGH");

    agch.setSocialcharacteristics(s);

    Cognition c=new Cognition();
    c.setCreativity("MEDIUM");
    if (getLocalName().equals("WORKER1") )
    {
        c.setExperience("HIGH");
        c.setRole("COORDINATOR");
    }
    else
    if (getLocalName().equals("WORKER2") )
    {
        c.setExperience("MEDIUM");
        c.setRole("SPECIALIST");
    }
    else
    {
        c.setExperience("LOW");
        c.setRole("TECHNICIAN");
    }

    agch.setCognitioncharacteristics(c);

    ACLMessage reply = msg.createReply();
    reply.setPerformative(ACLMessage.INFORM);

```

```

        reply.setLanguage(codec.getName());
        reply.setOntology(ProyectoOntology.ONTOLOGY_NAME);
        reply.setInReplyTo(msg.getReplyWith());
        manager.fillContent(reply,agch);
        send(reply);
        System.out.println(getLocalName()+" : envia el informe mensaje "+
reply.getContent());

    }

else if ("WHAT TASK".equals(msg.getContent()))
{

    // Prepare the INFORM message
    ACLMessage replyInf = msg.createReply();
    replyInf.setPerformative(ACLMessage.INFORM);
    replyInf.setInReplyTo(msg.getReplyWith());
    replyInf.setContent("ID "+new Integer(task.getId()).toString());
    send(replyInf);
    System.out.println(getLocalName()+" : envia el informe mensaje "+
replyInf.getContent());

}

else

    {

        ContentElement ce= manager.extractContent(msg);
        Concept action=((Action)ce).getAction();
        AssignsTask at= (AssignsTask) action;
        Tasks t=at.getTask();
        if (action instanceof AssignsTask)
        {
            if (msg.getSender().getLocalName().equals("ASSIGNER"))
            {

```

```

        //eliminar agentes de la lista
        for(int c=0;c<listAgents.size();c++)
            listAgents.remove(c);

        task=t;

        // Prepare the REQUEST message (qué agentes tienen esa tarea)
        doWait(2000);
        ACLMessage msgReqTask = new
        ACLMessage(ACLMessage.REQUEST);
        msgReqTask.setSender(getAID());
        msgReqTask.setSender(getAID());
        msgReqTask.setContent("WHAT TASK");
        msgReqTask.setReplyWith(getLocalName()+"WHAT TASK");
        for (int c=0;c<numAgents;c++)
        {
            if (getLocalName().compareTo("WORKER"+c)!=0)
                msgReqTask.addReceiver(new
        AID("WORKER"+c,AID.ISLOCALNAME));
        }
        send(msgReqTask);
    }

}

else System.out.println("Mensaje desconocido");
}

break;

case ACLMessage.INFORM:
    StringTokenizer st2 = new StringTokenizer(msg.getContent());
    String str= st2.nextToken();
    str= st2.nextToken();
    String cad2="ID "+str;

```

```

        if (cad2.equals(msg.getContent()))
        {
            int id_t=Integer.parseInt(str);
// comparar si tiene el mismo id y guardar en la lista
            if(task.getId()== id_t)
            {
                listAgents.add(msg.getSender().getLocalName());
            }
            numReq++;

            if(numReq==numAgents-1)
            {
                if (listAgents.size()>0)
                {
                    // Prepare the REQUEST message

                    ACLMessage msgReq = new ACLMessage(ACLMessage.REQUEST);
                    msgReq.setSender(getAID());
                    msgReq.setContent("INTERNAL-STATE");
                    msgReq.setReplyWith(getLocalName()+"Internal-State");
                    for (int c=0;c<listAgents.size();c++)
                    {
                        msgReq.addReceiver(new
AID(listAgents.get(c).toString(),AID.ISLOCALNAME));
                        System.out.println(getLocalName()+" envia el mensaje "+
msgReq.getContent()+listAgents.get(c));
                    }
                    send(msgReq);
                }

                if(listAgents.size()==0)
                {
                    finished=true;
                }
            }
        }
    }
}

```

```

    }
    }
    else{
        num++;
        if(num==listAgents.size())
            // Si el Agent Worker recibe el estado interno del agente
            { finished=true;
            }

        System.out.println(getLocalName()+" recibe mensaje
\n"+"("+msg.getContent()+ ")\n");
    }

    break;
case ACLMessage.CFP:
    ContentElement ce2= manager.extractContent(msg);
    Concept action2=((Action)ce2).getAction();
    AssignsTask at2= (AssignsTask) action2;
    Tasks t4=at2.getTask();
    if (action2 instanceof AssignsTask)
    {
        System.out.println(getLocalName()+" recibe un CFP
"+"("+t4.getId()+")\n");

        ACLMessage replyCFP=msg.createReply();
        replyCFP.setInReplyTo(msg.getReplyWith());
        //int i=r.nextInt(2);

        //if (i==0)//no eliminar
        // {

        // Prepara el mensaje del tipo PROPOSE
        replyCFP.setPerformative(ACLMessage.PROPOSE);

```

```

        TaskResults tr3=new TaskResults();
        tr3.setTask(t4);
        manager.fillContent(replyCFP,tr3);

// }
/*else //no eliminar
{
    // Prepare the REFUSE message

    replyCFP.setPerformative(ACLMessage.REFUSE);
    replyCFP.setContent(getLocalName()+" :envia un REFUSE (No
disponible)" +t4.getId());
    }*/
    send(replyCFP);
}
else System.out.println("Mensaje desconocido");
break;
case ACLMessage.ACCEPT_PROPOSAL:
    ContentElement ce3= manager.extractContent(msg);
    if (ce3 instanceof TaskResults){
        TaskResults tr4= (TaskResults) ce3;
        Tasks t5=tr4.getTask();
        System.out.println(getLocalName()+ ": recibe mensaje con el contenido
(ACCEPT_PROPOSAL)\n"+t5.getId()+"\n");
        int i=r.nextInt(2);

        if (i==0)
        {
            // Prepare the INFORM message
            ACLMessage msgInfD2 = new ACLMessage(ACLMessage.INFORM);
            msgInfD2.setSender(getAID());
            msgInfD2.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));
            msgInfD2.setLanguage(codec.getName());
            msgInfD2.setOntology(ProyectoOntology.ONTOLOGY_NAME);

```

```

//msgInfD2.setContent("ID "+t5.getId());
manager.fillContent(msgInfD2,tr4);
send(msgInfD2);
System.out.println(getLocalName()+" envia mensaje con el
contenido\n"+"(inform-done (Task: (Id:"+t5.getId()+"))));

// Prepare the INFORM message
ACLMessage msgInfIn = new ACLMessage(ACLMessage.INFORM);
msgInfIn.setSender(getAID());
msgInfIn.addReceiver(new AID("RESULTER", AID.ISLOCALNAME));
msgInfIn.setLanguage(codec.getName());
msgInfIn.setOntology(ProyectoOntology.ontology_NAME);

Agent_Performance ap=new Agent_Performance();
ap.setIndividual_level("MEDIUM");
ap.setGoals_achievement("HIGH");
ap.setSupervision_level("MEDIUM");
ap.setTask_duration("LOW");
ap.setTask_quality("HIGH");
ap.setTeam_level("MEDIUM");
ap.setId_task(t5.getId());
ap.setAgent(getLocalName());
BehaviourResults br=new BehaviourResults();
br.setInformation_performance(ap);

manager.fillContent(msgInfIn,br);

send(msgInfIn);
System.out.println(getLocalName()+" envia mensaje
("+msgInfIn.getContent()+")\n");
}
else
{
// Prepara el mensaje del tipo FAILURE

```

```

        ACLMessage msgFa = new ACLMessage(ACLMessage.FAILURE);
        msgFa.setSender(getAID());
        msgFa.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));
        msgFa.setContent("NOT-AVAILABLE "+t5.getId());
        send(msgFa);
        System.out.println(getLocalName()+" : envia un FAILURE
("+msgFa.getContent()+"\n");

        // Prepare the INFORM message
        ACLMessage msgInfFa = new ACLMessage(ACLMessage.INFORM);
        msgInfFa.setSender(getAID());
        msgInfFa.addReceiver(new AID("RESULTER", AID.ISLOCALNAME));
        msgInfFa.setLanguage(codec.getName());
        msgInfFa.setOntology(ProyectoOntology.ONTOLOGY_NAME);

        Agent_Performance ap=new Agent_Performance();
        ap.setIndividual_level("LOW");
        ap.setGoals_achievement("LOW");
        ap.setSupervision_level("HIGH");
        ap.setTask_duration("HIGH");
        ap.setTask_quality("LOW");
        ap.setTeam_level("LOW");
        ap.setId_task(t5.getId());
        ap.setAgent(getLocalName());
        BehaviourResults br=new BehaviourResults();
        br.setInformation_performance(ap);

        manager.fillContent(msgInfFa,br);
        send(msgInfFa);
        System.out.println(getLocalName()+" : envia mensaje
"+msgInfFa.getContent()+"\n");
    }
}
else System.out.println("Mensaje desconocido");

```

```

        break;
    case ACLMessage.REJECT_PROPOSAL:
        System.out.println(getLocalName()+ ": recibe un REJECT_PROPOSAL
"+"("+msg.getContent()+")\n");

        break;
    }
    if(numReq==numAgents-1 && finished)
    {
        int i=r.nextInt(2);

        if (i==0)
        {
            // Prepare the AGREE message
            ACLMessage msgA = new ACLMessage(ACLMessage.AGREE);
            msgA.setSender(getAID());
            msgA.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));
            msgA.setContent("AGREE");
            send(msgA);
            System.out.println(getLocalName()+": envia el mensaje "+
msgA.getContent());
        }
        else
        {
            // Prepare the REFUSE message

            ACLMessage msgR = new ACLMessage(ACLMessage.REFUSE);
            msgR.setSender(getAID());
            msgR.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));
            msgR.setContent("REFUSE");
            send(msgR);
            System.out.println(getLocalName()+": envia el mensaje "+
msgR.getContent());

```

```

    }

    i=r.nextInt(2);

    if (i==0)
    {
        // Prepare the INFORM message
        ACLMessage msgInfD = new ACLMessage(ACLMessage.INFORM);
        msgInfD.setSender(getAID());
        msgInfD.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));
        msgInfD.setLanguage(codec.getName());
        msgInfD.setOntology(ProyectoOntology.ontology_NAME);

        TaskResults tr2=new TaskResults();
        tr2.setTask(task);
        manager.fillContent(msgInfD,tr2);
        send(msgInfD);
        System.out.println(getLocalName()+" : envia mensaje con el
contenido\n"+"(inform-done (Task: (Id:"+msgInfD.getContent()+")))");

        // Prepare the INFORM message
        ACLMessage msgInfI = new ACLMessage(ACLMessage.INFORM);
        msgInfI.setSender(getAID());
        msgInfI.addReceiver(new AID("RESULTER", AID.ISLOCALNAME));
        msgInfI.setLanguage(codec.getName());
        msgInfI.setOntology(ProyectoOntology.ontology_NAME);
        Agent_Performance ap=new Agent_Performance();
        ap.setIndividual_level("MEDIUM");
        ap.setGoals_achievement("HIGH");
        ap.setSupervision_level("MEDIUM");
        ap.setTask_duration("LOW");
        ap.setTask_quality("HIGH");
        ap.setTeam_level("MEDIUM");
        ap.setId_task(task.getId());
    }
}

```

```

        ap.setAgent(getLocalName());
        BehaviourResults br=new BehaviourResults();
        br.setInformation_performance(ap);
        manager.fillContent(msgInfI,br);
        send(msgInfI);
        System.out.println(getLocalName()+ ": envia mensaje
"+ap.getIndividual_level()+ " Individual level");
    }
    else
    {
        // Prepare the FAILURE message
        ACLMessage msgF2 = new ACLMessage(ACLMessage.FAILURE);
        msgF2.setSender(getAID());
        msgF2.addReceiver(new AID("ASSIGNER", AID.ISLOCALNAME));
        msgF2.setLanguage(codec.getName());
        msgF2.setOntology(ProyectoOntology.ONTOLOGY_NAME);
        TaskResults tr=new TaskResults();
        tr.setTask(task);
        manager.fillContent(msgF2,tr);
        send(msgF2);
        System.out.println(getLocalName()+": envia mensaje con el
contenido\n"+(failure (Task: "+msgF2.getContent()+"))");

        // Prepare the INFORM message
        ACLMessage msgInfF = new ACLMessage(ACLMessage.INFORM);
        msgInfF.setSender(getAID());
        msgInfF.addReceiver(new AID("RESULTER", AID.ISLOCALNAME));
        msgInfF.setLanguage(codec.getName());
        msgInfF.setOntology(ProyectoOntology.ONTOLOGY_NAME);

        Agent_Performance ap=new Agent_Performance();
        ap.setIndividual_level("LOW");
        ap.setGoals_achievement("LOW");
        ap.setSupervision_level("HIGH");

```

```

        ap.setTask_duration("HIGH");
        ap.setTask_quality("LOW");
        ap.setTeam_level("LOW");
        ap.setId_task(task.getId());
        ap.setAgent(getLocalName());
        BehaviourResults br=new BehaviourResults();
        br.setInformation_performance(ap);

        manager.fillContent(msgInfF,br);
        send(msgInfF);
        System.out.println(getLocalName()+ ": envia mensaje
"+ap.getIndividual_level()+ " Individual level");
    }
    numReq=0;
    num=0;
    finished=false;
}

    }

    else
    {
        block();
        System.out.println(getLocalName()+ " esta bloqueado");
    }

} catch (Exception e){
    e.printStackTrace();
}

}
});
}
}

```

ProyectoOntology.java:

```
import jade.content.onto.*;
```

```

import jade.content.schema.*;

/** file: ProyectoOntology.java
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:12
 */
public class ProyectoOntology extends Ontology {

    //NAME
    public static final String ONTOLOGY_NAME = "Proyecto";
    // The singleton instance of this ontology
    private static Ontology theInstance = new ProyectoOntology();
    public static Ontology getInstance() {
        return theInstance;
    }

