UNIVERSIDAD POLITÉCNICA DE VALENCIA

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

TESIS DOCTORAL

espea

Aplicación de Técnicas de Negociación en la Arquitectura *ARTIS*

memflta

Patricia V. Maldonado Cárdenas

Director:
Dr. Vicente Botti Navarro

Valencia, Marzo 2005.

Agradecimientos

En el desenlace de esta etapa de mi vida debo agradecer a mucha gente que ha estado conmigo, me a brindado su apoyo y amistad.

Quiero comenzar agradeciendo a Vicente, que ha llegado a ser un amigo con el cual siempre he podido contar. Agradecerle su paciencia y compresión para conmigo, al escucharme y guiarme en todas las etapas de esta tesis. Especialmente al final, cuando más apoyo necesitaba de su parte.

Agradezco también a Pedro, mi esposo que me dió todo su apoyo y paciencia; por estar a mi lado en las buenas y no tan buenas, dándome la fuerza y su amor necesarios para terminar con éxito esta etapa de mi vida. Gracias por estar aquí conmigo, a mi lado.

Agradecer a mi familia y en especial a mis padres, quienes siempre han velado mis avances con su habitual fanatismo poco objetivo por mi. Garcias por darme la vida. A mi abuelita, quien no llegó a presenciar este final pero sé que está aquí, conmigo.

Agradezco también al grupo GTI-IA por brindarme tan calurosa acogida desde mi llegada a España, ese cariño que siempre es bien recibido cuando estás lejos de tu tierra natal. Gracias también por proporcionarme las herramientas que necesitaba para desarrollar esta tesis doctoral.

A Bernabé, que gracias a su desinteresada amistad y sus diligentes asistencias pude comenzar este camino de perfección.

A Carlos Carrascosa, que sabe justamente lo que me costó llegar al final

de esta tesis. Gracias por soportar mis constantes acedios y persecuciones. Al final llegué.

A la Universidad de Magallanes y en particular a mis colegas del Departamento de Computación. Ellos son los que han absorbido mis horas de trabajo para que yo pueda terminar esta loca aventura.

Somos una piedra en bruto, y cada uno de nuestros logros lima alguna de nuestras aristas, tratando de descubrir la forma final.

Resumen

"*Cogito ergo sum*", *Descartes (1596-1650)*

Desde que el hombre fue consciente de su existencia y de sus múltiples necesidades, la negociación pasó a formar parte integral de nuestras vidas. Esto debido a que muchas veces, los escasos recursos e insuficiencia de habilidades, lo empujaban a buscar ayuda con el fin de cubrir esas necesidades. Sin embargo, esta ayuda no la obtenía de forma gratuita, sino que debía trocar algo a cambio.

Los investigadores de los sistemas multi-agentes, en su afán de recrear la interacción de las sociedades humanas, han desarrollado una línea que trata la negociación entre los agentes que los componen. Para esto fue necesario adaptar las normas, reglas, estrategias y protocolos de las sociedades humanas a las interacciones de negociación entre los agentes.

Así, las primeras investigaciones sobre negociaciones entre agentes estaban orientadas a dilucidar las conductas y reglas generales que deben tener los agentes al momento de negociar. En estas primeras definiciones influyen las características básicas de los agentes tales como: no mienten, son sociables, etc. No obstante, dependiendo del entorno donde esté situado el agente, sus estrategias de negociación variarán.

En esta tesis doctoral se presentan las técnicas de negociación como una forma de solucionar el problema que debe resolver un agente $\mathcal{ARTIS}$ (*An Real-Time Intelligence System*). Cabe decir que el agente $\mathcal{ARTIS}$ trabaja en

entornos de tiempo real estrictos, con lo cual sus percepciones y acciones están restringidas temporalmente.

Para resolver sus problemas, el agente lo divide en sub-problemas los que serán resueltos por sus entidades internas (*in-agents*). De esta forma varios *in-agents* pueden dar una solución distinta a un mismo sub-problema, o también puede suceder que un *in-agent* puede tener dar solución a varios sub-problemas. Es en este marco de acción donde proponemos la inserción de las técnicas de negociación como una forma de mejorar las soluciones proporcionadas por las entidades internas del agente $\mathcal{ARTIS}$ también llamadas *in-agents*.

Se plantean las normas, reglas y estrategias que deberán seguir los *in-agents* durante el proceso de negociación, así como el planteamiento de los diversos protocolos que deberán cumplir para establecer y mantener dichas negociaciones.

Expondremos también las diversas experiencias hechas con los métodos de negociación implementados en la arquitectura del agente $\mathcal{ARTIS}$, los resultados obtenidos y las conclusiones finales.

Abstract

Since the man was conscientious of his existence and its multiple necessities, the negotiation happened to comprise integral of our lives. This because often, the limited resources and insufficiency of abilities, guided to look for it aid with the purpose of covering those necessities. Nevertheless, this aid did not obtain it free, but that had to change for this help.

The systems multi-agents investigators, in their efforts to recreate the interaction of the human societies, have developed a line that treats the negotiation among the agents who compose them. To do this was necessary to adapt the norms, rules, strategies and protocols of the human societies to the interactions of negotiation among the agents.

Thus, the first research on negotiations among agents were oriented to explain the general conducts and rules that must have the agents at the time of negotiating. In these first definitions influence the basic characteristics of the agents such as: they do not lie, they are sociable, etc. However, depending on the environment where the agent is located, their strategies of negotiation will varies.

In this doctoral thesis, the negotiation techniques appear as a form to solve the problem that must solve an $\mathcal{ARTIS}$ agent. In this way, the $\mathcal{ARTIS}$ agent works in hard real-time environment, with which its perceptions and actions are restringuidas temporarily.

In order to solve its problems, the agent divides each problem in sub-problems those to it that will be solved by their internal entities (*in-agents*).

Of this form, several *in-agents* can give a solution different from a same sub-problem, or also it can happen that an *in-agent* can have to give solution to several sub-problems. In this frame of action where we propose the insertion of the negotiation techniques as a form to improve the solutions provided by the internal organizations of agent $\mathcal{ARTIS}$ also called *in-agents*.

We will raise the norms, rules and strategies that will have to follow *in-agents* during the negotiation process, as well as the exposition of the diverse protocols that will have to fulfill to establish and to maintain these negociones. We will also expose the diverse experiences done with the implemented methods of negotiation in the obtained architecture of agent $\mathcal{ARTIS}$, results and the final conclusions.

Resum

Des de què l'home va ser conscient de la seua existència i de les seues múltiples necessitats, la negociació va passar a formar part integral de les nostres vides. Açò unit a què moltes vegades, la manca de recursos i la insuficiència d'habilitats, l'espentaven a cercar ajuda amb la finalitat de cobrir estes necessitats. En canvi, esta ajuda no l'obtenia forma gratuïta, sino què havia de pagar alguna cosa a canvi.

Els investigadors dels sistemes multi-agents, en el seu interés de reproduir la interacció de les societats humanes, han desenvolupat una línia que tracta la negociació entre els agents que la componen. Per este motiu, cal adaptar les normes, les regles, les estratègies i els protocols de les societats humanes a les interaccions de negociació entre els agents. Així, les primeres recerques sobre negociacions entre agents estaven orientades a resoldre les conductes i les regles generals que han de tindre els agents durant la negociació. En estes primeres definicions influeixen les característiques bàsiques dels agents, com ara: no menteixen, són sociables, etc. No obstant això, depenent de l'entorn on estiga situat l'agent, les seues estratègies de negociació variaran.

En esta tesis doctoral es presenten les tècniques de negociació com una forma de solventar el problema que ha de resoldre un agent ARTIS (An Real-Time Intelligent System). Cal dir, que l'agent ARTIS treballa amb entorns de temps real estrictes, amb la qual cosa, les seues percepcions i les seues accions estan restringides temporalment.

Per a resoldre el seus problemes, l'agent els divideix en subproblemes, els

quals, seran resolst per les seus entitats internes (in-agents). D'esta forma, diversos in-agent poden donar una solució diferent a un mateix sub-problema, o també, pot succeir que un in-agent puga donar solució a diversos subproblemes. En este marc d'acció on proposem la inserció de les tècniques de negociació com una forma de millorar les solucions proporcionades per les entitats internes de l'agent ARTIS, també anomenat in-agents.

Es plantegen les normes, les regles i les estratègies que hauran de seguir els in-agents durant el procés de negociació, així com el plantejament dels diversos protocols que hauran d'acomplir per establir i per a mantindre les esmentades negociacions.

Expondrem també les diverses experiències fetes amb els mètodes de negociació implementades en l'arquitectura de l'agent ARTIS, els resultats obtesos i les conclusions finals.

Índice general

Índice de figuras

Capítulo 1

Introducción

El utilizar agentes para dar solución a diversos tipos de problemas esta siendo cada vez más extendido. Este hecho con lleva, la mayoría de las veces, a que los agentes deban buscar ayuda en otros agentes para dar una óptima solución a su problema, ya que por sí mismos no podrían alcanzarla. Para la tarea de "*buscar ayuda*", los agentes necesitan poder relacionarse e interactuar entre sí. Este proceso implica que el agente:

- Pueda comunicarse con otros agentes.
- Pueda entender a otros agentes, es decir que posea un lenguaje común.
- Posea planes de acción y/o estrategias a seguir, dependiendo de los distintos escenarios que se le puedan presentar durante la interacción.

Lo anterior supone que el agente posea la capacidad de coordinarse con otros agentes, utilizando un protocolo de comunicación común, acerca de las posibles acciones que deberán tomar ambas partes para lograr sus objetivos.

No obstante lo anterior la forma de interactuar con los demás variará dependiendo del entorno del agente. Estos entornos pueden ser:

- *Entornos hostiles*, lo que implica que los objetivos de los agentes del sistema no son compatibles, por lo tanto cada agente deberá luchar por alcanzar sus objetivos.

- *Entornos colaborativos*, lo que supone que los objetivos de los agentes del sistema son compatibles entre sí, es decir, los agentes del sistema pueden trabajar en conjunto por el objetivo general del sistema, pudiendo además cada uno de ellos tener sus propios objetivos cuyos logros no implican necesariamente la obstaculización de los demás.

Esta forma, la interacción entre ellos cobrará mayor importancia a medida que el número de agentes que intervenga en los *Sistemas Multi-Agentes* aumente.

Dentro de las formas o métodos de interacción entre agentes está la negociación, la cual está considerada como una efectiva técnica de interacción y coordinación para resolver sus problemas tales que las partes involucradas logren sus objetivos alcanzando al mismo tiempo utilidades óptimas cuando ambas partes logran llegar a un acuerdo. Estas negociaciones son utilizadas principalmente cuando los agentes están faltos de recursos o habilidades para alcanzar por sí solos sus metas ó bien cuando los recursos del entorno al que pretencen son escasos.

Actualmente los desarrollos prácticos sobre el tema de agentes utilizando negociaciones como un método para solucionar sus problemas están enfocados principalmente en temas de *e-commerce*, y en particular las subastas on-line. Siendo estas últimas las que han tenido un mayor desarrollo por su eficacia y rapidez.

En esta tesis doctoral, vamos a presentar la negociación como una forma de interacción entre las entidades del agente $\mathcal{ARTIS}$, las cuales van a cooperar entre si para dar solución al problema del agente al que pertenecen.

1.1. Motivación

Los *Sistemas Multi-Agentes* son utilizados frecuentemente para desarrollar tareas tales como: búsquedas, cooperación, colaboración, negociaciones, etc.

Por otro lado, los *Sistemas de Tiempo-Real* son utilizados para tareas que requieren precisión en sus cálculos de ejecución, es decir, se utilizan para tareas que necesitan garantizar su correcta ejecución dentro de unos plazos preestablecidos (restricciones temporales).

$\mathcal{ARTIS}$ (*An Real-Time Intelligence System*) es una arquitectura para crear agentes inteligentes con las características de los *Sistemas de Tiempo-Real* (*STR*). Fue implementada en el Grupo de Tecnología Informática (GTI) de la Universidad Politécnica de Valencia – Valencia – España, y sobre la cual se han desarrollado múltiples proyectos tanto para el desarrollo de esta arquitectura como para el desarrollo de aplicaciones utilizando la arquitectura. Esto último implica que $\mathcal{ARTIS}$, al ir creciendo su aplicabilidad en el mundo real, vaya necesitando de más herramientas que le proporcionen una mayor inter-operabilidad entre estos.

La evolución natural de $\mathcal{ARTIS}$ nos guía a la ampliación de sus herramientas para darle mayor interoperabilidad, no solo porque lo necesita, sino porque no hay herramientas las satisfagan.

Es en este punto que enfocamos la incorporación de las técnicas de negociación en el agente $\mathcal{ARTIS}$. Estas técnicas pretender ayudarle a deliberar en los momentos donde se necesite maximizar su utilidad tanto a nivel interno como externo.

1.2. Objetivos

El objetivo principal de esta tesis doctoral es presentar una contribución en el ámbito de las negociaciones entre agentes que trabajen en entornos de *tiempo real estricto*. Para esto se incorporarán técnicas de negociación en la arquitectura del agente $\mathcal{ARTIS}$ tales que sus entidades internas puedan utilizarlas y

así obtener soluciones óptimas a sus problemas.

La integración de estos métodos de negociación en $\mathcal{ARTIS}$ suponen los siguientes sub-objetivos:

- Establecer el entorno donde se producirán las negociaciones, es decir, entornos de sistemas multi-agentes trabajando en conjunto con los sistemas de tiempo-real estricto y las restricciones que estos imponen.

- La comunicación entre los agentes que participen de la negociación tiene un papel preponderante ya que, por medio de esta los agentes obtendrán el conocimiento necesario para desarrollar sus acciones con una visión global y así poder sincronizarse con el resto de agentes en forma de negociaciones. Sin embargo, una excesiva comunicación puede dar lugar a una sociedad de agentes burocráticos, cuya sobrecarga de comunicación sea mayor que el trabajo efectivo realizado.
  Para este caso en particular, la comunicación entre las entidades del agente $\mathcal{ARTIS}$ se hace a través de una pizarra temporal llamada "*blackboard*", la cual se utilizará como vía de transporte de los mensajes que se produzcan entre los participantes de la negociación y mantendrá en todo momento el estado de las mismas, además de cualquier tipo de información relevante que las entidades o el mismo agente necesite.

- La negociación se utilizará como una forma de coordinación entre las entidades del agente $\mathcal{ARTIS}$ quienes utilizarán sus interacciones para resolver sus problemas y poder alcanzar un acuerdo sobre cómo trabajar conjuntamente obteniendo los mejores beneficios para ambas partes, ergo, para el agente al que pertencen.

- Las entidades participantes de las negociaciones compartirán información para que el sistema al que pertenecen (agente $\mathcal{ARTIS}$) pueda construir un plan de cómo y cuándo deben trabajar, distribuyendo y siguiendo este plan durante la resolución del problema del agente.

- Se integrará el razonamiento sobre las acciones y creencias de las entidades del sistema, con el razonamiento global sobre la resolución del problema, de forma que las decisiones de coordinación sean parte de las decisiones globales en vez de una capa separada sobre la resolución local del problema.

1.3. Organización del Documento

La organización de este documento será la siguiente:

- En el capítulo 2 se expondrá una síntesis del estado del arte de los *agentes*, los *Sistemas Multi-Agentes* y los *Sistemas de Tiempo-Real*.
- En el capítulo 3 se expone una revisión de los métodos de negociación en Sistemas Multi-Agentes, comenzando con definiciones básicas de tema, extendiéndolas a los agentes y diversas aplicaciones actuales.
- En el capítulo 4, se describe la arquitectura de un agente $\mathcal{ARTIS}$ así como sus actuales características y aplicaciones.
- En el capítulo 5 se expondrá la incorporación de las técnicas de negociaciones la arquitectura $\mathcal{ARTIS}$.
- En el capítulo 6 se presentan los resultados obtenidos con la incorporación de estas técnicas en pruebas simuladas y experimentales.
- En el capítulo 7 se entregan las conclusiones finales y trabajos futuros.

Capítulo 2

Sistemas Multi-Agentes

En este capítulo se presenta una síntesis del estado del arte del área donde se implementa esta tesis doctoral, es decir: *agentes*, *Sistemas Multi-Agentes* (*SMA*) y *Sistemas de Tiempo Real* (*STR*).

Esta revisión se divide en cuatro partes:

- Definiciones y clasificaciones de agentes.
- Introducción a los *Sistemas Multi-Agentes* y sus características.
- Los *Sistemas Multi-Agentes* y los *Sistemas de Tiempo-Real.*
- Interacciones en los *Sistemas Multi-Agentes.*

2.1. Agentes

2.1.1. Definición

Aunque no existe un consenso general para definir el concepto de *agente*, en la literatura se pueden encontrar muchas aproximaciones de su definición, características, y entornos donde actúan.

Una aproximación a la definición de *agente* la dan Russell and Norving (1996) que dicen:

"Un agente es todo aquello que puede percibir su ambiente mediante sensores y que responde o actúa en tal ambiente por medio de efectores"

Sin embargo, los agentes han llegado a ser entidades más complejas, cuyas aplicaciones abarcan múltiples áreas: industriales (control de procesos, tráfico aéreo, robótica); comerciales (gestión de la información, e-commerce, robot de búsqueda); médicas (control y atención de pacientes); entretenimiento (juegos interactivos, robots, cine interactivo); entre otras. Dada su gran diversidad, fue necesario darle al *agente* una definición más específica que considerara todos sus aspectos.

Una definición mas extensa se encuentra en (Wooldridge and Jennings, 1995). Ellos consideran un *agente* como un sistema computacional capaz de actuar de forma autónoma y flexible sobre un entorno. Entendiendo por *"flexible"* que el *agente* sea:

- *Reactivo*, que sea capaz de responder a los cambios de su entorno por medio de acciones específicas;
- *Pro-activo*, que esas acciones estén orientadas a cumplir con sus planes y objetivos primarios;
- *Social*, que el agente pueda comunicarse con otros agentes utilizando algún *lenguaje de comunicación de agentes* común entendible para ambos.

