PlanInteraction : Interacción multiagente para planificación

Pablo Castejón Navarro

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
Y COMPUTACIÓN

Dirigido por:
Eva Onaindía de la Rivaherrera

Trabajo Final de Máster desarrollado dentro del Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Valencia, Junio 2013

A mi madre,
porque los planes se tuercen pero
la experiencia nos ayuda a seguir adelante.

Agradecimientos

La elaboración de esta tesina es el resultado de un plan largo y complejo y las interacciones que se han llevado a cabo durante su ejecución. Quisiera dedicarle unas líneas de agradecimiento a todos esos "agentes" cuya intervención, pequeña o grande, ha sido imprescindible para la consecución del mismo.

Al extinto Ministerio de Ciencia e Innovación (MICINN) por la financiación para la realización del proyecto PlanInteraction, sin la cual no hubiera sido posible llevar a cabo mi investigación. A los grupos de investigación de las universidades UC3M y UGR por aportar su experiencia y maestría en la consolidación del proyecto. Y a todos los compañeros que han participado en las fases de diseño, implementación y depuración de la arquitectura PlanInteraction.

A mi directora de tesina, la doctora Eva Onaindía, por su confianza en mí desde el primer momento, su tenacidad en el trabajo y su comprensión más allá del deber. A los compañeros de laboratorio que me han ayudado a crecer profesionalmente. A mis compañeros del máster, porque remar en compañía es más grato y sencillo.

A mis familiares y amigos, siempre curiosos, conscientes e interesados por mi trabajo. Y toda la gente que me ha acogido fuera de mi ciudad natal, especialmente a Marc, Lucian, Elizabeth, Sergio E., Victor, Alejandro, Sergio P. y Moisés, gracias por hacerme sentir como en casa.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se enmarca el trabajo desarrollado introduciendo cuál es el problema que se pretende resolver, por qué se pretende resolver y cuáles son los objetivos principales a alcanzar. Al final del capítulo se hace un resumen del contenido de cada uno de los capítulos del trabajo.

1.1. Descripción del problema, motivación y objetivos

El presente Trabajo Fin de Máster (TFM) se enmarca dentro del proyecto de investigación TIN2011-27652-C03 bajo el nombre de "PlanInteraction: Multiagent Interaction for Planning". El proyecto PlanInteraction es un trabajo coordinado con otros dos grupos de investigación de las universidades UC3M[1] y UGR[2]. El objetivo de PlanInteraction consiste en desarrollar nuevas técnicas y tecnologías basadas en dinámicas sociales para el diseño e implementación de una plataforma de planificación multiagente compuesta de entidades de planificación autónomas y posiblemente heterogéneas. La investigación en PlanInteraction está en sintonía con los esfuerzos recientes en teoría de la decisión, teoría de juegos, planificación y scheduling automático, aprendizaje y otros campos de investigación.

[1] Universidad Carlos III de Madrid
[2] Universidad de Granada

El diseño de PlanInteraction se concibe a tres niveles:

a) Nivel de agente. El elemento fundamental es el agente de planificación con capacidades para planificar, actuar y aprender. Más específicamente, un agente es una plataforma que integra una colección de tecnologías de planificación, incluyendo sensorización, ejecución, monitorización, replanificación e incluso aprendizaje de experiencias pasadas.

b) Nivel de interacción. La esencia del comportamiento de los agentes es cómo éstos interactúan con el entorno.

c) Nivel de organización. La idea principal es la estructura de agentes en organizaciones, esto es, cómo conjuntos de agentes agrupados en estructuras organizacionales o coaliciones que representan una posición más ventajosa para conseguir un determinado conjunto de objetivos de planificación o llevar a cabo complejas tareas de planificación.

Los objetivos específicos del proyecto PlanInteraction incluyen:

1) Partiendo de un proyecto anterior en el que se diseñó una arquitectura de planificación y ejecución llamada PELEA, un primer objetivo de PlanInteraction es adaptar las técnicas de planificación monoagente desarrolladas en PELEA a un entorno multiagente, teniendo así un modelo de planificación multiagente.

2) Asimismo, como resultado de otro proyecto anterior, se dispone de una plataforma denominada Magentix2 para el desarrollo de sistemas multiagente abiertos de gestión optimizada y segura. Un segundo objetivo del proyecto PlanInteraction es la integración de la arquitectura de planificación monoagente PELEA en Magentix2, dando como resultado una plataforma de código abierto que permite a los usuarios desarrollar de forma sencilla aplicaciones de planificación multiagente complejas.

3) Por último, el tercer objetivo de PlanInteraction es definir y desarrollar nuevas tecnologías de agentes para la coordinación e interacción de agentes de planificación, teniendo como resultado un conjunto de nuevas técnicas para planificación multiagente.

En este TFM se afronta el diseño y desarrollo de PlanInteraction en sus dos primeros niveles, nivel de agente e interacción; y se cubren parcialmente los tres objetivos del proyecto, adaptación e integración de las tecnologías previas y desarrollo de técnicas sociales de interacción para la coordinación entre agentes. Para llevar a cabo el trabajo desarrollado en este TFM dentro del marco del proyecto PlanInteraction se han realizado las siguientes tareas:

1) Un estudio previo de las características y problemas fundamentales de arquitecturas de planificación y arquitecturas multiagente existentes, así como algunos de los intentos previos de unificación de ambos tipos de tecnologías en arquitecturas de planificación multiagente o sistemas multiagente para planificación.

2) Diseño y desarrollo de los mecanismos fundamentales para la configuración y funcionamiento de un sistema multiagente: modelo de agentes, sistema de comunicaciones, control del sistema, depuración, etc.

3) A un nivel más concreto se han desarrollado herramientas de diseño e implementación de métodos de razonamiento o comportamientos de agentes, un conjunto de tipologías de agentes, protocolos de comunicación, APIs de comunicación con librerías heredadas, un prototipo de interfaz gráfica para configuración, ejecución, análisis y depuración de experimentos, etc. Todo ello teniendo presente como principio básico la generalidad en el diseño de las herramientas. El objetivo es permitir la reutilización, extensión y fácil utilización de las herramientas por parte del usuario final así como por los desarrolladores.

4) Por último se presenta ejemplos de aplicación de la arquitectura PlanInteraction a varios casos de estudio que han servido tanto para validar la plataforma a nivel funcional como para desarrollar investigaciones paralelas incidiendo en el tercer objetivo del proyecto PlanInteraction.

Ejemplos de dominios en los que estas tecnologías están empezando a ser necesarias surgen de la reciente experiencia de los grupos

involucrados en el proyecto en la resolución de problemas de planificación y desarrollo de aplicaciones del mundo real.

1.2. Estructura de la tesis

Además del actual capítulo introductorio, el presente trabajo consta de otros cinco capítulos y un apéndice, los cuales se describen a continuación:

- **Capítulo 2** Estado del arte.

  Este capítulo describe el estado del arte en sistemas de planificación multiagente. En primer lugar se introduce el concepto de planificación automática así como algunas aproximaciones previas. A continuación se define un sistema multiagente, sus principales características y algunos de los trabajos previos relacionados. Por último se presentan tres de los sistemas de planificación multiagente que se han tenido en cuenta para el desarrollo de la arquitectura PlanInteraction.

- **Capítulo 3** PlanInteraction.

  Este capítulo presenta la arquitectura PlanInteraction, su diseño conceptual, así como el proceso y técnicas empleadas para su desarrollo. Primero se da una perspectiva global de la arquitectura para, posteriormente, ir detallando cada uno de los componentes de la misma. A continuación, el capítulo se centra en el diseño e implementación a nivel de agente y sus funcionalidades. Finalmente se introducen problemáticas de la interacción entre agentes y cómo se han resuelto, así como un prototipo de interfaz gráfica para el análisis y depuración de simulaciones.

- **Capítulo 4** Ejemplos de implementación.

  En este capítulo se incide sobre las características observadas para determinar la validación de la arquitectura y posteriormente se introducen y detallan tres casos de estudio implementados sobre la arquitectura PlanInteraction. El primer caso de

estudio considera dominios de planificación cooperativa poniendo en práctica la construcción de planes de forma coordinada. El segundo caso de estudio propone otra aproximación a la construcción de planes coordinados en un dominio médico real. Por último, se expone un caso de estudio de los *Rovers* empleados por la *NASA* para la exploración espacial, empleando técnicas de coordinación entre agente a nivel de ejecución.

- **Capítulo 5** Conclusiones, trabajos futuros y publicaciones.

  Este capítulo recoge las conclusiones derivadas de la realización del presente proyecto, así como las líneas de investigación abiertas y los trabajos futuros previstos dentro del proyecto PlanInteraction. Finalmente se listan las publicaciones a las que ha contribuído este trabajo de investigación.

Capítulo 2

Estado del arte

En este capítulo se hace una introducción al problema de la planificación automática y de los sistemas multiagente o MAS[1] como base para los sistemas de planificación multiagente o MAPS[2]. En primer lugar se define la planificación automática, conceptos básicos y arquitecturas de planificación y ejecución. A continuación se introducen conceptos básicos de los MAS, así como algunas de las principales plataformas. Y por último se presentan tres de los MAPS más relevantes en la actualidad comparables con PlanInteraction.

2.1. Planificación Automática

La planificación automática es una disciplina de la inteligencia artificial que se ocupa de la generación de estrategias o secuencias de acciones que posteriormente serán ejecutadas por robots autónomos, agentes inteligentes o vehículos no tripulados. La planificación automática se ha aplicado a multitud de dominios como robótica, cadenas de montaje, redes de transporte y otros tipos de tareas de gran complejidad.

2.1.1. Conceptos básicos

Tal y como se describe en [25], la formulación clásica de un problema de planificación se compone de tres elementos (1) una descripción inicial del estado del mundo en algún lenguaje formal, (2)

[1] Multi-Agent Systems
[2] Multi-Agent Planning Systems

una descripción de los objetivos y (3) una descripción de las acciones que se pueden realizar. A partir de estos elementos la resolución del problema consiste en la búsqueda, a partir del estado inicial, de una secuencia de acciones tal que se alcance un estado en el que se cumplan los objetivos planteados. Dicha secuencia de acciones es lo que se conoce como *plan*.

A medida que las técnicas de planificación automática han avanzado, también lo han hecho los lenguajes para la definición de problemas de planificación. Desde las IPC[3] que comenzaron en el año 1998 y se celebran bi-anualmente en el marco de la conferencia ICAPS[4], el lenguaje que se utiliza de manera generalizada para la definición de problemas de planificación es PDDL[5]. Existen otros paradigmas de planificación que utilizan su propio lenguaje de definición como es el caso de la planificación jerárquica o HTN[6].

Por lo general, en un problema de planificación, el estado del mundo viene representado por una secuencia de variables que toman un determinado valor de entre los posibles valores para el dominio de cada variable. Las acciones son funciones de transformación que se definen con una serie de precondiciones que deben cumplirse en un estado y los efectos que se producen en el estado o mundo resultante. Los objetivos son valores concretos que se pretenden alcanzar para un subconjunto de las variables del problema.

Existen multitud de técnicas para construir un plan como: *forward planning* (partiendo del estado inicial), ,*backward planning* (partiendo del estado final) o *least-commitment planning* (añadiendo acciones de forma no secuencial). Otras consideraciones que se pueden tener en cuenta a la hora de realizar la planificación son la incertidumbre, el tiempo o la limitación de recursos (gasolina, capacidad, dinero, batería, etc.).

[3]International Planning Competition
[4] International Conference on Automated Planning and Scheduling
[5]Planning Domain Definition Language
[6]Hierarchical Task Network

2.1.2. Arquitecturas de planificación y ejecución

Las primeras aproximaciones para resolver problemas de planificación consistían en sistemas especializados para cada problema, es decir, soluciones dependientes del dominio. Sin embargo, con el paso del tiempo se han generalizado las técnicas de planificación de forma que se puedan abordar un mayor número de problemas de diversa tipología. Por este motivo, las investigaciones más recientes se centran en el diseño de técnicas de planificación independientes del dominio de aplicación.

El proceso de planificación se refiere al proceso de construcción de un plan para resolver un problema determinado; el proceso de ejecución de un plan se refiere a la modificación del estado del mundo tras la ejecución de las acciones contenidas en un plan. De este modo las arquitecturas de planificación y ejecución pueden combinar ambos procesos, o bien estar más orientadas hacia uno de los dos. Las arquitecturas deliberativas usan planificación y razonamiento o métodos de inferencia que realizan un proceso de búsqueda para la construcción del plan. Las arquitecturas reactivas proporcionan una reacción inmediata ante cambios en el estado que son captados por los sensores, y las decisiones se toman en tiempo de ejecución en base a información precompilada. También se contemplan arquitecturas híbridas que combinan las ventajas de las dos anteriores y arquitecturas interactivas en las que la coordinación de los agentes se realiza mediante comunicación.

Existen numerosas arquitecturas de planificación y/o ejecución, entre las que destacan arquitecturas fundamentalmente reactivas como *PRS* [12] o *Steels*[20] o híbridas como *Plan-Based Control of Robotics Agents*[6], *Reactive Action Packages*[9, 10](RAPs), *GOFER*[7], *Maes*[14] o *Continual Planning and Execution Framework* (CPEF) [15].

PELEA

PELEA[7] [17] es una arquitectura de planificación y ejecución resultado de un proyecto de investigación conjunto entre los grupos GRPS-AI y GTI-IA de la UPV[8], el grupo de Planning and Learning de la UC3[9] y el grupo de Sistemas Inteligentes de la Universidad de Granada y financiado por el proyecto MICIIN TIN2008-06701-C03.

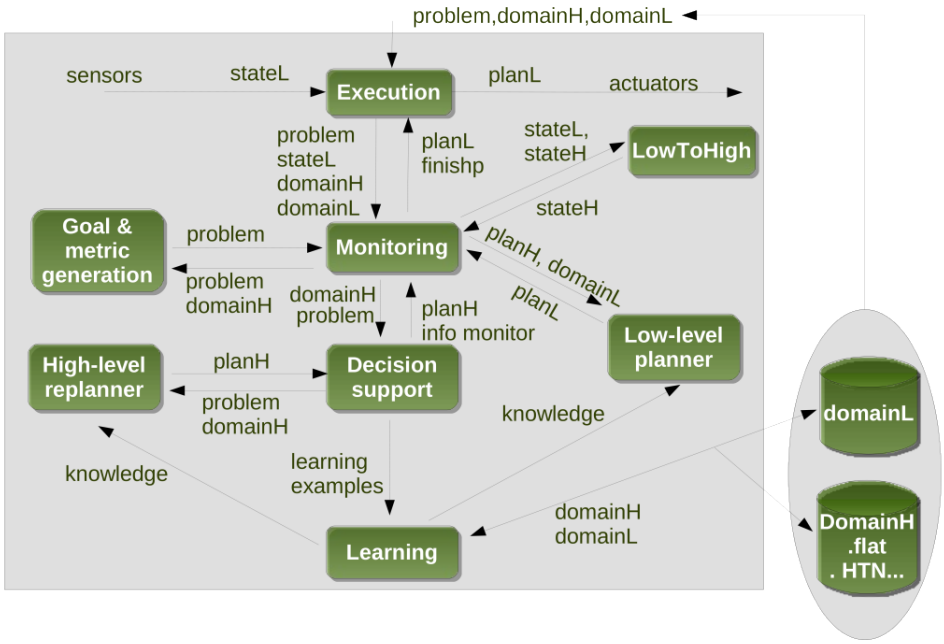

Figura 2.1: Arquitectura PELEA.