    // VOCABULARY
    public static final String
BEHAVIOURRESULTS_INFORMATION_PERFORMANCE="information_perform
ance";

    public static final String BEHAVIOURRESULTS="BehaviourResults";
    public static final String TASKRESULTS_TASK="task";
    public static final String TASKRESULTS="TaskResults";
    public static final String ASSIGNSTASK_TASK="task";
    public static final String ASSIGNSTASK="AssignsTask";
    public static final String COGNITION_EXPERIENCE="experience";
    public static final String COGNITION_CREATIVITY="creativity";
    public static final String COGNITION_ROLE="role";
    public static final String COGNITION="Cognition";
    public static final String EMOTION_DESIRE="desire";
    public static final String EMOTION_ANXIETY="anxiety";
    public static final String EMOTION_INTEREST="interest";
    public static final String EMOTION_DISGUST="disgust";

```

```

public static final String EMOTION="Emotion";
public static final String TASKS_DIFFICULTY="difficulty";
public static final String TASKS_NUM_PARTICIPANTS="num_participants";
public static final String TASKS_PRIORITY="priority";
public static final String TASKS_QUALITY="quality";
public static final String TASKS_DURATION="duration";
public static final String TASKS_COST="cost";
public static final String TASKS_ID="id";
public static final String TASKS_DEADLINE="deadline";
public static final String TASKS_TYPE="type";
public static final String TASKS="Tasks";
public static final String SOCIAL_EXTROVERTED="extroverted";
public static final String SOCIAL_WORK_TEAM="work_team";
public static final String SOCIAL_INTROVERTED="introverted";
public static final String SOCIAL_WORK_ALONE="work_alone";
public static final String SOCIAL="Social";
public static final String PERSONALITY_EXPRESSIVE="expressive";
public static final String PERSONALITY_DRIVER="driver";
public static final String PERSONALITY_ANALYTICAL="analytical";
public static final String PERSONALITY_AMIABLE="amiable";
public static final String PERSONALITY="Personality";
public static final String AGENT_PROPERTIES="Agent_Properties";
public static final String
AGENT_PERFORMANCE_TASK_QUALITY="task_quality";
    public static final String AGENT_PERFORMANCE_TEAM_LEVEL="team_level";
    public static final String
AGENT_PERFORMANCE_SUPERVISION_LEVEL="supervision_level";
    public static final String
AGENT_PERFORMANCE_GOALS_ACHIEVEMENT="goals_achievement";
    public static final String
AGENT_PERFORMANCE_INDIVIDUAL_LEVEL="individual_level";
    public static final String
AGENT_PERFORMANCE_TASK_DURATION="task_duration";
    public static final String AGENT_PERFORMANCE="Agent_Performance";

```

```

    public static final String AGENT_PERFORMANCE_ID_TASK="id_Task";
    public static final String AGENT_PERFORMANCE_AGENTE="agent";
        public static final String
ASSIGNSCHARACTERISTICS_PERSONALITYCHARACTERISTICS="personality
characteristics";
        public static final String
ASSIGNSCHARACTERISTICS_SOCIALCHARACTERISTICS="socialcharacteristic
s";
        public static final String
ASSIGNSCHARACTERISTICS_COGNITIONCHARACTERISTICS="cognitionchara
cteristics";


        public static final String
ASSIGNSCHARACTERISTICS="AssignsCharacteristics";
        public static final String TEAMRESULTS_LISTRESULTS="listresults";
        public static final String TEAMRESULTS="TeamResults";
        public static final String AGENTCHARACTERISTICS_STRESS="stress";
        public static final String
AGENTCHARACTERISTICS_NAME_AGENT="name_agent";
        public static final String
AGENTCHARACTERISTICS_PERSONALITYCHARACTERISTICS="personalitych
aracteristics";
        public static final String
AGENTCHARACTERISTICS_SOCIALCHARACTERISTICS="socialcharacteristics";
        public static final String
AGENTCHARACTERISTICS_COGNITIONCHARACTERISTICS="cognitioncharact
eristics";

        public static final String AGENTCHARACTERISTICS="AgentCharacteristics";
        public static final String AGENT_PROPERTIES_NAME_AGENT="name_agent";
        public static final String AGENT_PROPERTIES_STRESS="stress";


    /**
     * Constructor
    */

```

```

private ProyectoOntology(){
    super(ONTOLOGY_NAME, BasicOntology.getInstance());
    try {

        ConceptSchema tasksSchema = new ConceptSchema(TASKS);
        add(tasksSchema, Tasks.class);

        ConceptSchema agent_PerformanceSchema = new
ConceptSchema(AGENT_PERFORMANCE);
        add(agent_PerformanceSchema, Agent_Performance.class);

        ConceptSchema agent_PropertiesSchema = new
ConceptSchema(AGENT_PROPERTIES);
        add(agent_PropertiesSchema, Agent_Properties.class);

        ConceptSchema personalitySchema = new ConceptSchema(PERSONALITY);
        add(personalitySchema, Personality.class);

        ConceptSchema socialSchema = new ConceptSchema(SOCIAL);
        add(socialSchema, Social.class);

        ConceptSchema emotionSchema = new ConceptSchema(EMOTION);
        add(emotionSchema, Emotion.class);

        ConceptSchema cognitionSchema = new ConceptSchema(COGNITION);
        add(cognitionSchema, Cognition.class);


        // adding AgentAction(s)
        AgentActionSchema assignsTaskSchema = new
AgentActionSchema(ASSIGNSTASK);
        add(assignsTaskSchema, AssignsTask.class);


        // adding Predicate(s)
        PredicateSchema agentCharacteristicsSchema = new
PredicateSchema(AGENTCHARACTERISTICS);
        add(agentCharacteristicsSchema, AgentCharacteristics.class);


        PredicateSchema taskResultsSchema = new PredicateSchema(TASKRESULTS);
        add(taskResultsSchema, TaskResults.class);

```

```

    PredicateSchema behaviourResultsSchema = new
PredicateSchema(BEHAVIOURRESULTS);
    add(behaviourResultsSchema,BehaviourResults.class);
    PredicateSchema teamResultsSchema = new PredicateSchema(TEAMRESULTS);
    add(teamResultsSchema,TeamResults.class);
    // adding fields
    agent_PerformanceSchema.add(AGENT_PERFORMANCE_TASK_DURATION,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

    agent_PerformanceSchema.add(AGENT_PERFORMANCE_INDIVIDUAL_LEVEL,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

    agent_PerformanceSchema.add(AGENT_PERFORMANCE_GOALS_ACHIEVEMENT,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

    agent_PerformanceSchema.add(AGENT_PERFORMANCE_SUPERVISION_LEVEL,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
    agent_PerformanceSchema.add(AGENT_PERFORMANCE_TEAM_LEVEL,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

    agent_PerformanceSchema.add(AGENT_PERFORMANCE_TASK_QUALITY,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
    agent_PerformanceSchema.add(AGENT_PERFORMANCE_AGENTE,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
    agent_PerformanceSchema.add(AGENT_PERFORMANCE_ID_TASK,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));

    agent_PropertiesSchema.add(AGENT_PROPERTIES_NAME_AGENT,
(PrimitiveSchema)getSchema(BasicOntology.STRING),ObjectSchema.OPTIONAL);
    agent_PropertiesSchema.add(AGENT_PROPERTIES_STRESS,
(PrimitiveSchema)getSchema(BasicOntology.STRING),ObjectSchema.OPTIONAL);

    tasksSchema.add(TASKS_TYPE,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

```

```

        tasksSchema.add(TASKS_DEADLINE,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
        tasksSchema.add(TASKS_ID,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
        tasksSchema.add(TASKS_COST,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
        tasksSchema.add(TASKS_DURATION,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
        tasksSchema.add(TASKS_QUALITY,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        tasksSchema.add(TASKS_PRIORITY,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        tasksSchema.add(TASKS_NUM_PARTICIPANTS,
(PrimitiveSchema)getSchema(BasicOntology.INTEGER));
        tasksSchema.add(TASKS_DIFFICULTY,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

        assignsTaskSchema.add(ASSIGNSTASK_TASK, tasksSchema);
        taskResultsSchema.add(TASKRESULTS_TASK, tasksSchema);

behaviourResultsSchema.add(BEHAVIOURRESULTS_INFORMATION_PERFORM
ANCE, agent_PerformanceSchema);
        teamResultsSchema.add(TeamRESULTS_LISTRESULTS,
agent_PerformanceSchema, 1, ObjectSchema.UNLIMITED);

        personalitySchema.add(PERSONALITY_AMIABLE,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        personalitySchema.add(PERSONALITY_ANALYTICAL,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        personalitySchema.add(PERSONALITY_DRIVER,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        personalitySchema.add(PERSONALITY_EXPRESSIVE,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

```

```

        socialSchema.add(SOCIAL_WORK_ALONE,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        socialSchema.add(SOCIAL_INTROVERTED,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        socialSchema.add(SOCIAL_WORK_TEAM,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        socialSchema.add(SOCIAL_EXTROVERTED,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

        emotionSchema.add(EMOTION_DISGUST,
(PrimitiveSchema)getSchema(BasicOntology.STRING),ObjectSchema.OPTIONAL);
        emotionSchema.add(EMOTION_INTEREST,
(PrimitiveSchema)getSchema(BasicOntology.STRING),ObjectSchema.OPTIONAL);
        emotionSchema.add(EMOTION_ANXIETY,
(PrimitiveSchema)getSchema(BasicOntology.STRING),ObjectSchema.OPTIONAL);
        emotionSchema.add(EMOTION_DESIRE,
(PrimitiveSchema)getSchema(BasicOntology.STRING),ObjectSchema.OPTIONAL);

        cognitionSchema.add(COGNITION_CREATIVITY,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        cognitionSchema.add(COGNITION_EXPERIENCE,
(PrimitiveSchema)getSchema(BasicOntology.STRING));
        cognitionSchema.add(COGNITION_ROLE,
(PrimitiveSchema)getSchema(BasicOntology.STRING));

agentCharacteristicsSchema.add(AGENTCHARACTERISTICS_COGNITIONCHARACTERISTICS, cognitionSchema);

agentCharacteristicsSchema.add(AGENTCHARACTERISTICS_SOCIALCHARACTERISTICS, socialSchema);

agentCharacteristicsSchema.add(AGENTCHARACTERISTICS_PERSONALITYCHARACTERISTICS, personalitySchema);

```

```

        agentCharacteristicsSchema.add(AGENTCHARACTERISTICS_STRESS,
(TermSchema)getSchema(BasicOntology.STRING));

        agentCharacteristicsSchema.add(AGENTCHARACTERISTICS_NAME_AGENT,
(TermSchema)getSchema(BasicOntology.STRING));
        // adding inheritance
        personalitySchema.addSuperSchema(agent_PropertiesSchema);
        socialSchema.addSuperSchema(agent_PropertiesSchema);
        emotionSchema.addSuperSchema(agent_PropertiesSchema);
        cognitionSchema.addSuperSchema(agent_PropertiesSchema);

    } catch (java.lang.Exception e) {e.printStackTrace();}
}
}

```

AssignsTasks.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;
/**
 * Protege name: AssignsTask
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:13
 */
public class AssignsTask implements AgentAction {
    /**
     * Protege name: task
     */
    private Tasks task;
    public void setTask(Tasks value) {
        this.task=value;  }
    public Tasks getTask() {
        return this.task;  }
}

```

TeamResults.java:

```
import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

/**
 * Protege name: TeamResults
 * @author ontology bean generator
 * @version 2005/04/12, 16:29:58
 */
public class TeamResults implements Predicate {

    /**
     * Protege name: listresults
     */
    private List listresults = new ArrayList();

    public void addListresults(Agent_Performance elem) {
        listresults.add(elem);
    }

    public boolean removeListresults(Agent_Performance elem) {

        boolean result = listresults.remove(elem);
        return result;
    }

    public void clearAllListresults() {

        listresults.clear();
    }

    public Iterator getAllListresults() {return listresults.iterator(); }
    public List getListresults() {return listresults; }
    public void setListresults(List l) {listresults = l; }
```

```
}
```

TaskResults.java:

```
import jade.content.*;
```

```
import jade.util.leap.*;
```

```
import jade.core.*;
```

```
/**
```

```
 * Protege name: TaskResults
```

```
 * @author ontology bean generator
```

```
 * @version 2005/04/6, 20:45:13
```

```
 */
```

```
public class TaskResults implements Predicate {
```

```
    /**
```

```
     * Protege name: task
```

```
    */
```

```
    private Tasks task;
```

```
    public void setTask(Tasks value) {
```

```
        this.task=value;
```

```
    }
```

```
    public Tasks getTask() {
```

```
        return this.task;
```

```
    }
```

```
}
```

BehaviourResults.java:

```
import jade.content.*;
```

```
import jade.util.leap.*;
```

```
import jade.core.*;
```

```
/**
```

```
 * Protege name: BehaviourResults
```

```
 * @author ontology bean generator
```

```
* @version 2005/04/6, 20:45:13
```

```
*/
```

```
public class BehaviourResults implements Predicate {  
    /**  
    * Protege name: information performance  
    */  
    private Agent_Performance information_performance;  
    public void setInformation_performance(Agent_Performance value) {  
        this.information_performance=value; }  
    public Agent_Performance getInformation_performance() {  
        return this.information_performance; }  
}
```

AgentCharacteristics.java:

```
import jade.content.*;
```

```
import jade.util.leap.*;
```

```
import jade.core.*;
```

```
/**
```

```
    * Protege name: AgentCharacteristics
```

```
    * @author ontology bean generator
```

```
    * @version 2005/04/12, 21:32:14
```

```
*/
```

```
public class AgentCharacteristics implements Predicate {  
    /**  
    * Protege name: stress  
    */  
    private String stress;  
    public void setStress(String value) {  
        this.stress=value;  
    }  
    public String getStress() {
```

```

        return this.stress;
    }

/**
 * Protege name: name_agent
 */
    private String name_agent;
    public void setName_agent(String value) {
        this.name_agent=value;
    }
    public String getName_agent() {
        return this.name_agent;
    }

/**
 * Protege name: cognitioncharacteristics
 */
    private Cognition cognitioncharacteristics;
    public void setCognitioncharacteristics(Cognition value) {
        this.cognitioncharacteristics=value;
    }
    public Cognition getCognitioncharacteristics() {
        return this.cognitioncharacteristics;
    }

/**
 * Protege name: socialcharacteristics
 */
    private Social socialcharacteristics;
    public void setSocialcharacteristics(Social value) {
        this.socialcharacteristics=value;
    }
    public Social getSocialcharacteristics() {
        return this.socialcharacteristics;
    }