Figura 2.1: Agente Según Parunak

Sin embargo,Parunak and Odell (1999) plantean una definición aún más flexible, comparando a un *agente* con una navaja del ejercito suizo (Figura 2.1), es decir, el *agente* tendría una definición básica la cual considera los aspectos de percepción, acción y cognición planteados por Wooldridge and Jennings (1995), y dependiendo de su entorno, a la definición de *agente* se le incorporarán las características que necesite.

Otra definición planteada por Julián and Botti (2000) muestra al *agente* como una:

"Entidad capaz de aprender nuestros gustos y actuar tal y como lo haríamos nosotros, pero adelantándose y realizando tareas que nosotros podríamos realizar si dispusiésemos de más tiempo".

Para el ámbito de esta tesis y aprovechando las definiciones mencionadas, se considerará al *agente* como:

Una entidad capaz de razonar y actuar sobre su entorno según sus percepciones y objetivos, pudiendo aprender de estas acciones y utilizar este conocimiento para realizar sus tareas.

2.1.2. Características

Además de las características mencionadas por Wooldridge and Jennings (1995), otros investigadores (Wooldridge and Jennings, 1995; Franklin and Graesser, 1996; Nwana, 1996) exponen la necesidad de ampliarlas para dar mayor flexibilidad a la definición. Con esto, no se quiere decir que el agente debe poseer todas estas características, sino que se pueden contemplar dependiendo de su entorno. Por lo tanto, a las características anteriores se le agregan:

- *Racional*: las acciones del agente se regirán por la racionalidad, para alcanzar sus metas u objetivos si estos son viables.
- *Adaptablilidad*: el agente debe ser capaz de adaptarse a su entorno.
- *Movilidad*: el agente sea capaz de moverse en la red telemática.
- *Veracidad*: es la asunción que el agente no miente, ni comunica falsa información a propósito.
- *Benevolencia*: el agente podría ayudar a otros agentes, siempre que esto no entre en conflicto con sus creencias o metas.

Resumiendo, las características hasta aquí mencionadas serán las guías cuando se defina un agente, pudiendo considerarlas o no, según sea la necesidad del mismo *(Parunak and Odell, 1999)*.

2.1.3. Arquitecturas de Agentes

Como se explicó en la sección anterior, los agentes deben tener como mínimo las características de: *percepción*, *cognición* y *acción*.

El entorno de un agente y las habilidades que necesite para desenvolverse en dicho entorno, determinarán la arquitectura interna del mismo. Existen diversas clasificaciones de los agentes según su arquitectura y dependiendo de sus características, con diversas versiones, las cuales se exponen a continuación.

Nwana (1996) clasifica los agentes según su topología en:

1. *Agentes móviles*, son aquellos que pueden moverse por la red, los subdivide en estáticos y dinámicos.

2. *Agentes deliberativos*, son aquellos que tienen un modelo interno del mundo y, utilizando planes elaborados, reaccionan a los distintos eventos de su entorno. Incluyen la capacidad de coordinación con otros agentes.

3. *Agentes reactivos*, son aquellos agentes que reaccionan según sus percepciones y por lo general no mantienen un mapa simbólico de su entorno. Principalmente utilizan el principio de: *estímulo-respuesta*.

4. *Agentes de información/Internet*, son aquellos encargados de buscar información en la red

5. *Agentes híbridos*, que puede combinar dos o más filosofías de un agente simple.

6. *Agentes colaborativos*, aquí distingue varios tipos de agentes que deben poseer ciertas características mínimas, tales como: autónomo; poseer habilidades sociales para interactuar con otros agentes o bien con humanos; poseer inteligencia (capacidad de deducción y aprendizaje).

Otra forma de clasificar a los agentes propuesta por Franklin and Graesser (1996) es considerar las características "*naturales*" de los agentes y agruparlo bajo un concepto más amplio. Por ejemplo, los agentes que son capaces de aprender podrían ser subconjunto de los agentes móviles.

Sin embargo una forma de clasificación más extendida y utilizada es la propuesta por Gerard (1999) que clasifica a los agentes basándose en el tipo de arquitectura sobre la cual se definen, así se tienen básicamente tres tipos:

1. *Arquitectura por Capas*. En esta arquitectura el razonamiento del agente será por niveles o capas y cada una tendrá una razón de ser, es decir podrá cumplir una acción específica dentro del agente. Con esta característica base, el agente podrá aun sub-dividir su arquitectura en dos tipos:

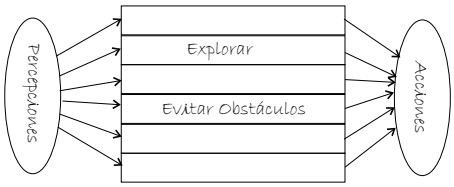


Figura 2.2: Arquitectura por Capas Horizontales

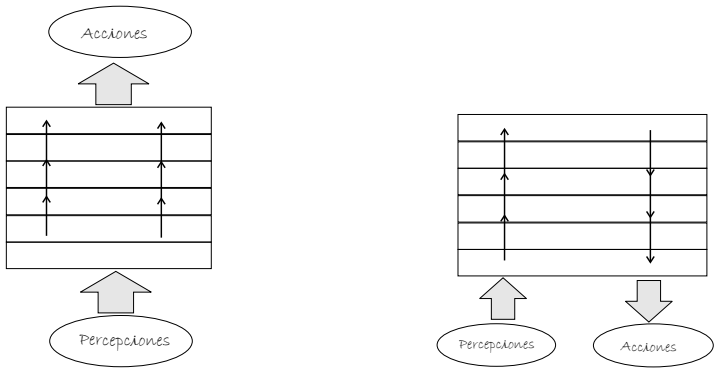
a) *Arquitectura por Capas Horizontal.* (Figura 2.2) En este tipo de arquitectura las capas del agente trabajan paralelamente pudiendo acceder cada una de ellas a los sensores y actuadores y procesar de forma independiente la información obtenida. No obstante, sus acciones siempre estarán coordinadas por los objetivos del agente.

b) *Arquitectura por Capas Vertical.* (Figura 2.3) En este tipo de arquitecturas solo las capas de más bajo nivel podrán acceder a los sensores y actuadores del agente, transmitiendo esta información a las capas de más alto nivel. Estas útlimas procesan la información recibida y envían los resultados de sus procesos (decisiones) a las capas de mas bajo nivel para que actúen según sus directrices. Existen dos tipos de arquitecturas:

1) *Arquitectura por Capas Vertical de una pasada.* (Figura 2.3(a)).

2) *Arquitectura por Capas Vertical de dos pasadas.* (Figura2.3(b)).

2. *Arquitecturas reactivas*. La principal características de este tipo de arquitectura es que el agente reacciona según sus percepciones (principio de *percepción-acción*). Es decir, dada una percepción (captada por el sensor), el agente podrá actuar (*reacción*) de forma intuitiva o "inteligente" frente a dicho evento (Figura 2.4). Generalmente los procesos internos de este tipo de agentes están jerarquizados.



(a) Arq. capas verticales, una pasada (b) Arq. capas verticales, dos pasadas

Figura 2.3: Arquitectura por Capas Verticales

Por ejemplo en las aplicaciones de robots, la información que entregan sus sensores tienen mayor prioridad que las demás tareas ya que ellas deben velar por la integridad del mismo.

3. *Arquitecturas deliberativas.* Estas arquitecturas utilizan el paradigma de los sistemas clásicos de planificación de *IA* (*percepción-planificación-acción*). El modelo más representativo es *BDI* (*Believe, Desired and Intentions*) presentada por Rao and Georgeff (1995) que caracterizan a los agentes con "*actitudes mentales*" como: *creencias*, *deseos* e *intenciones* (Figura 2.5).

   a) Las *creencias de un agente* corresponden al conocimiento que posee sobre su entorno.

   b) Los *deseos de un agente* serán los objetivos del agente.

   c) Las *intenciones de un agente* corresponden a las estrategias o acciones que debe realizar para alcanzar sus deseos u objetivos.

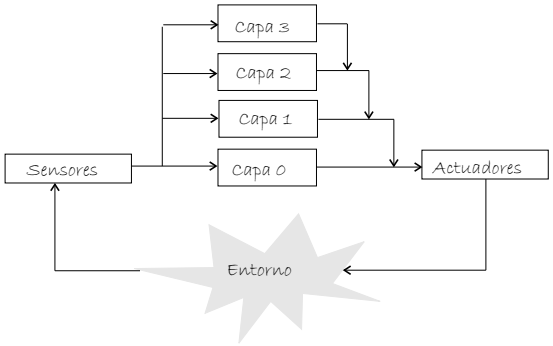


Figura 2.4: Arquitectura Reactiva

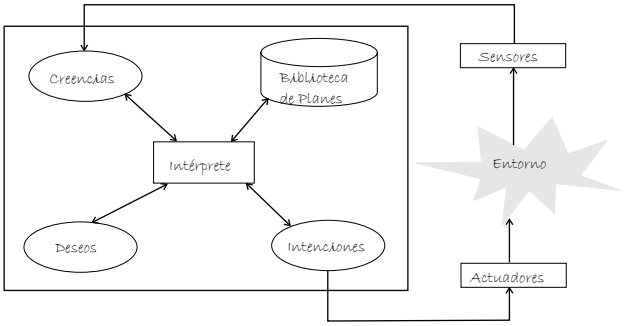


Figura 2.5: Arquitecturas Deliberativas

2.2. Sistemas Multi-Agentes

Cuando un agente necesite acceder a recursos o habilidades que no posee para lograr sus metas, necesitará utilizar la característica que hasta aquí no se ha mencionado: *sociabilidad*. Siguiendo esta línea de razonamiento Gerard (1999) y Durfee and Montgomery (1989) definen al *Sistema Multi-Agente* (*SMA*) como una agrupación de agentes autónomos que necesitan resolver un determinado problema, puesto que ellos no poseen la capacidad suficiente para lograrlo sin esta ayuda.

Se puede intuir que, las áreas de aplicación de agentes agrupados en *SMA* crece. Ferber (1999) destaca cinco áreas:

1. *Resolución de problemas*; aplicadas como una alternativa para centralizar la resolución de problemas aplicando técnicas distribuidas para la resolución de problemas. Donde agentes de distintos tipos se organizan para solucionarlo.

2. *Simulación utilizando SMA*; esta área es ampliamente utilizada para simular conductas humanas y/o naturales (por ejemplo: (Bura et al., 1993; Cambier, 1994; Castelfranchi et al., 1998, 1999)).

3. *Construcción de mundos virtuales*; aquí se representan habitualmente animaciones las cuales aspiran de alguna forma a dar pautas de validación para ciertas estrategias, como por ejemplo la robótica (Brooks, 1990; Jakobi, 1998).

4. *Robots como SMA*, donde cada robot se considera como un agente autónomo que podría unirse para trabajar bajo el mismo objetivo (por ejemplo: coordinación de equipo (Barnes et al., 1997; Han and Veloso, 1998; Portinale and Torasso, 1999; Veloso and Stone, 1998; Werger and Mataric, 2000)).

5. *Diseño de programas*, donde se utiliza los *SMA* para modelar programas eficientes.

2.3. Sistemas de Tiempo Real y Sistemas Multi-Agentes

2.3.1. Sistemas de Tiempo-Real (*STR*)

En (Stankovic, 1988) se define un *Sistema de Tiempo Real* (*STR*) como "*un sistema informático en el cual el buen funcionamiento del mismo no sólo depende de los resultados lógicos del proceso de computación, sino que también depende del tiempo en el que se obtienen dichos resultados*".

La caracterización mas relevante de los *STR* es asegurar la correcta ejecución de aquellas tareas que tienen restricciones temporales impuestas por los usuarios del sistema o bien por su entorno. Para que los *STR* puedan cumplir esta característica, las tareas deben dar a conocer sus restricciones antes de la ejecución del mismo.

Las restricciones temporales que se le solicitan a estas tareas por lo general son:

- *Deadline*, indica el plazo máximo de ejecución para que la tarea entregue una respuesta y el *STR* debe respetar. Es posible diferenciar, según sea más o menos rígido el cumplimiento de los deadlines de estas tareas, dos tipos de sistemas (Stankovic, 1992):

  - ***Sistemas de Tiempo Real Estricto***: son aquellos sistemas en que la ejecución de una tarea no debe sobrepasar su deadline, pues de lo contrario su resultado no sería útil al sistema. Este tipo de tareas se conocen como *tareas críticas.*
  - ***Sistemas de Tiempo Real no-Estricto***: son aquellos sistemas en que la ejecución de una tarea que sobrepasa su deadline se refleja en la disminución de la calidad de la respuesta que esta entrega. Este tipo de tareas se conocen como *tareas acríticas.*

- *Período*, es la frecuencia de la activación de la tarea durante la ejecución del sistema. Los *STR* utiliza este período para aplicar diversos tests que garantizarán la ejecución armónica de las tareas.

- *Tiempo de ejecución en el peor caso*, este se utiliza para enfrentarlo con el deadline y saber si las tareas críticas (que son las importantes) tienen o no tiempo para ejecutarse antes de que venza su plazo.

- *Una prioridad asignada para su ejecución*, la cual es utilizada por algunos tests de planificabilidad de las tareas críticas.

De esta forma, en los *STR* se debe tener presente que tipo de tareas se están ejecutando: críticas o acríticas, para que se les pueda planificar.

Por lo general en un *STR* se encuentran ambos tipos de tareas y su coexistencia depende de que se realice una correcta distribución del tiempo de CPU. Esta distribución dependerá en gran parte de las características de la aplicación que se esté desarrollando, no obstante lo anterior, en los *STR* siempre se debe asegurar la ejecución de las tareas críticas dentro de sus plazos (deadlines) establecidos (Stankovic, 1988). Para esto se le debe aplicar al sistema un test de planificabilidad a-priori.

Actualmente se puede encontrar diversas políticas de planificación, siendo la planificación por prioridades fijas expulsivas de (Audsley et al., 1991) la que nos interese en este trabajo, pues es la utilizada por $\mathcal{ARTIS}$. Esta política ordena las tareas presentes en el sistema según una prioridad fija asignada a priori generando con esto el *test de planificabilidad* mencionado antes. Con este test, el *STR* conocerá con anticipación cual de todas las tareas activas en el sistema es necesario ejecutar.

2.3.2. Unión de *STR* y *SMA*

La unión de los *Sistemas de Tiempo Real* (*STR*) y los *Sistemas de Inteligencia Artificial* (*SIA*) se ha presentado en los últimos años, cuando la comunidad científica vió la necesidad de aplicar a los *STR* técnicas utilizadas en los sistemas de *IA* y aportarles una mayor deliberación. Sin embargo estaba el problema de que muchos de los sistemas de *IA* son impredecibles pues sus acciones dependerán del entorno donde se implemente. El problema se extiende a los *SMA* con agentes reactivos.

Una forma de abordar este problema la propone Stankovic and Ramamritham (1993) diciendo que la nueva generación de los *STR* estrictos deberían poder flexibilizar sus características y adaptarse al empleo de las técnicas de *IA*.

Parece razonable entonces, la extensión estas características '*mas flexibles*'

desde los *SMA* hacia los *STR*, tales que las entidades que los compongan puedan tomar decisiones y cooperar entre sí, pero considerando las restricciones temporales propias de los *STR*. Este tipo de sistemas se encuentran en la literatura bajo el nombre de *Sistemas de Inteligencia Artificial en Tiempo-Real – SIA-TR* (*Real-Time Artificial Intelligence System – RT-AIS*) (Stankovic, 1992) cuyo objetivo, según Musliner et al. (1995) es desarrollar '*inteligencia en tiempo real*'. En otras palabras, aplicar las planificabilidad de los *STR*, garantizando la mejor respuesta inteligente (técnicas de *IA*) condicionada a las restricciones temporales de sus tareas en un período de tiempo dado.

Algunas aproximaciones sobre esta línea de investigación son: los algoritmos anytime de Boddy (1991), pensamiento aproximado de (Lesser et al., 1988) y el desarrollo de arquitecturas apropiadas para *STR* como: *AIS* (*Adaptative Intelligent Systems*) de (Hayes-Roth, 1995); *CIRCA* (*Cooperative Intelligent Real-Time Control Architecture*) de (Musliner et al., 1993; Ha and Musliner, 2002); los trabajos del grupo de Victor Lesser en (Lesser et al., 1988; Raja and Lesser, 2001; Horling et al., 2002); $\mathcal{ARTIS}$ (*An Real-Time Intelligence System*) de (García-Fornes, 1996).

2.4. Interacciones en los Sistemas Multi-Agentes

Como se ha expuesto hasta aquí, las aplicaciones orientadas a agentes se han extendido en diversos ámbitos. Esto impulsa a los agentes a poseer capacidades cada vez mas específicas, lo que implica invariablemente, que necesitarán de otros agentes para solventar aquellas tareas que no puedan realizar.

Los *SMA* son la respuesta a esta necesidad, reuniendo en ellos a estos agentes bajo características comunes tales como: acciones, objetivos, recursos, habilidades, etc.

Como los *SMA* son las sociedades que acogen a estas entidades, requerirán de metodologías y protocolos que normen sus interacciones. Estas *normas* van a depender directamente del tipo de interacción que tengan los agentes. El

Tabla 2.1: *Situaciones de Interacción.*

Tipo de situación	Objetivos	Recursos	Habilidad
Independiente	*C*	*S*	*S*
Colaboración simple	*C*	*S*	*I*
Obstrucción	*C*	*I*	*S*
Colaboración coordinada	*C*	*I*	*I*
Competición individual	*I*	*S*	*S*
Competición colectiva	*I*	*S*	*I*
Conflictos indiv.de recursos	*I*	*I*	*S*
Conf.Colectivo de recursos	*I*	*I*	*I*

C: Compatible *S*:Suficiente *I*:Insuficiente

tipo de situación e interacción entre los agentes va a depender de tres factores (Gerard, 1999):

- **Objetivos**. Los objetivos de los agentes de un *SMA* pueden influir en el tipo de interacción entre ellos, dependiendo de la compatibilidad o incompatibilidad. Un objetivo del agente A será incompatible con el objetivo del agente B cuando:
  $Objetivo(A) = \neg Objetivo(B)$.