PELEA está concebida como una arquitectura de planificación de propósito general e independiente del dominio con capacidades de ejecución, monitorización, planificación, replanificación, reparación y aprendizaje de forma integrada. PELEA diferencia entre dos tipos de razonamiento, uno más reactivo (bajo nivel) y otro más deliberativo (alto nivel) para dominios temporales y no temporales.

Como se muestra en la figura 2.1 la arquitectura está diseñada de forma modular, donde cada módulo se encarga de una tarea es-

[7]Planning, Execution and LEarning Architecture
[8]Universitat Politècnica de Valencia
[9]Universidad Carlos III de Madrid

pecífica. De esta forma se consigue una gran independencia entre las funcionalidades de cada fase del proceso de planificación y ejecución facilitando el desarrollo y sustitución de nuevas implementaciones.

El módulo *Execution* es el encargado de recibir el dominio y problema a resolver, hacer la captura del estado del mundo mediante sensores y enviar las acciones que se tienen que ejecutar en cada instante a los actuadores.

El módulo *Monitoring* comprueba la corrección del estado del mundo esperado y la consecución de los objetivos del problema.

El módulo *Decision support* selecciona las variables que hay que monitorizar, toma la decisión de reparar o replanificar cuando se detecta un error en la monitorización y proporciona la información de entrada para el módulo *Learning*.

El módulo *High-level replanner* genera y/o repara planes de alto nivel. La modularidad de PELEA hace posible la independencia del planificador, pudiendo usar planificadores actuales como LPG-TD, SG-PLAN, CRICKEY, TFD, etc., independientemente, siempre que se utilicen las APIs definidas para la comunicación entre módulos.

El módulo *Goal&metric generation* selecciona automáticamente el nuevo conjunto de objetivos y métricas a ser utilizadas para el estado actual de la ejecución.

El módulo *LowToHigh* traduce el estado del mundo de bajo nivel a una representación de alto nivel para su monitorización y obtención de planes.

El módulo *Low-level planner* genera acciones de bajo nivel para ser ejecutadas por el módulo *Execution*. Un ejemplo de acciones de bajo nivel puede ser las coordenadas de un robot o los grados de desplazamiento de un brazo robótico.

El módulo *Learning* infiere conocimiento a partir de un conjunto de entrenamiento que puede usarse para modificar el modelo del dominio de planificación o mejorar el proceso de planificación de forma heurística.

En una simulación simplificada, asumiendo que se trabaja únicamente a alto nivel, no se usa aprendizaje, ni selección de métricas y objetivos, ni se simulan fallos de ejecución de acciones. El flujo principal de funcionamiento sería: (1) una captación inicial del estado, dominio y problema, por el módulo *Execution*; (2) envío de la información pertinente a través del *Monitoring* y *Decision support* hasta el *High-level replanner* para la generación de un plan de alto nivel; (3) comunicación del plan al *Decision support* para generar las variables a monitorizar; (4) comunicación del plan y las variables a monitorizar al módulo *Monitoring*; (5) comunicación al *Execution* de las siguientes acciones del plan a ejecutar; (6) ejecución de las acciones mediante los actuadores; (7) sensorización del estado del mundo por el módulo *Execution*; y (8) monitorización del estado volviendo al paso 5 hasta que se alcancen los objetivos.

Las comunicaciones entre los módulos se realizan en formato XML y la descripción de los dominios y problemas de planificación pueden modelarse con el lenguaje PDDL o en un lenguaje para descripción de planificación jerárquica.

Tal y como se ha visto en la descripción de PELEA, se trata de una arquitectura de planificación monoagente, que trabaja con dominios de forma centralizada concentrando toda la información de un problema y el proceso completo de resolución y ejecución en un único agente. Esto supone varios problemas; en primer lugar, el conjunto de problemas abordables se reduce conforme crece la talla del problema. En la fase de planificación, al tratarse de un proceso de búsqueda sobre un espacio de soluciones, problemas con gran cantidad de información pueden causar que los planificadores sean incapaces de encontrar una solución factible o de un modo eficiente en caso de trabajar con sistemas de tiempo real. Otro problema es la ejecución, puesto que los problemas de alta complejidad congre-

gan multitud de elementos con diversas capacidades de actuación y no tiene mucho sentido atribuir todas esas capacidades a un único agente. En caso de subdividir un problema según las capacidades de cada agente y resolverlo de forma independiente por varias instancias o agentes PELEA, no se dispone de los mecanismos adecuados para la coordinación o combinación de las soluciones obtenidas por cada agente PELEA. Estos son dos de los principales problemas que se pretenden resolver mediante la extensión multiagente de PELEA en PlanInteraction.

PELEA constituye la base de planificación de la arquitectura PlanInteraction, siendo cada uno de los módulos de su arquitectura herramientas al servicio de los agentes de PlanInteraction. En el capítulo 3 se detallará la forma en que se han empleado dichos módulos.

2.2. Sistemas multiagente

No existe una definición universal del concepto de agente software, sin embargo, sí que hay un consenso sobre las características que debe poseer: *persistencia*, *autonomía*, *sociabilidad* y *reactividad*. Un agente está en **constante ejecución** en un determinado entorno en **contacto** con otros agentes y es capaz de **reaccionar** ante cambios en el entorno actuando según su **razonamiento**.

Un agente es social tanto en cuanto es capaz de comunicarse con otros agentes y colaborar con ellos en la realización de alguna tarea. La componente social de un agente comprende también otras características como reputación, confianza, negociación, argumentación, etc.

La comunicación entre agentes permite la solicitud o contratación de servicios. Esto está relacionado con la autonomía del agente, puesto que a un agente se le puede decir “qué“ hacer, pero no ”cómo“ hacerlo. El agente debe tener unos métodos de razonamiento propios que le permitan resolver los procesos en los que se vea envuelto. La autonomía implica también una cierta proactividad o

capacidad para marcarse sus propios objetivos y tomar decisiones para alcanzarlos.

Un MAS es pues el entorno compartido por un conjunto de agentes donde interaccionan para la resolución de algún conjunto de tareas, ya sean compartidas o privadas. Estos sistemas a su vez deben poseer una serie de características tales como autonomía de los agentes, distribución de la información y descentralización.

Los MAS se emplean para resolver tareas que serían demasiado complejas de llevar a cabo en un sistema monolítico, bien sea por las técnicas empleadas o por la ingente cantidad de información que se maneja. Es por ello que la especialización de agentes en determinadas tareas y la distribución de la información relevante para cada una de ellas simplifica la resolución del problema en conjunto. Esta distribución de la información conlleva una visión parcial de la información por parte de cada agente. En determinados casos la comunicación entre los agentes será necesaria para la resolución del problema debido a su visión parcial. Un sistema será descentralizado siempre que no exista un agente coordinador que tome decisiones por el conjunto de los agentes involucrados.

2.2.1. Plataformas multiagente

A continuación se presentan algunas de las plataformas multiagente existentes, así como sus características más representativas, para terminar introduciendo Magentix2 que es la plataforma que se ha usado como base de PlanInteraction.

AgentScape

AgentScape es una capa *middleware* que soporta MAS de gran escala desarrollado por la universidad tecnológica de Delft [2]. El fundamento tras las decisiones de diseño son: (i) proporcionar una plataforma para MAS de gran escala, (ii) soportar múltiples códigos y sistemas operativos, y (iii) la interoperabilidad con otras plataformas de agentes. La filosofía de diseño general es “menos es más”, es

decir, el *middleware* AgentScape debe proporcionar un apoyo mínimo pero suficiente para aplicaciones de agentes, y "una misma talla no sirve para todos", es decir, el *middleware* debe ser adaptable o reconfigurable de tal manera que se puede ajustar a una aplicación específica (clase) o sistema operativo/plataforma hardware. Los agentes y los objetos son entidades básicas en AgentScape. Una localización es un "lugar" en el que los agentes y los objetos pueden residir. Los agentes son entidades activas en AgentScape que interactúan entre sí mediante la comunicación de paso de mensajes. Además, la migración de agentes es compatible. Los objetos son entidades pasivas que sólo se emplean en cálculos de forma reactiva por iniciativa de un agente. Además de los agentes, objetos y localizaciones, el modelo también define servicios. Los servicios proporcionar información o actividades en favor de los agentes o el *middleware* AgentScape.

JADE

JADE[10] es un entorno software de desarrollo de aplicaciones de agentes conforme a las especificaciones FIPA para sistemas inteligentes multiagente interoperables [4]. El objetivo es simplificar el desarrollo asegurando el cumplimiento de los estándares mediante un conjunto de servicios y agentes del sistema. JADE implementa una plataforma de agentes y un entorno de desarrollo. La configuración puede cambiarse incluso en tiempo de ejecución moviendo agentes de un equipo a otro, cómo y cuándo se requiera. JADE se ocupa de todos los aspectos que no son propios de la configuración de los agentes y que son independientes de las aplicaciones, tales como transporte de mensajes, codificación y traducción, o el ciclo de vida del agente. JADE está implementado completamente en Java y el requisito mínimo de sistema es la versión 1.4 de Java (la máquina virtual o el JDK). La última versión es la 4.3.0 liberada el 29/03/2013.

[10] Java Agent DEvelopment Framework

JASON

Jason [5] es un intérprete para una versión ampliada del lenguaje de definición de agentes AgentSpeak [18]. Implementa la semántica operacional de ese lenguaje y proporciona una plataforma para el desarrollo de MAS, incluyendo muchas características personalizables por el usuario. Jason es de código abierto y se distribuye bajo licencia GNU LGPL. Soporta asunciones de mundo abierto y cerrado; anotaciones en creencias y planes; meta-eventos, anotaciones declarativas de objetivos, organizaciones usando el modelo Moise+; funciones de confianza; un IDE en forma de plugin de Eclipse; etc.

EVE

EVE es una plataforma de agentes polivalente basada en web [3]. El objetivo del proyecto EVE es desarrollar un protocolo abierto de comunicación entre agentes software. EVE está diseñado como un sistema escalable descentralizado para agentes autónomos. EVE usa internet como plataforma y se sirve de protocolos de intercambio de datos (HTTP) y mensajería (JSON-RPC) existentes. Cada agente se identifica y es accesible a través de una *url* única pudiendo actuar como capa superior de aplicaciones conectándolas al entorno compartido entre los agentes del sistema. La plataforma EVE la ha desarrollado Almende, una empresa de Rotterdam especializada en tecnologías de la información y comunicación.

MAGENTIX2

Magentix2 es una plataforma de agentes para Sistemas multiagente abiertos [1, 11]. Su objetivo principal es llevar la tecnología de agentes a dominios reales tales como negocios, industria, logística, comercio electrónico, etc.

Magentix2 da soporte a tres niveles: organizacional, interacción y agente. El nivel organizacional hace referencia al conjunto de tecnologías relacionadas con sociedades de agentes. El nivel de interacción se ocupa de las comunicaciones entre los agentes. Y el último nivel comprende las técnicas de agentes tales como razonamiento y aprendizaje.

Magentix2 ofrece funciones de seguridad y confianza, un sistema de trazas, integración con THOMAS para la oferta de servicios y gestión de organizaciones, gestor de comunicaciones Apache Qpid, una API de argumentación y APIs de desarrollo para agentes conversacionales.

A la fecha del presente trabajo se han explotado las capacidades de Magentix2 para la gestión de las comunicaciones y para el desarrollo e implementación de agentes guiados por conversaciones. En el capítulo 3 se proporcionarán más detalles sobre estos aspectos.

2.3. Sistemas de planificación multiagente

Los MAPS se pueden entender como entornos CDPS[11]. La diferencia entre los CDPS y los MAS es el carácter cooperativo de los agentes frente a un comportamiento más autointeresado. En un CDPS los agentes tienen, por lo general, una serie de objetivos comunes y por tanto tienen una mayor predisposición para ayudarse unos a otros sin recibir ningún tipo de compensación.

A continuación se introducen los tres MAPS más relevantes actualmente: *GPGP/TÆMS*, *RETSINA* y *MPA*. A pesar de estar destinados a cubrir la misma necesidad, cada uno está desarrollado desde una perspectiva diferente. PlanInteraction comparte algunas de sus características como modularidad, generalidad, adaptabilidad, ejecución, etc., y añade otras propias de los MAS que se expondrán en el capítulo 3.

2.3.1. GPGP/TÆMS

GPGP[12] y su representación asociada como red de tareas jerárquica TÆMS[13] [8, 13, 24] se desarrollaron como entornos independientes del dominio para la coordinación online de actividades de tiempo real de pequeños equipos de agentes cooperativos que traba-

[11] Cooperative Distributed Problem Solving
[12] Generalized Partial Global Planning
[13] Task Analysis, Environment Modeling and Simulation

jan para alcanzar una serie de objetivos de alto nivel. GPGP proporciona un nivel de abstracción para evaluar algoritmos de control centralizados, paralelos y distribuidos, estrategias de negociación y diseños organizacionales.

TÆMS provee una meta-estructura para la función de transición de estados del agente que se divide en cuatro partes: control, recopilación de información, comunicación y ejecución de métodos. Primero los mecanismos de control establecen acciones de las otras tres partes y posteriormente se ejecutan secuencialmente, tras lo cual se repite el ciclo de meta-estados. Las acciones de recopilación de información preestablecidas incluyen acceso a nuevas estructuras de tareas subjetivas recibidas y descubrir relaciones de coordinación con tareas de otros agentes. Las acciones de comunicación preestablecidas incluyen transmitir el resultado de la ejecución de una tarea, partes de la estructura de la tarea y comunicaciones de meta-nivel.

En TÆMS las tareas se pueden descomponer en subtareas formando una jerarquía con un nodo raíz llamado grupo de tareas. Pueden existir múltiples grupos de tareas al mismo tiempo. Las otras relaciones entre tareas son activadores, facilitadores y obstaculizadores. Para evaluar la ejecución de un conjunto de tareas se tienen en cuenta el tiempo transcurrido y la calidad de la ejecución de las tareas. Un agente tiene creencias, lo que significa que cree la parte de la estructura global de la tarea que puede ver. Los agentes se pueden comprometer con otros agentes en la realización de tareas.

Cada agente tiene un scheduler local que organiza los recursos computacionales del agente, es decir, determina qué tareas del agente se van a realizar y cuándo. El propósito de los mecanismos de coordinación es asegurar que el scheduler local dispone de la mejor entrada posible que le permita construir un programa de gran utilidad. En otras palabras, los mecanismos de coordinación se usan para permitir a un agente hacer un buen plan.

2.3.2. RETSINA

RETSINA es un sistema multiagente abierto que provee infraestructura para diferentes tipos de agentes deliberativos dirigidos por objetivos [16, 21]. En este sentido la arquitectura RETSINA presenta algunas de las ideas de los agentes BDI[14]. Los agentes de RETSINA están compuestos por cuatro módulos funcionales autónomos: un módulo de comunicación, un planificador, un scheduler y un monitor de la ejecución. Las comunicaciones utilizan el formato KQML. Cada uno de los módulos está implementado como un hilo de ejecución para permitir la concurrencia. Además las acciones de los planes generados se ejecutan también como hilos independientes de forma concurrente.