```

```

/**
 * Protege name: personalitycharacteristics
 */
private Personality personalitycharacteristics;
public void setPersonalitycharacteristics(Personality value) {
    this.personalitycharacteristics=value;
}
public Personality getPersonalitycharacteristics() {
    return this.personalitycharacteristics;
}
}

```

Agent_Properties.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

/**
 * Protege name: Agent Properties
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:13
 */
public class Agent_Properties implements Concept {

    /**
     * Protege name: name_agent
     */
    private String name_agent;
    public void setName_agent(String value) {
        this.name_agent=value; }
    public String getName_agent() {
        return this.name_agent; }

}

```

```

    * Protege name: stress
    */
    private String stress;
    public void setStress(String value) {
        this.stress=value;
    }
    public String getStress() {
        return this.stress;
    }
}

```

Cognition.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

/**
 * Protege name: Cognition
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:13
 */
public class Cognition extends Agent_Properties{

    /**
     * Protege name: creativity
     */
    private String creativity;
    public void setCreativity(String value) {
        this.creativity=value; }
    public String getCreativity() {
        return this.creativity; }

    /**
     * Protege name: experience

```

```

*/
private String experience;
public void setExperience(String value) {
    this.experience=value; }
public String getExperience() {
    return this.experience; }

/**
 * Protege name: role
 */
private String role;
public void setRole(String value) {
    this.role=value;
}
public String getRole() {
    return this.role;
}
}

```

Personality.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

/**
 * Protege name: Personality
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:13
 */
public class Personality extends Agent_Properties{
    /**
     * Protege name: amiable
     */
    private String amiable;

```

```

public void setAmiable(String value) {
    this.amiable=value;
}

public String getAmiable() {
    return this.amiable;
}

/**
 * Protege name: analytical
 */
private String analytical;
public void setAnalytical(String value) {
    this.analytical=value;
}
public String getAnalytical() {
    return this.analytical;
}

/**
 * Protege name: driver
 */
private String driver;
public void setDriver(String value) {
    this.driver=value;
}
public String getDriver() {
    return this.driver;
}

/**
 * Protege name: expressive
 */
private String expressive;
public void setExpressive(String value) {

```

```

        this.expressive=value;
    }
    public String getExpressive() {
        return this.expressive;
    }
}

```

Social.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

/**
 * Protege name: Social
 * @author ontology bean generator
 * @version 2005/05/21, 19:28:46
 */
public class Social extends Agent_Properties{

    /**
     * Protege name: work_alone
     */
    private String work_alone;
    public void setWork_alone(String value) {
        this.work_alone=value;
    }
    public String getWork_alone() {
        return this.work_alone;
    }

    /**
     * Protege name: introverted
     */
    private String introverted;

```

```

public void setIntroverted(String value) {
    this.introverted=value;
}
public String getIntroverted() {
    return this.introverted;
}

/**
 * Protege name: work_team
 */
private String work_team;
public void setWork_team(String value) {
    this.work_team=value;
}
public String getWork_team() {
    return this.work_team;
}

/**
 * Protege name: extroverted
 */
private String extroverted;
public void setExtroverted(String value) {
    this.extroverted=value;
}
public String getExtroverted() {
    return this.extroverted;
}
}

```

Emotion.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

```

```
/**
 * Protege name: Emotion
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:13
 */
public class Emotion extends Agent_Properties{
```

```
/**
 * Protege name: disgust
 */
private String disgust;
public void setDisgust(String value) {
    this.disgust=value; }
public String getDisgust() {
    return this.disgust; }
```

```
/**
 * Protege name: interest
 */
private String interest;
public void setInterest(String value) {
    this.interest=value; }
public String getInterest() {
    return this.interest; }
```

```
/**
 * Protege name: anxiety
 */
private String anxiety;
public void setAnxiety(String value) {
    this.anxiety=value; }
public String getAnxiety() {
    return this.anxiety; }
```

```

/**
 * Protege name: desire
 */
private String desire;
public void setDesire(String value) {
    this.desire=value; }
public String getDesire() {
    return this.desire; }
}

```

Tasks.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;

```

```

/**
 * Protege name: Tasks
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:13
 */
public class Tasks implements Concept {

```

```

/**
 * Protege name: type
 */
private String type;
public void setType(String value) {
    this.type=value; }
public String getType() {
    return this.type; }

```

```

/**
 * Protege name: deadline

```

```
*/  
private int deadline;  
public void setDeadline(int value) {  
    this.deadline=value; }  
public int getDeadline() {  
    return this.deadline; }
```

```
/**  
 * Protege name: id  
 */
```

```
private int id;  
public void setId(int value) {  
    this.id=value; }  
public int getId() {  
    return this.id; }
```

```
/**  
 * Protege name: cost  
 */
```

```
private int cost;  
public void setCost(int value) {  
    this.cost=value; }  
public int getCost() {  
    return this.cost; }
```

```
/**  
 * Protege name: duration  
 */
```

```
private int duration;  
public void setDuration(int value) {  
    this.duration=value; }  
public int getDuration() {  
    return this.duration; }
```

```

/**
 * Protege name: quality
 */
private String quality;
public void setQuality(String value) {
    this.quality=value;
}
public String getQuality() {
    return this.quality;
}

/**
 * Protege name: priority
 */
private String priority;
public void setPriority(String value) {
    this.priority=value; }
public String getPriority() {
    return this.priority; }

/**
 * Protege name: num participants
 */
private int num_participants;
public void setNum_participants(int value) {
    this.num_participants=value; }
public int getNum_participants() {
    return this.num_participants;
}

/**
 * Protege name: difficulty
 */
private String difficulty;

```

```

public void setDifficulty(String value) {
    this.difficulty=value; }
public String getDifficulty() {
    return this.difficulty; }
}

```

Agent_Performance.java:

```

import jade.content.*;
import jade.util.leap.*;
import jade.core.*;
import java.util.*;

/**
 * Protege name: Agent Performance
 * @author ontology bean generator
 * @version 2005/04/6, 20:45:13
 */
public class Agent_Performance implements Concept {
    /**
     * Protege name: task duration
     */
    private String task_duration;
    public void setTask_duration(String value) {
        this.task_duration=value; }
    public String getTask_duration() {
        return this.task_duration; }

    /**
     * Protege name: individual level
     */
    private String individual_level;
    public void setIndividual_level(String value) {
        this.individual_level=value; }
    public String getIndividual_level() {

```

```

        return this.individual_level; }

/**
 * Protege name: goals achievement
 */
private String goals_achievement;
public void setGoals_achievement(String value) {
    this.goals_achievement=value; }
public String getGoals_achievement() {
    return this.goals_achievement; }

/**
 * Protege name: supervision level
 */
private String supervision_level;
public void setSupervision_level(String value) {
    this.supervision_level=value; }
public String getSupervision_level() {
    return this.supervision_level; }

/**
 * Protege name: team level
 */
private String team_level;
public void setTeam_level(String value) {
    this.team_level=value; }
public String getTeam_level() {
    return this.team_level; }

/**
 * Protege name: task quality
 */
private String task_quality;
public void setTask_quality(String value) {

```

```

        this.task_quality=value;  }
    public String getTask_quality() {
        return this.task_quality;  }

    /**
     * Protege name: id task
     */
    private int id_task;
    public void setId_task(int value) {
        this.id_task=value;  }
    public int getId_task() {
        return this.id_task;  }

    /**
     * Protege name: agent
     */
    private String agent;
    public void setAgent(String value) {
        this.agent=value;  }
    public String getAgent() {
        return this.agent;  }
}

```

ConfigurationBehaviourRule.java:

```

package interfaz;
import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.border.*;
import java.util.*;

    /**
     * Title:      Interfaz Gráfica

```

* Description: Panel de configuración
* Copyright: Copyright (c) 2005
* Company:
* @author Laura Férrez
* @version 1.0
*/

```
public class ConfigurationBehaviourRule extends JFrame{  
    JLabel jLabel1 = new JLabel();  
    JLabel jLabel2 = new JLabel();  
    JComboBox jComboBox1;  
    JComboBox jComboBox2;  
    JComboBox jComboBox3;  
    JComboBox jComboBox4;  
    JComboBox jComboBox6;  
    JComboBox jComboBox7;  
    JComboBox jComboBox8;  
    JComboBox jComboBox9;  
    JComboBox jComboBox10;  
    JComboBox jComboBox11;  
    JComboBox jComboBox12;  
    JComboBox jComboBox13;  
    JComboBox jComboBox14;  
    JComboBox jComboBox15;  
    JComboBox jComboBox16;  
    JComboBox jComboBox17;  
    JLabel jLabel3 = new JLabel();  
    JLabel jLabel4 = new JLabel();  
    JLabel jLabel5 = new JLabel();  
    JLabel jLabel6 = new JLabel();  
    JLabel jLabel7 = new JLabel();  
    JLabel jLabel8 = new JLabel();  
    JLabel jLabel9 = new JLabel();  
    JLabel jLabel10 = new JLabel();  
}
```

```

JLabel jLabel11 = new JLabel();
JLabel jLabel12 = new JLabel();
JLabel jLabel13 = new JLabel();
JLabel jLabel14 = new JLabel();
JLabel jLabel15 = new JLabel();
JLabel jLabel16 = new JLabel();
JLabel jLabel17 = new JLabel();
JLabel jLabel18 = new JLabel();
JPanel jPanel1 = new JPanel();
TitledBorder titledBorder1;
JButton jButton1 = new JButton();
JButton jButton2 = new JButton();
String cad_user=""; // cadena para visualizar las reglas generadas
String name_rule=""; // cadena que guarda el nombre de la regla de una configuración
String[] Typelist1;
String[] Typelist3;
String[] Typelist4;
String[] Typelist2;
String names_rules=""; // cadena que guarda todos los nombres de las reglas
configuradas
BufferedWriter file;
boolean modified=false;

JButton jButton3 = new JButton();
TitledBorder titledBorder2;
Border border1;
Border border2;
Border border3;
Border border4;
Border border5;
Border border6;
Border border7;
TitledBorder titledBorder3;
Border border8;

```

```

Border border9;
Border border10;
menu menu1;

public ConfigurationBehaviourRule(menu menu1) {
    try {
        this.menu1=menu1;
        jbInit();

    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

//Inicializacion de todos los campos
void init()
{
    jComboBox1.setSelectedItem(" ");
    jComboBox6.setSelectedItem(" ");
    jComboBox7.setSelectedItem(" ");
    jComboBox8.setSelectedItem(" ");
    jComboBox9.setSelectedItem(" ");
    jComboBox2.setSelectedItem(" ");
    jComboBox3.setSelectedItem(" ");
    jComboBox4.setSelectedItem(" ");
    jComboBox10.setSelectedItem(" ");
    jComboBox11.setSelectedItem(" ");
    jComboBox12.setSelectedItem(" ");
    jComboBox13.setSelectedItem(" ");
    jComboBox14.setSelectedItem(" ");
    jComboBox15.setSelectedItem(" ");
    jComboBox16.setSelectedItem(" ");
    jComboBox17.setSelectedItem(" ");
}

```

```

}

//Creación de la interfaz
void jbInit() throws Exception {
    titledBorder1 = new
TitledBorder(BorderFactory.createEtchedBorder(Color.white,new Color(156, 156,
158)), "Configuration");
    titledBorder2 = new TitledBorder("");
    border1 = BorderFactory.createEmptyBorder();
    border2 = new MatteBorder(null);
    border3 = BorderFactory.createCompoundBorder(new
EtchedBorder(EtchedBorder.RAISED,Color.white,new Color(178, 178, 178)),border2);
    border4 = BorderFactory.createEmptyBorder();
    border5 = BorderFactory.createEmptyBorder();
    border6 = BorderFactory.createEmptyBorder();
    border7 =
BorderFactory.createCompoundBorder(BorderFactory.createEmptyBorder(),border3);
    titledBorder3 = new TitledBorder("");
    border8 = BorderFactory.createEmptyBorder();
    border9 = BorderFactory.createEmptyBorder();
    border10 = BorderFactory.createEmptyBorder();
    this.getContentPane().setLayout(null);
    this.setTitle("Behaviour Rules");
    this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    this.addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosed(WindowEvent e) {
            if(modified)
                endInformationRules();
            menu1.show();
        }
    });
}