- **Recursos**. La disponibilidad de los recursos determinan la interacción de los agentes, en el caso de que no estén disponibles ellos deberán buscar la forma de acceder a estos recursos considerando también el entorno en que se encuentran.

- **Habilidad**. La situación que se presente dependerá de la capacidad de los agentes para realizar las distintas tareas que le permiten alcanzar sus objetivos.

La combinación de los factores de interacción antes mencionados (Tabla

2.1), determina la situación que se le presenta al agente y por lo tanto las tácticas y/o protocolos que utilice para actuar en el *SMA*. Según esta combinación el agente podrá actuar básicamente de dos formas:

1. *Acción individual.* Cada agente del *SMA* tendrá sus propias metas y objetivos sin que estos formen necesariamente, parte del objetivo global del *SMA* al que pertenecen. Para optar por una estrategia individualista, el agente deberá contar con los recursos y habilidades suficientes para alcanzar sus objetivos sin la intervención de otros agentes. Cuando esto no ocurra, el agente se ve en la necesidad de buscar ayuda y podrá optar por dos tipos de acciones:

   a) *Colaborativa*: esta se producirá cuando los objetivos de los agentes involucrados sean compatibles. Ellos cooperarán y colaborarán entre sí para alcanzar sus respectivas metas (*protocolos de cooperación*).

   b) *Competitiva*: esta se producirá cuando los objetivos del los agentes involucrados sean incompatibles y los recursos en el *SMA* sean escasos, tales que los agentes deberán competir por los recursos (*protocolos competitivos).*

2. *Acción conjunta.* En este caso los agentes actúan agrupándose bajo un objetivo común. No obstante, cada agente podrá tener sus propios objetivos (sub-objetivos del *SMA*) que formarán parte del objetivo global del *SMA* al que pertenecen. En un mismo entorno pueden existir varios *SMA* y podría ocurrir la falta de recursos en uno de ellos, o los objetivos de los mismos sean incompatibles, o bien las habilidades de sus agentes sean insuficientes; en estos casos se podrán tener dos tipos de interacciones entre sus agentes:

   a) *Colaborativa – Cooperativa*: Se producirá cuando los objetivos de los agentes del mismo o bien distintos *SMA* sean compatibles. En este caso ellos podrán actuar en forma conjunta y coordinada para

alcanzar sus objetivos particulares, y así lograr los objetivos del *SMA* al que pertenecen (*protocolos de cooperación y coordinación*).

b) *Colaborativa – Competitiva*: Se producirá cuando los agentes de distintos *SMA* se reunan según la compatibilidad de sus objetivos (Gerard, 1999) pudiendo así *"atacar"* a aquellos agentes que tengan objetivos incompatibles con el grupo al que pertenecen (*protocolos de competición*).

Resumiendo, en los *SMA* los protocolos de interacción serán necesarios solo cuando el agente necesite interactuar con otros agentes. De esta forma encontramos en la bibliografía los siguientes protocolos que rigen estas interacciones (Gerard, 1999):

1. **Protocolos de cooperación**. Estos protocolos son utilizados por los agentes cuando los recursos y/o habilidades son insuficientes y sus objetivos son compatibles (Stone et al., 1999; Sen and Durfee, 1994; Kaminka and Tambe, 1999; Kloos et al., 2001; Werger and Mataric, 2000). Una estrategia básica de intercambio usada por los protocolos de cooperación es la descomposición y distribución de ta-reas también llamada "*divide y vencerás*", pueden reducir la complejidad de las tareas (las subtareas requieren una menor capacidad de parte del agente y utiliza menos recursos). Sin embargo esta descomposición deberá considerar las capacidades de cada agente y los recursos disponibles en el sistema. Algunos métodos utilizados en los protocolos de cooperación son:

   a) *Agrupación*: los agentes se agrupan de acuerdo a intereses comunes para alcanzar sus objetivos.

   b) *Especialización*: existen agentes especialistas en tareas específicas para el beneficio del *SMA*.

   c) *Distribución*: colaboran en la distribución y asignación de recursos y tareas.

d) *Coordinación*: los *SMA* mantienen una coordinador y un planificador de las tareas y recursos del sistema.

e) *Resolución*: los *SMA* resuelven los conflictos con mediadores y/o negociaciones.

2. **Protocolos de Coordinación**. Igual que el anterior, este protocolo se utiliza cuando las habilidades y/o recursos son escasos en el *SMA*, los objetivos de los agentes son compatibles (Barbuceanu and Fox, 1966, 1996; Pir).
   En este tipo de protocolos, el agente tiene un grado de autonomía para decidir sobre sus acciones. Sin embargo, esto implica que desconocen las acciones de los demás agentes del sistema, dificultando la coordinación del mismo. Lo anterior provoca que los agentes tengan una visión parcial del sistema y por lo tanto una perspectiva imprecisa del mismo. Por esta razón, estos protocolos deben considerar todas las tareas a realizar y coordinarlas de tal forma que no se ejecuten acciones indeseables. En otras palabras, se necesita una planificación a-priori de todas tareas para predecir los comportamientos de los agentes e intercambiar resultados que ayuden a la solución del problema general del *SMA*.
   La coordinación de un *SMA* se puede separar en:

   a) *Coordinación global*, en las cuales existe un plan global que abarcará las acciones de todos los agentes del *SMA*.

   b) *Coordinación individual*, aquí existe un plan global pero cada agente lo ajusta a sus necesidades.

   c) *Coordinación por sincronización*, existe una secuencia básica de las acciones a seguir por los agentes. Los posibles conflictos entre ellos se resuelven a medida que se presenten.

   d) *Coordinación por planes*, se generan planes para coordinar la secuencia de acciones de los agentes y así determinar a-priori posibles conflictos.

e) *Coordinación reactiva*, se generan los planes de las acciones de los agentes, y se resuelven los conflictos en tiempo de ejecución.

f) *Coordinación por regulación*, se determinan a-priori una serie de reglas que "solucionarían" los posibles conflictos entre los agentes del sistema.

3. **Protocolos de Competición**. Este tipo de protocolo se utiliza principalmente cuando los objetivos de los agentes son incompatibles y los recursos existentes son escasos. Puede haber dos tipos de competición: *colectiva* e *individual*. Ambas se basan principalmente en la Teoría de Juegos (Raiffa, 1982; Cabrales, 2001; Kraus, 2001b; Axelrod, 1997; Krause, 2002). En esta teoría los agentes actúan según unas estrategias seleccionadas para obtener la mejor utilidad con cada acción que tome, de igual forma el oponente seleccionará sus estrategias a seguir. Como ambas partes buscan lo mejor para ellos, el sistema completo alcanza un estado de equilibrio. Se pueden encontrar en la literatura dos tipos de "*juegos*":

   a) Juego estratégico o no cooperativo (por ejemplo: ajedrez).

   b) Juego cooperativo (por ejemplo: fútbol de Veloso and Stone (1998) entre otros).

4. **Protocolos de Negociación**. Estos protocolos se utilizan cuando cualquiera de los tres factores planteados: objetivos, recursos o habilidades, es insuficiente en el *SMA* (Tabla 2.1).
   Este protocolo está orientado a que los agentes participantes alcancen un acuerdo beneficioso para ambas partes, utilizando distintos métodos y técnicas de negociación.
   La cantidad mínima de agentes para establecer una negociación serán dos, pudiendo extenderse dependiendo del tipo de negociación involucrado (Raiffa, 1982).

En el siguiente capítulo se explicará con mayor detalle el estado del arte de este tipo de protocolos, poniendo énfasis a las técnicas de negociación entre agentes inteligentes.

Capítulo 3

Negociación en *SMA*

En este capítulo se introducirán algunos conceptos generales que se utilizarán en las técnicas de negociación que se expondrán más adelante.

Se comienza explicando el concepto de negociación como tal, y como éste se introduce en los *Sistemas Multi-Agentes* (*SMA*) utilizando y adaptando estos conceptos a sus respectivos entornos.

También se explican las distintas estrategias que se pueden encontrar y utilizar en negociaciones de *SMA*, para terminar con algunas aplicaciones y líneas de investigación que se pueden encontrar actualmente en la comunidad científica.

3.1. Conceptos Generales

La negociación es uno de los métodos más extendidos y utilizados para resolver de forma racional todo tipo de conflictos entre los seres humanos.

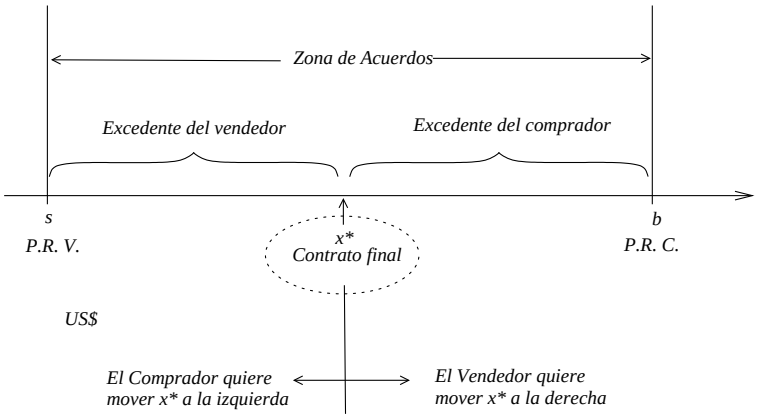


Figura 3.1: Zona de acuerdos entre los compradores

En (Enc, 2005) definen negociación como "*el conjunto de reuniones, gestiones, discusiones que se dan entre las partes implicadas y cuyo objetivo final es conseguir un acuerdo que sea lo más positivo posible para el conjunto de las partes*", considerando a las "*partes implicadas*" aquellas que están en algún tipo de conflicto.

En toda negociación hay dos partes involucradas: *vendedor y comprador*, y para que se produzca un acuerdo efectivo entre las partes deberá existir una *"zona de acuerdos"* (Raiffa, 1982). Esta zona de acuerdos estará comprendida entre dos puntos (ver Figura 3.1) los cuales son:

- *Precio Reservado del Vendedor (PRV)*: que está definido como el menor valor que aceptará el vendedor por su recurso; y
- *Precio Reservado del Comprador (PRC)*: que será el mayor valor que ofrecerá el comprador por lo que desea.

Es en esta zona donde finalmente se encontrará el valor del *contrato final* (x^*) del acuerdo alcanzado por ambas partes.

Cuando las partes involucradas deciden negociar, se deben establecer ciertos parámetros antes de comenzar con la negociación en sí (Raiffa, 1982), estos son:

1. *El Número de Integrantes* involucrados en la negociación deben ser al menos dos: comprador y vendedor. Con las siguientes posibles combinaciones:

   a) un vendedor - un comprador,

   b) un vendedor - varios compradores,

   c) varios vendedores - un comprador,

   d) varios vendedores - varios compradores.

2. *El Tipo de Negociación* que se establecerá entre ambas partes determinará el tipo y forma de seleccionar la estrategia que van seguir. Algunos tipos de negociación son: subastas, redes de contratos, juego de la escalada o subastas del tipo ascendente, negociación con media-dor, negociación por persuasión, negociación con argumentación.

3. *La Presencia de Terceros en la negociación*, será necesaria en caso de que una de las partes no sea la oficial, es decir, es representante del interesado oficial, y es éste último quien dará en definitiva su aprobación o rechazo a un determinado acuerdo.

4. *Las Restricciones de Tiempo* (si las hubiese) para dar respuestas a las distintas ofertas y contra-ofertas que lleguen, así como también los posibles *límites de coste* en que podrán variar las ofertas, y si las negociaciones serán *públicas* o *privadas*.

5. *La Necesidad de un Mediador para la negociación* si las partes involucradas lo estiman conveniente. Por ejemplo en el caso de subastas puede haber un mediador, que no necesariamente sería el vendedor, llamado *martillero*.

6. *Los Tipos de Estrategias* que se utilizarán en la negociación van a depender de los puntos anteriores. Ellos determinarán si la estrategia es pasiva o competitiva.

Una vez que han aclarado estos puntos, se deben establecer las reglas y normas que regirán la negociación, estas últimas también dependerán de los puntos anteriores, variando desde *cooperativa – colaborativa* a *competitiva – no-colaboratotiva.*

3.2. Negociación en Sistemas Multi-Agentes

Los modelos de negociación que utilizan los *SMA* están orientados a replicar las negociaciones humanas y sus interacciones. Por esta razón, no es irracional pensar que necesitarán, al igual que las sociedades humanas, reglas que normen sus negociaciones y regulen las interacciones tales que puedan alcanzar acuerdos beneficiosos para ambas partes.

Como se mencionó en el capítulo anterior, un protocolo de negociación es un conjunto de reglas públicas las cuales dictan las conductas entre los agentes cuando están involucrados en una negociación.

Una de las metas de los investigadores de esta área fue dar sentido a la frase *"negociación entre agentes"*. Algunas definiciones que se pueden encontrar son:

- Meyer (1998) define la *negociación entre agentes* como una comunicación interactiva para coordinar actividades.

- Lomuscio et al. (2001) la definen como un proceso de comunicación entre dos grupos de agentes por el cual tratan de llegar a un acuerdo aceptable y beneficioso para ambas partes.

- Faratin (2000) la define como un proceso por el cual dos o mas partes llegan a una decisión.

- Los autores (Wooldridge and Parsons, 2000; Reilly and Bates, 1995; Barbuceanu, 1999; Jennings et al., 2000; Nwana et al., 1996) coinciden en que la *negociación entre agentes* es un tipo de coordinación entre agentes que están en conflicto, de tal forma que *negociando entre ellos* alcancen un acuerdo común y beneficioso para ambas partes. Aunque también se puede presentar que este "*acuerdo beneficioso*" sea que no llegasen a ningún acuerdo.

Para efecto de este trabajo de tesis, se definirá *"negociación entre agentes"* como:

> *Una interacción entre dos o mas agentes para resolver algún problema común. Este proceso de negociación deberá tener restricciones temporales, reglas y normas pre-establecidas, tales que se pueda determinar la duración máxima de la misma.*

Todo esto con el fin de insertar apropiadamente estas negociaciones en la arquitectura del agente $\mathcal{ARTIS}$.

3.2.1. Estrategias de negociación

Como se adelantaba en la sección anterior, los agentes necesitarán establecer y utilizar reglas y normas para alcanzar acuerdos y obtener beneficios mutuos. En otras palabras, el agente necesitará saber como generar o seleccionar una secuencia de acciones tales que el proceso de negociación le sea favorable y alcance una utilidad máxima, siguiendo dichas reglas o normas.

Para resolver un problema determinado, un agente puede tener varias secuencias de acciones posibles a seguir, y cada una de estas secuencias producirá diferentes resultados. De esta forma, el principal problema de un agente racional se transforma en responder las preguntas:

- *¿Cómo generar esta "secuencia de acciones"?*

- *¿Cómo seleccionar una de esas secuencias de acciones?*

La secuencia de acciones que tiene disponible el agente se conoce como estrategia (Kraus, 2001a,b), y la selección de una determinada estrategia estará siempre orientada a la utilidad que el agente desea conseguir. No obstante, para tomar la mejor decisión sobre que estrategia seguir, también deberá considerar:

- *Racionalidad*: es una característica que calcula a-priori la *utilidad esperada* con una determinada acción. Así, se elige la acción "*mas óptima*" considerando maximizar la *utilidad esperada*.
- *Eficacia*: es una característica calculada a-posteriori y dependerá del esfuerzo involucrado en la acción y la *utilidad efectiva* que se obtiene al aplicar dicha acción sobre el entorno. La acción mas eficaz es la que con el menor esfuerzo produce una utilidad mayor.
- *Viabilidad*: representa la viabilidad de realizar una acción y el coste asociado a cada una de ellas.
- *Eficiencia*: es el grado de calidad de la operativización de los problemas. Mide la relación existente entre las acciones seleccionadas y las que se podrían haber seleccionado para obtener una mayor utilidad.
- *Consistencia*: representa la calidad en la instrumentalización de los problemas, y expresa la relación entre las reacciones esperadas y las reacciones producidas.
- *Valor de una acción*: expresa la relación entre la utilidad esperada y la utilidad real obtenida.
- *Distribución*: expresa que la decisión de la selección podría ser distribuida.
- *Simplicidad*: para los *SMA* los procesos de decisión simples y eficientes son preferibles sobre los complejos, pues los primeros toman menor tiempo en su procesamiento.

- *Estabilidad*: se considera que una determinada estrategia de negociación es estable, si esta fue seleccionada de un conjunto de estrategias conocida por todos los agentes tales que puedan calcular *a-priori* sus funciones de utilidad esperadas.

En general, las selección de determinada estrategia por parte de los agentes esta basada en sus beneficios personales que esta estrategia les proporcione. Es decir, un agente selecciona una determinada estrategia, si y solo si, esta estrategia le produce algún tipo de beneficio. Sin embargo, como todos los agentes participantes de una misma negociación tienden a la misma línea de selección (maximizar la utilidad de cada uno) al final el sistema alcanza un estado de equilibrio, este se conoce con el nombre de *Equilibrio de Nash* (Kraus, 2001a; Cabrales, 2001; Krause, 2002).

Sin embargo para algunos *SMA*, alcanzar este estado de equilibrio no siempre es posible ya que las negociaciones entre agentes autónomos se podría realizar en dominios dinámicos del mundo real, con lo cual tener un conjunto finito y determinado de estrategias a seguir por los agentes del *SMA* no sería viable.