Los agentes disponen de tres bases de datos principales: de objetivos, de tareas y de creencias. La base de datos de objetivos está implementada como una cola de prioridad. Los objetivos más prioritarios se resuelven primero. El módulo de comunicaciones o el planificador pueden añadir nuevos objetivos a la cola cuando objetivos complejos se descomponen en otros más simples. La base de datos de tareas contiene las acciones de un plan. El módulo planificador genera las tareas y el scheduler las retira cuando están listas para ejecución. Y la base de datos de creencias almacena el conocimiento del agente sobre el dominio en que se va a ejecutar el plan. El planificador consulta las creencias durante la planificación como fuente de hechos que afectan a sus decisiones de planificación. Las acciones pueden cambiar las creencias cambiando hechos del dominio.

2.3.3. MPA

MPA[15] es una arquitectura diseñada específicamente para integrar diversas tecnologías que permitan resolver problemas complejos de planificación, problemas que requieren ser resueltos de forma coordinada por un conjunto diverso de expertos [26]. MPA dispone de protocolos de comunicación para la compartición de información entre los agentes. MPA es capaz de generar multiples planes alterna-

[14]Believes, Desires and Intentions.
[15]Multiagent Planning Architecture

tivos en paralelo. Los agentes comparten interfaces de especificación uniformes bien definidas de forma que se pueden explorar diferentes maneras de reconfigurar, reimplementar y añadir nuevas capacidades al sistema de planificación.

Los agentes de planificación surgen de la descomposición de sistemas de planificación y scheduling en módulos independientes, siendo cada uno de estos módulos un agente. Para facilitar la construcción de los agentes a partir de los sistemas heredados se proporcionan librerías en C y Lisp.

La definición de unidades organizacionales entre los agentes permite explorar políticas de control flexibles en la generación de planes. Se definen *celdas básicas* y *meta-celdas* de planificación, donde las celdas básicas proporcionan soluciones de forma secuencial, mientras que las meta-celdas usan las celdas básicas para la generación de soluciones cualitativamente diferentes de forma concurrente. Las meta-celdas de planificación permiten explorar de forma rápida el espacio de soluciones de un determinado problema de planificación. Los meta-agentes de planificación (agentes que controlan otros agentes) definen diferentes estrategias de control sobre los agentes de planificación a su cargo.

MPA se ha usado en el desarrollo de varios sistemas de resolución de problemas de gran escala para el dominio de planificación de campañas aéreas[16].

[16] Air Campaign Planning (ACP).

Capítulo 3

PlanInteraction

En este capítulo se detalla el diseño de la arquitectura PlanInteraction y la filosofía de desarrollo empleada desde la visión más general de la arquitectura hasta los elementos más concretos. En primer lugar se ofrece una perspectiva global de los componentes de la arquitectura. Seguidamente se introduce el modelo de agente empleado y los distintos componentes que lo conforman, así como las distintas topologías de agentes que se proporcionan con la arquitectura. A continuación se explica cómo se han resuelto otras problemáticas típicas de los MAS como son las comunicaciones, sincronización y coordinación entre agentes. Y por último se comenta brevemente un prototipo de interfaz gráfica desarrollada para el uso de la plataforma.

3.1. Arquitectura

Lo primero que se debe tener en cuenta para entender la aproximación de planificación multiagente adoptada en PlanInteraction es su origen en los poyectos predecesores, la arquitectura PELEA y la plataforma Magentix2 expuestos en el capítulo 2. La idea inicial es adaptar un sistema de planificación monoagente de próposito general a un sistema de planificación multiagente.

El proceso de adaptación de un sistema monoagente a uno multiagente conlleva introducir toda una serie de elementos para permitir la coexistencia de varios agentes en un mismo entorno y su interope-

rabilidad. Primero hay que disponer de un canal de comunicaciones, independientemente de los protocolos u ontologías que se utilicen, para el intercambio de información entre los agentes. Por otro lado se requieren sistemas de control para garantizar la unicidad de agentes en el sistema, su correcto funcionamiento y el suministro de información del sistema a los agentes emergentes. En segundo lugar, se tienen que idear métodos de sincronización entre los distintos procesos que puedan estar ejecutando los agentes. Finalmente hay que determinar la constitución de los agentes como entidades sociales, proactivas y reactivas. Durante el presente capítulo se dará respuesta a cómo se han solucionado estas incógnitas.

Diseño

Desde la visión más general de la arquitectura, podemos entender PlanInteraction como un entorno en el que coexisten un conjunto de agentes con capacidades de planificación y/o ejecución, una serie de elementos de control, un simulador y una interfaz de interacción con el mundo real, tal y como se muestra el la figura 3.1.

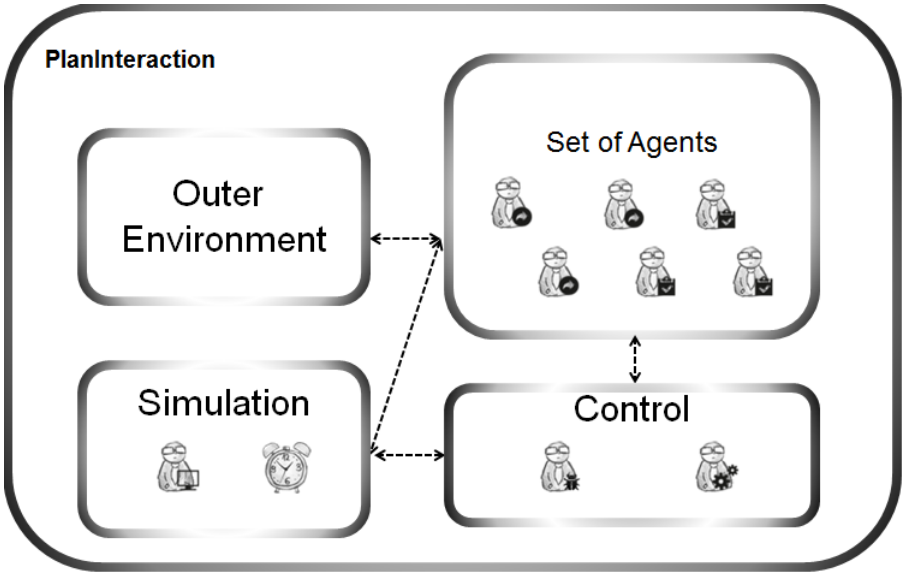

Figura 3.1: Arquitectura PlanInteraction.

A grandes rasgos, una posible instanciación completa de la arquitectura dispondría de (1) un agente de control que gestiona el registro en el sistema de los demás agentes, controlando también el momento de su finalización al cumplir sus objetivos; (2) un agente de simulación con capacidades para mantener una representación del mundo para un problema dado e interpretar y ejecutar sobre dicho mundo las acciones que reciba de los agentes de planificación, (3) un agente reloj que marca los pasos de ejecución para la simulación y (4) una serie de agentes de planificación/ejecución que reciben como entrada un determinado dominio y problema de planificación e interaccionan según su propio criterio para generar un plan de acciones y ejecutarlo sobre el simulador o el entorno real.

La arquitectura PlanInteraction permite crear múltiples configuraciones según el tipo de problema que se vaya a tratar o las técnicas que se quieran estudiar. La aproximación general que se ha adoptado para la resolución de los problemas de planificación supone la división de las tareas de planificación y ejecución, teniendo así un conjunto de agentes de planificación y un conjunto de agentes de ejecución como se muestra en la figura 3.2.

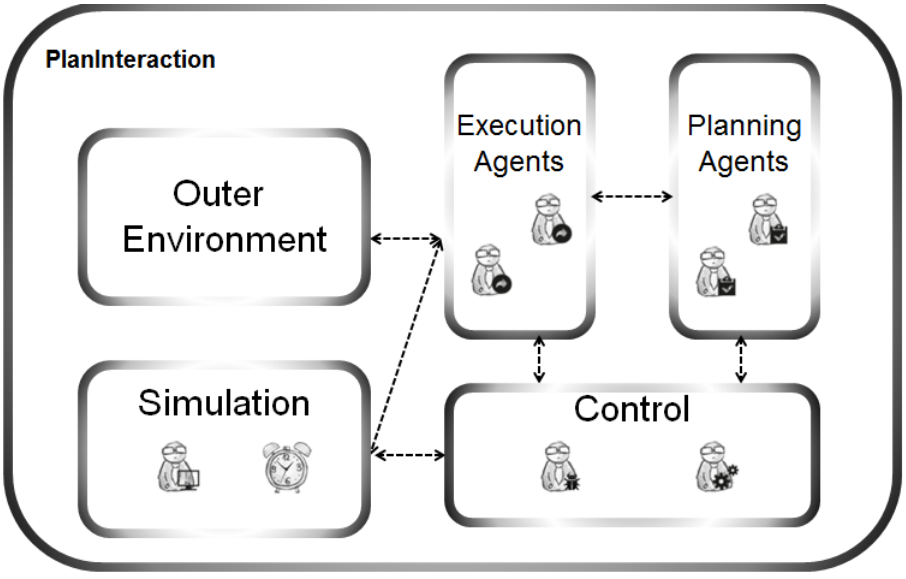

Figura 3.2: Arquitectura PlanInteraction diferenciando planificación y ejecución.

La división en dos conjuntos de agentes de ejecución y agentes de planificación hace posible independizar el desarrollo de nuevas técnicas de coordinación para cada uno de ellos. Así, pues, sería posible prescindir de la ejecución para centrarse únicamente en la planificación y la coordinación entre los agentes previa ejecución. O podrían probarse técnicas de coordinación y resolución de conflictos para agentes de ejecución que reciben un plan de acciones para ser ejecutado. En la sección 3.2 se definen de manera precisa la estructura y funcionalidades de estos agentes y en la sección 4 se verán varios casos de instanciaciones concretas probadas sobre la arquitectura.

Componentes de desarrollo

Como se muestra en la figura 3.3, la implementación de PlanInteraction dispone de un núcleo principal alrededor del cual se han desarrollado multitud de funcionalidades, todas ellas con la mayor sencillez y generalidad posibles para facilitar su utilización o futuras implementaciones y extensiones.

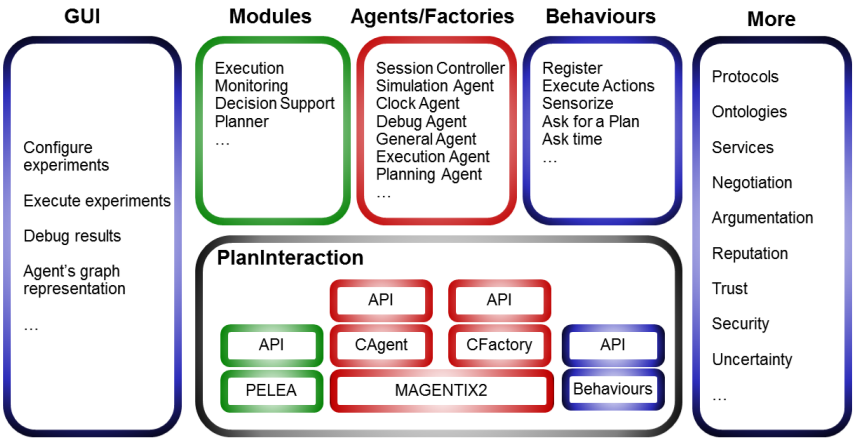

Figura 3.3: Componentes de desarrollo de PlanInteraction.

El núcleo principal de PlanInteraction está compuesto por las librerías de Magentix2 y PELEA, así como las APIs desarrolladas para simplificar su utilización. Adicionalmente se ha desarrollado para este conjunto de herramientas un modelo de especificación de comportamientos de los agentes, así como las APIs oportunas.

A partir de las librerías de PELEA y sus APIs se han desarrollado una serie de módulos que aportan las funcionalidades de planificación y ejecución a los agentes (caja "Modules" en la figura 3.3). Estos módulos se detallan en la sección 3.2.3.

Las librerías de Magentix2 y sus APIs aportan todo el sistema de comunicaciones y de especificación de agentes (caja "Agents/Factories" en la figura 3.3). En las secciones 3.2 y 3.2.1 se profundiza en ello.

Además de una serie de comportamientos necesarios para el control del sistema, para cada tarea típica del proceso de resolución de un problema de planificación, así como para las comunicaciones necesarias, se han desarrollado un conjunto de comportamientos que se expondrán en la sección 3.2.4 (caja "Behaviours" en la figura 3.3).

Todo lo anterior supone el núcleo principal de la arquitectura PlanInteraction, sin embargo se ha desarrollado una primera versión de interfaz de usuario para la configuración de experimentos, ejecución y depuración. En la sección 3.5 se puede ver en qué consiste exactamente (caja "GUI" en la figura 3.3).

Por último, aunque no se trata de elementos específicamente desarrollados para la arquitectura, se permite el desarrollo e incorporación de todo tipo de técnicas sociales relacionadas con la computación multiagente como: protocolos, servicios, negociación, confianza, etc (caja "More" en la figura 3.3).

3.2. Agentes

En la sección 2.2 se establecieron las características que debe tener un agente software para poder ser considerado como tal. A continuación se plantea el diseño que se ha utilizado para proporcionar esas característcas. En la figura 3.4 se puede apreciar la composición de un agente genérico de PlanInteraction (GA)[1]. Los GA están

[1]General Agent of PlanInteraction Architecture

compuestos principalmente por una capa de comunicaciones, una base de datos de conocimiento (KDB)[2], una base de datos de comportamientos (BDB)[3], una cola de prioridad de comportamientos (BPQ)[4] y un conjunto de módulos con funcionalidades de planificación/ejecución.

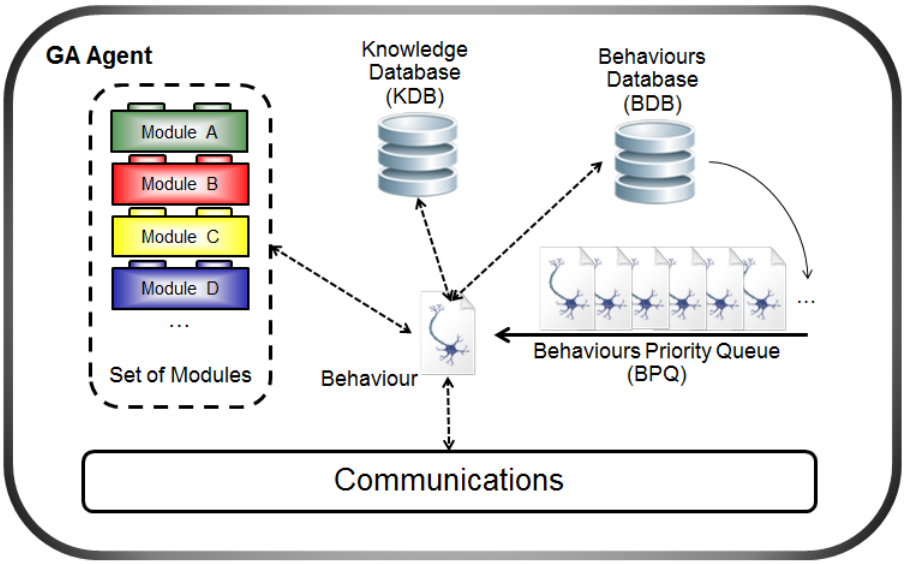

Figura 3.4: Composición de un agente GA.