```

//modificar ruta

```
copy("D:/JBuilder4/Interfaz/DefinitionsBehaviourRules.txt","D:/JBuilder4/Interfaz/BehaviourRules.txt");
```

```
jLabel1.setFont(new java.awt.Font("Dialog", 1, 12));  
jLabel1.setToolTipText("");  
jLabel1.setText("Antecedents");  
jLabel1.setBounds(new Rectangle(175, 10, 89, 40));  
jLabel2.setFont(new java.awt.Font("Dialog", 1, 12));  
jLabel2.setText("Consequences");  
jLabel2.setBounds(new Rectangle(436, 10, 89, 40));
```

```
Typelist1 = new String[4];  
Typelist1[0] = new String("Low");  
Typelist1[1] = new String("Medium");  
Typelist1[2] = new String("High");  
Typelist1[3] = new String(" ");
```

```
jLabel3.setText("Experience");  
jLabel3.setVerticalAlignment(SwingConstants.TOP);  
jLabel3.setVerticalTextPosition(SwingConstants.TOP);  
jLabel3.setBounds(new Rectangle(50, 46, 66, 20));
```

```
jComboBox2= new JComboBox(Typelist1);  
jComboBox2.setBounds(new Rectangle(141, 44, 143, 21));  
jComboBox2.setSelectedItem(" ");  
jComboBox2.setName("experience");  
jComboBox2.setBorder(border1);
```

```
jLabel4.setVerticalTextPosition(SwingConstants.TOP);  
jLabel4.setVerticalAlignment(SwingConstants.TOP);  
jLabel4.setText("Drive");  
jLabel4.setBounds(new Rectangle(50, 68, 66, 16));
```

```
jComboBox3= new JComboBox(Typelist1);
jComboBox3.setBounds(new Rectangle(141, 66, 143, 21));
jComboBox3.setSelectedItem(" ");
jComboBox3.setName("drive");
jComboBox3.setBorder(border1);
```

```
jLabel5.setText("Influence");
jLabel5.setVerticalAlignment(SwingConstants.TOP);
jLabel5.setVerticalTextPosition(SwingConstants.TOP);
jLabel5.setBounds(new Rectangle(50, 91, 89, 22));
```

```
jComboBox4= new JComboBox(Typelist1);
jComboBox4.setBounds(new Rectangle(141, 88, 143, 21));
jComboBox4.setSelectedItem(" ");
jComboBox4.setName("influence");
jComboBox4.setBorder(border1);
```

```
jLabel6.setVerticalTextPosition(SwingConstants.TOP);
jLabel6.setVerticalAlignment(SwingConstants.TOP);
jLabel6.setText("Stainess");
jLabel6.setBounds(new Rectangle(50, 114, 89, 19));
```

```
jComboBox13= new JComboBox(Typelist1);
jComboBox13.setBounds(new Rectangle(141, 110, 143, 21));
jComboBox13.setSelectedItem(" ");
jComboBox13.setName("stainess");
jComboBox13.setBorder(border1);
```

```
jLabel7.setText("Compliance");
jLabel7.setVerticalAlignment(SwingConstants.TOP);
jLabel7.setVerticalTextPosition(SwingConstants.TOP);
jLabel7.setBounds(new Rectangle(50, 136, 89, 20));
```

```
jComboBox10= new JComboBox(Typelist1);
```

```
jComboBox10.setBounds(new Rectangle(141, 132, 143, 21));
jComboBox10.setSelectedItem(" ");
jComboBox10.setName("compliance");
jComboBox10.setBorder(border1);
jLabel8.setVerticalTextPosition(SwingConstants.TOP);
jLabel8.setVerticalAlignment(SwingConstants.TOP);
jLabel8.setText("Alone Team");
jLabel8.setBounds(new Rectangle(50, 160, 89, 22));
```

```
jComboBox11= new JComboBox(Typelist1);
jComboBox11.setBounds(new Rectangle(141, 154, 143, 21));
jComboBox11.setSelectedItem(" ");
jComboBox11.setName("aloneteam");
jComboBox11.setBorder(border1);
```

```
jLabel9.setText("Introverted");
jLabel9.setVerticalAlignment(SwingConstants.TOP);
jLabel9.setVerticalTextPosition(SwingConstants.TOP);
jLabel9.setBounds(new Rectangle(50, 181, 89, 22));
```

```
jComboBox12= new JComboBox(Typelist1);
jComboBox12.setBounds(new Rectangle(141, 176, 143, 21));
jComboBox12.setSelectedItem(" ");
jComboBox12.setName("introextro");
jComboBox12.setBorder(border1);
```

```
Typelist2 = new String[6];
Typelist2[0] = new String("HighAdvance");
Typelist2[1] = new String("LowAdvance");
Typelist2[2] = new String("Normal");
Typelist2[3] = new String("LowDelay");
Typelist2[4] = new String("HighDelay");
Typelist2[5] = new String(" ");
```

```

jLabel10.setText("Duration");
jLabel10.setVerticalAlignment(SwingConstants.TOP);
jLabel10.setVerticalTextPosition(SwingConstants.TOP);
jLabel10.setBounds(new Rectangle(50, 202, 89, 22));

jComboBox14= new JComboBox(Typelist2);
jComboBox14.setBounds(new Rectangle(141, 198, 143, 21));
jComboBox14.setSelectedItem(" ");
jComboBox14.setName("advance");
jComboBox14.setBorder(border1);

jLabel11.setText("Task type"); //nivel de especialización requerido
jLabel11.setVerticalAlignment(SwingConstants.TOP);
jLabel11.setVerticalTextPosition(SwingConstants.TOP);
jLabel11.setBounds(new Rectangle(50, 224, 89, 22));

jComboBox15= new JComboBox(Typelist1);
jComboBox15.setBounds(new Rectangle(141, 220, 143, 21));
jComboBox15.setSelectedItem(" ");
jComboBox15.setName("typetask");
jComboBox15.setBorder(border1);

jLabel12.setText("Task difficult");
jLabel12.setVerticalAlignment(SwingConstants.TOP);
jLabel12.setVerticalTextPosition(SwingConstants.TOP);
jLabel12.setBounds(new Rectangle(50, 245, 89, 24));

jComboBox16= new JComboBox(Typelist1);
jComboBox16.setBounds(new Rectangle(141, 242, 143, 21));
jComboBox16.setSelectedItem(" ");
jComboBox16.setName("kindoftask");
jComboBox16.setBorder(border1); //nivel de dificultad

Typelist3 = new String[6];

```

```
Typelist3[0] = new String("HighDown");
Typelist3[1] = new String("Down");
Typelist3[2] = new String("Equal");
Typelist3[3] = new String("Up");
Typelist3[4] = new String("Highup");
Typelist3[5] = new String("    ");
```

```
jLabel13.setVerticalTextPosition(SwingConstants.TOP);
jLabel13.setVerticalAlignment(SwingConstants.TOP);
jLabel13.setText("Desire");
jLabel13.setBounds(new Rectangle(333, 45, 89, 22));
```

```
jComboBox1= new JComboBox(Typelist3);
jComboBox1.setBounds(new Rectangle(423, 43, 114, 22));
jComboBox1.setSelectedItem("    ");
jComboBox1.setName("desire");
jComboBox1.setBorder(border1);
```

```
jLabel14.setText("Interest");
jLabel14.setVerticalAlignment(SwingConstants.TOP);
jLabel14.setVerticalTextPosition(SwingConstants.TOP);
jLabel14.setBounds(new Rectangle(333, 68, 89, 19));
```

```
jComboBox6= new JComboBox(Typelist3);
jComboBox6.setBounds(new Rectangle(423, 66, 114, 22));
jComboBox6.setSelectedItem("    ");
jComboBox6.setName("interest");
jComboBox6.setBorder(border1);
```

```
jLabel15.setText("Disgust");
jLabel15.setVerticalAlignment(SwingConstants.TOP);
jLabel15.setVerticalTextPosition(SwingConstants.TOP);
jLabel15.setBounds(new Rectangle(333, 92, 89, 22));
```

```
jComboBox7= new JComboBox(Typelist3);
jComboBox7.setBounds(new Rectangle(423, 89, 114, 22));
jComboBox7.setSelectedItem(" ");
jComboBox7.setName("disgust");
jComboBox7.setBorder(border1);
```

```
jLabel16.setText("Anxiety");
jLabel16.setVerticalAlignment(SwingConstants.TOP);
jLabel16.setVerticalTextPosition(SwingConstants.TOP);
jLabel16.setBounds(new Rectangle(333, 112, 54, 22));
```

```
jComboBox8= new JComboBox(Typelist3);
jComboBox8.setBounds(new Rectangle(423, 111, 114, 22));
jComboBox8.setSelectedItem(" ");
jComboBox8.setName("anxiety");
jComboBox8.setBorder(border1);
```

```
jLabel17.setBounds(new Rectangle(333, 136, 89, 22));
jLabel17.setVerticalTextPosition(SwingConstants.TOP);
jLabel17.setVerticalAlignment(SwingConstants.TOP);
jLabel17.setText("Stress");
```

```
jComboBox9= new JComboBox(Typelist3);
jComboBox9.setBounds(new Rectangle(423, 133, 114, 21));
jComboBox9.setSelectedItem(" ");
jComboBox9.setName("stress");
jComboBox9.setBorder(border1);
```

```
Typelist4 = new String[6];
Typelist4[0] = new String("PROJECT MANAGER");
Typelist4[1] = new String("COORDINATOR");
Typelist4[2] = new String("SPECIALIST Sr.");
Typelist4[3] = new String("TECHNICIAN");
Typelist4[4] = new String("ASSISTANT");
```

```

Typelist4[5] = new String("    ");

jLabel18.setBounds(new Rectangle(51, 266, 89, 24));
jLabel18.setVerticalTextPosition(SwingConstants.TOP);
jLabel18.setVerticalAlignment(SwingConstants.TOP);
jLabel18.setText("Role");

jComboBox17= new JComboBox(Typelist4);
jComboBox17.setBounds(new Rectangle(141, 264, 143, 21));
jComboBox17.setSelectedItem("    ");
jComboBox17.setName("typeagent");
jComboBox17.setBorder(border1);

jPanel1.setBorder(titledBorder1);
jPanel1.setBounds(new Rectangle(16, 3, 537, 297));
jButton1.setBorder(BorderFactory.createEtchedBorder());
jButton1.setText("Add");
jButton1.addActionListener(new java.awt.event.ActionListener(){
    public void actionPerformed(ActionEvent e){
        add_actionPerformed(e);
    }
});
jButton1.setBounds(new Rectangle(61, 309, 72, 22));
jButton2.setBorder(BorderFactory.createEtchedBorder());
jButton2.setText("View");
jButton2.addActionListener(new java.awt.event.ActionListener(){
    public void actionPerformed(ActionEvent e){
        view_actionPerformed(e);
    }
});
jButton2.setBounds(new Rectangle(414, 308, 80, 22));
jButton3.setBounds(new Rectangle(242, 309, 72, 22));
jButton3.addActionListener(new java.awt.event.ActionListener(){
    public void actionPerformed(ActionEvent e){

```

```

        close_actionPerformed(e);
    }
});
jButton3.setText("Close");
jButton3.setBorder(BorderFactory.createEtchedBorder());
this.getContentPane().add(jComboBox1);
this.getContentPane().add(jComboBox6);
this.getContentPane().add(jComboBox7);
this.getContentPane().add(jComboBox8);
this.getContentPane().add(jComboBox9);
this.getContentPane().add(jLabel4);
this.getContentPane().add(jLabel3);
this.getContentPane().add(jLabel5);
this.getContentPane().add(jLabel6);
this.getContentPane().add(jLabel7);
this.getContentPane().add(jLabel9);
this.getContentPane().add(jLabel8);
this.getContentPane().add(jLabel10);
this.getContentPane().add(jLabel14);
this.getContentPane().add(jLabel15);
this.getContentPane().add(jLabel17);
this.getContentPane().add(jLabel1);
this.getContentPane().add(jComboBox13);
this.getContentPane().add(jComboBox3);
this.getContentPane().add(jComboBox4);
this.getContentPane().add(jComboBox2);
this.getContentPane().add(jComboBox10);
this.getContentPane().add(jComboBox11);
this.getContentPane().add(jComboBox12);
this.getContentPane().add(jComboBox14);
this.getContentPane().add(jComboBox15);
this.getContentPane().add(jComboBox16);
this.getContentPane().add(jLabel11);
this.getContentPane().add(jLabel12);

```

```

this.getContentPane().add(jLabel2);
this.getContentPane().add(jComboBox17);
this.getContentPane().add(jLabel18);
this.getContentPane().add(jLabel16);
this.getContentPane().add(jLabel13);
this.getContentPane().add(jPanel1);
this.getContentPane().add(jButton2);
this.getContentPane().add(jButton3);
this.getContentPane().add(jButton1);
this.setResizable(false);
this.setSize(575,380);
this.setLocation(350,200);
this.setVisible(true);
}

```

//Definiciones guardadas al final del fichero cuando se han acabado de añadir todas las reglas

```

void endInformationRules ()
{
try{
file=new BufferedWriter (new FileWriter("BehaviourRules.txt",true));
file.newLine();
file.newLine();
file.write("(defrule defuzzify" + ""+"low salience to allow all rules to fire and do global
contribution"+"");
file.newLine();
file.write(" (declare (salience -100))");
file.newLine();
file.write(" (change_desire ?chdes)");
file.newLine();
file.write(" (change_interest ?chint)");
file.newLine();
file.write(" (change_disgust ?chdisg)");
file.newLine();

```

```

file.write(" (change_anxiety ?chanx)");
file.newLine();
file.write(" (change_stress ?chstrs)");
file.newLine();
file.write(" =>");
file.newLine();
file.write(" (bind ?desire-to-add (?chdes momentDefuzzify))");
file.newLine();
file.write(" (bind ?interest-to-add (?chint momentDefuzzify))");
file.newLine();
file.write(" (bind ?disgust-to-add (?chdisg momentDefuzzify))");
file.newLine();
file.write(" (bind ?anxiety-to-add (?chanx momentDefuzzify))");
file.newLine();
file.write(" (bind ?stress-to-add (?chstrs momentDefuzzify))");
file.newLine();
file.write(" (store SUMDESIRE ?desire-to-add)");
file.newLine();
file.write(" (store SUMINTEREST ?interest-to-add)");
file.newLine();
file.write(" (store SUMDISGUST ?disgust-to-add)");
file.newLine();
file.write(" (store SUMANXIETY ?anxiety-to-add)");
file.newLine();
file.write(" (store SUMSTRESS ?stress-to-add)");
file.newLine();
file.write(" (store RULESTHATFIRED ?*rulesThatFired*)");
file.newLine();
file.write("");
file.close();
this.hide();
} catch (IOException ex){
    System.out.println("Error");
}

```

```

}

//Cerrar la ventana de configuración
void close_actionPerformed(ActionEvent e){

    if (modified==false) //si no se ha añadido ninguna regla al fichero
        this.hide();
    else endInformationRules(); //guardar definiciones al final del fichero
    menu1.show();
}

//Visualizar las reglas generadas
//cad_user contiene las reglas generadas
//names_rules contiene el nombre de las reglas generadas
//Separadas por un asterisco para separar las reglas
void view_actionPerformed(ActionEvent e) {
    panel p= new panel();
    StringTokenizer st = new StringTokenizer(cad_user,"*");
    StringTokenizer st2 = new StringTokenizer(names_rules,"*");
    while (st2.hasMoreTokens()&&st.hasMoreTokens()) {
        p.jTextArea1.append(" RULE "+st2.nextToken()+" :
\n"+st.nextToken()+"\n"+"
\n");
    }
}

//Crea un fichero nuevo y copia el contenido del fichero de definiciones en este nuevo
void copy(String fuente, String destino) throws IOException {
    FileInputStream fis=new FileInputStream(fuente);
    FileOutputStream fos=new FileOutputStream(new File(destino));
    int c;
    while((c=fis.read())!=-1) // -1 indica fin
        fos.write(c);
    fis.close();
    fos.close();
}