En esta tesis se definirá la *selección de la estrategia* como la maximización de la *utilidad esperada* del agente participante, considerando los siguientes factores:

- Las estrategias utilizadas por los agentes participantes de las negociaciones son finitas y conocidas
- El proceso de negociación tendrá un límite de tiempo
- El cálculo de la utilidad esperada considerará las restricciones temporales de las entidades del agente $\mathcal{ARTIS}$.

Definición.
Sean:
$E = \{e_1, e_2, \ldots, e_k\}$, *el conjunto de posibles entornos de un agente.*

$Ac = \{ac_1, ac_2, \ldots, ac_n\}$, *el conjunto de posibles acciones que puede realizar el agente sobre un determinado entorno.*

$R = \{r_1, r_2, \ldots, r_n\}$, *las posibles reacciones del entorno dada una determinada acción(es) del agente.*
La función para seleccionar una determinada estrategia, f, *estará definida por la ecuación (3.1).*

$$\forall r_j \in R, \wedge a_i \in A, \exists\, f(r_i, a_i) = \begin{cases} a_i; & ssi\, la\, utilidad\, esperada\, se\, maximiza \\ 0 & en\, otro\, caso \end{cases} \tag{3.1}$$

□

Estas técnicas de negociación, protocolos y estrategias a seguir se explicarán con mayor detalle en el capítulo 5.

3.3. Teorías de Negociación Aplicada en Agentes

En la literatura se pueden encontrar algunas aproximaciones que desarrollan teorías sobre la negociación entre agentes. Una de ellas es la *Teoría Formal de Negociaciones* basada en la *Teoría de Juegos* (Kraus, 2001b; Axelrod, 1997). Los investigadores de esta área analizan las distintas situaciones que puede tener un agente, el resultado de estas situaciones proporcionan los datos para seleccionar la estrategia que se va a utilizar. No obstante, esta estrategia solo se puede aplicar entre agentes que sigan protocolos de negociación con reglas rígidas, es decir, que el entorno no presente grandes cambios (por ejemplo: juego de ajedrez).

Otra línea de investigación esta basada en la *Teoría Informal de Negociación*. En esta teoría se pueden identificar los posibles beneficios generales de utilizar una determinada estrategia de negociación (Parsons et al., 1998; Sierra et al., 1997; Sycara, 1990).

3.3.1. Teoría de Juegos

Esta teoría se utiliza para la toma de decisiones racionales en situaciones de conflicto, tratando siempre de alcanzar la máxima utilidad con las acciones seleccionadas, considerando que las estrategias y acciones seleccionadas por los demás jugadores también maximizan sus respectivas utilidades.

En otras palabras, la *Teoría del Juego* utiliza herramientas matemáticas que analizan las interrelaciones entre dos o más agentes y busca un modelo donde la selección de la siguiente acción sea óptima.

Este tipo de teoría trabaja sobre los supuestos de que todos los participantes buscan la *máxima utilidad* con cada una de sus acciones (Ecuación 3.1).

La *Teoría del Juego* presenta dos variantes:

1. ***Estratégico o no-cooperativo***. También llamado competitivos o de suma cero. En este tipo de juegos se detallan claramente las reglas del mismo. Cada agente buscará una estrategia óptima y sus decisiones son individuales, sin embargo en ellas está incorporada la relación que tenga con los demás agentes: cooperación o rivalidad (Kraus, 2001a; Guttman and Maes, 1998).

2. ***Cooperativo o Coalicional***. Este tipo de teoría describe la conducta óptima en juegos con muchos jugadores. La formación de estos modelos cooperativos son consistentes con conductas racionales. Los agentes eligen e implementan acciones en conjunto (Axelrod, 1997; Kraus, 1997; Veloso and Stone, 1998).

Los elementos esenciales del juego son:

1. ***Agentes-Jugadores***. Son los agentes que toman las decisiones tratando de obtener el mejor resultado posible.
   La meta de cada *agente-jugador* es maximizar su utilidad por medio de sus acciones.

2. ***Acciones***. Son todas las acciones que el *agente-jugador* tiene disponible para alcanzar sus objetivos.

3. ***Información***. Es el conocimiento que tiene un *agente-jugador* en un determinado momento de los valores de las diferentes variables en juego de su entorno.

4. ***Estrategia***. Es un conjunto de acciones en particular que puede tomar un *agente-jugador* en cada momento del juego dada la información disponible. Estas estrategias también las podrá *ordenar* o *combinar* para conseguir la máxima utilidad con sus acciones.
   Para esto, el *agente-jugador* tiene a su disposición:

   a) *Conjunto de estrategias*, son todas las estrategias disponibles en un determinado momento.

   b) *Perfil de estrategias*, es un conjunto de acciones de una estrategia por cada uno de los jugadores del juego.

5. ***Utilidad esperada***, es la que motiva la acción. Representa la utilidad que espera alcanzar un *agente-jugador* dada la selección de una determinada estrategia.

6. ***Utilidad alcanzada***, es la utilidad que reciben los *agentes-juadores* al terminar el juego. En otras palabras, es la evaluación posterior sobre si el objetivo buscado fue alcanzado.

7. ***Resultados***, estos a estar relacionados con la utilidad que perciba o espera percibir el *agente-jugador*, utilizando una determinada estrategia.

8. ***Equilibrio***, es el perfil integrado por las mejores estrategias de cada *agente-jugador*.

   a) *Equilibrio Económico*, será el conjunto de precios que equilibra el mercado.

b) *Equilibrio de la teoría de juegos*, es el perfil de estrategias que genera un resultado de equilibrio.

9. ***Concepto o solución de equilibrio***, estas son las normas que generan el equilibrio basadas en los prefiles disponibles y los resultados esperados de los *agentes-jugadores*. Consideraremos tres tipos:

   a) ***Estrategia Dominante***. Es cuando la mejor estrategia del *agente-jugador* no depende de las acciones del resto de los *agentes-jugadores*. En otras palabras, es la mejor respuesta de un *agente-jugador* frente a cualquier estrategia seleccionada por sus oponentes, ya que la utilidad que obtendrá será siempre la máxima.

   b) ***Juegos sin Estrategia Dominante***. Es cuando los *agentes-jugadores* no tienen estrategias dominantes, es decir la selección de sus estrategias dependerá en todo instante de las estrategias que utilicen los demás *agentes-jugadores*.

   c) ***Equilibrio de Nash***. Se presenta cuando la estrategia seleccionada por cada uno de los *agentes-jugadores* es la óptima, es decir, maximiza la utilidad del *agente-jugador* dado que el resto de los *agentes-jugadores* también seleccionaron sus estrategias siguiendo las mismas reglas (maximizando sus propias utilidades).

3.3.2. Modelos de Negociaciones Bilaterales

En el modelo de negociación bilateral (Raiffa, 1982) se puede identificar las dos partes involucradas en la negociación llamadas *participantes*, sobre algún tema o recurso llamado *objeto de la negociación*. Esto generará un *proceso de negociación* entorno a un estado particular del mundo y un *modelo de negociación*. Con este escenario se puede definir una serie de posibles estados del mundo a los que puede llegar el agente según las estrategias utilizadas.

También se puede diferenciar dos tipos de negociación bilateral:

1. *Negociación bilateral genérica.* En este tipo de negociación encontramos:

   a) *offered*: es el objeto de la negociación no puede ser cambiado.

   b) *proposed*: son las propuestas hechas por los participantes.

   c) *requested*: son las respuestas a las propuestas, permite enviar: una propuesta, oferta o sugerencia.

2. *Negociación bilateral expandida.* En algunos tipos de negociaciones las características del modelo anterior serán insuficientes, y deberán expandir las posibilidades de interacción entre los agentes participantes en la negociación. Por ejemplo, algunas posibilidades de expansión sería dotar al agente del conocimiento necesario para la argumentación de sus posturas, y así influenciar en las creencias del otro agente para alcanzar ambos sus metas (Parsons et al., 1998; Sierra et al., 1998).

3.3.3. Negociación Multi-lateral

Este tipo de negociaciones se utiliza principalmente cuando hay mas de dos participantes. En el modelo de negociación Multi-lateral se definen protocolos para reunir diversas mociones las cuales serán expuestas y evaluadas mediante un quórum por medio de las votaciones de los agentes pertenecientes a la comunidad.

3.3.4. Subastas

Evidentemente, en toda negociación deberá existir una *moneda de cambio* o *sistema monetario*(Kraus, 2001a; Engelbrecht-Wigganns, 1983) que permita a los participantes valorar su participación y a la vez utilizarla como sistema de intercambio para resolver sus conflictos.

Las subastas son un ejemplo perfecto de este intercambio monetario. En ellas se puede apreciar que, frente a un determinado producto, se produce una

interacción entre compradores y vendedor tales que el producto es adjudicado finalmente al comprador con la mejor oferta[1].

Actualmente podemos encontrar aplicaciones virtuales (Guttman and Maes, 1998), así como electrónicas (E-Bay, 2001), que tienen un elevado interés en las subastas.

Se pueden diferenciar dos tipos de patrones en las subastas:

- El protocolo de subastas *uno-a-muchos*, es el más común, consiste en que un agente inicia una subasta y otros agentes pueden participar en ella (Andersson and Sandholm, 1998; Funes et al., 1998; Sandholm, 1993).
- El protocolo de subastas *muchos-con-muchos,* consiste de varios agentes iniciando la subasta y varios agentes participando de ella (Wurman et al., 1998).

En ambos casos, primero se necesita determinar el tipo de protocolo que se utilizará para la subasta, como por ejemplo los protocolos propuestos por FIPA (FIPA00031; FIPA00029; FIPA00032).

Una vez establecido el protocolo a seguir, el agente necesita decidirse por una determinada estrategia para generar sus ofertas.

Por lo general, en los protocolos de subastas *uno-a-muchos* los agentes utilizan: Subasta Inglesa, Subasta del Sobre Cerrado, Subasta de Vickrey, Subasta Holandesa, también conocida como Subasta Alemana o Subasta a la baja.

Y cuando hablamos del protocolo de subastas *muchos-a-muchos*, como una aproximación al modelo de negociación Multi-lateral, entonces utilizan un mediador (martillero) que la dirige.

Tanto los modelos de negociación bilateral, multilateral y subastas, pueden ser expandidos según la necesidad de los participantes. Los agentes negociadores pueden utilizar teorías formales como motor de inferencias para la aplicación de los protocolos de negociación.

[1] Subasta es una "*venta pública en la que se exponen objetos de valor que son adjudicados al que más dinero ofrece por ellos*"Enc (2005).

Estos protocolos de negociación estarán compuestos por un conjunto de estrategias (públicas o privadas) de las cuales el agente participante podrá optar. Para esto contará con diversos modelos o teorías en las cuales siempre primará la estrategia que maximice su utilidad.

No obstante, la estrategia seleccionada podría tener derivaciones dependiendo si se trata de un modelo formal de negociación o informal. Por esta razón es importante conocer a-priori el modelo o protocolo a utilizar, pues de ello dependerá la siguiente acción a tomar por el agente participante.

3.4. Modelos de Negociación Desarrollados en *SMA*

Existen varias aproximaciones de modelos de negociación para la resolución de problemas distribuidos, donde los agentes están cooperando en un *SMA* con intereses propios. Estos trabajos están orientados a las posibles soluciones de planificación distribuida utilizando la negociación.

- Moehlman et al. (1992) y Lander and Lesser (1992) utiliza la negociación como herramienta para la planificación distribuida: cada agente tiene ciertas restricciones y trata de encontrar las posibles soluciones utilizando procesos de negociación. Estos procesos son búsquedas que son traducidas a una negociación de múltiples estados considerando la cooperación de los agentes mientras buscan y resuelven conflictos entre ellos.

- Rosenschein and Zlotkin (1994); Zlotkin and Rosenschein (1992, 1996) identifican tres tipos de dominios para la negociación entre agentes, estos son: *dominios orientados a tareas* (DOT), *dominios orientados a valores* (DOV), y *dominios orientados a estados* (DOE):

  - *Dominios orientados a tareas*: es decir las actividades del agente estarán guiadas por tareas sin importar la interferencia de otros agentes. El agente cuenta además con los recursos necesarios, pudiendo redistribuir las tareas para una mayor eficiencia. En el caso

de negociación en este tipo de dominio, se establecerán un conjunto de reglas (protocolo) y se propone un plan (acciones o tareas a seguir), este plan se acepta cuando maximice la utilidad del agente; si no es así se propone otro plan. Así hasta que alcance la utilidad máxima para el agente o bien rehace la propuesta en cuyo caso solo ejecutará sus tareas.

- *Dominios orientados a valores*: En este tipo de dominios los agentes asignan un valor a cada estado potencial que representa el mayor o menor grado de '*deseo*' de dicho estado para el agente. La ventaja principal de este tipo de dominios es que la función de valor permite que los agentes se comprometan con sus metas, tales que podrán relajar sus metas con la finalidad de conseguir coordinar sus acciones.
- *Dominios orientados a estados*: En este caso es necesario considerar la presencia de otros agentes, pues las actividades de los agentes se definen como planes conjuntos de los mismos motiva por ejemplo por la falta de recursos entonces deberán organizarse para lograr sus metas.

- Sycara (1987, 1990) presenta en sus trabajos un modelo de negociación que combina razonamiento y optimización. Los agentes involucrados tratan de influenciar en las metas e intenciones de sus oponentes utilizando la persuasión.
- Kraus and Lehmann (1995) desarrollan sus trabajos basándose en la *"teoría del juego"* utilizando la *"diplomacia"* para llegar a acuerdos razonables para ambas partes.
- Sierra et al. (1998); Lesser (1995); Parsons et al. (1998) presenta un modelo de negociación entre agentes autónomos que *razonan* y *llegan a acuerdos* utilizando *argumentos* en sus discusiones y ofrecimientos para obtener un determinado servicio.

- Zeng and Sycara (1996) considera la negociación en un entorno de mercado con un proceso de aprendizaje en el cual el comprador y el vendedor actualizan sus creencias sobre los precios reservados (Raiffa, 1982) utilizando reglas bayesianas.

- Sandholm and Lesser (1995) abordan problemas de consolidación que se presentan en la negociación automática entre agentes interesados en si mismos, cuya racionalidad está limitada por la complejidad computacional.

- En trabajos de Chavez and Maes (1996); Sandholm and Lesser (1995); Tsvetovatyy et al. (1997); Wurman et al. (1998); Camarinha-Matos and Afsarmanesh (2001), se pueden encontrar aplicaciones de negociación en comercio electrónico. Algunos utilizan un agente llamado *"Asistente Personal"* para determinar las preferencias del usuario y buscar en la web según esas preferencias. Esto permite que el agente negocie los términos del pre-contrato ya que solo el usuario puede dar la aceptación final.

Capítulo 4

Arquitectura $\mathcal{ARTIS}$

En este capítulo se explicará con detalle la arquitectura y funcionamiento de un agente $\mathcal{ARTIS}$ ($\mathcal{AA}$) puesto que es en esta arquitectura donde se incorporan nuestras técnicas de negociación. Este capítulo se estructura de la siguiente forma:

- Génesis de la arquitectura $\mathcal{ARTIS}$.
- Características del agente $\mathcal{ARTIS}$.
- Descripción de la arquitectura del agente $\mathcal{ARTIS}$.

4.1. Génesis de $\mathcal{ARTIS}$

$\mathcal{ARTIS}$ (*An Real-Time Intelligence System*) es una arquitectura implementada por el Grupo de Tecnología Informática (GTI) de la Universidad

Politécnica de Valencia, propuesta inicialmente en la tesis doctoral de García-Fornes (1996).

Sobre esta arquitectura se han desarrollado múltiples proyectos tanto para el desarrollo de aplicaciones como para el desarrollo complementario de la arquitectura en si (Carrascosa et al., 2003b; Soler-Bayona, 2003; Soler et al., 2000; Julián et al., 2000).

$\mathcal{ARTIS}$ une las dos líneas de investigación mencionadas en el capítulo 2, estas son las técnicas de la Inteligencia Artificial (IA) con los Sistemas de Tiempo-Real (STR). Las aplicaciones desarrolladas con esta arquitectura son aquellas que mezclan tareas que tienen restricciones temporales críticas con tareas propias del sistema al que pertenecen.

Como se comentó en la sección 2.3.1, lo mas importante en los STR es poder garantizar la correcta ejecución de las tareas críticas del sistema en armonía con las tareas no-críticas. Para garantizar esto, $\mathcal{ARTIS}$ aplica un test de planificabilidad (García-Fornes et al., 1997) a estos dos tipos de tareas tal que asegura que su ejecución estará dentro de los tiempos pre-establecidos.

De esta forma un *agente* $\mathcal{ARTIS}$ contendrá estos dos tipos de tareas y deberá distribuir el tiempo de CPU entre ellas, asignando mayor prioridad a la ejecución de las tareas críticas del *agente* $\mathcal{ARTIS}$, cuyos tiempos de ejecución son generalmente bajos, para luego ejecutar las tareas del sistema.

En otras palabras, con $\mathcal{ARTIS}$ se pueden construir aplicaciones de IA utilizando Agentes Inteligentes de Tiempo Real Estricto.

4.2. Características de un Agente $\mathcal{ARTIS}$

La principal característica de un *agente* $\mathcal{ARTIS}(\mathcal{AA})$ es que garantiza el cumplimiento de las restricciones temporales definidas por el usuario para sus componentes o tareas críticas (deadline, períodos). También proporciona el soporte necesario para la ejecución de sus componentes opcionales, los cuales mejorarán la calidad de la respuesta entregada por el $\mathcal{AA}$ (Botti et al., 1999;

Carrascosa et al., 1997).