Los GA están desarrollados sobre los agentes conversacionales proporcionados por Magentix2 (CAgents). Los CAgents ya implementan toda una serie de mecanismos de comunicación tales como protocolos, recepción y envío de mensajes y gestión de múltiples conversaciones simultáneas. Esta capa de comunicaciones es la que se muestra en la parte inferior de la estructura del GA en la figura 3.4. Otra de las características heredadas de los CAgents es el flujo de ejecución dirigido por las conversaciones, i.e., los agentes disponen de una serie de grafos de estados con capacidades de envío, recepción, espera y rechazo de mensajes (además de otros de propósito general), que permiten definir las divergencias o puntos de decisión sobre las acciones a realizar por el agente tras un evento comunicativo. Una última característica a tener en cuenta de los CAgents es que los agentes pueden interpretar dos roles en las conversacio-

[2]Knowledge DataBase.
[3]Behaviours DataBase
[4]Behaviours Priority Queue

nes: iniciador o participante. En la sección 3.2.1 se especifica más claramente en qué consiste esta representación en forma de grafo, así como las posibilidades que ofrece; las diferencias entre los roles de iniciador y participante y lo que conlleva el uso de un rol u otro; y algunos de los protocolos de comunicación desarrollados.

Cada GA dispone de una KDB interna. Esta base de datos almacena toda aquella información que el agente percibe del entorno, ya sea mediante sensorización o por comunicación con otros agentes. También puede almacenarse información auxiliar relativa a procesos internos del agente.

La BDB es una colección de los comportamientos que puede desempeñar un determinado agente. Los comportamientos consisten en secuencias de estados (como los definidos para las comunicaciones), con un propósito bien definido, a modo de tareas o procesos. Los comportamientos están relacionados con procesos cognitivos del agente. Un ejemplo de comportamiento podría ser la obtención de un plan, ya sea mediante generación propia o por solicitud a un agente del sistema. El agente dispone de una BPQ de donde se van seleccionando comportamientos para su ejecución consecutivamente. En la sección 3.2.4 se puede ver más información sobre los comportamientos.

Por último, cada agente posee una serie de **módulos** que le aportan funcionalidades de planificación/ejecución. Estos módulos se han desarrollado a partir de las APIs de PELEA. Los GA están diseñados de la forma más flexible posible, de tal manera que puedan incluir el subconjunto de módulos que el diseñador escoja. Las posibles combinaciones de módulos que contiene un agente da lugar a múltiples tipologías de agentes que se detallarán en la sección 3.2.5.

3.2.1. Comunicaciones

Como se ha comentado en la sección 3.2, la especificación y control de las comunicaciones viene heredado de los mecanismos proporcionados por Magentix2. El broker de comunicaciones que se utiliza es Apache Qpid y los mensajes intercambiados entre los agentes si-

guen el estándar FIPA-ACL.

Headers	Meta-info
ConvId	Identificador de la conversación
Performative	Performativa del mensaje
Language	Lenguaje del mensaje
Ontology	Ontología del mensaje
Protocol	Protocolo del mensaje
Sender	Identificador del agente emisor del mensaje
Receivers	Lista de identificadores de los receptores
Content	Contenido del mensaje

Tabla 3.1: Formato de mensaje FIPA-ACL.

El broker de comunicaciones establece una dirección única para cada agente a partir de su identificador de agente, es por ello que cada agente debe tener un identificador único. Los mensajes recibidos por el broker son redirigidos a la cola de mensajes del receptor correspondiente. Cada agente procesa los mensajes de su cola de mensajes de forma secuencial y asícrona.

Roles

Los GA son una extensión de los agentes conversacionales de Magentix2. Como se ha comentado previamente, estos agentes están guiados por las conversaciones, es decir, su ejecución se basa en la sucesión de diversos protocolos comunicativos y la toma de decisiones frente a la recepción de mensajes. Los agentes pueden interactuar en múltiples conversaciones simultáneamente y las conversaciones pueden incluir a más de dos agentes. Los agentes conversacionales, deben representar un rol en las conversaciones, deben ser **iniciadores** o **participantes**.

Un agente iniciador es el que envía el primer mensaje de una conversación al resto de agentes participantes de la conversación. Durante su ejecución, un agente puede cumplir los dos roles, tanto de iniciador como de participante, para distintas conversaciones. Para un determinado agente que cumpla el rol de participante, las conversaciones no se crean hasta que se recibe un mensaje.

Las conversaciones tienen un identificador generado a partir del identificador del agente iniciador y un número de secuencia. Cada conversación que mantiene un agente resulta en un hilo de ejecución diferente. Los mensajes intercambiados entre los agentes deben incorporar este identificador para que los agentes puedan redirigir los mensajes de la cola de mensajes a la conversación correspondiente.

Grafos de estados

Los protocolos de comunicación/ejecución que van a seguir los agentes se definen como grafos de estados. Magentix2 proporciona toda una serie de estados con funcionalidades predefinidas y las herramientas para su definición. Una de las aportaciones de PlanInteraction ha sido simplificar el proceso de definición de estos estados y transiciones mediante unas APIs de más alto nivel. Los tipos de estados permitidos se pueden consultar en la tabla 3.2.1

BEGIN	Estado inicial
FINAL	Estado final
ACTION	Estado de propósito general
WAIT	Estado de espera de recepción de mensajes (puede tener *timeout*)
RECEIVE	Estado de recepción y manejo de mensajes
SEND	Estado de construcción y envío de un mensaje
NAM[5]	Estado de absorción de mensajes

Tabla 3.2: Estados conversacionales de Magentix2.

En la definición de estados únicamente hace falta indicar el tipo de estado que se quiere definir, el método que va a realizar y, dependiendo del tipo de etado, una información de configuración. La información de configuración puede ser el *timeout* de un estado *WAIT* o filtros de aceptación de mensajes en los estados *RECEIVE*.

Para definir correctamente el flujo de ejecución de un protocolo de comunicación se han de establecer las transiciones entre los estados. Las transiciones son arcos unidireccionales entre un estado origen y un estado destino. Un estado puede tener múltiples arcos de entrada y de salida. Si un estado tiene varios arcos de salida, la

[5]Not Accepted Message

decisión de transición se toma en el método a ejecutar en dicho estado. Existen unas ciertas relaciones de precedencia predefinidas entre los estados. Los estados *RECEIVE* deben estar enlazados siempre a continuación de estados de tipo *WAIT*.

Grafo básico de un agente

Los GA interpretan tanto el rol de iniciador como el de participante e incorporan un grafo básico de estados (BSG)[6] conversacionales predefinido. Este grafo de estados se muestra en la figura 3.5:

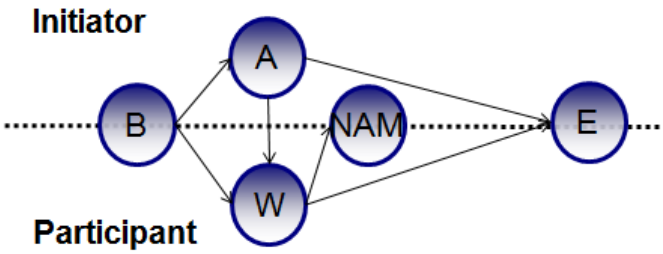

Figura 3.5: Grafo básico de un agente GA.

El BSG de un agente está compuesto por un estado inicial (B), un estado de propósito general (A), un estado de espera (W), un estado de absorción de mensajes (NAM) y un estado final (F). Cada vez que un agente crea una conversación, ya sea como iniciador o como participante, se crea una instancia de su grafo asociado. El comportamiento por defecto asociado al estado B de los GA es transitar al estado A o al estado W según el rol que juegue el agente. Si el agente es iniciador se transita al estado A, mientras que si es participante se transita al estado W. El estado A del BSG es un estado especial del agente en el que se decide el siguiente comportamiento a ejecutar, como se verá en la sección 3.2.4. El estado W es un estado de espera general que prepara al agente para la recepción de cualquier mensaje. De esta forma queda reflejada la división entre el funcionamiento proactivo del agente y el funcionamiento reactivo.

[6]Basic State Graph

3.2.2. Conocimiento

A lo largo de la vida de un agente, éste va adquiriendo información de toda índole que posteriormente le servirá para razonar y decidir sus acciones futuras. Esta información puede venir de tres fuentes principales: el agente la conoce desde un principio, la obtiene mediante sensorización o se la comunican otros agentes. Toda esta información debe ser procesada y almacenada de forma adecuada para su posterior recuperación y consulta. Es por ello que los agentes disponen de una KDB.

La KDB está relacionada con las creencias de un agente BDI[7]. Los agentes de las arquitecturas GPGP/TÆMS y RETSINA también disponen de una. Los GA implementan esta base de datos de forma sincronizada para evitar problemas de múltiples accesos simultáneos debido al paralelismo entre conversaciones.

La información contenida en la KDB se identifica como creencias en la medida en que se considera cierta. Esto es así debido a que los agentes se encuentran en entornos muy dinámicos que evolucionan con el paso del tiempo y es posible que un agente no disponga de información actualizada del mundo.

3.2.3. Módulos

Uno de los objetivos del proyecto PlanInteraction consiste en la adaptación de las técnicas de planificación monoagente de la arquitectura PELEA a un sistema multiagente, concretamente a uno integrado con Magentix2. Para realizar esta adaptación se han encapsulado los módulos de la arquitectura PELEA descritos en la sección 2.1.2 y se han desarrollado unas APIs de comunicación para su utilización. Concretamente, los módulos integrados actualmente en PlanInteraction son *Execution*, *Monitoring*, *Decision support* y *High-level replanner*. Aunque no es una parte integral de un agente PELEA, el simulador de ejecución de la arquitectura PELEA también se ha implementado como un módulo funcional de un GA.

[7]Believes, Desires, Intentions.

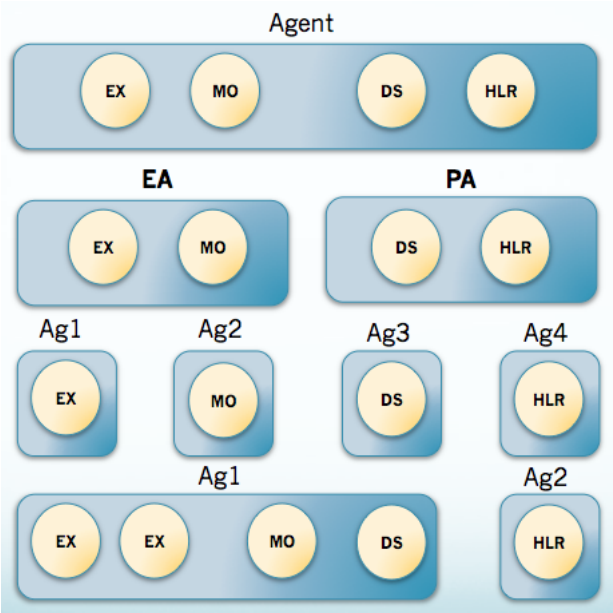

Figura 3.6: Configuraciones de agentes según módulos funcionales.

Los GA pueden incorporar el número de módulos funcionales que el diseñador estime conveniente. Esta flexibilidad en la composición de los agentes hace de la arquitectura PlanInteraction un entorno con multitud de configuraciones posibles a nivel de agente. Se puede definir desde un agente que incorpore todos los módulos, siendo una representación directa de un agente PELEA, hasta un agente que incorpore un único módulo, como agente especializado en una determinada tarea de planificación y/o ejecución, pasando por todas las posibilidades intermedias. La figura 3.6 ilustra estas posibilidades.

Un GA puede incorporar nuevos módulos funcionalidades que no estén directamente relacionadas con la planificación o ejecución. Para ello es preciso que el desarrollador extienda de un módulo genérico para implementar las nuevas funcionalidades.

Con independencia de la generalidad en la configuración de agentes, desde un punto de vista práctico se plantean dos configuraciones concretas, una orientada a ejecución y otra a planificación. Esta clasificación es la que se refleja en la figura 3.6 como los agentes EA[8]

[8]Execution Agent

y PA[9]. En la sección 3.2.5 se detallan más estos agentes y otros proporcionados por la arquitectura PlanInteraction.

3.2.4. Comportamientos

Como se ha comentado en las secciones previas, los GA tienen un grafo de estados de ejecución dirigido por las comunicaciones (véase 3.2.1), un conjunto de conocimiento que les permite razonar (véase 3.2.2) y un conjunto de módulos funcionales que les dota de capacidades de planificación o ejecución (véase 3.2.3). Sin embargo, no se ha comentado la forma en que se definen los procesos concretos que siguen los GA para interactuar, es decir, los comportamientos.

Un comportamiento es una rutina bien definida con unos objetivos determinados. Los GA disponen de una BDB que almacena todos los comportamientos que puede realizar el agente. La arquitectura PlanInteraction proporciona unas APIs de definición de comportamientos como grafos de estados. Los estados que se emplean para definir los comportamientos son los mismos que se especifican en 3.2.1.

Los comportamientos se pueden abstraer como cajas negras para las que hay que definir un estado de entrada, estados de salida y todos los estados intermedios, así como las transiciones entre ellos. Así mismo, se pueden definir comportamientos mediante anidamiento de subgrafos, haciendo todavía más flexible la definición de los mismos e independizando procesos internos a un comportamiento.

Desde cada estado de un comportamiento se puede acceder a la KDB del agente, así como a las funcionalidades que proporcionan los módulos de planificación/ejecución.

La ejecución de un determinado comportamiento puede estar motivado por una decisión propia del agente o por una reacción ante una interacción con otros agentes. De esta forma distinguimos entre **comportamientos proactivos** y **comportamientos reactivos**.

[9]Planning Agent

Los GA disponen de una BPQ donde se añaden los comportamientos proactivos que se van a ejecutar. Los métodos de razonamiento del agente son los que determinan en qué orden se tienen que ejecutar. Si el agente no implementa una lógica de decisión para la selección de comportamientos, la BPQ funciona según la técnica *First in, First out*. Una alternativa a este funcionamiento sería gestionar la BPQ conforme a un orden de prioridades preestablecido. Otra decisión a tener en cuenta en el diseño de un GA es la acción que se debe realizar cuando la BPQ queda vacía.

Los comportamientos reactivos se ejecutan de forma asíncrona bajo petición expresa de otros agentes. Normalmente son peticiones de información o realización de procesos a modo de servicios. La ejecución de un comportamiento reactivo puede provocar la incorporación de un comportamiento proactivo a la cola de comportamientos.

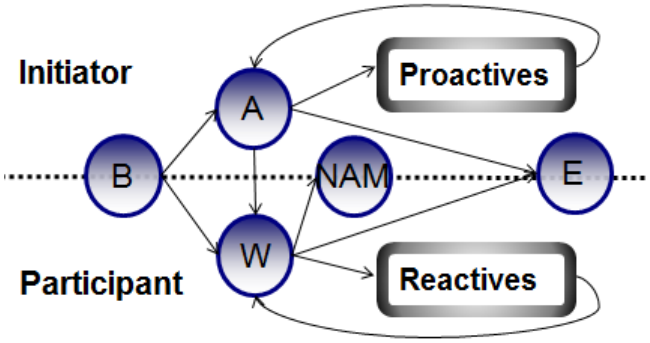

Figura 3.7: Grafo básico de un agente GA con comportamientos.

Los comportamientos se pueden definir de forma independiente a un agente concreto, favoreciendo la reusabilidad, modificación y extensión de los mismos. En la definición de los GA se debe indicar cuáles son los comportamientos concretos que incorpora. Cuando se añade un comportamiento al GA éste queda almacenado en la BDB y se enlaza de manera adecuada al BSG del agente. Los comportamientos proactivos se enlazan con el estado de propósito general (A-Action) del BSG y los comportamientos reactivos al estado de espera (W-Wait) del BSG. En el estado A es donde el agente tiene que decidir qué comportamiento de la cola de comportamientos

proactivos se va a ejecutar a continuación.