```

```
}
```

//Comprobación: que se hayan seleccionado alguno de los campos de la lista de desplegables en el caso de antecedentes

```
boolean selectedAntecedents(){
```

```
    boolean selected=false;
```

```
    if(selectedComboBoxAntec(jComboBox2)||selectedComboBoxAntec(jComboBox3)||selectedComboBoxAntec(jComboBox4)||selectedComboBoxAntec(jComboBox13)||selectedComboBoxAntec(jComboBox10)||selectedComboBoxAntec(jComboBox11)||selectedComboBoxAntec(jComboBox12)||selectedComboBoxAntec(jComboBox14)||selectedComboBoxAntec(jComboBox15)||selectedComboBoxAntec(jComboBox16)||selectedComboBoxAntec(jComboBox17))
```

```
        selected=true;
```

```
    return selected;
```

```
}
```

//Comprobación: que se hayan seleccionado todos los campos de la lista de desplegables en el caso de las consecuencias

```
boolean selectedConsequences(){
```

```
    boolean selected=false;
```

```
    if(selectedComboBoxCons(jComboBox1)&&selectedComboBoxCons(jComboBox6)&&selectedComboBoxCons(jComboBox7)&&selectedComboBoxCons(jComboBox8)&&selectedComboBoxCons(jComboBox9))
```

```
        selected=true;
```

```
    return selected;
```

```
}
```

//Comprobación: que se haya seleccionada alguno de los elementos de la lista menos el campo en blanco

//la lista es para el campo advance

```
boolean toCheckLista2(JComboBox combo){
```

```
    boolean selected=false;
```

```

for(int i=0;i<Typelist2.length-1;i++){
    if (combo.getSelectedItem().equals(Typelist2[i])){
        selected=true;
    }
}
return selected;
}

```

//Comprobación: que se haya seleccionada un elemento de la lista menos el campo en blanco

//la lista es para el estado interno del agente y el tipo y dificultad de la tarea del apartado de antecedentes menos el advance

```

boolean toCheckLista1(JComboBox combo){
    boolean selected=false;
    for(int i=0;i<Typelist1.length-1;i++){
        if (combo.getSelectedItem().equals(Typelist1[i])){
            selected=true;
        }
    }
    return selected;
}

```

//Comprobación: que se haya seleccionada un elemento de la lista menos el campo en blanco

//lista de tipos de rol

```

boolean toCheckLista4(JComboBox combo){
    boolean selected=false;
    for(int i=0;i<Typelist4.length-1;i++){
        if (combo.getSelectedItem().equals(Typelist4[i])){
            selected=true;
        }
    }
    return selected;
}

```

//Comprobación: tratar los diferentes campos según el tipo de lista al que pertenecen en el apartado de antecedentes

```
boolean selectedComboBoxAntec(JComboBox combo)
{
    boolean selected=false;

    if (combo.getName().compareTo("advance")==0)
        selected=toCheckLista2(combo);        //método para la lista2
    else if (combo.getName().compareTo("typeagent")==0)
        selected=toCheckLista4(combo);        //método para la lista4
    else
        selected=toCheckLista1(combo); //método para la lista1
    return selected;
}
```

//Comprobación: que se haya seleccionado un campo de la lista desplegable en el apartado de las consecuencias

```
boolean selectedComboBoxCons(JComboBox combo)
{
    boolean selected=false;
    for(int i=0;i<Typelist3.length-1;i++){
        if (combo.getSelectedItem().equals(Typelist3[i])){
            selected=true;
        }
    }
    return selected;
}
```

//Añadir las reglas generadas al fichero

//Hacer comprobaciones conforme se han seleccionado los campos de sus listas

//Escribir en el fichero las reglas

```
void add_actionPerformed(ActionEvent e) {
    try{
```

```

file=new BufferedWriter (new FileWriter("BehaviourRules.txt",true));

int c=0;
String cad_drive="";
String cad_influence="";
String cad_stainess="";
String cad_compliance="";
String cad_aloneteam="";
String cad_intro="";
String cad_duration="";
String cad_task="";
String cad_task_dif="";
String cad_role="";
String cad_experience="";
String aux="";

if (selectedComboBoxAntec(jComboBox2)&& selectedConsequences())
{
    c++;
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox2.getSelectedItem()+"_"+jComboBox2.getNa
me();
        names_rules=names_rules+"
"+jComboBox2.getSelectedItem()+"_"+jComboBox2.getName();
    }
    else
    {
        name_rule=jComboBox2.getSelectedItem()+"_"+jComboBox2.getName();
        names_rules=names_rules+name_rule;
    }
}

```

```

        cad_user=cad_user+" "+jLabel3.getText().toUpperCase()+":
"+jComboBox2.getSelectedItemAt()+"\n';
        cad_experience=" (" + jComboBox2.getName()+" ?ex&:(fuzzy-match ?ex
"+""+jComboBox2.getSelectedItemAt()+""+""))";
    }
    if (selectedComboBoxAntec(jComboBox3)&& selectedConsequences())
    {
        c++;
        if (name_rule!="")
        {

name_rule=name_rule+"_"+jComboBox3.getSelectedItemAt()+"_"+jComboBox3.getNa
me();
        names_rules=names_rules+"
"+jComboBox3.getSelectedItemAt()+"_"+jComboBox3.getName();
        }
    else
    {
        name_rule=jComboBox3.getSelectedItemAt()+"_"+jComboBox3.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+" "+jLabel4.getText().toUpperCase()+":
"+jComboBox3.getSelectedItemAt()+"\n';
    cad_drive=" (" +jComboBox3.getName()+" ?dr&:(fuzzy-match ?dr
"+""+jComboBox3.getSelectedItemAt()+""+""))";
    }
    if(selectedComboBoxAntec(jComboBox4)&& selectedConsequences())
    {
        if (name_rule!="")
        {

name_rule=name_rule+"_"+jComboBox4.getSelectedItemAt()+"_"+jComboBox4.getNa
me();

```

```

        names_rules=names_rules+"
"+jComboBox4.getSelectedItemAt()+"_"+jComboBox4.getName();
    }
    else
    {
        name_rule=jComboBox4.getSelectedItemAt()+"_"+jComboBox4.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel5.getText().toUpperCase()+":
"+jComboBox4.getSelectedItemAt()+"\n';
    cad_influence=" (" +jComboBox4.getName()+" ?inf&:(fuzzy-match ?inf
"+""+jComboBox4.getSelectedItemAt()+""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox13)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox13.getSelectedItemAt()+"_"+jComboBox13.getN
ame();
        names_rules=names_rules+"
"+jComboBox13.getSelectedItemAt()+"_"+jComboBox13.getName();
    }
    else
    {
        name_rule=jComboBox13.getSelectedItemAt()+"_"+jComboBox13.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel6.getText().toUpperCase()+":
"+jComboBox13.getSelectedItemAt()+"\n';
    cad_stainess=" (" +jComboBox13.getName()+" ?stss&:(fuzzy-match ?stss
"+""+jComboBox13.getSelectedItemAt()+""+"))";
    c++;

```

```

}
if(selectedComboBoxAntec(jComboBox10)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox10.getSelectedItem()+"_"+jComboBox10.getN
ame();
        names_rules=names_rules+"
"+jComboBox10.getSelectedItem()+"_"+jComboBox10.getName();
    }
    else
    {
        name_rule=jComboBox10.getSelectedItem()+"_"+jComboBox10.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel7.getText().toUpperCase()+":
"+jComboBox10.getSelectedItem()+"\n";
    cad_compliance=" (" +jComboBox10.getName()+" ?cmpl&:(fuzzy-match ?cmpl
"+""+jComboBox10.getSelectedItem()+""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox11)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox11.getSelectedItem()+"_"+jComboBox11.getN
ame();
        names_rules=names_rules+"
"+jComboBox11.getSelectedItem()+"_"+jComboBox11.getName();
    }
    else
    {

```

```

        name_rule=jComboBox11.getSelectedItemAt()+"_"+jComboBox11.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel8.getText().toUpperCase()+":
"+jComboBox11.getSelectedItemAt()+"\n";
    cad_aloneteam=" (" +jComboBox11.getName()+" ?at&:(fuzzy-match ?at
"+""+jComboBox11.getSelectedItemAt()+""+""));
    c++;
}
if(selectedComboBoxAntec(jComboBox12)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox12.getSelectedItemAt()+"_"+jComboBox12.getN
ame();
        names_rules=names_rules+"
"+jComboBox12.getSelectedItemAt()+"_"+jComboBox12.getName();
    }
    else
    {
        name_rule=jComboBox12.getSelectedItemAt()+"_"+jComboBox12.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel9.getText().toUpperCase()+":
"+jComboBox12.getSelectedItemAt()+"\n";
    cad_intro=" (" +jComboBox12.getName()+" ?intext&:(fuzzy-match ?intext
"+""+jComboBox12.getSelectedItemAt()+""+""));
    c++;
}
if(selectedComboBoxAntec(jComboBox14)&& selectedConsequences())
{
    if (name_rule!="")
    {

```

```

name_rule=name_rule+"_"+jComboBox14.getSelectedItem()+"_"+duration";
names_rules=names_rules+" "+jComboBox14.getSelectedItem()+"_"+duration";
}
else
{
name_rule=jComboBox14.getSelectedItem()+"_"+duration";
names_rules=names_rules+name_rule;
}
cad_user=cad_user+" "+jLabel10.getText().toUpperCase()+":
"+jComboBox14.getSelectedItem()+"\n";
cad_duration=" (" +jComboBox14.getName()+" ?a&:(fuzzy-match ?a
"+""+jComboBox14.getSelectedItem()+"")");
c++;
}
if(selectedComboBoxAntec(jComboBox15)&& selectedConsequences())
{
if (name_rule!="")
{

name_rule=name_rule+"_"+jComboBox15.getSelectedItem()+"_"+jComboBox15.getN
ame();
names_rules=names_rules+"
"+jComboBox15.getSelectedItem()+"_"+jComboBox15.getName();
}
else
{
name_rule=jComboBox15.getSelectedItem()+"_"+jComboBox15.getName();
names_rules=names_rules+name_rule;
}
cad_user=cad_user+" "+jLabel11.getText().toUpperCase()+":
"+jComboBox15.getSelectedItem()+"\n";
cad_task=" (" +jComboBox15.getName()+" ?kt&:(fuzzy-match ?kt
"+""+jComboBox15.getSelectedItem()+"")");
c++;

```

```

}
if(selectedComboBoxAntec(jComboBox16)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox16.getSelectedItem()+"_"+jComboBox16.getN
ame();
        names_rules=names_rules+"
"+jComboBox16.getSelectedItem()+"_"+jComboBox16.getName();
    }
    else
    {
        name_rule=jComboBox16.getSelectedItem()+"_"+jComboBox16.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel12.getText().toUpperCase()+":
"+jComboBox16.getSelectedItem()+"\n";
    cad_task_dif=" ("+jComboBox16.getName()+" ?tt&:(fuzzy-match ?tt
"+""+jComboBox16.getSelectedItem()+"""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox17)&& selectedConsequences())
{
    String convert;
    convert=jComboBox17.getSelectedItem().toString();
    convert=convert.replace(' ','_');
    if (name_rule!="")
    {
        name_rule=name_rule+"_"+convert.toLowerCase()+"_"+jComboBox17.getName();
        names_rules=names_rules+"
"+convert.toLowerCase()+"_"+jComboBox17.getName();
    }
    else

```

```

{
    name_rule=convert.toLowerCase()+"_"+jComboBox17.getName();
    names_rules=names_rules+name_rule;
}

cad_user=cad_user+"    "+jLabel18.getText().toUpperCase()+":
"+jComboBox17.getSelectedItem()+"\n";
    cad_role=" (" +jComboBox17.getName()+" ?tag&:(eq ?tag
"+""+jComboBox17.getSelectedItem()+""+"))";
    c++;
}

if(selectedAntecedents() && selectedConsequences()) //si se ha seleccionado alguno
de los campos de los antecedentes
                                //y todos los campos de las consecuencias
{
    if (JOptionPane.showConfirmDialog(null,"¿Are you sure?","Confirmation
message",JOptionPane.OK_CANCEL_OPTION)== JOptionPane.OK_OPTION)
    {
        names_rules=names_rules+'*';
        file.newLine();
        file.write("(defrule ");
        file.write(name_rule);
        file.newLine();

        cad_user=cad_user+"    "+"=>"+"\n"+"    "+jLabel13.getText().toUpperCase()+":
"+jComboBox1.getSelectedItem()+"\n"+"    "+jLabel14.getText().toUpperCase()+":
"+jComboBox6.getSelectedItem()+"\n"+"    "+jLabel15.getText().toUpperCase()+":
"+jComboBox7.getSelectedItem()+"\n"+"    "+jLabel16.getText().toUpperCase()+":
"+jComboBox8.getSelectedItem()+"\n"+"    "+jLabel17.getText().toUpperCase()+":
"+jComboBox9.getSelectedItem()+ '*';
        if (c>1) //si hay más de un campo seleccionado
        {
            file.write(" (and");
        }
    }
}

```

```

if (cad_experience!="")
{
    file.write(cad_experience);
    file.newLine();
    file.write("    ");
}
if (cad_drive!="")
{
    file.write(cad_drive);
    file.newLine();
    file.write("    ");
}
if (cad_influence!="")
{
    file.write(cad_influence);
    file.newLine();
    file.write("    ");
}
if (cad_stainess!="")
{
    file.write(cad_stainess);
    file.newLine();
    file.write("    ");
}

if (cad_compliance!="")
{
    file.write(cad_compliance);
    file.newLine();
    file.write("    ");
}
if (cad_aloneteam!="")
{
    file.write(cad_aloneteam);

```

```

file.newLine();
file.write("    ");
}
if (cad_intro!="")
{
file.write(cad_intro);
file.newLine();
file.write("    ");
}
if (cad_duration!="")
{
file.write(cad_duration);
file.newLine();
file.write("    ");
}
if (cad_task!="")
{
file.write(cad_task);
file.newLine();
file.write("    ");
}
if (cad_task_dif!="")
{
file.write(cad_task_dif);
file.newLine();
file.write("    ");
}
if (cad_role!="")
{
file.write(cad_role);
file.newLine();
file.write("    ");
}
if (c>1) //si hay más de un campo seleccionado se debe cerrar el paréntesis del and