$\mathcal{ARTIS}$ es una extensión del modelo *Blackboard* (Nii, 1986b,a) adaptado para trabajar en entornos de Tiempo-Real Estricto (*Hard Real-Time*). Dentro de la clasificación general de arquitecturas de agentes comentada en la sección 2.1.3, $\mathcal{ARTIS}$ se podría clasificar como arquitectura híbrida vertical, puesto que puede gestionar dos tipos de tareas: críticas y no-críticas en un mismo sistema.

Un $\mathcal{AA}$ se puede describir desde tres puntos de vistas (Carrascosa et al., 2003a):

1. *Modelo formal*, que se refiere a la especificación formal de las entidades que conforman un $\mathcal{AA}$.

2. *Modelo de usuario*, que se refiere a la filosofía conceptual que existe detrás del $\mathcal{AA}$.

3. *Modelo de sistema*, aquí se refiere a la ejecución del $\mathcal{AA}$.

4.3. Modelo Formal

El modelo formal define y explica formalmente la arquitectura interna del $\mathcal{AA}$ junto con las respectivas operaciones que se utilizan sobre él.

El desarrollo del modelo formal de un $\mathcal{AA}$ así como de una metodología de diseño orientado a $\mathcal{AA}$ se explica más extensamente en (Julián et al., 1999), sin embargo aquí se presentará un breve descripción del mismo por su relevancia en esta tesis.

Definición.
Un Agente $\mathcal{ARTIS}$ es una estructura de la forma:

$$\mathcal{AA}_i = \{Behav_i, G_i, Blv_i, \Gamma_i\}$$

Donde:

- $Behav_i$: *es un conjunto de diferentes comportamientos del* $\mathcal{AA}_i$ *para diferentes situaciones complejas del entorno.*
- $G_i = \left\{ g_1^{[t_1,t_2]}, g_2^{[t_2,t_3]}, \ldots, g_n^{[t_n,t_m]} \right\}$: *un conjunto de objetivos a alcanzar por el agente. Cada objetivo es un estado deseable y puede estar acotado temporalmente,* $[t_i, t_j]$ *con* $i < j$*, en cuyo caso se denotaría el intervalo temporal en el cual el objetivo debe ser alcanzado.*
- Blv_i: *es el conjunto de creencias (believes) del* $\mathcal{AA}_i$ *en el instante actual. Todos los datos en este conjunto disponen de una etiqueta temporal mediante el empleo de una lógica temporal tal y como se expresa en* (Crespo et al., 1994) *y* (Barber et al., 2003).
- $fblv_i$: *es una función de selección que determina el comportamiento deseado del* $\mathcal{AA}_i$*en función de sus objetivos y creencias actuales (implementada por el Módulo de Control del* $\mathcal{AA}$*).*□

Además, cada *in-agent* a_{ij} estará definido como:

Definición.
Un <u>*in-agent*</u> *es una estructura de la forma:*

$$a_{ij} = \langle \Gamma_{ij},\ Ar_{ij},\ \beta_{ij},\ fr_{ij},\ C_{ij},\ fc_{ij},\ D_{ij},\ T_{ij} \rangle$$

Donde:

- Γ_{ij} : *es el conjunto de percepciones del* $\mathcal{AA}_i$ *y las utiliza para actualizar temporalmente sus creencias que representan el estado interno y el entorno del in-agent.*

- *Ar_{ij} : es la lista de todas las posibles acciones reflejas conocidas por el in-agent.*
- *β_{ij}: es el conjunto de creencias que representan los estados del entorno e interno del in-agent. Estas creencias complementan las creencias del $\mathcal{AA}_i$ (Blv_i) al que pertenece.*
- *fr_{ij} : es la función que selecciona una acción refleja desde la lista Ar_{ij}.*
- *C_{ij} : es la lista de todas las posibles acciones cognitivas (deliberativas) conocidas por el in-agent.*
- *fc_{ij} : es la función que selecciona una acción deliberativa desde la lista C_{ij} según sea el estado percibido en β_{ij}, D_{ij} y una acción inicial resultante de fr_{ij}.*
- *D_{ij} : es el deadline del in-agent, y corresponde al plazo máximo de ejecución en que el in-agent debe haber ejecutado una acción.*
- *T_{ij} : es el Período del in-agent, y determina la frecuencia de activación del in-agent. Este período determina la duración de los valores del estado interno del in-agent, pues con cada activación estos valores renuevan el conjunto β_{ij} del in-agent.*□

4.4. Modelo de Usuario

El *modelo de usuario* del $\mathcal{AA}$ permiten definirlo utilizando lenguajes de alto nivel. Aquí se introduce el conocimiento necesario para que el $\mathcal{AA}$ pueda interactuar con su entorno y resolver los problemas que se le presenten. Así, el conocimiento de un $\mathcal{AA}$ esta dividido en dos tipos:

- *Conocimiento del Dominio*. Este conocimiento representa la información del entorno que posee el $\mathcal{AA}$.

- *Conocimiento de la resolución del problema.* Este conocimiento está relacionado en con la estructura interna del $\mathcal{AA}$ para resolver sus problemas.

4.4.1. Conocimiento del Dominio

Está claro que un único agente no podría tener una visión completa de su entorno, por lo tanto este conocimiento será reducido, tomado principalmente de sus percepciones y del conocimiento previo necesario para inferir soluciones. Este tipo de conocimiento lo almacena en una "*pizarra temporal*" o *KDM* (*Knowledge Data Manager*) que está estructurada por *frames* los cuales contienen la información captada por el agente durante un período limitado de tiempo. Dada esta forma de trabajar, las soluciones que entrega son del tipo incremental, es decir, entrega una una solución cuya calidad podría mejorar de tener mas tiempo para ejecutarla. Es esta característica, la que aprovecharemos en las técnicas que se propondrán en el capítulo 5.

El conocimiento de este dominio permite identificar los indicios de situaciones significativas para que el $\mathcal{AA}$ pueda actuar según eso, tratándola por medio de eventos hacia el módulo de control. Existen dos tipos de eventos:

1. Eventos asociados a las partes opcionales de los *in-agent* críticos. Estos eventos permiten al *Servidor Reflejo* (*RS*) comunicarle al *Servidor Deliberativo* (*DS*) la disponibilidad de ejeutar las partes opcionales de un determinado *in-agent*.

2. Eventos asociados al los cambios en el *KDM*, los cuales podrán indicar cualquier modificación a las información interna del $\mathcal{AA}$.

4.4.2. Conocimiento de Resolución de Problemas

Un $\mathcal{AA}$ esta organizado internamente por entidades las cuales ayudarán a modelar el conocimiento del mismo para dar solución al problema principal (división de la complejidad y reutilización del código (Botti et al., 1999)). Estas entidades internas están organizadas por niveles jerárquicos tales que,

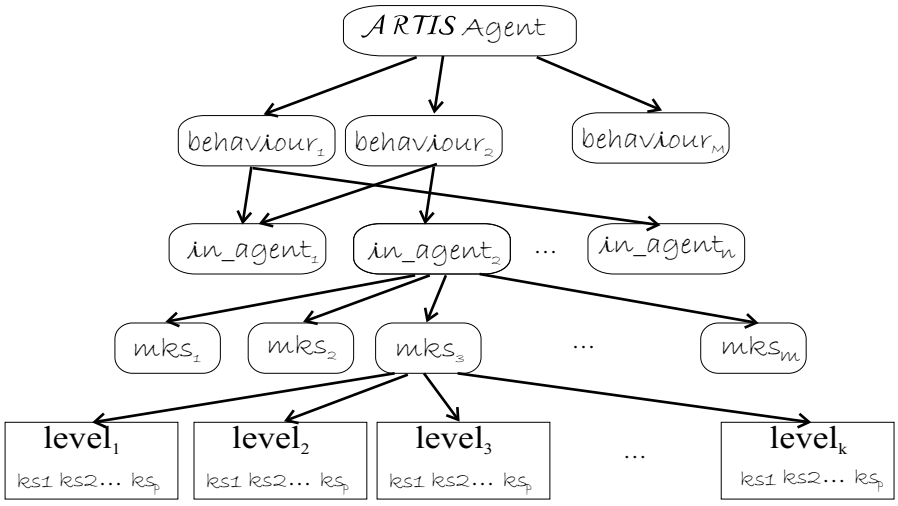


Figura 4.1: Jerarquía de las entidades de *ARTIS*

cada una de ellas deberá "*responder*" a la entidad inmediatamente superior de sus progresos y soluciones (Figura 4.1).

Las partes que componen un *AA* son:

- El *AA*, que es la entidad más alta dentro la jerarquía de entidades de *ARTIS* y tiene la visión completa del problema y el entorno.
- *Comportamientos* (*behaviour*), los cuales están compuestos de un conjunto de *in-agent* junto con la condiciones en las que se va a activar. Cada comportamiento esta compuesto de un comportamiento reflejo junto con un comportamiento deliberativo que lo refina.
- *Agentes internos* (*in-agent*), en ellos está el conocimiento parcializado tales que puedan resolver determinadas partes del problema principal.
- *Fuente de conocimiento de múltiples niveles* (*MKS – Multiple-level Knowledge Source*), cada *in-agent* a su vez, tiene múltiples niveles interrumpibles tales que puedan siempre entregar una primera solución a un sub-

problema, pudiéndola mejorar en el caso de tener el tiempo suficiente para ello.

- *Fuentes de conocimiento* (*KS – Knowledge Source*), es la entidad de mas bajo nivel y la encargada de ejecutar en la solución seleccionada por los niveles superiores. Esta solución podría estar en forma procedural, basado en reglas o alguna otra técnica propia de *Inteligencia Artificial* (*IA*).

No obstante, y como se comentó en capítulos anteriores, las técnicas de *IA* no siempre se pueden acotar temporalmente lo cual imposibilita su utilización en los *STR*. Para solucionar este problema, $\mathcal{ARTIS}$ permite acotar las ejecuciones de sus *MKS* indicando tiempos de cómputo en el peor caso (Carrascosa et al., 1997) y así integrar estas técnicas a la arquitectura para solucionar problemas de tiempo real críticos.

A continuación se detallará cada una de estas entidades.

4.4.2.1. Agente $\mathcal{ARTIS}$ ($\mathcal{AA}$)

Un $\mathcal{AA}$ debe cumplir con las definiciones planteadas en 2.1.1, por lo tanto es: autónomo, pro-activo y reactivo. Sin embargo, el agregarle mas características dependerá directamente del diseñador, por ejemplo: comunicación, la cual necesitaremos para entablar nuestras técnicas de negociación.

Un $\mathcal{AA}$ está compuesto por sub-entidades que modelan sus conductas, comportamiento, entorno, etc. Está formado por (Botti et al., 1999)(Figura 4.2):

1. Un conjunto de *sensores y actuadores* que permiten al agente interactuar con el entorno con las restricciones de tiempo que tiene impuestas por el Sistema de Tiempo-Real al que pertenece.

2. Un conjunto de *comportamiento* (*behaviours*) compuesto cada uno de un conjunto de *in-agents*. Cada *in-agent* se ocupa de solucionar una parte del

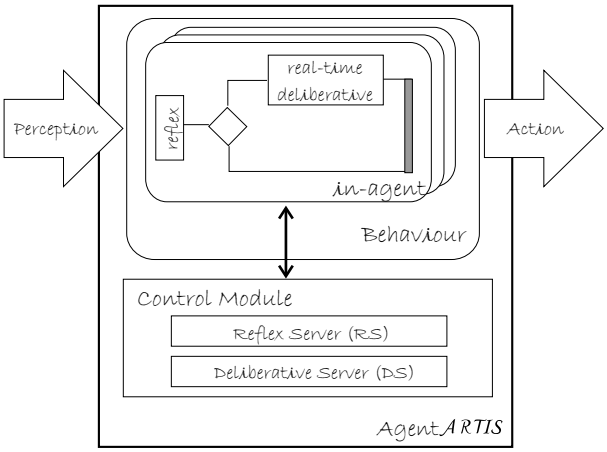


Figura 4.2: *Modelo de Usuario* del *Agente* $\mathcal{ARTIS}$

problema del $\mathcal{AA}$, activándose de forma periódica tal que todos cooperan para solucionarlo. Existen dos tipos de *in-agents*:

a) ***In-agent críticos*** están caracterizados por un período y un deadline. Su ejecución está garantizada bajo cualquier circunstancia en tiempo de ejecución (*run-time*). Tienen dos capas:

1) Una *capa refleja* que asegura una respuesta al problema del $\mathcal{AA}$ de calidad mínima en un tiempo de ejecución garantizado.
2) Una *capa deliberativa de tiempo-real* que trata de mejorar la calidad de la respuesta alcanzada por la capa refleja.

b) ***In-agents no-críticos*** forman parte de la deliberación del $\mathcal{AA}$. Este tipo de *in-agent* no tienen garantizada su ejecución, pero el agente trata de ejecutar tantos *in-agents* no-críticos como le sea posible con el objetivo de maximizar la calidad global de la solución a su problema.

3. Un conjunto de *creencias* (*believes*) que componen el estado mental del

in-agent que son: *un modelo del mundo* y su *estado interno*. Este conjunto de creencias está almacenado en un '*frame-base*' *blackboard* (Barber et al., 2003) el cual es accesible por todos los *in-agent*.

4. Un *módulo de control* que es responsable de la ejecución de los *in-agents* que componen la actual conducta del $\mathcal{AA}$. Como los *in-agents* están compuestos de dos capas, reactiva y deliberativa, el módulo de control está divido en dos sub-módulos (o servidores) encargados de ejecutar cada una de ellas, estos son:

   a) *Servidor Reflejo* (*RS*). Este servidor se encarga de ejecutar los componentes críticos del conjunto de *in-agents* activos del $\mathcal{AA}$. Con esto asegura que el $\mathcal{AA}$ obtendrá una solución inmediata al problema que enfrenta.

   b) *Servidor Deliberativo* (*DS*). Este servidor se encarga de mejorar la solución encontrada por el *RS* cuando existe tiempo para ello.

4.4.2.2. Agentes internos o *in-agent*

En los *in-agent* se encuentra el conocimiento del $\mathcal{AA}$ que necesita para solucionar un sub-problema determinado. Para la manipulación de este conocimiento, podría tener incorporado técnicas de *IA*.

Los *in-agent*, son las entidades con mayor jerarquía definidas dentro de un $\mathcal{AA}$. Se caracterizan por tener características y restricciones temporales como período, frecuencia y deadlines, dentro de los cuales deben ejecutar las acciones más relevantes del $\mathcal{AA}$ (tareas críticas).

El *in-agent* obtiene una primera respuesta a un sub-problema del $\mathcal{AA}$ y dependiendo del vencimiento o no de su deadline, puede entregar una respuesta más refinada.

Los *in-agents* organizan su conocimiento en tres niveles (Figura 4.3):

- *Nivel de Percepción*, este nivel adquiere la información significativa del entorno donde está situado el $\mathcal{AA}$.

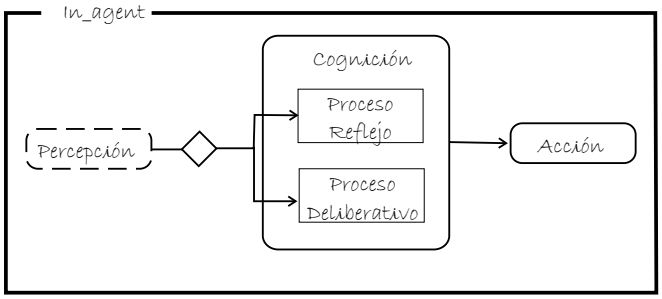


Figura 4.3: Estructura interna del *in_agent*

- *Nivel de Cognición*, este nivel obtiene una respuesta a partir de las percepciones. Esta respuesta puede obtenerla mediante dos tipos de procesos:
  - un *proceso reflejo*, el cual siempre entrega una respuesta de calidad mínima al sub-problema del agente.
  - un *proceso deliberativo en tiempo real*, en el cual se calcula una respuesta razonada a través de un proceso deliberativo.
- *Nivel de Acción*, que realiza las acciones calculada en el nivel cognitivo.

La forma en que el *in-agent* calcule la respuesta que entregará finalmente al agente (refleja o deliberativa), dependerá del tiempo que disponga para calcular dicha respuesta, una vez que se ha activado el $\mathcal{AA}$.

Cada *in-agent* está formado por una secuencia de *MKS* (Figura 4.1).

4.4.2.3. MKS – Multiple-level Knowledge Source

Los *MKS* ó fuente de conocimiento de múltiples niveles, es la entidad que implementa el concepto de algoritmo *anytime* o de un método múltiple (Boddy, 1991). Esto significa que cada *MKS* puede entregar diferentes soluciones a un mismo problema con distintos tiempos de cómputo y distintas calidades.

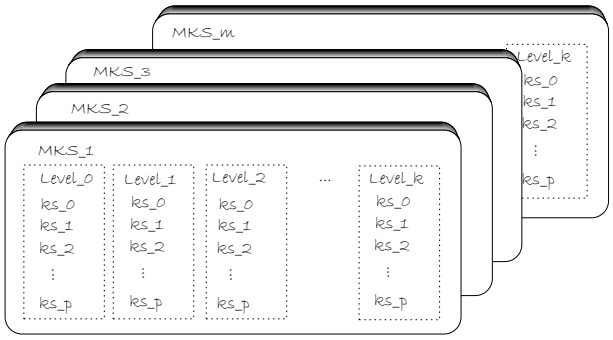


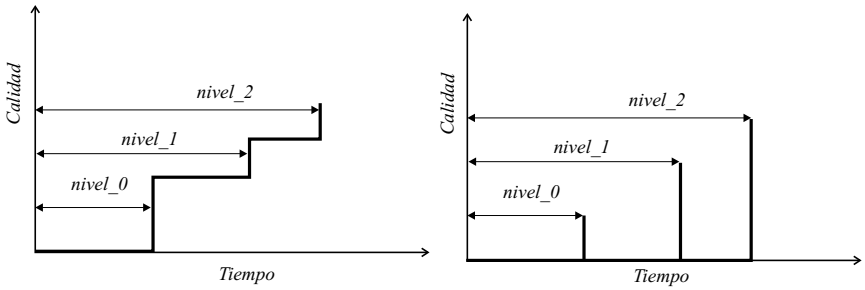
Figura 4.4: Estructura interna de un *MKS*

Como cada *MKS* puede poseer múltiples niveles (Figura 4.4), cada nivel proporcionaría un tipo de solución al problema del $\mathcal{AA}$. De esta forma, los *MKS*s pueden ofrecer dos forma de obtener la solución a un determinado problema:

1. *MKS anytime o de refinamientos sucesivos.* Consiste en que el *MKS* ofrece una primera solución al problema, la cual se calcula en su nivel más crítico con una determinada calidad. Sin embargo, si el *in-agent* al que pertenece no ha alcanzado su deadline, la calidad de esta solución se puede mejorar con la ejecución de sus siguientes niveles (Figura 4.5(a)).