Los GA incorporan un conjunto de comportamientos predefinidos del sistema como: registro en el sistema, petición del tiempo del sistema, petición de alarmas, petición de información sobre los agentes en la simulación, etc. La figura 3.8 muestra el comportamiento proactivo de un agente para registrarse en el sistema y la figura 3.9 muestra el comportamiento reactivo del agente controlador de sesión. Este último agente se detalla en la sección 3.2.5.

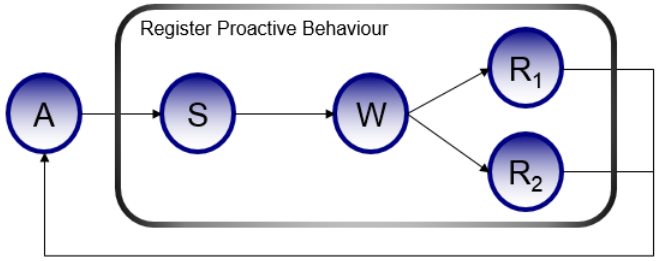

Figura 3.8: Comportamiento proactivo de registro.

Cuando un agente llega al sistema, el primer comportamiento que ejecuta proactivamente es el de registro. En el estado S se envía una petición de registro al agente controlador de sesión junto con la información característica del agente. Una vez enviada la petición, se transita a un estado de espera W. Los estados R_1 y R_2 son estados de recepción de mensajes, cada uno con un filtro especificado para tratar una respuesta diferente del agente controlador de sesión. Así, en el estado R_1 se trataría una aceptación del registro y se procesaría la información enviada por el agente controlador de sesión y en el estado R_2 se trataría un rechazo de registro en el sistema. Una vez se ha tratado la respuesta correspondiente, se transita de nuevo al estado A del BSG para decidir el comportamiento que se va a ejecutar a continuación.

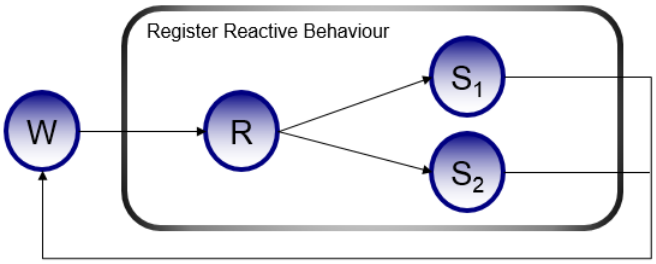

Figura 3.9: Comportamiento reactivo de registro.

El agente controlador de sesión crea una conversación como participante cada vez que recibe una petición de registro por parte de otro agente. En primer lugar se transita hasta el estado de espera W del BSG. La primera vez que se llega a este estado no se realiza espera puesto que se dispone de un mensaje en la cola de mensajes del agente. Así pues, se transita al estado de recepción que puede tratar dicho mensaje, esto es el estado R perteneciente al comportamiento reactivo de registro de agentes. En este estado se obtiene la información del agente que solicita el registro, se procesa y se decide si aceptar o rechazar. En caso de aceptar el registro se transita al estado de envío S_1, donde se construye un mensaje de aceptación añadiendo la información de los agentes que comparten sesión con el nuevo agente. En caso de rechazo se transita al estado de envío S_2 donde se envía un mensaje de rechazo. Finalmente se transita al estado W donde se mantendrá a la espera de una nueva petición.

3.2.5. Agentes predefinidos

La arquitectura PlanInteraction dispone de unos agentes predefinidos tanto para control del sistema como para llevar a cabo experimentos de planificación y ejecución. Estos agentes se han desarrollado como extensiones de un GA y pueden ser extendidos y modificados por los desarrolladores para otorgarles nuevas funcionalidades.

Agentes del sistema

El agente principal del sistema es el **agente controlador de sesión** (SCA)[10]. El SCA es el encargado de registrar a los agentes,

[10]Session Controller Agent

suministrar información sobre los agentes en la plataforma y enviar mensajes de finalización a los agentes. Los agentes tienen que registrarse con el SCA para que se les asigne un identificador de sesión y poder conocer a los agentes con los que comparten sesión. Una sesión viene identificada por el dominio y problema de planificación que se va a solucionar, de esta forma se pueden ejecutar distintas simulaciones en la plataforma sin que colisionen ni interaccionen los agentes y la información de cada simulación. Algunos agentes pueden ofrecer funcionalidades independientes del dominio y problema a solucionar, por lo que podrían intervenir en distintas sesiones simultáneamente.

El segundo agente más relevante del sistema es el **agente de simulación** (SA)[11]. Este agente es el encargado de mantener la representación del estado del mundo y aplicar sobre él las acciones de los agentes de ejecución. Incorpora la posibilidad de generar eventos exógenos para provocar fallos en las acciones de los agentes de ejecución, propiciando así comportamientos de replanificación o reparación en los agentes de planificación/ejecución.

El SA está íntimamente relacionado con el **agente de reloj** (ClkA)[12]. El ClkA proporciona el tiempo global del sistema, informa al SA tras cada ciclo de reloj para la actualización del paso de ejecución, y atiende peticiones de alarmas para pasos de ejecución futuros. Por cada sesión gestionada por el SCA existe un SA y un ClkA, siendo el SCA quien establece la frecuencia del ciclo de reloj para el ClkA.

El último agente del sistema que se proporciona con la arquitectura PlanInteraction es el **agente de depuración** (DbgA)[13]. Este agente se emplea para facilitar la depuración de la información relativa a cada agente y las interacciones producidas durante la resolución de un problema de planificación. El DbgA es creado por el SCA cuando este último se instancia en modo depuración. Cuando un agente se inicializa en modo de depuración, todos los estados

[11] Simulation Agent
[12] Clock Agent
[13] Debug Agent

de los comportamientos que incoropora se encapsulan en estados de tipo *SEND*, de esta forma se puede enviar la información que nos interese monitorizar del agente en cada uno de sus estados. El DbgA recibe los mensajes de depuración de los demás agentes, los procesa y los guarda en un fichero de depuración para posteriormente ser interpretado. El contenido del fichero de depuración se almacena en XML para facilitar su procesamiento.

Agentes de planificación y ejecución

Como se indica en la sección 3.2.3, se ha optado por una división de las tareas de planificación y ejecución estableciendo de esta forma dos tipos de agentes: un **agente de planificación** (PA)[14] y un **agente de ejecución** (EA)[15].

El EA es un agente con razonamiento reactivo, trabaja en un nivel más cercano al mundo y por tanto sus respuestas tienen que ser más rápidas. Entre los módulos funcionales que incorpora se cuentan el módulo *Execution* y el módulo *Monitoring* de PELEA. Los EA se identifican más fácilmente con elementos del mundo real puesto que son los que pueden modificar el mundo mediante sus acciones. Los EA son los encargados de cargar los ficheros donde se especifican el dominio y problema de planificación a resolver. Cada EA tiene sus propios ficheros de dominio y problema. En el fichero de dominio se especifican sus capacidades mientras que en el fichero de problema se especifican su conocimiento inicial (estado actual del mundo) y sus objetivos. Puesto que no disponen de capacidades de planificación, deben comunicarse con un PA para obtener un plan que posteriormente puedan ejecutar.

El PA es un agente deliberativo, utiliza técnicas heurísticas de búsqueda para encontrar soluciones factibles a un problema de planificación dado un estado del mundo. Este tipo de agente incorpora los módulos *Decision support* y *High-level replanner* de PELEA. Una de las características más relevantes de estos agentes es la capacidad heredada de PELEA de incorporar el planificador que se

[14] Planning Agent
[15] Execution Agent

desee. Estos agentes funcionan a modo de proveedores de servicios ofreciendo sus capacidades de planificación a los EA.

En los experimentos llevados a cabo se han implementado otros agentes con funcionalidades extendidas a partir de los agentes PA y EA. En el capítulo 4 se detalla de qué forma se han adaptado estos agentes para casos concretos.

3.3. Sincronización e Interacción

Los agentes son entidades autónomas, proactivas y reactivas, por tanto, pueden realizar acciones en su entorno en cualquier momento, ya sea por decisión propia o como respuesta a la interacción con otro agente (véase 3.2.4). La proactividad de los agentes se define mediante los comportamientos que ejecutan con el rol de iniciador (véase 3.2.1). La reactividad, por otro lado, se refleja en los comportamientos que se ejecutan con el rol de participante.

Los agentes pueden involucrarse en procesos que requieran distintos protocolos de interacción, desde una simple petición a otro agente para la realización de una acción (*Request Interaction Protocol*), hasta solicitud de propuestas para la realización de una acción bajo determinadas condiciones (*Contract Net*) o diversos tipos de subastas (*English Auction, Dutch Acution, etc.*).

La importancia de los protocolos de interacción estriba en la sincronización para la compartición, procesamiento y análisis de información, así como la ejecución conjunta entre los agentes. Los comportamientos de un agente pueden estar diseñados a partir de un determinado protocolo de interacción. En tal caso, las acciones del protocolo donde interviene el iniciador constituyen un comportamiento proactivo y las diferentes actuaciones de los participantes constituyen comportamientos reactivos.

Diseñar los protocolos de interacción de esta forma facilita la sincronización entre los agentes, limitándola únicamente a gestionar las respuestas recibidas por el iniciador. Esto es así debido a

que un agente sólo puede ejecutar un hilo como iniciador, lo que implica que solo puede manterner una conversación como iniciador simultáneamente con el resto de agentes. Por tanto, todos los mensajes dirigidos a dicha conversación se tratarán en la medida en que el protocolo de interacción, representado como grafo de estados, contemple estados *WAIT* y *RECEIVE* para gestionar los mensajes de forma secuencial. Si los protocolos de los agentes no están bien diseñados puede ocurrir que un mensaje que llega a la cola de mensajes se procese en un punto del protocolo que no tiene sentido teniendo como resultado un funcionamiento inesperado y/o erróneo.

Si las intervenciones de los participantes son simples, i.e., tan solo se comunican con el iniciador, no supone ningún problema de sincronización puesto que siempre están preparados para servir peticiones. Tan solo pueden surgir problemas de sincronización en protocolos en los que el agente participante deba comunicarse con otros participantes, con lo cual pasaría a cumplir el rol de iniciador en otra conversación.

La arquitectura PlanInteraction permite la participación de más de un iniciador en la misma conversación, pero se desaconseja el uso de este diseño debido a que aumenta considerablemente la cantidad de sincronizaciones necesarias en los protocolos de interacción de los agentes. Un ejemplo de sincronización, utilizando un protocolo de tipo *paso de testigo*, se muestra en la figura 3.10. En este ejemplo el que tiene el testigo hace una petición a los demás agentes de la conversación enviando un mensaje de tipo *broadcast*. A continuación espera la respuesta de cada agente y cuando recibe el último mensaje continúa con su ejecución. El resto de agentes que actúan como participantes reciben la petición, la procesan, responden con la información oportuna y quedan a la espera de una nueva petición.

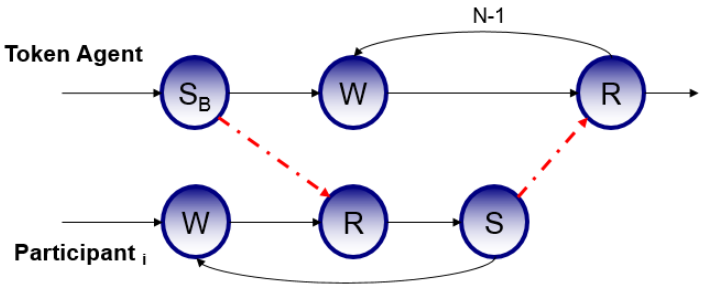

Figura 3.10: Sincronización simple de un broadcast de agente central.

En caso de que el resto de agentes tuviera el rol de iniciadores en lugar de participantes, requerirían de una segunda fase de sincronización, en la que el agente con el testigo confirma que ha recibido todos los mensajes y les despierta de su estado de espera para que continúen con su comportamiento. Este ejemplo se puede ver en la figura 3.11.

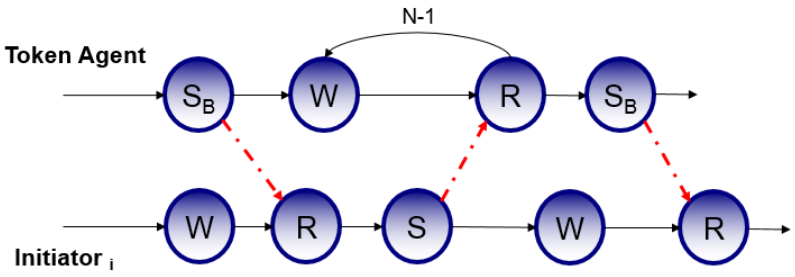

Figura 3.11: Sincronización doble de un broadcast de agente central.

3.4. Coordinación

Las interacciones entre los agentes de un MAS pueden estar motivadas por un interés común, dotando a los agentes de un perfil cooperativo, o por intereses privados que requieren de la participación de otros agentes para su consecución, es decir, un perfil más competitivo o simplemente colaborativo. En cualquier caso, la interacción entre estos agentes, además de sincronización de procesos (véase 3.3), requiere una coordinación entre los mismos.

La coordinación entre agentes pretende evitar conflictos entre las acciones que van a realizar los agentes en un espacio compartido. En el ámbito de los MAPS las acciones que realizarán los agentes vienen definidas por el plan generado para cada agente. Teniendo esto en cuenta, se pueden distinguir tres puntos principales de coordinación:

- *Pre-planning.* La coordinación entre los agentes se produce mediante algún proceso de reparto de objetivos previa generación del plan, i.e., los objetivos comunes de los agentes se reparten de tal manera que los planes generados para cada agente sean independientes.
- *During planning.* En este caso la coordinación se produce durante la elaboración del plan. Los agentes pueden construir el plan añadiendo las acciones de forma consensuada siguiendo algún protocolo de generación y aceptación de propuestas entre los agentes. Como resultado se obtiene un plan común para los agentes conteniendo las acciones que debe realizar cada uno de ellos.
- *Post-planning.* La coordinación después de la planificación procura resolver los posibles conflictos existentes entre planes, a priori independientes, de un conjunto de agentes mediante algún algoritmo de *plan merging*.

Cabe destacar que ninguno de los tres métodos presentados es mejor que los demás, sino que cada uno es adecuado para un determinado tipo de dominios y problemas de planificación teniendo en cuenta las características sociales de los agentes. Otra consideración a tener en cuenta es que la coordinación *pre-planning* no garantiza la ausencia de conflictos entre los planes generados posteriormente para cada agente. Por tanto, puede resultar interesante la combinación de métodos de coordinación como *pre-planning* y *post-planning*.

En PlanInteraction también se incluyen tareas de ejecución además de las tareas de planificación. En la ejecución de los planes es precisamente donde se pueden producir fallos debido a los conflictos entre las acciones de los agentes. Por ello, también podemos encontrar varios puntos de coordinación asociados a la ejecución:

- *Pre-execution.* La coordinación antes de la ejecución de una acción puede estar asociada a la reserva de recursos. Mediante la reserva de recursos los agentes saben antes de realizar una determinada acción si los recursos necesarios para llevarla a cabo están disponibles.
- *During execution.* En este punto se incluyen los casos en los que, por algún motivo, el agente no puede alcanzar el estado previsto mediante la realización de una acción y se solicita ayuda a otro agente.
- *Post-execution.* Este tipo de coordinación puede ser necesario en el caso de comunicados de finalización de acciones entre agentes, para garantizar la consecución de objetivos.