```



```

        file.close();
        init();
        modified=true;
        name_rule=""; //inicializar la variable para guardar el nombre de la siguiente regla
        que se vaya a generar
    }
}
else {
    JOptionPane.showMessageDialog(null,"Select an item","Error
Message",JOptionPane.ERROR_MESSAGE);
}
} catch (IOException ex){
    System.out.println("Error");
}
}
}
}

```

ConfigurationTaskRule.java:

```

import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.*;
import java.io.*.*;
import javax.swing.border.*;
import java.util.*;
/**
 * Title:    interfaz grafica
 * Description:
 * Copyright: Copyright (c) 2005
 * Company:
 * @author laura
 * @version 1.0
 */

```

```

public class ConfigurationTaskRule extends JFrame{
    JComboBox jComboBox1;
    JComboBox jComboBox2;
    JComboBox jComboBox3;
    JComboBox jComboBox4;
    JComboBox jComboBox5;
    JComboBox jComboBox6;
    JComboBox jComboBox7;
    JComboBox jComboBox8;
    JComboBox jComboBox9;
    JComboBox jComboBox10;
    JComboBox jComboBox11;
    JComboBox jComboBox12;
    JComboBox jComboBox13;
    JComboBox jComboBox14;
    JComboBox jComboBox15;
    JComboBox jComboBox16;
    JComboBox jComboBox17;
    JComboBox jComboBox18;
    JComboBox jComboBox19;
    JLabel jLabel1 = new JLabel();
    JLabel jLabel2 = new JLabel();
    JLabel jLabel3 = new JLabel();
    JLabel jLabel4 = new JLabel();
    JLabel jLabel5 = new JLabel();
    JLabel jLabel6 = new JLabel();
    JLabel jLabel7 = new JLabel();
    JLabel jLabel8 = new JLabel();
    JLabel jLabel9 = new JLabel();
    JLabel jLabel10 = new JLabel();
    JLabel jLabel11 = new JLabel();
    JLabel jLabel12 = new JLabel();
    JLabel jLabel13 = new JLabel();
    JLabel jLabel14 = new JLabel();

```

```

JLabel jLabel15 = new JLabel();
JLabel jLabel16 = new JLabel();
JLabel jLabel17 = new JLabel();
JLabel jLabel18 = new JLabel();
JLabel jLabel19 = new JLabel();
JLabel jLabel20 = new JLabel();
JLabel jLabel21 = new JLabel();
Border border1;
TitledBorder titledBorder1;
String[] listaTipo1;
String[] listaTipo2;
String[] listaTipo3;
BufferedWriter file;
JButton jButton1 = new JButton();
Border border2;
String name_rule=""; // cadena que guarda el nombre de la regla de una configuración
String names_rules=""; // cadena que guarda todos los nombres de las reglas
configuradas
String cad_user=""; // cadena para visualizar las reglas generadas
boolean modified=false;
JButton jButton3 = new JButton();
JButton jButton2 = new JButton();
JPanel jPanel1 = new JPanel();
TitledBorder titledBorder2;
menu menu1;

public ConfigurationTaskRule(menu menu1) {
    try {
        this.menu1=menu1;
        jbInit();

    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

```

```
}  
}
```

```
//Inicializacion de todos los campos
```

```
void init()
```

```
{  
jComboBox1.setSelectedItem(" ");  
jComboBox2.setSelectedItem(" ");  
jComboBox3.setSelectedItem(" ");  
jComboBox4.setSelectedItem(" ");  
jComboBox5.setSelectedItem(" ");  
jComboBox6.setSelectedItem(" ");  
jComboBox7.setSelectedItem(" ");  
jComboBox8.setSelectedItem(" ");  
jComboBox9.setSelectedItem(" ");  
jComboBox10.setSelectedItem(" ");  
jComboBox11.setSelectedItem(" ");  
jComboBox12.setSelectedItem(" ");  
jComboBox13.setSelectedItem(" ");  
jComboBox14.setSelectedItem(" ");  
jComboBox15.setSelectedItem(" ");  
jComboBox16.setSelectedItem(" ");  
jComboBox17.setSelectedItem(" ");  
jComboBox18.setSelectedItem(" ");  
jComboBox19.setSelectedItem(" ");  
}
```

```
//Creación de la interfaz
```

```
private void jbInit() throws Exception {  
    border1 = BorderFactory.createEmptyBorder();  
    titledBorder1 = new TitledBorder("");  
    border2 = new EtchedBorder(EtchedBorder.RAISED,Color.white,new Color(156,  
156, 158));
```

```

        titledBorder2 = new
TitledBorder(BorderFactory.createEtchedBorder(Color.white,new Color(156, 156,
158)), "Configuration");
        this.getContentPane().setLayout(null);

        listaTipo1 = new String[4];
        listaTipo1[0] = new String("Low");
        listaTipo1[1] = new String("Medium");
        listaTipo1[2] = new String("High");
        listaTipo1[3] = new String(" ");

        jLabel1.setText("Creativity");
        jLabel1.setBounds(new Rectangle(37, 37, 89, 22));

        jComboBox1= new JComboBox(listaTipo1);
        jComboBox1.setBounds(new Rectangle(137, 38, 123, 21));
        jComboBox1.setBorder(border1);
        jComboBox1.setSelectedItem(" ");
        jComboBox1.setName("creativity");

        jLabel2.setBounds(new Rectangle(37, 58, 89, 22));
        jLabel2.setText("Experience");

        jComboBox2= new JComboBox(listaTipo1);
        jComboBox2.setBounds(new Rectangle(137, 60, 123, 21));
        jComboBox2.setBorder(border1);
        jComboBox2.setSelectedItem(" ");
        jComboBox2.setName("experience");

        jLabel3.setBounds(new Rectangle(37, 79, 89, 22));
        jLabel3.setText("Drive");

        jComboBox3= new JComboBox(listaTipo1);
        jComboBox3.setBorder(border1);

```

```

jComboBox3.setSelectedItem(" ");
jComboBox3.setBounds(new Rectangle(137, 82, 123, 21));
jComboBox3.setName("drive");

jLabel4.setBounds(new Rectangle(37, 101, 89, 22));
jLabel4.setText("Influence");

jComboBox4= new JComboBox(listaTipo1);
jComboBox4.setBorder(border1);
jComboBox4.setSelectedItem(" ");
jComboBox4.setBounds(new Rectangle(137, 104, 123, 21));
jComboBox4.setName("influence");

jLabel5.setBounds(new Rectangle(37, 126, 89, 22));
jLabel5.setText("Stainess");

jComboBox5= new JComboBox(listaTipo1);
jComboBox5.setBorder(border1);
jComboBox5.setSelectedItem(" ");
jComboBox5.setBounds(new Rectangle(137, 126, 123, 21));
jComboBox5.setName("stainess");

jLabel6.setBounds(new Rectangle(37, 150, 89, 22));
jLabel6.setText("Compliance");

jComboBox6= new JComboBox(listaTipo1);
jComboBox6.setBorder(border1);
jComboBox6.setSelectedItem(" ");
jComboBox6.setBounds(new Rectangle(137, 148, 123, 21));
jComboBox6.setName("compliance");

jLabel7.setBounds(new Rectangle(37, 171, 89, 22));
jLabel7.setText("Desire");

```

```
jComboBox7= new JComboBox(listaTipo1);
jComboBox7.setBorder(border1);
jComboBox7.setSelectedItem(" ");
jComboBox7.setBounds(new Rectangle(137, 170, 123, 21));
jComboBox7.setName("desire");
```

```
jLabel8.setText("Interest");
jLabel8.setBounds(new Rectangle(37, 193, 89, 22));
```

```
jComboBox8= new JComboBox(listaTipo1);
jComboBox8.setBorder(border1);
jComboBox8.setSelectedItem(" ");
jComboBox8.setBounds(new Rectangle(137, 192, 123, 21));
jComboBox8.setName("interest");
```

```
jLabel9.setText("Disgust");
jLabel9.setBounds(new Rectangle(37, 213, 89, 22));
```

```
jComboBox9= new JComboBox(listaTipo1);
jComboBox9.setBorder(border1);
jComboBox9.setSelectedItem(" ");
jComboBox9.setBounds(new Rectangle(137, 214, 123, 21));
jComboBox9.setName("disgust");
```

```
jLabel10.setText("Anxiety");
jLabel10.setBounds(new Rectangle(37, 235, 89, 22));
```

```
jComboBox10= new JComboBox(listaTipo1);
jComboBox10.setBorder(border1);
jComboBox10.setSelectedItem(" ");
jComboBox10.setBounds(new Rectangle(137, 236, 123, 21));
jComboBox10.setName("anxiety");
```

```
jLabel11.setBounds(new Rectangle(37, 257, 89, 22));
```

```
jLabel11.setText("Stress");
```

```
jComboBox11= new JComboBox(listaTipo1);  
jComboBox11.setBorder(border1);  
jComboBox11.setSelectedItem(" ");  
jComboBox11.setBounds(new Rectangle(137, 258, 123, 21));  
jComboBox11.setName("stress");
```

```
jLabel12.setBounds(new Rectangle(36, 279, 89, 22));  
jLabel12.setText("Task type");
```

```
jComboBox12= new JComboBox(listaTipo1);  
jComboBox12.setBorder(border1);  
jComboBox12.setSelectedItem(" ");  
jComboBox12.setBounds(new Rectangle(137, 280, 123, 21));  
jComboBox12.setName("tasktype");
```

```
jLabel13.setBounds(new Rectangle(37, 302, 89, 22));  
jLabel13.setText("Task difficult");
```

```
jComboBox13= new JComboBox(listaTipo1);  
jComboBox13.setBorder(border1);  
jComboBox13.setSelectedItem(" ");  
jComboBox13.setBounds(new Rectangle(137, 302, 123, 21));  
jComboBox13.setName("taskdifficult");
```

```
listaTipo2 = new String[6];  
listaTipo2[0] = new String("High_Advance");  
listaTipo2[1] = new String("Medium_Advance");  
listaTipo2[2] = new String("Normal");  
listaTipo2[3] = new String("Medium_Delay");  
listaTipo2[4] = new String("High_Delay");  
listaTipo2[5] = new String(" ");
```

```

jLabel14.setText("Change Task Days");
jLabel14.setBounds(new Rectangle(310, 40, 174, 22));

jComboBox14= new JComboBox(listaTipo2);
jComboBox14.setBorder(border1);
jComboBox14.setSelectedItem(" ");
jComboBox14.setBounds(new Rectangle(498, 41, 146, 21));
jComboBox14.setName("change_task_days");

listaTipo3 = new String[6];
listaTipo3[0] = new String("High_Decrease");
listaTipo3[1] = new String("Medium_Decrease");
listaTipo3[2] = new String("Normal");
listaTipo3[3] = new String("Medium_Increase");
listaTipo3[4] = new String("High_Increase");
listaTipo3[5] = new String(" ");

jLabel15.setText("Change Task Quality");
jLabel15.setBounds(new Rectangle(310, 64, 174, 22));

jComboBox15= new JComboBox(listaTipo3);
jComboBox15.setBorder(border1);
jComboBox15.setSelectedItem(" ");
jComboBox15.setBounds(new Rectangle(498, 64, 146, 21));
jComboBox15.setName("change_task_quality");

jLabel16.setText("Change Goals Achievement");
jLabel16.setBounds(new Rectangle(310, 88, 174, 22));

jComboBox16= new JComboBox(listaTipo3);
jComboBox16.setBorder(border1);
jComboBox16.setSelectedItem(" ");
jComboBox16.setBounds(new Rectangle(498, 86, 146, 21));
jComboBox16.setName("change_goals_achievement");

```

```
jLabel17.setText("Change Team Collaboration");  
jLabel17.setBounds(new Rectangle(310, 110, 174, 22));
```

```
jComboBox17= new JComboBox(listaTipo3);  
jComboBox17.setBorder(border1);  
jComboBox17.setSelectedItem(" ");  
jComboBox17.setBounds(new Rectangle(498, 109, 146, 21));  
jComboBox17.setName("change_team_collaboration");
```

```
jLabel18.setText("Change Individual Contribution");  
jLabel18.setBounds(new Rectangle(310, 134, 174, 22));
```

```
jComboBox18= new JComboBox(listaTipo3);  
jComboBox18.setBorder(border1);  
jComboBox18.setSelectedItem(" ");  
jComboBox18.setBounds(new Rectangle(498, 132, 146, 21));  
jComboBox18.setName("change_individual_contribution");
```

```
jLabel19.setText("Change Required Supervision");  
jLabel19.setBounds(new Rectangle(310, 156, 174, 22));
```

```
jComboBox19= new JComboBox(listaTipo3);  
jComboBox19.setBorder(border1);  
jComboBox19.setSelectedItem(" ");  
jComboBox19.setBounds(new Rectangle(498, 155, 146, 21));  
jComboBox19.setName("change_required_supervision");
```

```
jLabel20.setBounds(new Rectangle(160, 7, 97, 39));  
jLabel20.setText("Antecedents");  
jLabel20.setToolTipText("");  
jLabel20.setFont(new java.awt.Font("Dialog", 1, 12));  
jLabel21.setFont(new java.awt.Font("Dialog", 1, 12));  
jLabel21.setToolTipText("");
```

```

jLabel21.setText("Consequences");
jLabel21.setBounds(new Rectangle(524, 7, 90, 39));
jButton1.setBorder(BorderFactory.createEtchedBorder());
jButton1.setText("Add");
jButton1.setBounds(new Rectangle(76, 345, 77, 21));
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        add_actionPerformed(e);
    }
});
//modificar ruta
copy("D:/JBuilder4/Interfaz/DefinitionsTaskRules.txt", "D:/JBuilder4/Interfaz/TasksRules.txt");

jButton3.setBounds(new Rectangle(303, 346, 72, 22));
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        close_actionPerformed(e);
    }
});
jButton3.setText("Close");
jButton3.setBorder(BorderFactory.createEtchedBorder());
jButton2.setBorder(BorderFactory.createEtchedBorder());
jButton2.setText("View");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        view_actionPerformed(e);
    }
});
jButton2.setBounds(new Rectangle(494, 346, 80, 22));
jPanel1.setBorder(titledBorder2);
jPanel1.setBounds(new Rectangle(11, 0, 655, 338));
this.setTitle("Tasks Rules");
this.getContentPane().add(jLabel9);