2. *MKS de Métodos Múltiples.* En este caso, cada nivel del *MKS* puede ofrecer soluciones con distintas calidades para un mismo problema. Sin embargo, tanto la calidad de la respuesta como el tiempo de cómputo estimado en el peor caso será igual o superior al de los niveles anteriores a él (Figura 4.5(b)). La decisión de cual solución se utilizará se resolverá en el nivel de acción del *in-agent*.

La ejecución de un *MKS* puede ser interrumpida en cualquier momento, retornando siempre la solución calculada en su último nivel que se haya ejecutado completamente.



(a) MKS de Refinamientos Sucesivos

(b) MKS de Métodos Múltiples

Figura 4.5: Tipos de soluciones ofrecidas por los *MKSs*

Cuando la ejecución de un *MKS* es esencial para el buen funcionamiento del sistema, este se indica como *MKS crítico*, y se exige que el primer nivel proporcione una solución en un tiempo de cómputo predecible y realista es decir que tiene un primer nivel crítico.

Cada *MKS* a su vez, están compuestos de varios *KS* (*KS – Knowledge Source*), como se ve en la Figura 4.4, los cuales se ejecutan en forma conjunta y con un orden determinado.

4.4.2.4. KS – Knowledge Source

Los *KS* o fuente de conocimiento, son las entidades de más bajo nivel dentro de la jerarquía de la arquitectura de un $\mathcal{AA}$, y representa el verdadero código fuente ejecutable del $\mathcal{AA}$ (procedural o basado en reglas).

Las *KS* se distinguen según la operaciones que realice en el sistema, estas operaciones pueden ser:

- *KS de percepción*, este tipo de *KS* percibe valores desde el entorno del agente y proporciona este conocimiento en forma de variables en la memoria común (*KDM – Knowledge Data Memory*).

- *KS de cognición*, este tipo de *KS* procesa el conocimiento tanto interno como percibido, proporcionando la o las posibles respuestas al problema en forma de variables internas.
- *KS de acción*, este tipo de *KS* actúa sobre su entorno a partir de los valores de las variables de salida previamente calculadas, leídos desde el *KDM*.

4.5. Modelo del Sistema

Para que $\mathcal{ARTIS}$ pueda ejecutar un $\mathcal{AA}$ como el descrito en el *Modelo de Usuario*, debe '*traducirlo*' a tareas de bajo nivel tales que sean entendibles por el sistema donde esta siendo implementado. Con esta traducción se obtiene el *Modelo de Sistema*, cuyos componentes resultantes se ejecutan directamente sobre el sistema operativo de tiempo real *RT-Linux* denominado *Flexible RT-Linux* (Terrasa, 2000).

Las principales características que tiene este modelo son (Terrasa et al., 2002):

- Test de planificabilidad, es un test *off-line* de la planificabilidad de las tareas del agente.
- Modelo de tareas que garantiza el cumplimiento de las restricciones temporales críticas del sistema.
- Método de extracción de holgura *on-line* que calcula el tiempo disponible para la ejecución de niveles deliberativos de tiempo real.
- Conjunto de extensiones sobre el sistema operativo Real-Time para facilitar las características de tiempo real (*Flexible RT-Linux* (Terrasa, 2000)).

De esta forma, la equivalencia a una entidad de bajo nivel de cada una de las partes del $\mathcal{AA}$ del modelo de usuario es:

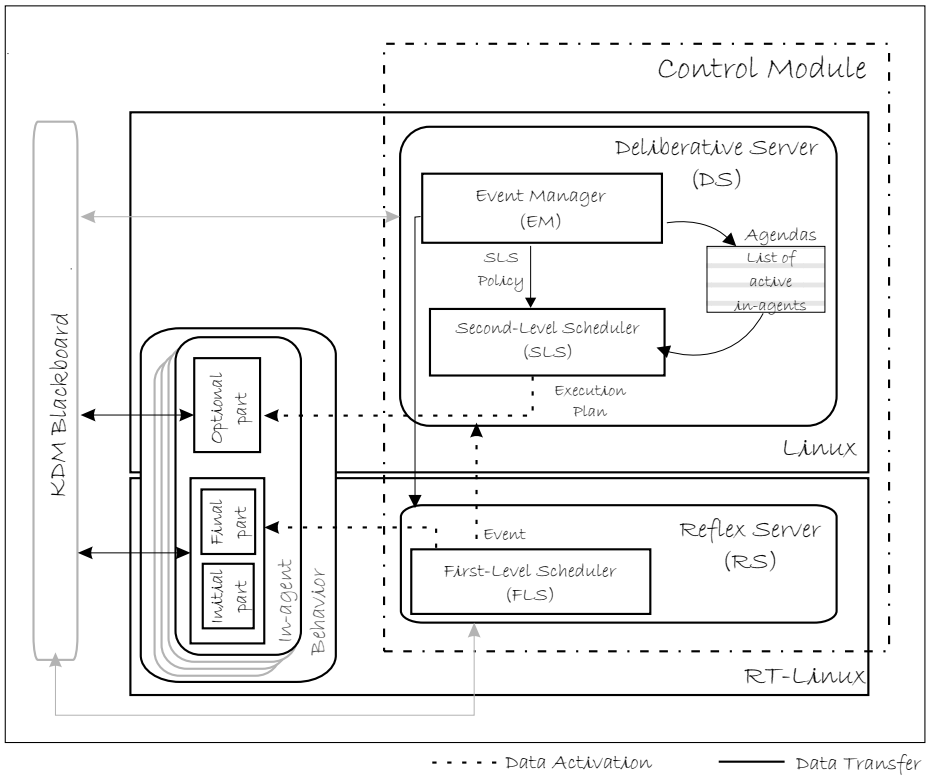


Figura 4.6: *Modelo de Sistema* de $\mathcal{AA}$.

1. Los *sensores* y *actuadores* del modelo de usuario se corresponden con una librería para acceder a los distintos dispositivos de hardware.

2. Los *in-agents* del modelo de usuario son traducidos a tareas de bajo nivel en el modelo de sistema. Cada una de estas tareas deberán tener predefinido sus restricciones temporales: prioridad, tiempo de cómputo en el peor caso, período y deadline. Los períodos y deadlines de estas tareas serán comunes al *in-agent* al que pertenecen. Así, cada tarea deberá tener tres partes:

   a) Una *parte inicial* (*mandatory*). Esta es la parte refleja del *in-agent*, se debe ejecutar siempre y obtiene una primera respuesta refleja al problema del $\mathcal{AA}$ la cual tiene una baja calidad. Esta parte inicial incluye la parte de percepción del *in-agent*.

   b) Una *parte opcional*. Esta es la parte deliberativa del *in-agent*. Estos componentes opcionales incrementan la calidad de la respuesta calculada en la parte inicial implementando el proceso cognitivo del *in-agent*. Para esto, se utilizan técnicas de *IA* y se ejecutan entre las partes inicial y final del *in-agent* correspondiente.

   c) Una *parte final*. Esta parte entrega la respuesta que se generó en las partes previas (inicial y opcional) del *in-agent*. Esta parte se encarga de las acciones del *in-agent*.

3. Las *creencias* del $\mathcal{AA}$ (*believes*) se traducen en una memoria compartida (*blackboard*) accesible desde todas las tareas activas.

4. Las dos partes del *Módulo de Control* del modelo de usuario se traducen en (ver figura 4.7):

   a) El *Servidor Reflejo* (*RS*) incluye *Planificador de Primer Nivel (FLS)* para la tareas de *Real-Time* (*RT*). El *FLS* utiliza políticas de tiempo real en tiempo de ejecución para decidir que tarea ejecutar en

cada momento. Esta planificación ayuda al $\mathcal{AA}$ a que se adapte a los cambios en su entorno, así como ejecutar las tareas usando menos tiempo que el estimado para su ejecución en el peor caso. Esto lo realiza mediante un test de planificabilidad *off-line*, el que permite comprobar, en la fase de diseño, si el agente podrá cumplir las restricciones temporales impuestas por el usuario. Para este test solo considera la ejecución de sus partes obligatorias y finales de cada tarea.

Para la realización de este análisis se debe tener en cuenta que la ejecución de las partes críticas de las tareas (obligatoria y final) está determinada por el *planificador de primer nivel* (*First-Level Scheduler - FLS*) que pertenece al *Servidor Reflejo* (*Reflex Server - RS*), utilizando para ello una política de planificación basada en prioridades fijas expulsivas (*Fixed Priority Pre-emptive Scheduling Policy*) (Audsley et al., 1995). Mientras que la ejecución de las partes opcionales de las mismas estará determinada por le *Servidor Deliberativo* (*Deliberative Server - DS*) (Figura 4.6).

Una vez que las restricciones temporales de las tareas estén satisfechas, el procesador quedará con tiempo ocioso, momentos que el *RS* se lo cede al *Servidor Deliberativo* (*DS*) para la ejecución de las partes opcionales, indicándole el estado de las tareas críticas, pues solo se podrán ejecutar las partes opcionales de las tareas críticas que aún estén activas.

b) El *Servidor Deliberativo* (*Deliberative Server - DS*). Básicamente, el *DS* está basado en la esencia de las arquitecturas tipo *blackboard* (Hayes-Roth, 1995), es decir, dirigido por eventos. Gestiona la inteligencia del $\mathcal{AA}$ en el tiempo de holgura del sistema. Esta gestión es posible gracias a la manipulación de las partes opcionales del $\mathcal{AA}$ las cuales mejoran la calidad de la respuesta al problema del agente. Su funcionamiento está dividido en dos sub-módulos:

1) *Gestor de Eventos* (*EM*). El *EM* se activa con los mensajes y eventos que recibe desde *RTOS* el cual le indica que existe *tiempo de holgura* en el sistema. Entonces, *EM* genera una agenda con las tareas opcionales activas y vigentes en el sistema. Es decir, anota aquellas tareas que aun no han alcanzado su *deadline* o bien que su parte inicial recién ha comenzado en este ciclo. Por otro lado va a quitar de la agenda aquellas tareas cuyas partes finales hallan acabado, o bien la ejecución anterior no se completó con éxito, con lo cual no podrá continuar aunque no halla vencido su *deadline*.
2) *Planificador de Segundo Nivel* (*SLS*). El *SLS* va a recibir la agenda actualizada desde el *EM* y la tendrá que ordenar. Para esto cuenta con diversas políticas de planificación a las cuales podrán optar. Cabe destacar que, aunque consta de varias políticas implementadas, solo podrá ejecutar una hasta el final de la aplicación. Esta política la indica el usuario en la etapa de diseño.

Actualmente el *SLS* cuenta con las siguientes políticas:

1. **DM (Deadline Monotonic)**. Estrategia que utiliza el deadline de los *in-agent*. Prioriza aquellas tareas cuyo plazo máximo de ejecución sea más urgente.
2. **EDF (Earliest Deadline First)**. Las tareas opcionales activas son ordenadas según el vencimiento de sus deadlines. Escoge aquellas tareas cuya pronta finalización del plazo máximo de ejecución imposibilitará la ejecución de más niveles opcionales.
3. **ELDF (Earliest Level Deadline First)**. Es una variación de la anterior la cual intenta ajustar al máximo los deadlines asignados a las tareas. La idea principal de esta política es planificar el mayor número posible de niveles en base a las restricciones temporales de la misma.

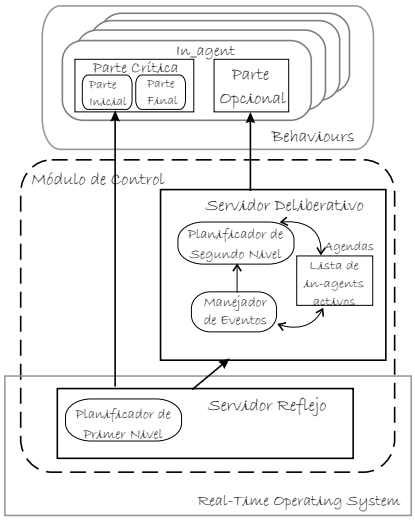


Figura 4.7: Módulo de Control del $\mathcal{AA}$

4. **HSF (High Slope First)**. Esta política ordena las tareas por la pendiente de su perfil de ejecución. Esta pendiente es el resultado de convertir los escalones del perfil de ejecución en una función lineal. Los niveles activos se ordenan por dicha pendiente. El objetivo de esta política es conseguir la mayor calidad posible, pero seleccionando primero a aquellas tareas cuyo tiempo de ejecución son menores.

5. **BIF (Best Importance First)**. Esta política ordena las tareas según la importancia asignada en el momento de diseño. El objetivo de esta política es que las tareas más importantes del sistema tengan prioridad sobre las demás.

La propuesta de esta tesis doctoral está enfocada a darle una opción más al *Servidor Deliberativo* para que al momento de decidir que partes opcionales ejecutar utilice métodos de negaciones entre los *in-agents* (tareas activas del sistema) del *agente* $\mathcal{ARTIS}$.

Capítulo 5

Técnicas de Negociación en $\mathcal{ARTIS}$

En este capítulo se expondrá y explicarán las técnicas de negociación implementadas en el agente $\mathcal{ARTIS}$. La estructura de este capítulo es la siguiente:

1. Motivaciones para la implementación de estas técnicas en el agente $\mathcal{ARTIS}$.
2. Definiciones de las negociaciones que se producen en el agente.
3. Formalización e Implementación de estas negociaciones en el agente $\mathcal{ARTIS}$.

5.1. Motivación

La arquitectura de agentes $\mathcal{ARTIS}$ ($\mathcal{AA}$) se puede utilizar en el desarrollo de diversas aplicaciones tales como: industriales, sociales, robóticas, control aé-

reo, etc.. Esto implica que los agentes $\mathcal{ARTIS}$ trabajan en dominios complejos y difíciles de resolver.

La negociación entre agentes ha sido, hasta el momento, la técnica más utilizada por los mismos para solucionar todo tipo de problemas. En particular cuando un agente del Sistema Multi-Agente necesita algo que no posee (recurso, habilidad, etc.) para alcanzar su objetivo final. Es entonces cuando corresponde al agente interesado exponer y ofrecer una serie de ventajas y beneficios en forma de ofertas y contra-ofertas, para solucionar sus problemas.

Los agentes $\mathcal{ARTIS}$ ($\mathcal{AA}$) no están exentos a este tipo de problemas, dado que las entidades del $\mathcal{AA}$ están en continua '*lucha*' por ejecutar sus tareas y así poder dar la mejor respuesta posible en el tiempo dispuesto para ello, como se expuso en el capítulo 4 (sección 4.4.2.2, pagina 58).

Puesto que la naturaleza del agente $\mathcal{ARTIS}$, como un agente que trabaja en entornos de tiempo real estrictos, es obtener *siempre* una respuesta en un tiempo mínimo. El agente $\mathcal{ARTIS}$ además incluye una mejora de esta respuesta cuando cuenta con el tiempo suficiente para esto. Es en este instante cuando las negociaciones que expondremos aquí cobran sentido, pues son las entidades del $\mathcal{AA}$ (los *in-agents*) las que aumentan esta calidad, pudiendo existir varios *in-agent* que den una mejora al problema. Esto implica que todos ellos desean ejecutarse pero, por lo general en este tipo de sistemas (STR), no hay tiempo suficiente para ejecutarlos a todos.

5.2. Negociación Entre las Entidades de un Agente $\mathcal{ARTIS}$

Como se dijo en el capítulo 3, para que una negociación se produzca deben:

1. Tener claros los motivos de la negociación.
2. Tener claras las partes involucradas: *una parte vendedora* y *una parte compradora*.

3. Conocer de antemano el tipo de negociación que realizarán (incluyendo con esto: normas, reglas, estrategias, etc.).

5.2.1. Motivos de la negociación

En nuestro caso, el motivo de la negociación entre las entidades de un agente $\mathcal{ARTIS}$ ($\mathcal{AA}$) será tratar de acceder a un recurso escaso en el sistema, *tiempo de CPU*, para que puedan ejecutarse y así dar mayor calidad a la respuesta que entregarán finalmente al agente al que pertenecen.

Esto nos da a entender que, si todas las entidades internas (*in-agents*) del $\mathcal{AA}$ cooperan y colaboran entre sí para obtener esta respuesta, podemos afirmar que el entorno donde se producirán las negociaciones es: *colaborativo – cooperativo.*

Para poder explicar de mejor forma la inclusión de las negociaciones en la arquitectura del $\mathcal{AA}$, recordaremos su funcionamiento desde el punto de vista del usuario (*modelo de usuario*) y ubicar allí nuestras negociaciones, para luego implementarlas en el *modelo de sistema* del agente.

En el *modelo de usuario* del agente $\mathcal{ARTIS}$ las conductas del agente están formadas por un conjunto de *in-agents*, donde cada uno de ellos resuelve un problema específico. Además, cada *in-agent* está compuesto de dos partes: refleja y deliberativa. La parte refleja entrega una respuesta rápida y de baja calidad al problema del agente. La parte deliberativa, está encargada de afinar la primera respuesta entregada por la parte refleja, aumentándole la calidad. Esta parte debe ejecutarse antes de que finalice completamente la ejecución del *in-agent* (i.e., antes que el *in-agent* ejecute su parte final, ver figura 5.1).