Las interacciones entre los agentes para la generación de los planes se realizan aplicando el mismo método de coordinación. Por tanto, los agentes que interactúan se encuentran en el mismo punto de procesamiento hasta el envío de los planes para ejecución. Los métodos de coordinación asociados a la ejecución hacen referencia a las medidas que puede adoptar cada agente antes, durante y después de la ejecución de cada una de sus acciones. Una vez se distribuyen los planes entre los agentes para su ejecución, cada agente sigue su propio proceso individual e iterativo sobre las acciones del plan, por lo que cada agente puede encontrarse en un punto de procesamiento diferente y aplicar una coordinación en ejecución diferente. Un ejemplo de ello sería la solicitud de ayuda de un agente A a un agente B para la realización de una acción de A. El agente A emplearía coordinación *during execution*, mientras que el agente B emplearía coordinación *post-execution* para comunicar a A que puede continuar con la ejecución de su acción.

3.5. Interfaz gráfica

Otra de las aportaciones realizadas al proyecto PlanInteraction es la implementación de un prototipo de interfaz gráfica para la configuración, ejecución y depuración de experimentos haciendo uso de las APIs de la arquitectura PlanInteraction. La interfaz gráfica se ha desarrollado mediante la API Swing de Java.

Configuración de simulaciones

Como se comentó en 2.1.1 y 3.2.5, los agentes de ejecución (EA) son los encargados de leer los ficheros de dominio y problema que van a tratar. En la configuración de un experimento hay que indicar cuántos EA van a intervenir y los ficheros de dominio y problema asociados a cada agente. Del mismo modo hay que indicar los agentes de planificación (PA) que van a dar soporte de planificación y el tipo de planificador que van a usar. La interfaz gráfica se encarga de lanzar la ejecución del agente controlador de sesión (SCA), el agente de simulación (SA) y el agente de reloj (ClkA) necesarios.

Una de las ventajas de la interfaz gráfica radica en la facilidad para definir y reconfigurar experimentos. Se puede modificar el número y el tipo de los agentes, las relaciones prestablecidas entre ellos, así como otras características individuales de cada agente. Otra posibilidad es la recuperación de experimentos preconfigurados que se hubieran guardado en el sistema.

Depuración de resultados

El análisis de resultados y la depuración de la ejecución de planificadores y MAS son tareas complejas y tediosas en la mayoría de los casos. Los planificadores habitualmente actúan como cajas negras que reciben un problema a resolver y devuelven un plan como solución. Los planes para un determinado problema pueden constar de innumerables acciones ordenadas temporalmente o causalmente. Para poder llevar a cabo un análisis de los resultados, rápido y sencillo, es importante ofrecer esta información de la forma más intuitiva posible. En el caso de los MAS las dificultades se presentan en el análisis de las interacciones entre los agentes. Esto incluye el

intercambio de mensajes y cómo dicho intercambio influye en la obtención de resultados.

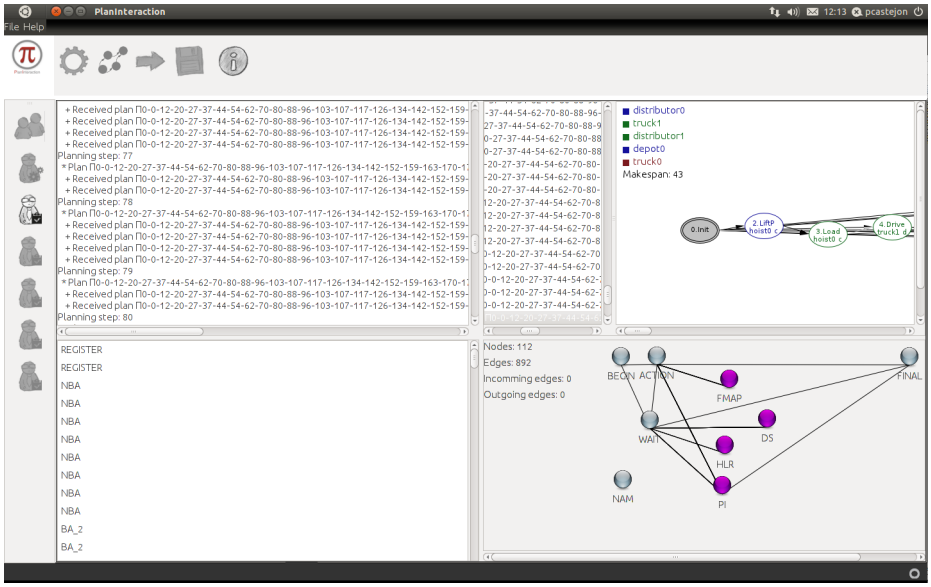

Figura 3.12: Interfaz gráfica.

La figura 3.12 muestra algunas de las opciones de análisis y depuración que se ofrecen con la interfáz gráfica. En el cuadro superior izquierda se muestran los planes parciales construidos de forma conjunta entre los agentes durante una simulación. Arriba a la derecha se muestra el plan seleccionado a modo de grafo. Cada nodo del grafo representa una acción del plan y el color determina el agente que la tiene que llevar a cabo. El cuadro inferior izquierda lista los mensajes intercambiados por el agente. Estos mensajes se pueden seleccionar para mostrar una ventana con información más detallada sobre la interacción. Por último, el cuadro inferior derecha es la representación del agente como un grafo de estados, es decir, muestra los comportamientos que contiene el agente.

Estas visualizaciones, unidas a información de estado de los agentes durante la ejecución, permiten llevar a cabo la depuración de la experimentación. Se puede consultar el estado del conocimiento de cada agente en cada paso de ejecución, el punto concreto de un comportamiento en el que se encuentra y los mensajes intercambiados con otros agentes paso a paso.

Capítulo 4

Ejemplos de implementación

En el presente capítulo se exponen algunos ejemplos concretos que se han implementado e integrado en la arquitectura PlanInteraction. Como se comentó en la sección 1 estos ejemplos de aplicación persiguen validar la arquitectura, así como desarrollar investigaciones paralelas sobre planificación multiagente.

La validación de la arquitectura tiene en cuenta tres aspectos: configuración de agentes, gestión de comportamientos y actos comunicativos. La configuración de agentes varía según el número de agentes, sus módulos funcionales y las relaciones existentes entre ellos. La gestión de comportamientos cubre la propia especificación de los comportamientos, su asignación a los agentes y la incorporación, selección y extracción de la cola de comportamientos de los agentes. Por último, los actos comunicativos deben asegurar el correcto envío y recepción de todos los mensajes y las sincronizaciones entre los agentes.

Típicamente, la planificación clásica ha trabajado con problemas mono-agente de modo que la mayoría de las colecciones de problemas de planificación son problemas definidos para un único agente. Para convertir un problema mono-agente a una especificación multiagente se requiere generar un fichero para cada agente con sus capacidades concretas y su conocimiento parcial sobre el mundo. En el

dominio se especifica las acciones permitidas y en el problema se especifica el estado inicial del mundo y los objetivos a alcanzar. Estos ficheros personalizados para cada agente son los que se suministran a cada uno de los EA.

Los ejemplos de aplicación que se presentan a continuación están diseñados para el desarrollo de técnicas de planificación multiagente. En la sección 4.1 se plantea un ejemplo de coordinación *during planning* como se planteó en la sección 3.4, para dominios clásicos de planificación usando el planificador *FMAP*. En la sección 4.2 se presenta otra aproximación a la coordinación *during planning* para un dominio médico real. Por último, en 4.3 se presenta un ejemplo de coordinación a nivel de ejecución en el dominio de los rovers, los robots de exploración de la *NASA*.

4.1. Planificación multiagente cooperativa

Una de las aproximaciones estudiadas en planificación multiagente asume dominios totalmente cooperativos. Esta asunción implica que los agentes persiguen un objetivo común sin tener en cuenta objetivos privados ni preferencias personales. Al no existir intereses particulares todos los agentes utilizan la misma función de utilidad sobre las acciones del plan conjunto. Sin embargo, es posible que los agentes dispongan de información privada sobre el estado del mundo y, por tanto, sea imprescindible la comunicación para la resolución del problema.

El planificador *FMAP* (Forward MAP) está diseñado para resolver problemas en un contexto cooperativo y es una versión mejorada de un planificador previo que funciona con un tipo de razonamiento regresivo o hacia atrás [22, 23]. *FMAP* consta de tres partes fundamentales: (1) captación del dominio y problema, (2) preproceso de información y (3) algoritmo de construcción del plan. El preproceso, a su vez, se puede dividir en tareas de *grounding* y compartición de grafos de información. El algoritmo de construcción del plan consiste en un proceso iterativo en el que los agentes proponen acciones que refinan un plan dado y consiguen los objetivos pendientes del plan.

FMAP puede funcionar tanto en entornos monoagente como en entornos multiagente. La diferencia entre unos y otros radica en la inclusión de actos comunicativos en el preproceso y la construcción del plan. Se ha llevado a cabo una adaptación de *FMAP* independizando estos actos comunicativos del resto del flujo de ejecución. De esta forma, se dispone del planificador *FMAP* como un módulo funcional de planificación y los actos comunicativos como parte de los comportamientos del agente en el proceso de planificación.

En la figura 4.1 se muestra un posible escenario en el que dos agentes (*Ag1* y *Ag2*) asumen el rol de agencias de transporte y un tercer agente (*Ag3*) gestiona el almacén. Los agentes de transporte reparten paquetes a través de una red de ciudades. Los paquetes pueden ser tanto materias primas como productos finales. El agente del almacén está al cargo del almacenaje y suministro de paquetes a los camiones. Los objetivos comunes que se persiguen consisten en llevar el paquete *p1* a la ciudad *cA* y el paquete *p3* a la ciudad *cE*.

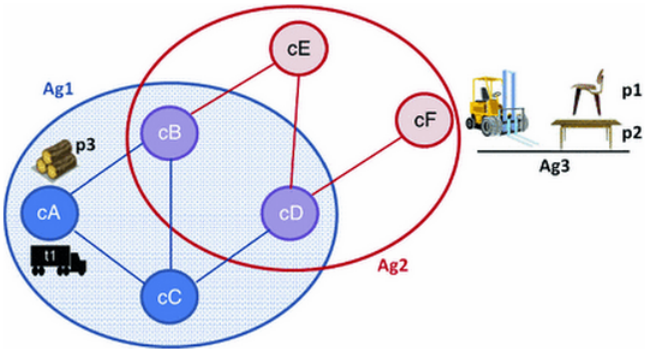

Figura 4.1: Escenario de transporte y almacenaje.

Este escenario incluye enlaces bidireccionales entre las ciudades que permiten a los agentes de transporte desplazar camiones de una ciudad a otra. Los agentes de transporte *Ag1* y *Ag2* pueden realizar tres acciones diferentes: pueden cargar y descargar paquetes de los camiones y pueden desplazar camiones de una ciudad a otra dentro de sus áreas de trabajo. Las áreas de trabajo de cada agente se indican con círculos de colores en la figura 4.1. De esta forma los agentes de transporte deben interactuar y cooperar para repartir los paquetes a un área de trabajo diferente.

Un posible plan para resolver este escenario consiste en que: (1) *Ag1* cargue la materia prima *p3* en el camión *t1*, (2) *Ag1* proporciona *t1* a *Ag2* en alguna de las ciudades *cB* o *cD* y (3) *Ag2* transporta *p3* hasta *cE*. Esto lleva a que el agente *Ag1* comparta con el agente *Ag2* la información sobre la posición del camión *t1* una vez alcance *cB* o *cD*.

El agente de almacenaje *Ag3* está al cargo de interactuar con los camiones para almacenar materias primas y suministrar productos finales. El almacén tiene un palet donde se pueden apilar y desapilar los paquetes. Los paquetes se intercambian en la ciudad en que está situado el almacén, es decir, la ciudad *cF*.

Los agentes *Ag2* y *Ag3* tienen distintas áreas de trabajo, por lo que para conseguir el primer objetivo del problema (transportar el producto *p1* a *cA*), es preciso que ambos agentes interactúen. *Ag3* suministrará *p1* a *cF*, informando a *Ag2* sobre la posición del paquete. Posteriormente, *Ag2* cargará el *p1* en el camión *t1* y lo conducirá hasta *cB* o *cD*. Finalmente, *Ag1* realizará el último transporte, repartiendo *p1* en la ciudad *cA*.

A la hora de integrar este ejemplo en la plataforma de PlanInteraction hay que determinar el conjunto de agentes que se va a utilizar. Como se indica en las secciones 3.2.3 y 3.2.5, se ha optado por una división de las tareas de planificación y ejecución en EAs y PAs. Los agentes *FMAP* se corresponden con agentes PA de PlanInteraction y cada uno de ellos estará asociado a un EA para la ejecución de sus planes. Los procesos descritos anteriormente, así como las interacciones entre agentes se llevarían a cabo entre los PAs. La configuración de agentes para el ejemplo anterior se muestra en la figura 4.2, donde los agentes *Ag1*, *Ag2* y *Ag3* se identifican con PA_1, PA_2 y PA_3 respectivamente.

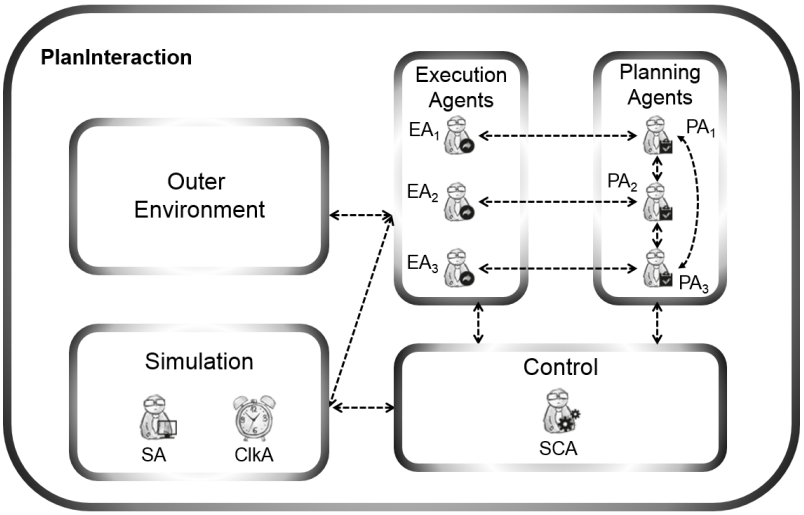

Figura 4.2: Configuración para el escenario de transporte y almacenaje.

Las interacciones que tienen lugar en este ejemplo de aplicación son las que siguen:

1. EA_i, $PA_i \rightarrow SCA$: Los agentes EA y PA solicitan registrarse en el sistema enviando un mensaje al SCA. El SCA les confirma el registro informándoles del resto de agentes presentes en la simulación.

2. $EA_i \rightarrow PA_i$: Cada agente de ejecución solicita un plan a su agente de planificación asociado, proporcionándole para ello el dominio y problema a resolver.

3. $PA_i \rightarrow PA_j$,$\forall i \neq j$: Los agentes de planificación interactúan unos con otros hasta que el plan común está formado.

4. $PA_i \rightarrow EA_i$: Cada agente de planificación comunica su parte del plan a su agente de ejecución asociado.