```

```
this.getContentPane().add(jLabel6);
this.getContentPane().add(jLabel5);
this.getContentPane().add(jLabel3);
this.getContentPane().add(jLabel2);
this.getContentPane().add(jLabel1);
this.getContentPane().add(jLabel7);
this.getContentPane().add(jLabel4);
this.getContentPane().add(jLabel8);
this.getContentPane().add(jLabel10);
this.getContentPane().add(jLabel11);
this.getContentPane().add(jLabel12);
this.getContentPane().add(jLabel13);
this.getContentPane().add(jComboBox10);
this.getContentPane().add(jComboBox5);
this.getContentPane().add(jComboBox2);
this.getContentPane().add(jComboBox3);
this.getContentPane().add(jComboBox1);
this.getContentPane().add(jComboBox4);
this.getContentPane().add(jComboBox6);
this.getContentPane().add(jComboBox7);
this.getContentPane().add(jComboBox8);
this.getContentPane().add(jComboBox9);
this.getContentPane().add(jComboBox11);
this.getContentPane().add(jComboBox12);
this.getContentPane().add(jComboBox13);
this.getContentPane().add(jComboBox14);
this.getContentPane().add(jComboBox15);
this.getContentPane().add(jComboBox16);
this.getContentPane().add(jComboBox17);
this.getContentPane().add(jComboBox18);
this.getContentPane().add(jComboBox19);
this.getContentPane().add(jLabel14);
this.getContentPane().add(jLabel15);
this.getContentPane().add(jLabel18);
```

```

this.getContentPane().add(jLabel19);
this.getContentPane().add(jLabel17);
this.getContentPane().add(jLabel16);
this.getContentPane().add(jLabel21);
this.getContentPane().add(jLabel20);
this.getContentPane().add(jPanel1);
this.getContentPane().add(jButton2);
this.getContentPane().add(jButton3);
this.getContentPane().add(jButton1);
this.setSize(685,410);
this.setLocation(300,200);
this.setVisible(true);
this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
this.addWindowListener(new java.awt.event.WindowAdapter() {
    public void windowClosed(WindowEvent e) {
        if(modified)
            endInformationRules();
        menu1.show();
    }
});
}

```

//Crea un fichero nuevo y copia el contenido del fichero de definiciones en este nuevo

```

void copy(String fuente, String destino) throws IOException {
    FileInputStream fis=new FileInputStream(fuente);
    File f=new File(destino);
    FileOutputStream fos=new FileOutputStream(new File(destino));
    int c;
    while((c=fis.read())!=-1) // -1 indica fin
        fos.write(c);

    fis.close();
    fos.close();
}

```

```

//Definiciones guardadas al final del fichero cuando se han acabado de añadir todas las
reglas
void endInformationRules ()
{
try{

file=new BufferedWriter (new FileWriter("TasksRules.txt",true));
file.newLine();
file.newLine();
file.write("(defrule defuzzify"+ ""+"low salience to allow all rules to fire and do global
contribution"+"");
file.newLine();
file.write(" (declare (salience -100))");
file.newLine();
file.write(" (change_task_days ?chd)");
file.newLine();
file.write(" (change_task_quality ?chq)");
file.newLine();
file.write(" (change_goals_achievement ?chga)");
file.newLine();
file.write(" (change_team_collaboration ?chte)");
file.newLine();
file.write(" (change_individual_contribution ?chic)");
file.newLine();
file.write(" (change_required_supervision ?chrs)");
file.newLine();
file.write(" =>");
file.newLine();
file.write(" (bind ?final-days (?chd momentDefuzzify))");
file.newLine();
file.write(" (bind ?final-quality (?chq momentDefuzzify))");
file.newLine();
file.write(" (bind ?final-goals (?chga momentDefuzzify))");

```

```

file.newLine();
file.write(" (bind ?final-collaboration (?chtc momentDefuzzify))");
file.newLine();
file.write(" (bind ?final-contribution (?chic momentDefuzzify))");
file.newLine();
file.write(" (bind ?final-supervision (?chrs momentDefuzzify))");
file.newLine();
file.write(" (store DIASPARALATAREA ?final-days)");
file.newLine();
file.write(" (store CALIDADTOTAL ?final-quality)");
file.newLine();
file.write(" (store METASALCANZADAS ?final-goals)");
file.newLine();
file.write(" (store COLABORACIONEQUIPO ?final-collaboration)");
file.newLine();
file.write(" (store CONTRIBUCIONINDIVIDUAL ?final-contribution)");
file.newLine();
file.write(" (store SUPERVISIONREQUERIDA ?final-supervision)");
file.newLine();
file.write(" (store RULESTHATFIRED ?*rulesThatFired*)");
file.newLine();
file.write("");

file.close();
this.hide();
} catch (IOException ex){
    System.out.println("Error");
}
}

```

//Comprobación: que se hayan seleccionado alguno de los campos de la lista de desplegables en el caso de antecedentes

```

boolean selectedAntecedents(){
    boolean selected=false;

```

```

if(selectedComboBoxAntec(jComboBox1)||selectedComboBoxAntec(jComboBox2)||selectedComboBoxAntec(jComboBox3)||selectedComboBoxAntec(jComboBox4)||selectedComboBoxAntec(jComboBox5)||selectedComboBoxAntec(jComboBox6)||selectedComboBoxAntec(jComboBox7)||selectedComboBoxAntec(jComboBox8)||selectedComboBoxAntec(jComboBox9)||selectedComboBoxAntec(jComboBox10)||selectedComboBoxAntec(jComboBox11)||selectedComboBoxAntec(jComboBox12)||selectedComboBoxAntec(jComboBox13))
    selected=true;
return selected;
}

```

//Comprobación: que se hayan seleccionado todos los campos de la lista de desplegables en el caso de las consecuencias

```

boolean selectedConsequences(){

```

```

    boolean selected=false;

```

```

if(selectedComboBoxCons(jComboBox14)&&selectedComboBoxCons(jComboBox15)
&&selectedComboBoxCons(jComboBox16)&&selectedComboBoxCons(jComboBox17)
&&selectedComboBoxCons(jComboBox18)&&selectedComboBoxCons(jComboBox19))

```

```

    selected=true;

```

```

return selected;

```

```

}

```

//Comprobación: que se haya seleccionada alguno de los elementos de la lista menos el campo en blanco

//la lista es para el campo change task days

```

boolean toCheckLista2(JComboBox combo){

```

```

    boolean selected=false;

```

```

for(int i=0;i<listaTipo2.length-1;i++){

```

```

    if (combo.getSelectedItem().equals(listaTipo2[i])){

```

```

        selected=true;

```

```

    }

```

```

    }
    return selected;
}

//Comprobación: que se haya seleccionada alguno de los elementos de la lista menos el
campo en blanco
//la lista es para los campos del apartado de consecuencias menos el campo change task
days
boolean toCheckLista3(JComboBox combo){
    boolean selected=false;
    for(int i=0;i<listaTipo3.length-1;i++){
        if (combo.getSelectedItem().equals(listaTipo3[i])){
            selected=true;
        }
    }
    return selected;
}

//Comprobación: tratar los diferentes campos según el tipo de lista al que pertenecen en
el apartado de consecuencias
boolean selectedComboBoxCons(JComboBox combo)
{
    boolean selected=false;

    if (combo.getName().compareTo("change_task_days")==0)
        selected=toCheckLista2(combo);    //método para la lista2
    else
        selected=toCheckLista3(combo); //método para la lista3
    return selected;
}

//Comprobación: que se haya seleccionado un campo de la lista desplegable en el
apartado de antecedentes
//la lista es la misma para todos los campos de los antecedentes

```

```

boolean selectedComboBoxAntec(JComboBox combo)
{
    boolean selected=false;
    for(int i=0;i<listaTipo1.length-1;i++){
        if (combo.getSelectedItem().equals(listaTipo1[i])){
            selected=true;
        }
    }
    return selected;
}

```

//Añadir las reglas generadas al fichero

//Hacer comprobaciones conforme se han seleccionado los campos de sus listas

//Escribir en el fichero las reglas

```

void add_actionPerformed(ActionEvent e) {

```

```

    try{

```

```

        file=new BufferedWriter (new FileWriter("TasksRules.txt",true));

```

```

        int c=0;

```

```

        String cad_experience="";

```

```

        String cad_creativity="";

```

```

        String cad_drive="";

```

```

        String cad_influence="";

```

```

        String cad_stainess="";

```

```

        String cad_compliance="";

```

```

        String cad_desire="";

```

```

        String cad_interest="";

```

```

        String cad_disgust="";

```

```

        String cad_anxiety="";

```

```

        String cad_stress="";

```

```

        String cad_task_type="";

```

```

        String cad_task_dif="";

```

```

        String aux="";

```

```

if (selectedComboBoxAntec(jComboBox1)&& selectedConsequences())
{
    c++;
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox1.getSelectedItem()+"_"+jComboBox1.getNa
me();
        names_rules=names_rules+"
"+jComboBox1.getSelectedItem()+"_"+jComboBox1.getName();
    }
    else
    {
        name_rule=jComboBox1.getSelectedItem()+"_"+jComboBox1.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+" "+jLabel1.getText().toUpperCase()+":
"+jComboBox1.getSelectedItem()+"\n";
    cad_creativity=" (" + jComboBox1.getName()+" ?c&:(fuzzy-match ?c
"+'"'+jComboBox1.getSelectedItem()+"'+")";
}
if (selectedComboBoxAntec(jComboBox2)&& selectedConsequences())
{
    c++;
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox2.getSelectedItem()+"_"+jComboBox2.getNa
me();
        names_rules=names_rules+"
"+jComboBox2.getSelectedItem()+"_"+jComboBox2.getName();
    }
    else

```

```

{
    name_rule=jComboBox2.getSelectedItemAt()+"_"+jComboBox2.getName();
    names_rules=names_rules+name_rule;
}

cad_user=cad_user+" "+jLabel2.getText().toUpperCase()+":
"+jComboBox2.getSelectedItemAt()+"\n";
    cad_experience=" (" +jComboBox2.getName()+" ?e&:(fuzzy-match ?e
"+""+jComboBox2.getSelectedItemAt()+""+"))";
}
if(selectedComboBoxAntec(jComboBox3)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox3.getSelectedItemAt()+"_"+jComboBox3.getNa
me();
        names_rules=names_rules+"
"+jComboBox3.getSelectedItemAt()+"_"+jComboBox3.getName();
    }
    else
    {
        name_rule=jComboBox3.getSelectedItemAt()+"_"+jComboBox3.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+" "+jLabel3.getText().toUpperCase()+":
"+jComboBox3.getSelectedItemAt()+"\n";
    cad_drive=" (" +jComboBox3.getName()+" ?dr&:(fuzzy-match ?dr
"+""+jComboBox3.getSelectedItemAt()+""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox4)&& selectedConsequences())
{
    if (name_rule!="")
    {

```

```

name_rule=name_rule+"_"+jComboBox4.getSelectedItemAt()+"_"+jComboBox4.getNa
me();
    names_rules=names_rules+"
"+jComboBox4.getSelectedItemAt()+"_"+jComboBox4.getName();
    }
    else
    {
        name_rule=jComboBox4.getSelectedItemAt()+"_"+jComboBox4.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel4.getText().toUpperCase()+":
"+jComboBox4.getSelectedItemAt()+"\n";
    cad_influence=" (" +jComboBox4.getName()+ " ?inf&:(fuzzy-match ?inf
"+""+jComboBox4.getSelectedItemAt()+""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox5)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox5.getSelectedItemAt()+"_"+jComboBox5.getNa
me();
        names_rules=names_rules+"
"+jComboBox5.getSelectedItemAt()+"_"+jComboBox5.getName();
        }
        else
        {
            name_rule=jComboBox5.getSelectedItemAt()+"_"+jComboBox5.getName();
            names_rules=names_rules+name_rule;
        }
        cad_user=cad_user+"    "+jLabel5.getText().toUpperCase()+":
"+jComboBox5.getSelectedItemAt()+"\n";

```

```

        cad_stainess=" (" +jComboBox5.getName()+" ?stss&:(fuzzy-match ?stss
"+""+jComboBox5.getSelectedItem()+""+""));
        c++;
    }
    if(selectedComboBoxAntec(jComboBox6)&& selectedConsequences())
    {
        if (name_rule!="")
        {

name_rule=name_rule+"_"+jComboBox6.getSelectedItem()+"_"+jComboBox6.getNa
me();
        names_rules=names_rules+"
"+jComboBox6.getSelectedItem()+"_"+jComboBox6.getName();
        }
        else
        {
            name_rule=jComboBox6.getSelectedItem()+"_"+jComboBox6.getName();
            names_rules=names_rules+name_rule;
        }
        cad_user=cad_user+"      "+jLabel6.getText().toUpperCase()+"":
"+jComboBox6.getSelectedItem()+"\n";
        cad_compliance=" (" +jComboBox6.getName()+" ?cmpl&:(fuzzy-match ?cmpl
"+""+jComboBox6.getSelectedItem()+""+""));
        c++;
    }
    if(selectedComboBoxAntec(jComboBox7)&& selectedConsequences())
    {
        if (name_rule!="")
        {

name_rule=name_rule+"_"+jComboBox7.getSelectedItem()+"_"+jComboBox7.getNa
me();
        names_rules=names_rules+"
"+jComboBox7.getSelectedItem()+"_"+jComboBox7.getName();

```

```

    }
else
{
    name_rule=jComboBox7.getSelectedItemAt()+"_"+jComboBox7.getName();
    names_rules=names_rules+name_rule;
}
cad_user=cad_user+"    "+jLabel7.getText().toUpperCase()+":
"+jComboBox7.getSelectedItemAt()+"\n';
    cad_desire=" (" +jComboBox7.getName()+" ?d&:(fuzzy-match ?d
"+""+jComboBox7.getSelectedItemAt()+""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox8)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox8.getSelectedItemAt()+"_"+jComboBox8.getNa
me();
        names_rules=names_rules+"
"+jComboBox8.getSelectedItemAt()+"_"+jComboBox8.getName();
    }
else
{
    name_rule=jComboBox8.getSelectedItemAt()+"_"+jComboBox8.getName();
    names_rules=names_rules+name_rule;
}
    cad_user=cad_user+"    "+jLabel8.getText().toUpperCase()+":
"+jComboBox8.getSelectedItemAt()+"\n';
    cad_interest=" (" +jComboBox8.getName()+" ?i&:(fuzzy-match ?i
"+""+jComboBox8.getSelectedItemAt()+""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox9)&& selectedConsequences())