El ejecutar o no la parte deliberativa del *in-agent* dependerá del tiempo de holgura disponible en el sistema (*slack disponible*) y estará coordinada por el *Servidor Deliberativo* (*DS*) del agente $\mathcal{ARTIS}$.

Como en este tiempo de slack el *DS* puede ejecutar cualquier parte deliberativa de cualquier *in-agent* activo, este deberá aplicar alguna política o heurística para planificar estas ejecuciones. Aquí es donde proponemos la in-

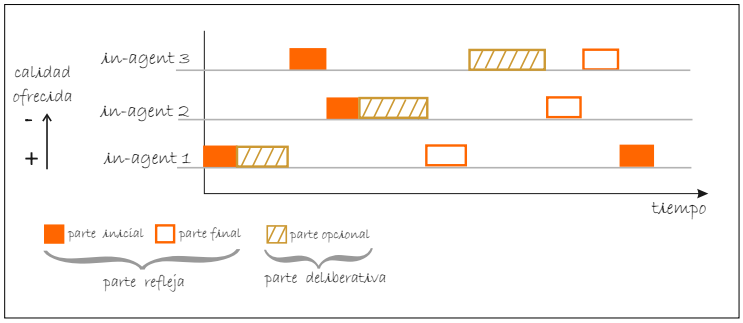


Figura 5.1: Ejecución de *in-agents* del agente *ARTIS*

serción de las negociaciones ya que los *in-agents* quieren ejecutar sus partes deliberativas pero es el *DS* quien deberá elegir cuales ejecutar y por qué las debe seleccionar (Figura 5.2).

Al traducir el *modelo de usuario* del agente *ARTIS* al *modelo de sistema* para que podamos implementar las negociaciones, nos queda el escenario que muestra la figura 5.3.

El *servidor deliberativo* (*DS*) se ejecuta cuando existe tiempo de holgura disponible en el sistema (*slack*) para ejecutar partes opcionales[1]. Esto se lo indica el *servidor reflejo* mediante un mensaje (evento). Cuando esto sucede, el *manejador de eventos* (*Event Manager - EM*) del *DS* genera una lista (agenda) con los *in-agent* activos y vigentes[2] en ese instante. Esta lista es referenciada por el *Planificador de Segundo Nivel* (*Second-level Scheduler - SLS*) quién enviará a ejecutar las partes opcionales del *in-agent* dependiendo del resultado de las negociaciones entre estas dos entidades (el tipo de negociación

[1] Al traducir cada *in-agent* a una tarea de bajo nivel (*Modelo de Sistema* del agente *ARTIS*), queda compuesto de tres partes: parte inicial, parte final y parte opcional. Las dos primeras son reactivas y se ejecutan siempre por el *servidor reflejo*. La tercera parte será la traducción de la parte deliberativa del *in-agent* y su ejecución la coordinará y planificará el *servidor deliberativo* bajo las condiciones antes mencionadas.

[2] Un *in-agent* activo y vigente es aquel cuya parte inicial ha finalizado y su parte final aún no empieza.

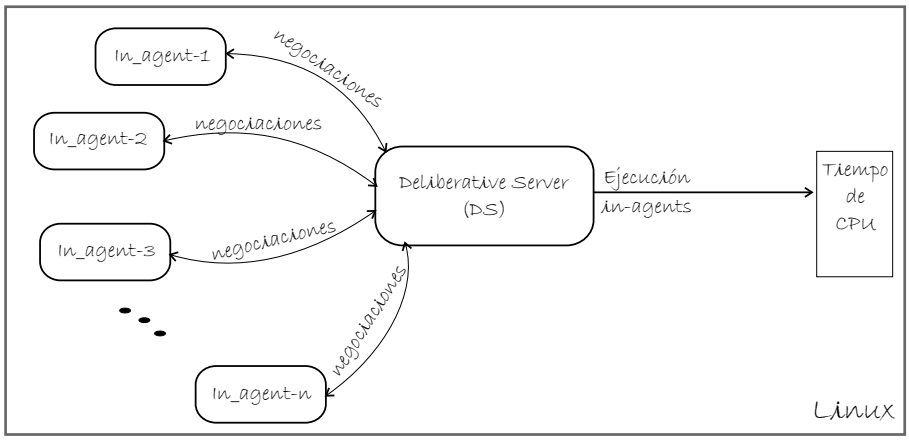


Figura 5.2: Negociación en *modelo de usuario* del agente *ARTIS*

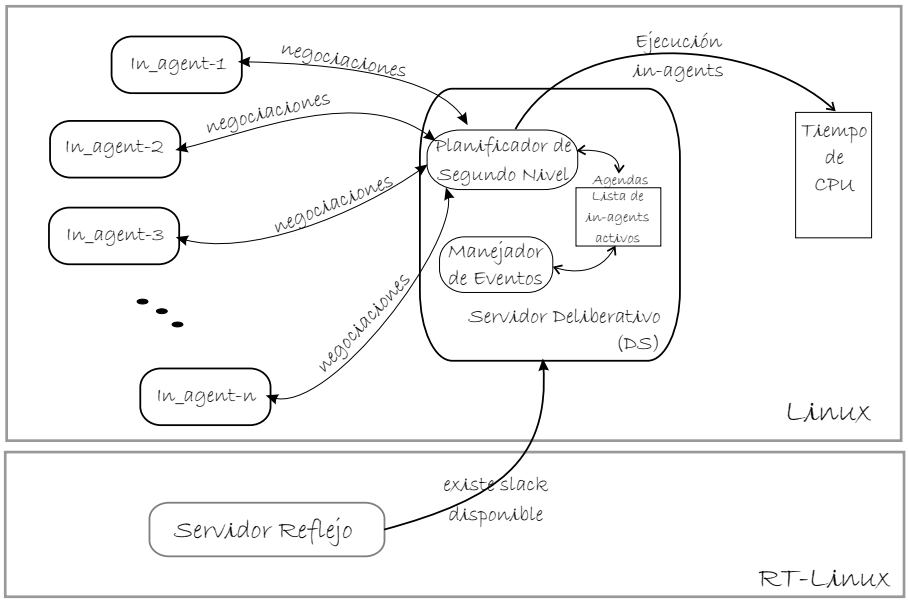


Figura 5.3: Negociación en *modelo de sistema* del agente *ARTIS*

Tabla 5.1: Situación para establecer la negociación entre entidades del $\mathcal{AA}$ y el *SLS*

SLS(Second Level Scheduler)	**Entidades del $\mathcal{AA}$*(in-agent)***
Quiere:	**Tiene:**
- *Solucionar un problema*	- *Una o varias soluciones al problema*
- *Calidad en esa solución**	- *Calidad asociada a esa solución*
- *Rapidez y eficacia en la solución**	- *Tiempo de ejecución promedio*
Tiene:	**Quiere:**
- *Tiempo de CPU*	- *Tiempo de CPU*
**: La mejor posible*	

es seleccionado por el usuario en tiempo de diseño).

La Tabla 5.1 muestra las situaciones que conducen a proponer una negociación entre ambas partes: *SLS* e *in-agents activos y vigentes*. En esta tabla se puede apreciar que el *SLS*:

- Cuenta con tiempo para ejecutar las partes opcionales (*deliberativas*) de los *in-agents* que están en la agenda que generó el *EM*.

- Deberá decidir que partes opcionales de los *in-agents* ejecutar tal que, la respuesta final al problema del agente, sea la mejor posible.

Por otra lado, están las entidades del $\mathcal{AA}$, los *in-agents*, los cuales poseen las respuestas al problema del agente y cada una de ellas está asociada a una calidad y tiempos de cómputos pre-determinados.

En definitiva, la finalidad de las negociaciones sería entonces permitir que ambas partes, *SLS* e *in-agent*, alcancen sus objetivos los cuales son compatibles con los objetivos generales del sistema al que pertenecen (el agente $\mathcal{ARTIS}$) que es dar la mejor solución a un determinado problema.

5.2.2. Partes involucradas en la negociación

Los motivos de la negociación explicados hasta aquí nos dan claras indicaciones de las partes que están involucradas en las negociaciones. Estas son:

- *SLS* (*planificador de segundo nivel*) del *servidor deliberativo*.
- *In-agents*

Sin embargo aun falta determinar que roles asumirá cada parte.

Como ya sabemos, en toda negociación existen dos partes: *comprador* y *vendedor*. El *comprador* es el que necesita de un recurso o habilidad determinada, para alcanzar sus objetivos. Por otro lado está el *vendedor* que es poseedor de este recurso o habilidad que busca el comprador. El grado de necesidad del recurso por parte del comprador estará determinado por la utilidad que finalmente obtenga con su adquisición, versus el gasto implícito que conlleva el negociar su compra con el *vendedor*.

Así, los roles que se asignarán para solucionar este conflicto serán:

Vendedor Este rol lo asumirá el *planificador de segundo nivel* (*SLS*). Ya que, es el *SLS* quien posee el recurso en conflicto y debe determinar a que *in-agents*, de entre los que están activos, entregárselo tal que ocupe de la mejor forma posible ese recurso (*tiempo de CPU*).

Comprador Este rol lo asumirán los *in-agents*. Ya que, son estos quienes necesitan del recurso '*tiempo*' para poder ejecutarse. La finalidad de los *in-agents* es entregar la mejor respuesta que posean. Esto implica que en las negociaciones influirá la calidad que puedan entregar según sus recursos y habilidades disponibles en ese instante.

5.2.3. Selección del Tipo de Negociación

La siguiente cuestión es, determinar *a-priori* que tipo de negociación sería la apropiada para estas entidades.

Como el fin último de estas negociaciones es alcanzar, en el menor tiempo posible, la mayor calidad en la respuesta al problema del $\mathcal{AA}$. Se deben evaluar las siguientes consideraciones:

- El *SLS* necesita obtener una respuesta rápida y eficiente a un determinado problema en un tiempo limitado.
- El *SLS* debe seleccionar la respuesta final desde un grupo de posibles respuestas. Para esto debe considerar:
  - La calidad de las respuesta ofrecidas por cada uno de los *in-agent* participantes.
  - Los tiempos de cómputos estimados de cada *in-agent* en entregar estas respuestas.
- El tiempo total destinado a las negociaciones entre el *SLS* y los *in-agents*, para seleccionar la respuesta está acotado.
- El tiempo destinado a generar las ofertas y contra-ofertas está limitado.

En resumen, existe un recurso en conflicto que es el *tiempo de CPU*. Este recurso lo posee el *SLS* (*vendedor*) quien deberá distribuirlo bajo las condiciones mencionadas antes. Existen también muchos *compradores*, los *in-agent* activos, que desean *adquirir* este recurso.

Por otro lado, en este sistema también podemos encontrar:

- En la arquitectura del agente $\mathcal{ARTIS}$ se hace necesario considerar la presencia de todos los *in-agents* activos en un instante de tiempo determinado.
- Los planes de los *in-agents* forman parte del plan general del agente $\mathcal{ARTIS}$. Por lo tanto siempre se ejecutará al menos uno.
- Los recursos del entorno común están limitados.

- Es necesario organizar las acciones de los *in-agent* para que el agente alcance su meta final: *solucionar su problema*.

- Una '*moneda común*'. En toda negociación, debe estar presente la '*moneda común*' (Kraus, 2001b), es decir, algo que puedan utilizar las partes involucradas en sus transacciones. En nuestro caso la '*moneda común*' será *la calidad* que es ofrecida al *SLS* por los *in-agents* participantes como parte de la solución al problema del agente.

- El proceso de adjudicación del recurso deberá ser rápido y ventajoso para ambas partes.

Teniendo presente todo lo expuesto hasta aquí, hemos considerado que la mejor opción para este tipo de dominios sería aplicar *las subastas* como técnicas de negociación, ya que el fin de las subastas es que el vendedor ofrezca su producto al que la haga la mejor propuesta u oferta respetando las normas y protocolos fijados para el proceso.

En nuestro caso, las subastas se llevarán a cabo bajo condiciones prefijadas considerando entre otras, como es el caso de *ARTIS*, los tiempos límites para cada una de las acciones e interacciones durante la misma. Estas condiciones determinarán las pujas y la asignación final de del recurso subastado.

5.2.3.1. Normas, Reglas y Estrategias

Una vez que se determina el tipo de negociación que seguirán los participantes, estableceremos las reglas, normas y estrategias generales que deberán considerar los participantes de las subastas. Estas son:

- Los roles que pueden tener los participantes en las subastas son: *vendedor* (*SLS*) y *compradores* (*in-agents*). Solo un participante hará el rol de vendedor mientras que los demás tendrán el rol de compradores.

- Cada *in-agent* hará sólo una propuesta por vez.

- El *SLS* acepta la propuesta, si con esta alcanza o excede su utilidad esperada. Para nuestras subastas la utilidad será proporcional a la calidad que espera recibir.
- Si el *SLS* rechaza la propuesta, significa que la utilidad que obtiene con esa propuesta es menor a la que espera recibir, y llama a presentar nuevas propuestas[3].
- Las nuevas propuestas deberán ofrecer mayores utilidades que las rechazadas.
- Si no se llegase a un acuerdo al término de las subastas (i.e. el *SLS* no ha aceptado ninguna de las propuestas), el *SLS* dispondrá de su recurso como le sea más conveniente.
- Es una negociación cooperativa o coalicional. Es decir, los *in-agents* participantes trabajan en coordinación y cooperan entre sí para alcanzar los objetivos del sistema al que pertenecen, agente $\mathcal{ARTIS}$.
- El recurso siempre se asigna al pujador con la valoración mas elevada.
- Los acuerdos alcanzados se mantendrán y cumplirán según las condiciones establecidas antes del comienzo de la subasta.

Definidas las reglas generales, se hace necesario determinar el tipo de protocolo que utilizarán los participantes. Esta estará determinado por el tipo de subasta que realicen, y se basarán en los protocolos estándar utilizados para subastas entre agentes(FIPA00031, FIPA; FIPA00032, FIPA). Estos los explicaremos con mayor detalle, en las siguientes secciones.

Resuelto el protocolo de la negociación y normas generales que utilizarán las entidades del agente $\mathcal{ARTIS}$, especificaremos las estrategias a seguir por

[3]Esta nueva llamada a propuestas dependerá del tipo de subastas que se estén llevando a cabo.

los *in-agents* (*compradores*). Estas le indican al *in-agent* que acción seguir o tomar en cada momento.

Se describirán estrategias generales para cada una de las subastas, en las siguientes secciones describiremos con mayor nivel de detalle cada una de ellas y su aplicación en el agente *ARTIS*.

Una estrategia común a todas las subastas será la forma que tendrán los *in-agents* de valorar sus pujas, estas son:

- Sus valoraciones serán *independientes o privadas*, es decir que los compradores conocen toda la información sobre lo que se está vendiendo (*tiempo de CPU*), y las ofertas o valoraciones de los demás participantes no les afectará directamente.
- Sus valoraciones serán *simétricas*, es decir, derivan de las mismas funciones de distribución, lo que implica que el *SLS* (vendedor) percibe de forma similar a todos los posibles *in-agents* compradores.
- Sus valoraciones serán directamente proporcionales a sus utilidades, es decir, cuanto mayor sea la utilidad esperada por el *in-agent* mayor será la valoración o puja.

Sin embargo existirán estrategias particulares a cada subasta.

Subastas Inglesa

La estrategia dominante[4] de la *Subasta Inglesa* o ascendente, consiste en que el *in-agent* permanezca en la subasta hasta que el precio de venta del recurso iguale o supere su *máximo precio reservado*. En ese momento el *in-agent* en cuestión se retirará de la subasta.

[4] Las *estrategias dominantes* son aquellas en que la selección de la mejor estrategia para el agente no depende de las estrategias elegidas por el resto de los participantes, y esta selección siempre le produce la mejor utilidad para él.

Las *estrategias débilmente dominante*, entrega la acción cuya utilidad no están pequeña como si seleccionara cualquier otra, independiente de la estrategia que seleccionen los demás compradores.

En este sentido, el *SLS* (vendedor) se asegura de que el ganador de la subasta es el *in-agent* que más utilidades le proporcionará (le dará una mayor *calidad*).

Subasta del Sobre Cerrado al Primer Precio

En este caso, cada *in-agent* tendrá una única oportunidad de ofrecer una buena puja por el recurso. Así, su oferta deberá considerar las pujas de los demás *in-agent*.

La valoración del *in-agent* comprador se podría establecer considerando las mejores pujas que podrían entregar los demás *in-agents*, basándose en que su propia valoración será la más alta. Es decir, solo necesitan de su propia valoración para calcular su "*puja óptima*". Como en nuestro caso la moneda de cambio es la calidad, se supone que cada *in-agent* ofrecerá una calidad máxima del *100 %*.

Subasta del Sobre Cerrado al Segundo Precio

La estrategia dominante de esta subasta consiste en que el *in-agent* presentará una puja igual a su valoración. Es decir, la estrategia consiste en "*decir la verdad*" donde cada *in-agent* pujaría por su valor real.

Nuevamente, el *SLS* (vendedor) asignará el recurso al *in-agent* que más utilidades le proporcionará, basándose en que los *in-agents* participantes *"no mienten"*.

Subasta Holandesa

La estrategia en este caso es que los *in-agents* participantes deberá elegir su puja sin conocer las decisiones de los demás *in-agents*, y en caso de ganar, pagar gusto lo que ofertaron. Por lo tanto, la información con la que cuentan para adoptar su decisión será idéntica que en las subastas anteriores, siendo la decisión que tome independiente de las acciones de los otros *in-agents*

participantes.