5. $EA_i \rightarrow SA$: Cada agente de ejecución envía al agente de simulación las acciones a ejecutar en el instante de tiempo actual.

6. $EA_i \rightarrow ClkA$: Cada agente de ejecución solicita al agente de reloj una alarma de finalización de las acciones en ejecución.

7. $ClkA \rightarrow SA$: El agente de reloj informa al agente de simulación tras cada ciclo de reloj (paso de ejecución), para actualizar el estado del mundo.

8. $ClkA \rightarrow EA_i$: El agente de reloj informa a los agentes de ejecución oportunos del vencimiento de las alarmas solicitadas.

9. $EA_i \rightarrow SA$: Los agentes de ejecución avisados por el agente de reloj solicitan al agente de simulación el estado resultante de la sensorización.

Tras la sensorización realizada en el punto 9, los agentes de ejecución monitorizan la información necesaria para la consecución del plan. Como resultado de la monitorización pueden darse tres casos: (1) el objetivo se ha alcanzado, (2) no se ha alcanzado el objetivo y el estado es correcto o (3) no se ha alcanzado el objetivo y el estado es erróneo. En el primer caso la simulación termina. En el segundo caso, el agente de ejecución retorna al punto 5, enviando al agente de simulación sus siguientes acciones del plan. Y en el último caso, es necesario una reparación o replanificación, lo que supone una nueva interacción con otro EA o con su PA asociado.

Las interacciones que se llevan a cabo en el punto 3 surgen del propio funcionamiento de *FMAP*. En el preproceso de información los PA generan grafos de información llamados $DTGs$[1] a partir de su conocimiento del mundo. Puesto que el conocimiento del problema está distribuído, los PA incian un protocolo de compartición de los DTGs refinando el grafo propio hasta que ninguno añade nueva información a su DTG.

Durante el proceso de construcción del plan, las interacciones entre los agentes siguen un protocolo similar al de compartición de DTGs. Este protocolo consiste en una serie de rondas de refinamiento del plan. En cada ronda, los PA se turnan para proponer una acción propia. Para decidir la mejor acción para la consecución de los objetivos, se utilizan funciones heurísticas. Al igual que en el caso de los DTGs, el desconocimiento de información hace que los agentes interactúen para evaluar el valor de una determinada propuesta. Finalmente, se añade al plan la propuesta con mejor valor de utilidad y se pasa a la siguiente ronda de propuestas. El proceso finaliza cuando se han cubierto todos los objetivos del problema.

[1] *Domain-Transition Graph*

4.2. Comorbilidad

La comorbilidad hace referencia a aquellos pacientes que padecen múltiples enfermedades de forma simultánea. En el ámbito médico se dispone de guías clínicas (CPG)[2] que proveen una serie de pautas a seguir para el tratamiento de determinadas enfermedades. Normalmente estas guías están especializadas para el tratamiento de enfermedades relativas al mismo área de conocimiento. Esta distribución del conocimiento para la planificación de tratamientos, unido a la concurrencia de enfermedades en un mismo paciente, supone el escenario ideal para la aplicación de la arquitectura PlanInteraction.

El problema de adaptación de tratamientos para pacientes comórbidos requiere la conciliación de las acciones recomendadas por varias CPG independientes. Las interacciones que surgen entre estas acciones pueden ser de varios tipos, por ejemplo conflictos entre los medicamentos recomendados para cada enfermedad, efectos adversos en el tratamiento de otras enfermedades, o planificación ineficiente de actividades.

En la figura 4.3 se muestra un ejemplo de configuración de la arquitectura de PlanInteraction para la simulación de este escenario [19]. Este ejemplo se centra en la coordinación entre agentes de planificación. Cada agente de planificación representa a un especialista clínico con capacidades para (1) elaborar un plan de atención personalizado para una enfermedad (módulo *HTN Planner*), (2) coordinarse con otros especialistas compartiendo sus recomendaciones y experiencia (módulo *Coordination*) y (3) detectar y resolver conflictos entre sus recomendaciones y las de otros especialistas (módulo *Conflict Solver*).

El *Initiator Agent* se encarga de proporcionar a los agentes de planificación el problema global a resolver. Este problema viene especificado como una serie de tareas a realizar, de modo que el diseño de un tratamiento para un paciente con comorbilidad consiste en un proceso de planificación cooperativa que itera sobre cada una de es-

[2]Clinical Practice Guideline.

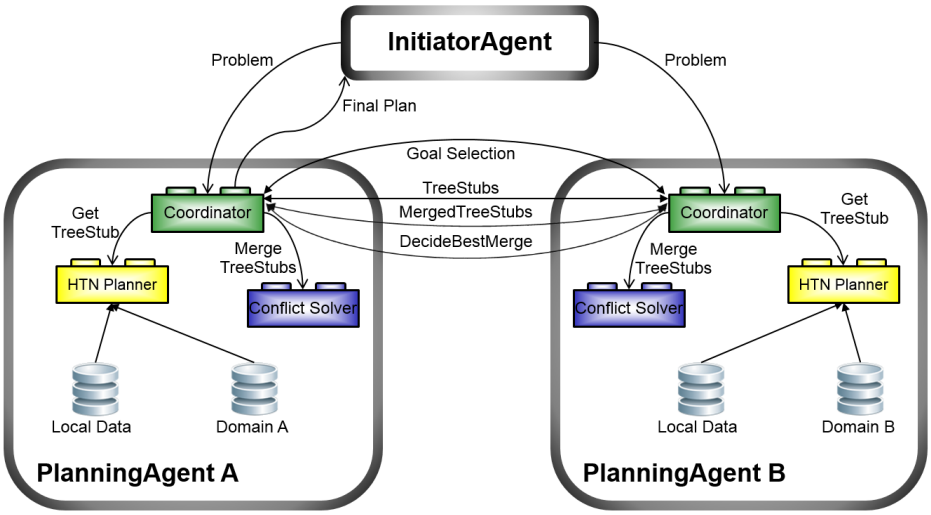

Figura 4.3: Configuración de agentes para el dominio de la comorbilidad.

tas tareas.

El módulo *HTN Planner* recibe la información del dominio y del problema y elabora un *TreeStub* compuesto por un plan personalizado para una enfermedad. El módulo *Conflict Solver* utiliza un proceso de *plan merging* combinando los dominios de dos *TreeStubs*. Esto quiere decir que cada resolutor de conflictos hace uso de dos fuentes de información diferente para la generación de un tratamiento combinado sin conflictos.

El módulo de coordinación implementa el comportamiento mostrado en la figura 4.4 (véase 3.2.4). En el primer estado el coordinador recive el problema global (*Receive Problem*). A continuación, tras seleccionar una tarea específica del problema y enviarla al otro agente (*Choose Goal*), espera la recepción de la tarea del otro agente (*Goal Agreement*). Cuando ambos agentes están conformes con la tarea a resolver, cada uno llama a su planificador HTN para generar un plan local (*Make Plan*). El planificador devuelve un *TreeStub*, que se envía al otro agente. En el siguiente estado (*Receive Plan*), el coordinador recive el *TreeStub* del otro agente y el *Conflict Solver* realiza un proceso de combinación con su plan local (*Share Merge*). A continuación los agentes comparten sus tratamientos combina-

dos y seleccionan el mejor como plan global final. El coordinador también analiza si el proceso de planificación ha terminado (*End*), en cuyo caso se envía el plan final en un mensaje ACL al *Initiator Agent*. En otro caso, se selecciona la siguiente tarea a realizar (*Choose Goal*).

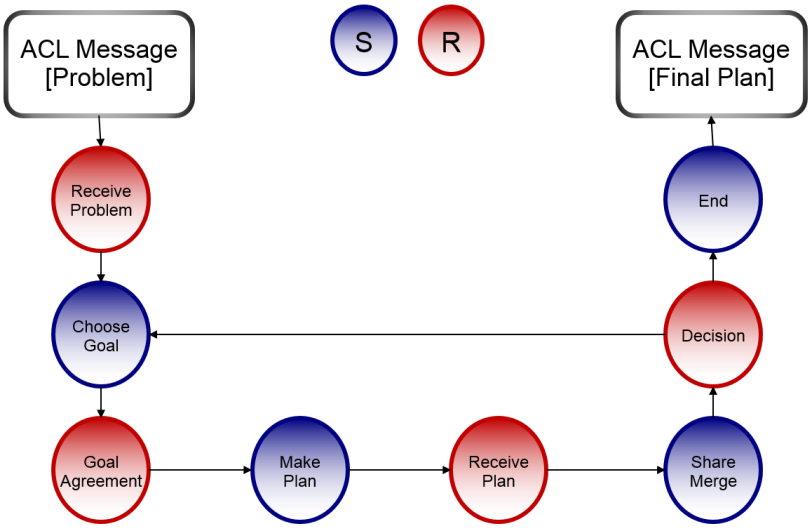

Figura 4.4: Comportamiento de coordinación.

El mejor plan se selecciona atendiendo a una función de utilidad que puede incorporar múltiples criterios clínicos: (1) Eficiencia desde el punto de vista institucional, optimizando el uso de recursos o el coste temporal; (2) Complejidad desde el punto de vista del paciente, cuantificando la complejidad del régimen de medicación y seleccionando el más sencillo.

4.3. *NASA Rovers*

Los ejemplos de aplicación mostrados previamente realizan coordinación entre los agentes a nivel de planificación. Estos procesos de coordinación, más deliberativos y de mayor abstracción, obtienen habitualmente mejores resultados sacrificando la reactividad del sistema. Sin embargo, existen dominios de gran dinamismo en los que la reactividad prima frente a la calidad. Las coordinaciones a nivel de ejecución son procesos reactivos que se basan en comportamientos e información precompilada.

La coordinación a nivel de ejecución se da cuando surge un conflicto entre los planes de los agentes de ejecución o cuando un agente de ejecución detecta un fallo en su plan. La primera situación ocurre, típicamente, cuando no se ha realizado coordinación a nivel de planificación y los EA ejecutan sus acciones en un entorno concurrente. El segundo caso ocurre cuando (1) el entorno cambia inesperadamente, (2) el resultado de una acción no obtiene los efectos esperados o (3) no se ha realizado una coordinación a nivel de ejecución adecuada, por ejemplo, para reservar un recurso. Mientras que la primera situación es más propia de agentes competitivos, la segunda puede ocurrir en cualquier momento, incluso con agentes cooperativos.

La *Mars Exploration Rover* (MER) es una misión de la *NASA* para la exploración de Marte. En esta misión se enviaron dos robots *Rovers* (*Spirit* y *Oportunity*) para explorar la superficie y geología de Marte. Los *Rovers* reciben sus instrucciones desde la Tierra. Estas instrucciones son secuencias de acciones a realizar generadas por un planificador central (CP). Al finalizar la ejecución de las acciones, los *Rovers* envían sus resultados a la Tierra.

Los *Rovers* están limitados por la energía, la capacidad de almacenamiento, las oportunidades de descarga y el tiempo y ancho de banda para completar sus observaciones. Estas limitaciones, unidas a la incertidumbre del entorno, pueden causar problemas imprevistos durante la misión y, por tanto, requerir la comunicación con la Tierra para la obtención de un nuevo plan de acciones del CP.

En este dominio el principal problema radica en la demora de las comunicaciones debido a la distancia entre la Tierra y Marte. Proporcionando a los *Rovers* capacidades de planificación y coordinación se solventa este problema. De este modo, un *Rover* puede decidir cómo actuar frente a un imprevisto, de forma individual o conjunta, en lugar de esperar una solución desde el CP de la Tierra.

El escenario de la figura figura 4.5 consta de dos *Rovers* (R_a y R_b), un spacecraft (D), tres *waypoints* (W_1, W_2 y W_3), el *lander* (L) y un planificador central. Inicialmente R_a, R_b y L se encuentran en W_2, D se encuentra en W_1 y el planificador central se encuentra en la Tierra.

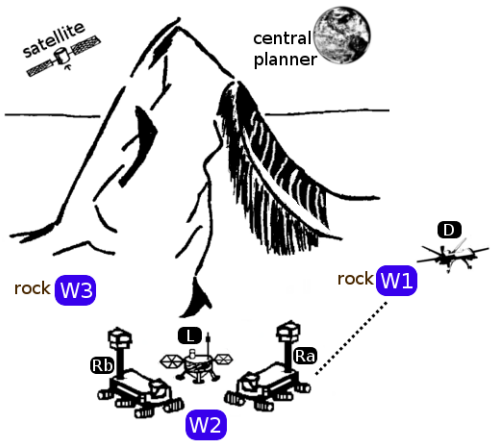

Figura 4.5: Escenario de *Rovers* en Marte.

El *Rover* R_a dispone de un brazo robótico y de una cámara para analizar rocas, mientras que R_b está equipado con una cámara panorámica para tomar imágenes de alta resolución del terreno circundante. Se pueden llevar a cabo comunicaciones con L desde cualquier *waypoint*.

La tarea de R_a consiste en identificar rocas, analizarlas y enviar los resultados a L. La tarea de R_b consiste en tomar imágenes panorámicas y comunicarlas a L. Por su parte, D tiene que sobrevolar el terreno y generar mapas de navegación para los *Rovers*.

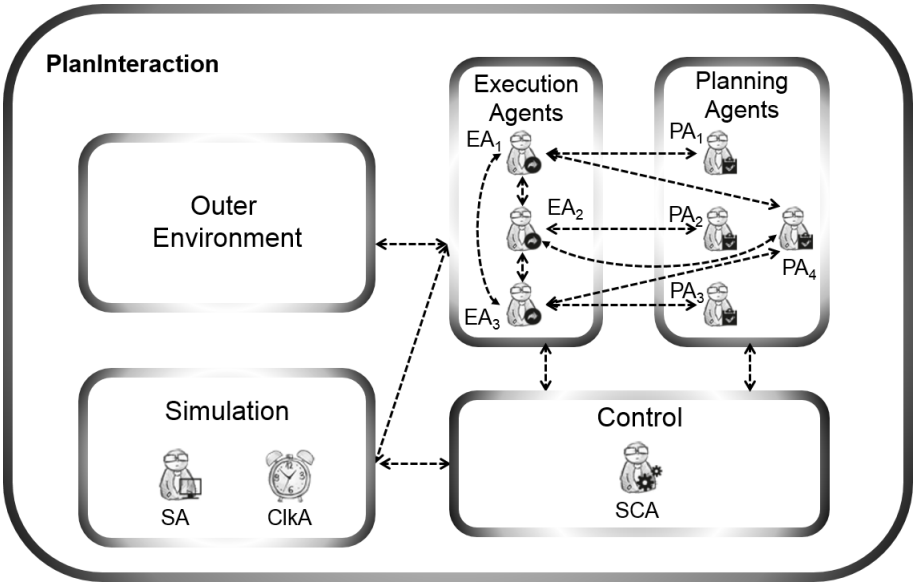

Figura 4.6: Configuración para el escenario de los *Rovers*.

Cada par de agentes $\{EA_i, PA_i\}$ de la configuración de la figura 4.6 se corresponde con uno de los *Rovers* o el *spacecraft*. El PA_4 representa al planificador central situado en la Tierra. Las tareas iniciales para la misión vienen dadas desde el centro de control en la Tierra (PA_4). A continuación, estas tareas se dividen en un conjunto de planes que se envían a R_a, R_b y D según sus capacidades.