```

```

{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox9.getSelectedItem()+"_"+jComboBox9.getNa
me();

        names_rules=names_rules+"
"+jComboBox9.getSelectedItem()+"_"+jComboBox9.getName();
    }
    else
    {
        name_rule=jComboBox9.getSelectedItem()+"_"+jComboBox9.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel9.getText().toUpperCase()+":
"+jComboBox9.getSelectedItem()+"\n';
    cad_disgust=" (" +jComboBox9.getName()+" ?di&:(fuzzy-match ?di
"+""+jComboBox9.getSelectedItem()+"'+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox10)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox10.getSelectedItem()+"_"+jComboBox10.getN
ame();

        names_rules=names_rules+"
"+jComboBox10.getSelectedItem()+"_"+jComboBox10.getName();
    }
    else
    {
        name_rule=jComboBox10.getSelectedItem()+"_"+jComboBox10.getName();
        names_rules=names_rules+name_rule;
    }
}

```

```

    }
    cad_user=cad_user+"    "+jLabel10.getText().toUpperCase()+":
"+jComboBox10.getSelectedItem()+"\n";
    cad_anxiety=" (" +jComboBox10.getName()+" ?a&:(fuzzy-match ?a
"+""+jComboBox10.getSelectedItem()+"")");
    c++;
}
if(selectedComboBoxAntec(jComboBox11)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox11.getSelectedItem()+"_"+jComboBox11.getN
ame();
        names_rules=names_rules+"
"+jComboBox11.getSelectedItem()+"_"+jComboBox11.getName();
    }
    else
    {
        name_rule=jComboBox11.getSelectedItem()+"_"+jComboBox11.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel11.getText().toUpperCase()+":
"+jComboBox11.getSelectedItem()+"\n";
    cad_stress=" (" +jComboBox11.getName()+" ?s&:(fuzzy-match ?s
"+""+jComboBox11.getSelectedItem()+"")");
    c++;
}
if(selectedComboBoxAntec(jComboBox12)&& selectedConsequences())
{
    if (name_rule!="")
    {

```

```

name_rule=name_rule+"_"+jComboBox12.getSelectedItemAt()+"_"+jComboBox12.getN
ame();
    names_rules=names_rules+"
"+jComboBox12.getSelectedItemAt()+"_"+jComboBox12.getName();
    }
    else
    {
        name_rule=jComboBox12.getSelectedItemAt()+"_"+jComboBox12.getName();
        names_rules=names_rules+name_rule;
    }
    cad_user=cad_user+"    "+jLabel12.getText().toUpperCase()+":
"+jComboBox12.getSelectedItemAt()+"\n";
    cad_task_type=" (" +jComboBox12.getName()+ " ?tstp&:(fuzzy-match ?tstp
"+""+jComboBox12.getSelectedItemAt()+""+"))";
    c++;
}
if(selectedComboBoxAntec(jComboBox13)&& selectedConsequences())
{
    if (name_rule!="")
    {

name_rule=name_rule+"_"+jComboBox13.getSelectedItemAt()+"_"+jComboBox13.getN
ame();
        names_rules=names_rules+"
"+jComboBox13.getSelectedItemAt()+"_"+jComboBox13.getName();
        }
        else
        {
            name_rule=jComboBox13.getSelectedItemAt()+"_"+jComboBox13.getName();
            names_rules=names_rules+name_rule;
        }
        cad_user=cad_user+"    "+jLabel13.getText().toUpperCase()+":
"+jComboBox13.getSelectedItemAt()+"\n";

```

```

        cad_task_dif=" (" +jComboBox13.getName()+" ?tsdiff&:(fuzzy-match ?tsdiff
"+""+jComboBox13.getSelectedItem()+""+"))";
        c++;
    }

    if(selectedAntecedents() && selectedConsequences())
    {
        if (JOptionPane.showConfirmDialog(null,"¿Are you sure?","Confirmation
message",JOptionPane.OK_CANCEL_OPTION)== JOptionPane.OK_OPTION)
        {
            names_rules=names_rules+'*';
            file.newLine();
            file.write("(defrule ");
            file.write(name_rule);
            file.newLine();

            cad_user=cad_user+"      "+"=>"+"\n"+"      "+jLabel14.getText().toUpperCase()+"":
            "+jComboBox14.getSelectedItem()+"\n"+"      "+jLabel15.getText().toUpperCase()+"":
            "+jComboBox15.getSelectedItem()+"\n"+"      "+jLabel16.getText().toUpperCase()+"":
            "+jComboBox16.getSelectedItem()+"\n"+"      "+jLabel17.getText().toUpperCase()+"":
            "+jComboBox17.getSelectedItem()+"\n"+"      "+jLabel18.getText().toUpperCase()+"":
            "+jComboBox18.getSelectedItem()+"\n"+"      "+jLabel19.getText().toUpperCase()+"":
            "+jComboBox19.getSelectedItem()+ '*';
            if (c>1)//si hay más de un campo seleccionado
            {
                file.write(" (and");
            }
            if (cad_creativity!="")
            {
                file.write(cad_creativity);
                file.newLine();
                file.write("      ");
            }
            if (cad_experience!="")

```

```

{
    file.write(cad_experience);
    file.newLine();
    file.write("    ");
}
if (cad_drive!="")
{
    file.write(cad_drive);
    file.newLine();
    file.write("    ");
}
if (cad_influence!="")
{
    file.write(cad_influence);
    file.newLine();
    file.write("    ");
}
if (cad_stainess!="")
{
    file.write(cad_stainess);
    file.newLine();
    file.write("    ");
}
if (cad_compliance!="")
{
    file.write(cad_compliance);
    file.newLine();
    file.write("    ");
}
if (cad_desire!="")
{
    file.write(cad_desire);
    file.newLine();
    file.write("    ");
}

```

```

}
if (cad_interest!="")
{
    file.write(cad_interest);
    file.newLine();
    file.write(" ");
}
if (cad_disgust!="")
{
    file.write(cad_disgust);
    file.newLine();
    file.write(" ");
}
if (cad_anxiety!="")
{
    file.write(cad_anxiety);
    file.newLine();
    file.write(" ");
}
if (cad_stress!="")
{
    file.write(cad_stress);
    file.newLine();
    file.write(" ");
}
if (cad_task_type!="")
{
    file.write(cad_task_type);
    file.newLine();
    file.write(" ");
}
if (cad_task_dif!="")
{
    file.write(cad_task_dif);

```

```

file.newLine();
file.write("    ");
}
if (c>1)//si hay más de un campo seleccionado
{
file.write("");
file.newLine();
}

file.write(" =>");
file.newLine();

file.write("  (assert (change_task_days (new FuzzyValue ?*ChangeTaskDays*
"+""+jComboBox14.getSelectedItemAt(""+""+"))));
file.newLine();
file.write("      (change_task_quality (new FuzzyValue ?*ChangeTaskQuality*
"+""+jComboBox15.getSelectedItemAt(""+""+"))));
file.newLine();
file.write("      (change_goals_achievement (new
FuzzyValue ?*ChangeGoalsAchievement*
"+""+jComboBox16.getSelectedItemAt(""+""+"))));
file.newLine();
file.write("      (change_team_collaboration (new
FuzzyValue ?*ChangeTeamCollaboration*
"+""+jComboBox17.getSelectedItemAt(""+""+"))));
file.newLine();
file.write("      (change_individual_contribution (new
FuzzyValue ?*ChangeIndividualContribution*
"+""+jComboBox18.getSelectedItemAt(""+""+"))));
file.newLine();
file.write("      (change_required_supervision (new
FuzzyValue ?*ChangeRequiredSupervision*
"+""+jComboBox19.getSelectedItemAt(""+""+"))));
file.newLine();

```

```

file.write(" ");
file.newLine();
file.write(" (bind ?*rulesThatFired* (str-cat ?*rulesThatFired*");
file.newLine();

aux=name_rule.replace('_', ' '); //comentario que aparece en la regla del fichero
file.write("      "+"'+aux.toUpperCase()+"'; "+"'+')");
file.newLine();
file.write(" ");
file.newLine();
file.write("");
file.close();
init();
modified=true;
name_rule=""; //inicializar la variable para guardar el nombre de la siguiente regla
que se vaya a generar
    }
}
else {
    JOptionPane.showMessageDialog(null,"Select an item","Error
Message",JOptionPane.ERROR_MESSAGE);
    }
    } catch (IOException ex){
        System.out.println("Error");
    }
}

//Cerrar la ventana de configuración
void close_actionPerformed(ActionEvent e) {
    if (modified==false)
        this.hide();
    else endInformationRules();
    menu1.show();
}

```

```

//Visualizar las reglas generadas
//cad_user contiene las reglas generadas
//names_rules contiene el nombre de las reglas generadas
//Separadas por un asterisco para separar las reglas
void view_actionPerformed(ActionEvent e) {
    panel p= new panel();
    StringTokenizer st = new StringTokenizer(cad_user,"*");
    StringTokenizer st2 = new StringTokenizer(names_rules,"*");
    while (st2.hasMoreTokens()&&st.hasMoreTokens()) {
        p.jTextArea1.append(" RULE "+st2.nextToken()+" :
\n"+st.nextToken()+"\n"+"
\n");
    }
}
}

```

Panel.java:

```

import javax.swing.*;
import java.awt.*;
import javax.swing.border.*;
import java.awt.event.*;

/**
 * Title:    interfaz grafica
 * Description:
 * Copyright: Copyright (c) 2005
 * Company:
 * @author laura
 * @version 1.0
 */

public class panel extends JFrame {
    JLabel jLabel1 = new JLabel();

```

```

JScrollPane jScrollPane1 = new JScrollPane();
JScrollPane jScrollPane3 = new JScrollPane();
JTextArea jTextArea1=new JTextArea();
JButton jButton1 = new JButton();

public panel() {
    try {
        jbInit();
    }
    catch(Exception e) {
        e.printStackTrace();
    }
}

private void jbInit() throws Exception {

    jLabel1.setHorizontalAlignment(SwingConstants.CENTER);
    jLabel1.setText("RULES");
    jLabel1.setBounds(new Rectangle(208, 4, 68, 33));

    jScrollPane1.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_
_ALWAYS);

    jScrollPane1.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AL
WAYS);

    jScrollPane1.getViewport().setBackground(Color.white);
    jScrollPane1.setBorder(BorderFactory.createLoweredBevelBorder());
    jScrollPane1.setBounds(new Rectangle(29, 38, 451, 246));

    this.getContentPane().setLayout(null);
    jButton1.setBorder(BorderFactory.createEtchedBorder());
    jButton1.setText("Ok");
    jButton1.setBounds(new Rectangle(227, 298, 59, 22));

```

```

jButton1.addActionListener(new java.awt.event.ActionListener(){
    public void actionPerformed(ActionEvent e){
        ok_actionPerformed(e);
    }
});
jTextArea1.setToolTipText("");
jTextArea1.setFont(new java.awt.Font("Dialog", 2, 12));
this.getContentPane().add(jScrollPane1);
jScrollPane1.getViewport().add(jTextArea1);
this.getContentPane().add(jLabel1);
this.getContentPane().add(jButton1);
this.setTitle("View");
this.setSize(500,380);
this.setLocation(380,250);
this.setVisible(true);
this.setResizable(false);
}

void ok_actionPerformed(ActionEvent e){
    this.hide();
}
}

```

Menu.java:

```

import javax.swing.*;
import java.awt.*;
import javax.swing.border.*;
import java.awt.event.*;
/**
 * Title:    interfaz grafica
 * Description:
 * Copyright: Copyright (c) 2005
 * Company:

```

```
* @author laura
* @version 1.0
*/
```

```
public class menu extends JFrame{
    JButton jButton1 = new JButton();
    JButton jButton2 = new JButton();
    Border border1;
    Border border2;
    TitledBorder titledBorder1;
    JButton jButton3 = new JButton();
    JPanel jPanel1 = new JPanel();

    public menu() {
        try {
            jbInit();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        menu opcion1 = new menu();
    }

    private void jbInit() throws Exception {
        border1 = BorderFactory.createEmptyBorder();
        border2 = BorderFactory.createEmptyBorder();
        titledBorder1 = new
TitledBorder(BorderFactory.createEtchedBorder(Color.white,new Color(156, 156,
158)), "Operations");
        this.getContentPane().setLayout(null);
        jButton1.setBorder(border1);
        jButton1.setHorizontalAlignment(SwingConstants.LEFT);
        jButton1.setText("Behaviour Rules");
    }
}
```

```

jButton1.setBounds(new Rectangle(71, 50, 134, 38));
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        behaviourRules_actionPerformed(e);
    }
});
jButton2.setBounds(new Rectangle(71, 80, 134, 38));
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        taskRules_actionPerformed(e);
    }
});
jButton2.setBorder(border1);
jButton2.setHorizontalAlignment(SwingConstants.LEFT);
jButton2.setText("Task Rules");
jButton3.setBorder(BorderFactory.createEtchedBorder());
jButton3.setText("Cancel");
jButton3.setBounds(new Rectangle(70, 180, 92, 20));
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(ActionEvent e) {
        cancel_actionPerformed(e);
    }
});
jPanel1.setBorder(titledBorder1);
jPanel1.setBounds(new Rectangle(9, 2, 213, 169));
this.getContentPane().add(jButton1, null);
this.getContentPane().add(jButton2, null);
this.getContentPane().add(jPanel1, null);
this.getContentPane().add(jButton3, null);

this.setTitle("Menu");
this.setSize(240,240);
this.setLocation(400,200);
this.setResizable(false);

```

```
        this.setVisible(true);
    }

    void cancel_actionPerformed(ActionEvent e) {
        System.exit(0);
    }

    void behaviourRules_actionPerformed(ActionEvent e) {
        this.hide();
        ConfigurationBehaviourRule cbr = new ConfigurationBehaviourRule(this);
    }

    void taskRules_actionPerformed(ActionEvent e) {
        this.hide();
        ConfigurationTaskRule ctr = new ConfigurationTaskRule(this);
    }
}
```