Las interacciones que se dan en este caso de estudio son las mismas que se exponen en la sección 4.1 con alguna salvedad. En este caso de estudio, independientemente de las interacciones que pudiera haber entre los agnetes de planificación, se introducen interacciones entre los agentes de ejecución ($EA_i \rightarrow EA_j$,$\forall i \neq j$). Un ejemplo de estas interacciones consiste en la solicitud de ayuda entre dos *Rovers* ante la imposibilidad de llevar a cabo una acción. Esta interacción se corresponde con el método de coordinación *during execution* de la sección 3.4. El agente que recibe la petición de ayuda intervendría en dos procesos de coordinación (1) evaluación de la petición para dar soporte previa ejecución de su siguiente acción (*pre-execution*) y (2) comunicado de finalización tras ayudar al primer *Rover* (*post-execution*).

La figura 4.7 representa el protocolo de solicitud de ayuda que implementan los agentes de la simulación.

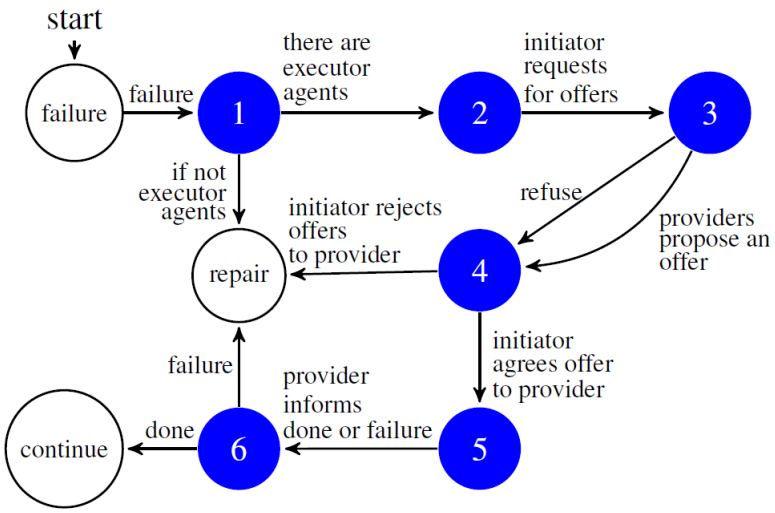

Figura 4.7: Protocolo de coordinación entre *Rovers*.

Cuando un agente de ejecución detecta un fallo en la ejecución de su plan se inicia el protocolo de coordinación entre *Rovers*. Si (**1**) el agente sabe de la existencia de otros EA entonces envía una propuesta de ayuda a todos (**2**), en caso contrario termina el protocolo iniciándose una interacción con el PA asociado para reparar el plan del agente (**repair**). En la petición de ayuda se proporciona la información relevante sobre el problema encontrado para que los EA puedan determinar si son capaces de solventar el problema. Una vez evaluada la solicitud, los EA confirman o deniegan su disponibilidad para reparar el fallo (**3**). Tras recibir todas las respuestas, el iniciador se compromete con uno solo de los posibles colaboradores (**4**) o, en caso de no existir compromiso de ayuda, el protocolo termina (**repair**). El EA colaborador que realiza la reparación informa de la correcta o incorrecta finalización de la ayuda (**5**). Por último, el agente iniciador puede continuar con su ejecución si la reparación se ha llevado a término correctamente (**continue**), o bien iniciar un protocolo de reparación (**repair**).

Capítulo 5

Conclusiones, trabajos futuros y publicaciones

En este capítulo se hace un compendio del trabajo realizado y se exponen las conclusiones extraídas del mismo. A continuación se enumeran las diferentes líneas de trabajo futuro que quedan abiertas. Por último se listan las publicaciones que han surgido de la realización del presente trabajo.

5.1. Conclusiones

La línea de desarrollo que sigue la ciencia de la computación, y en particular la inteligencia artificial, conduce al tratamiento de tareas de creciente complejidad y, a menudo, de manejo de grandes conjuntos de información. En la actualidad existen multitud de áreas en las que se utiliza planificación para optimizar el uso de recursos, mejorar la productividad o simplificar la ejecución de procesos: cadenas de montaje, planes de contingencia, autonomía robótica, estrategias comerciales, gestión empresarial y de grupos, etc. Las limitaciones computacionales y/o temporales para el tratamiento de grandes conjuntos de datos ha llevado al desarrollo de técnicas basadas en la distribución de la información para la resolución de tareas, siendo una de ellas las tecnologías multiagente.

Este trabajo introduce nociones básicas de planificación automática y tecnologías multiagente para abordar la planificación multiagente como combinación de ambas. Se ha hecho un resumen de las principales arquitecturas de planificación multiagente actuales, permitiendo caracterizar las funcionalidades comunes de estos sistemas y sus particularidades.

Este trabajo constituye la primera fase del proyecto PlanInteraction, la cual está orientada al desarrollo de una arquitectura de planificación multiagente para el estudio de dinámicas sociales en planificación y ejecución. La arquitectura PlanInteraction está concebida desde el inicio como una extensión multiagente de un sistema consolidado de planificación y ejecución (PELEA), en contraposición a otras arquitecturas de planificación multiagente tratadas en este trabajo. La principal ventaja heredada de PELEA consiste en la independencia del planificador concreto empleado. Del mismo modo, la arquitectura PlanInteraction se apoya en la plataforma Magentix2 para el desarrollo del entorno multiagente, lo que permite la reutilización de un gran conjunto de herramientas para entornos multiagente como sistemas de comunicación, diseño básico de agentes, sistemas de seguridad, sistemas de trazas, organizaciones, etc.

El grado de madurez alcanzado en la implementación de la plataforma permite el diseño y desarrollo de: agentes de planificación y/o ejecución, módulos funcionales generales o específicos para los agentes, comportamientos de agentes representados como grafos de estados, protocolos de comunicación, coordinación y sincronización. La arquitectura PlanInteraction proporciona también un conjunto de agentes y comportamientos predefinidos para una correcta simulación de planificación y ejecución. Así mismo, se dispone de una interfaz gráfica preliminar que permite la depuración y análisis de resultados de las simulaciones llevadas a cabo en la plataforma.

Los casos de estudio desarrollados han permitido comprobar las principales funcionalidades de la arquitectura en problemas reales. Cada uno persigue estudiar distintas técnicas de coordinación entre agentes a nivel de planificación/ejecución. El uso de estas técnicas

no garantiza una ejecución de los planes libre de errores, pero minimiza los fallos debidos a ejecuciones concurrentes. La coordinación a nivel de ejecución proporciona soluciones de menor calidad pero más rápidas y aptas para entornos de gran dinamismo. El uso de un tipo de coordinación no es excluyente sino incremental, es decir, se puede hacer coordinación *pre-planning* y reforzar la fiabilidad de la ejecución mediante coordinación *post-planning* y *during execution*.

5.2. Trabajos futuros

El trabajo realizado hasta la fecha dentro del proyecto PlanInteraction ha supuesto un gran esfuerzo y dedicación para sentar una base sólida sobre la que desarrollar nuevas líneas de investigación. Por otro lado, este proyecto de investigación recoge más objetivos de los que se han cubierto en el presente trabajo. A continuación se detalla alguno de los objetivos pendientes que se están desarrollando en la actualidad y las investigaciones en curso.

- La arquitectura PlanInteraction, como extensión multiagente de la arquitectura de planificación y ejecución PELEA, hace uso exclusivamente de las funcionalidades de alto nivel, dejando al margen la información de bajo nivel relacionada con el mundo físico. La incorporación del **tratamiento del bajo nivel** permitirá pasar de las simulaciones virtuales a las simulaciones reales en el mundo físico mediante el uso de robots.

- En el área de la planificación clásica, siempre se han tratado los problemas de planificación como problemas monoagente. Sin embargo, la complejidad de algunos problemas estudiados, unido a la gran cantidad de información/combinatoria que suponen, hace que sean inabordables por los planificadores clásicos. El uso de las tecnologías multiagente y la distribución de la información hacen posible el tratamiento de estos problemas. El desarrollo de un **método automático de agentización** de los problemas sería de gran utilidad para el estudio estandarizado de los problemas de planificación clásicos en entornos multiagente.

- En las simulaciones llevadas a cabo, las relaciones entre los agentes están predefinidas. Un comportamiento más realista de la plataforma, y en sintonía con otros sistemas multiagente, sería permitir la **publicación de servicios de los agentes**. De esta forma las relaciones e interacciones entre los agentes serían el resultado de procesos de decisión de agentes inteligentes. Un agente solo debería comunicarse con aquellos agentes que sabe que le pueden proporcionar un servicio determinado para no saturar el sistema de comunicaciones.
- El proyecto PlanInteraction prevee el **desarrollo de un *framework* completo** para el estudio de técnicas de interacción entre agentes de planificación y ejecución. La interfaz gráfica introducida en este trabajo supone la primera aproximación en esta línea.
- Las investigaciones en curso contemplan el desarrollo de **técnicas de interacción entre agentes** haciendo uso de teoría de juegos, negociación y argumentación. La incorporación de estas técnicas supondría la generación de planes más satisfactorios y seguros para los agentes, tanto individual como colectivamente.
- El resultado de un proceso de planificación para un problema dado, independientemente del método empleado, depende del conocimiento que se proporciona sobre el mismo. El conocimiento de los agentes sobre el mundo en el que interactúan puede provenir de los sensores que incorporan. De esta forma la visión de cada agente sobre el mundo es parcial y caduca, puesto que la veracidad de la información depende del instante de tiempo en el que se obtuvo en un entorno dinámico. El **tratamiento de la incertidumbre** de la información puede llevar al desarrollo de nuevas técnicas de planificación y/o coordinación entre agentes.
- Otro de los objetivos previstos dentro del proyecto PlanInteraction consiste en el estudio de la **asociatividad entre agentes** mediante la generación de coaliciones, *teamworks* y organizaciones. Actualmente se está diseñando un método de reparación de fallos a nivel de ejecución como fase previa a la interacción entre agentes dentro de un *teamwork*.

5.3. Publicaciones

Las investigaciones descritas en esta tesis han contribuido a la publicación de los siguientes artículos:

- I. Sánchez-Garzón, J. Fernández-Olivares, E. Onaindia, G. Milla, J. Jordán, and P. Castejón. A multi-agent planning approach for the generation of personalized treatment plans of comorbid patients. *In N. Peek, R. M. Morales, and M. Peleg, editors, AIME, volume 7885 of Lecture Notes in Computer Science, pages 23-27. Springer, 2013.* **(CORE A)**

- C. Guzmán, P. Castejón, E. Onaindía and J. Frank. Multi-agent reactive planning for solving plan failures. *International Conference on Hybrid Artificial Intelligence Systems pp. In Press. (2013)* **(CORE C)**

Bibliografía

[1] Magentix2. http://www.gti-ia.upv.es/sma/tools/magentix2/index.php, Apr. 2013.

[2] AgentScape. http://www.agentscape.org/about/, Apr. 2013.

[3] EVE. http://almende.github.io/eve/, Apr. 2013.

[4] JADE, Java Agent DEvelopment Framework. http://jade.tilab.com/, Apr. 2013.

[5] JASON. http://jason.sourceforge.net/wp/, Apr. 2013.

[6] M. Beetz. *Plan-based control of robotic agents: improving the capabilities of autonomous robots*. Springer-Verlag, Berlin, Heidelberg, 2002.

[7] W. Choi and J.-C. Latombe. A reactive architecture for planning and executing robot motions with incomplete knowledge. In *Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91. IEEE/RSJ International Workshop on*, pages 24–29 vol.1, 1991.

[8] K. Decker. TAEMS: A Framework for Environment Centered Analysis & Design of Coordination Mechanisms. *Foundations of Distributed Artificial Intelligence, Chapter 16*, pages 429–448, January 1996.

[9] R. J. Firby. *Adaptive execution in complex dynamic worlds*. PhD thesis, New Haven, CT, USA, 1989. AAI9010653.

[10] R. J. Firby. Building symbolic primitives with continuous control routines. In *Proceedings of the first international conferen-*

ce on Artificial intelligence planning systems, pages 62–69, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[11] R. L. Fogués, J. M. Alberola, J. M. Such, A. Espinosa, and A. García-Fornes. Towards Dynamic Agent Interaction Support in Open Multiagent Systems. In *Proceedings of the 13th International Conference of the Catalan Association for Artificial Intelligence*, volume 220, pages 89–98. IOS Press, 2010.

[12] M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the sixth National conference on Artificial intelligence - Volume 2*, AAAI'87, pages 677–682. AAAI Press, 1987.

[13] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja, R. Vincent, P. Xuan, and X. Zhang. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

[14] P. Maes. The dynamics of action selection. In *Proceedings of the 11th international joint conference on Artificial intelligence - Volume 2*, IJCAI'89, pages 991–997, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[15] K. L. Myers. Cpef: A continuous planning and execution framework. *AI Magazine*, 20(4):63–69, 1999.

[16] M. Paolucci, O. Shehory, K. Sycara, D. Kalp, and A. Pannu. A planning component for retsina agents. In *In Agent Theories, Architectures, and Languages*, pages 147–161. Springer Verlag.

[17] E. Quintero, V. Alcázar, D. Borrajo, J. Fernández-Olivares, F. Fernández, A. G. Olaya, C. Guzman, E. Onaindia, and D. Prior. Autonomous mobile robot control and learning with the pelea architecture. In *Automated Action Planning for Autonomous Mobile Robots*, volume WS-11-09 of *AAAI Workshops*. AAAI, 2011.

[18] A. S. Rao. Agentspeak(l): Bdi agents speak out in a logical computable language. In *Proceedings of the 7th European works-*

hop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away, MAAMAW '96, pages 42–55, Secaucus, NJ, USA, 1996. Springer-Verlag New York, Inc.

[19] I. Sánchez-Garzón, J. Fernández-Olivares, E. Onaindia, G. Milla, J. Jordán, and P. Castejón. A multi-agent planning approach for the generation of personalized treatment plans of comorbid patients. In N. Peek, R. M. Morales, and M. Peleg, editors, *AIME*, volume 7885 of *Lecture Notes in Computer Science*, pages 23–27. Springer, 2013.

[20] L. Steels. Cooperation between distributed agents through self-organisation. In *Intelligent Robots and Systems '90. 'Towards a New Frontier of Applications', Proceedings. IROS '90. IEEE International Workshop on*, pages 8–14 supl, 1990.

[21] K. Sycara, M. Paolucci, M. van Velsen, and J. Giampapa. The retsina mas infrastructure. Technical report, the special joint issue of Autonomous Agents and MAS, Volume 7, Nos. 1 and 2, 2001.

[22] A. Torreño, E. Onaindía, and O. Sapena. A Flexible Coupling Approach to Multi-Agent Planning under Incomplete Information. Springer, 2012. DOI: 10.1007/s10115-012-0569-7.

[23] A. Torreño, E. Onaindía, and O. Sapena. An Approach to Multi-Agent Planning with Incomplete Information. In *20th European Conference of Artificial Intelligence (ECAI 2012)*, volume 242, pages 762–767. IOS Press, 2012.

[24] M. D. Weerdt, A. T. Mors, and C. Witteveen. Multi-agent planning: An introduction to planning and coordination. Technical report, In: Handouts of the European Agent Summer, 2005.

[25] D. S. Weld. Recent advances in ai planning. *AI MAGAZINE*, 20:93–123, 1999.

[26] D. E. Wilkins and K. L. Myers. A multiagent planning architecture. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, pages 154–162, 1998.