Ventajas de las estrategias mencionadas son:

- La selección de las acciones será óptima incluso cuando en el caso de que se asignarán probabilidades positivas a que el resto de los *in-agents* participantes se puedan desviar de sus estrategias de equilibrio.
- Para el *SLS* implica no tener que recurrir al diseño de complejos modelos de selección ni reglas, ya que con las subastas se asegura de maximizar siempre sus ingresos, es decir maximizar la *calidad final esperada*.
- La simplificación en la presentación a las pujas, ya que los *in-agents* no tendrían que analizar exhaustivamente la información ni conjeturar sobre el comportamiento de los demás *in-agents* participantes. Esto se refleja en la forma de valorar su participación o puja dentro de la subasta.

Estos dos últimos puntos son esenciales, pues permiten que el agente *ARTIS* confeccione la planificación de sus tareas mencionada en el capítulo 4.

5.3. Descripción de las Subastas

El punto de partida de las negociaciones será (ver Figura 5.4) cuando el *Servidor Reflejo* (*RS*) indique al *Servidor Deliberativo* (*DS*) que hay tiempo disponible para la ejecución de una o varias partes deliberativas de los *in_ agent* (también llamadas partes opcionales del *in-agent* en el modelo de sistema del agente) que están activos en ese instante (capítulo 4, página 65).

El *DS* activará entonces el *Manejador de Eventos* (*EM*) quien debe verificar que los *in-agent* activos en ese instante estén vigentes, es decir, que cumplan con las siguientes condiciones:

- $t_i^d < t_{Sa}^f$, El instante del vencimiento de su próximo *deadline*, t_i^d, del *in-agent* participante i, debe estar antes del instante de término del tiempo que se está subastando t_{Sa}^f.

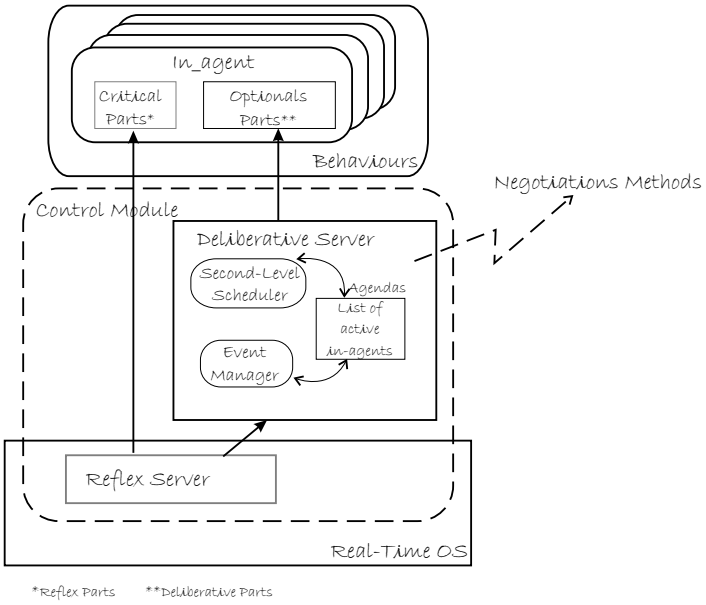


Figura 5.4: Implementación de Negociaciones en $\mathcal{ARTIS}$

- Que no hallan ejecutado todas sus partes opcionales.
- Que sus partes finales no hallan acabado.

Una vez que el *EM* actualiza la lista de *in-agent* activos y vigentes, la almacena en una agenda interna (figura 5.4) a la cual accede el *Planificador de Segundo Nivel* (*SLS*) quien deberá distribuir el tiempo de slack disponible entre las partes opcionales de los integrantes de esa lista utilizando las subastas.

Antes de comenzar las pujas por el *slack disponible*, el *SLS* deberá enviar la siguiente información a todos aquellos *in_agent* activos y vigentes:

- Sa: Duración total del *slack disponible*. Será el tiempo total de CPU que está disponible para la ejecución de las partes opcionales de los *participantes*.
- t^s_{Sa}: Instante de tiempo en que comienza el *slack disponible*. A este tiempo, se le ha descontado el tiempo de ejecución del *DS*, el cual es fijo y constante.
- t^f_{Sa}: Instante de tiempo en que termina el *slack disponible*.
- T^{offers}_{max}: Tiempo máximo destinado para generar ofertas dadas las características de la arquitectura de $\mathcal{ARTIS}$.
- BO^t_k: Dependiendo del tipo de subasta que se esté utilizando, el *SLS* deberá enviar la mejor oferta recibida, desde los *participantes*, en la llamada k en el momento t. Esta oferta comenzará en algún valor pre-establecido por el diseñador del $\mathcal{AA}$.

La participación de los *in_agent* en las subastas, estará determinada por las siguientes condiciones:

- Que los *participantes* no hallan ejecutado sus tareas opcionales.

- $mcet_i \leq Sa$, el tiempo de cómputo medio del *participante* i para generar y entregar una respuesta, debe ser menor o igual que el tamaño del *slack disponible* que está subastando el *SLS*.

Los *participantes* de la subasta recibirán y evaluarán esa información según sus creencias, limitaciones, características internas y el tipo de subasta en que se está participando.

Como las subastas se ejecutan en el servidor deliberativo del agente $\mathcal{ARTIS}$, el tiempo utilizado en las mismas es primordial para la correcta planificación y ejecución de sus *in-agents*. Esto implica que el tiempo dedicado a las negociaciones deberá estar siempre controlado, limitando el tiempo total dedicado a las subasta (T_{neg}), así como también el tiempo que tomen los *participantes* en generar sus ofertas (T_{max}^{offers}).

Conforme las directrices anteriores, se analizaron las siguientes subastas:

1. Subasta del Sobre Cerrado (primer y segundo precio).

2. Subasta Inglesa ("*subasta al alza*").

3. Subasta Alemana o también conocida por "*Subasta Holandesa*" ("*subasta a la baja*").

Se debe tener presente que cada una de las subastas aquí propuestas van a considerar los dos tipos de repuestas que proporcionan los *in_ agent* que son (sección 4.4.2.3):

1. *Refinamientos sucesivos.*

2. *Métodos múltiples.*

5.4. Formalización de las Subastas en un Agente $\mathcal{ARTIS}$

5.4.1. Definición las Subastas $\mathcal{ARTIS}$

El proceso de adjudicación en las subastas del agente $\mathcal{ARTIS}$ se definirá de la siguiente forma.

Definición.

Sean:

- $L_{in-agents}$ *: el conjunto con todos in-agent activos y vigentes en el sistema en un tiempo* $0 < t < t^f_{Sa}$.
- T^{offer}_{max} *: tiempo máximo destinado para generar una oferta.*
- P *: conjunto de todas las ofertas recibidas por SLS desde los participantes.*
- $p^t_i \in P$ *: la propuesta del in-agent* $i \in L_{in-agents}$ *al SLS en el tiempo* t.

El SLS seleccionará la mejor oferta en la llamada k *de la siguiente forma:*

$$BO^t_k = max\{p_i \in P\};\ con\, 0 < t < T^{offer}_{max}$$

5.4.2. Protocolo General para las Subastas en un Agente $\mathcal{ARTIS}$

Antes de continuar con las exposición de cada una de las subastas, se procederá a explicar el protocolo general utilizado para las subastas. Estos se basan en los protocolos propuestos por FIPA00032 (FIPA); FIPA00031 (FIPA).

Como se mencionó en la sección anterior, las subastas comienzan cuando el *SLS* envía una primera señal informando el comienzo de las negociaciones

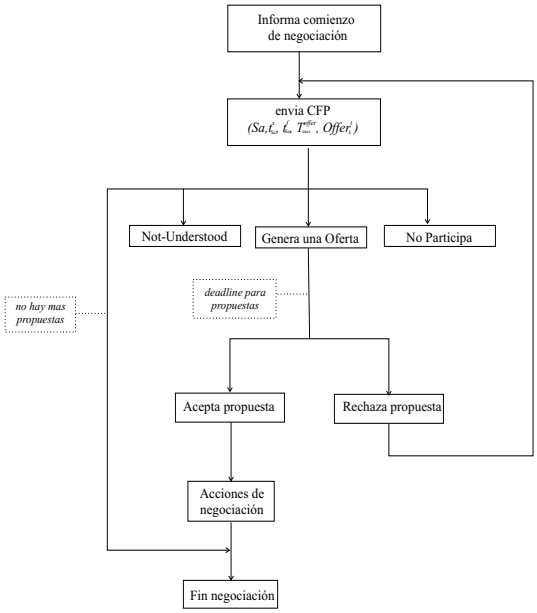


Figura 5.5: Diagrama general utilizado en las subastas

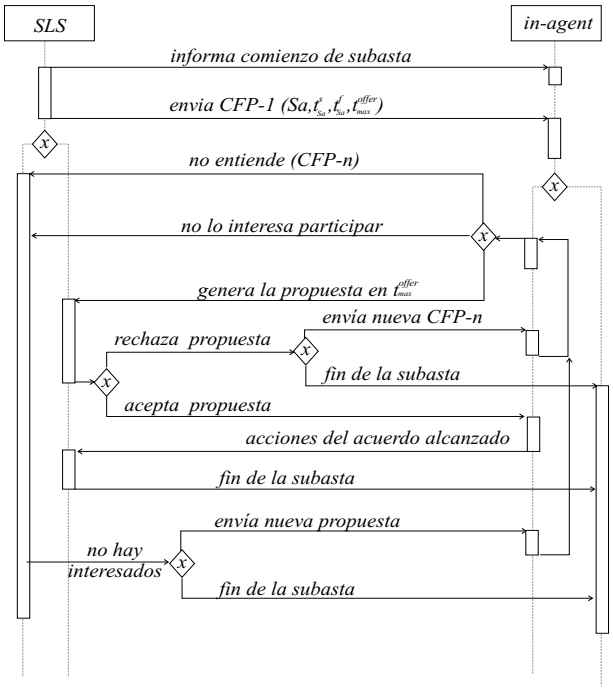


Figura 5.6: Protocolo general utilizado en las subastas

(Figura 5.5 y Figura 5.6). Luego envía a los *in-agent participantes* una llamada a propuestas (*Call For Proposes - CFP*) indicando: Sa, t^s_{Sa}, t^f_{Sa}, T^{offer}_{max}.

Como respuesta a esta *CFP* los *in_ agent* podrán:

- No entender la llamada (*not-understand*).
- No estar interesado en participar.
- Generar y enviar una propuesta.

En el tiempo destinado para generar ofertas (T^{offer}_{max}), los *in-agents* deberán responder al *SLS*. Este recibe todas las propuestas u ofertas hechas por los *in_ agent participantes*, y decide si las acepta o rechaza considerando para ello la calidad que ofrezcan con la solución que necesita, la prioridad del agente que la ofrece y la importancia de la solución. Si las acepta, el *SLS* y él o los *in-agents* ganadores proceden con las gestiones para comenzar la ejecución de la(s) solución o soluciones de los ganadores. De lo contrario, si las rechaza, hace una llamada a nuevas propuestas (*CFP*) indicando, en el caso de la subasta inglesa, la mejor oferta recibida hasta el momento, BO^t_k, y en el caso de la subasta alemana su siguiente oferta o propuesta.

Las subastas pueden terminar por una de las siguientes situaciones:

- Se ha alcanzado el tiempo destinado por el servidor deliberativo (*DS*) para las negociaciones (T_{neg}).
- No hay mas propuestas desde los *in-agents* participantes.
- No queda *slack disponible* para subastar.

5.5. Implementación de las Subastas en el Agente $\mathcal{ARTIS}$

Todas las subastas se implementarán en el servidor deliberativo (*DS*) del $\mathcal{AA}$ (ver figura 5.4).

Figura 5.7: Ejecución de las subastas.

Algoritmo 1 Implementación de subasta del sobre cerrado en el $\mathcal{AA}$.

```
0.    {
1.    DS_Activate;
2.    ListOf_ActiveIn_agents = EM_Search_ActiveIn_agents();
3.    ListOf_Participating = SLS_receive(ListOf_ActiveIn_agents);
4.    SLS_sendStartNegotiations(ListOf_Participating);
5.    SLS_Send_InfoToParticipating();
6.    While((NegotiationsTime>0)
7.          &&((length(ListOf_Participating)>0)
8.          &&(slack_time>0))
9.                ExecuteAuctions(ListOf_Participating);
10.   SLS_Execute_WinerInAgents();
11.   }
```

El proceso comienza cuando el servidor reflejo envía un mensaje al *DS* indicándole que existe tiempo para ejecutar partes opcionales de los *in-agent* activos en ese instante.

El algoritmo 1 muestra el código para insertar las subastas en la arquitectura del agente $\mathcal{ARTIS}$.

Como se puede apreciar, cuando el servidor deliberativo (*DS*) se activa con el mensaje enviado por el servidor reflejo que hay tiempo de *CPU* para ejecutar las partes opcionales de los *in-agent* activos (*línea 1*), el manejador de eventos

(*EM*) genera una agenda con la lista de aquellos *in-agents* que estén vigentes (*línea 2*), es decir, aquellos que:

- su *deadlines* no estén vencidos.
- que se hayan terminado de ejecutar correctamente en la anterior activación del *DS*.
- que las partes finales de los *in-agents* activos no haya terminado.

Aquellos *in-agent* que, luego de esta primera selección, sigan en la lista cumplirán con estar "*activos y vigentes*" pasarán a formar parte integral de la lista y podrán participar de la subasta. Esta lista es accedida por el planificador de segundo nivel (*SLS*) (*línea 3*) y envía un primer mensaje avisando el comienzo de las subastas a los integrantes de la misma, seguido de un mensaje con información para que los *in-agents* participantes envíen sus ofertas, propuestas o respuestas a esta llamada (*línea 4 y 5*). Después, mientras no se cumpla alguna de las condiciones de término de subastas mencionadas en el punto anterior, lleva a cabo la subasta en sí (*líneas 6, 7, 8 y 9*).

Finalmente, una vez terminados los procesos de subastas, el *SLS* procede a ejecutar aquellos *in-agents* que que ofertaron más por su ejecución (*línea 10*).

A continuación detallaremos con mas precisión la implementación de cada una de las subastas en la arquitectura interna del agente $\mathcal{ARTIS}$.

Antes de continuar con la implementación en sí de las subastas, cabe mencionar que hemos modificado algunas reglas de las subastas dadas las características especiales del agente $\mathcal{ARTIS}$ como:

- Si en una primera llamada no se presentase ningún *in-agent* comprador, el *SLS* no puede esperar hasta la siguiente activación del *Servidor Deliberativo* para subastar este slack, simplemente lo perdería. Como el perder un espacio de slack es inviable en un sistema de tiempo real, el SLS deberá hacer las gestiones necesarias para subastar íntegramente el slack siempre que existan *in-agents* activos en la agenda.

- Lo anterior implica que, si existen *in-agents* activos, las subastas no podrán declararse desiertas, siempre habrá al menos un ganador. Lo que las subastas determinarán será *quién* y *por cuánto tiempo* utilizará el slack.

Entonces, y según lo expuesto, las subastas que implementaremos tendrán las bases en las antes mencionadas (subasta del sobre cerrado, subasta inglesa y subasta holandesa) pero en ellas se subastarán múltiples unidades. En nuestro caso, cada unidad a subastar será un trozo o todo el slack disponible, y durarán hasta que no exista mas slack disponible, o se agote el tiempo destinado para el proceso de subastas, o bien no hayan más propuestas desde los *in-agents* activos.

De esta forma, a las reglas antes mencionadas, se le agregan:

- Para la subasta del sobre cerrado (en ambas versiones), las N unidades de slack subastadas se adjudicarán a los N *in-agents* con las pujas más altas.
- Para la subasta inglesa, el *in-agent* con la puja más alta se adjudica el slack. Si sobrase, el *SLS* llamará nuevamente a pujas.
- Para la subasta holandesa, el primer *in-agent* que acepte la propuesta del SLS se adjudica el slack. Al igual que la anterior, si sobrase, el *SLS* llamará nuevamente a pujas.

En particular, para las dos variantes de la subasta del sobre cerrado (primer y segundo precio) adecuaremos sus reglas y protocolos a las necesidades reales del agente $\mathcal{ARTIS}$ generando una versión alternativa de subasta del sobre cerrado la cual explicaremos a continuación.

5.5.1. Subasta del Sobre Cerrado

La estrategia de la subasta del sobre cerrado al primer precio es cada *in-agent* hará su valoración, considerando la valoración de los demás *in-agents*.

Por otro lado, la estrategia de la subasta del sobre cerrado al segundo precio es *decir siempre la verdad*. Como los *in-agents* de un agente $\mathcal{ARTIS}$ no mienten con respecto a sus pujas, podemos unir ambos tipos de subastas, considerando una única estrategia común: *cada in-agent valorará su puja según sus capacidades máximas y la utilidad que pueda darle a él y al agente al que pertenece.*

Así, resumiremos los dos tipos de subastas en una sola llamada en adelante *subasta del sobre cerrado* y en este tipo de subasta el *SLS* (vendedor) hace una única llamada a propuesta de ofertas por el slack disponible a los *in-agent participantes* (compradores). Una vez que el *SLS* recibe todas las ofertas, escogerá la mejor de todas para asignar el producto subastado (slack disponible).

5.5.1.1. Protocolo para *in-agents* con MKS de Refinamientos Sucesivos

El protocolo utilizado para las subastas del sobre cerrado está basado en el propuesto por FIPA (FIPA00031) (Figura 5.8) con la excepción de que, en el caso de recibir propuestas, no será repetitivo.

El *SLS* envía una *señal de comienzo* de la subasta a todos los *in-agent participantes* y *llama a propuestas* (*CFP*) de ofertas enviando en la llamada:

- S_a : duración del espacio a subastar
- t^s_{Sa} : instante de tiempo en que comenzará dicho espacio.
- t^f_{Sa} : instante de tiempo en que terminará dicho espacio.
- T^{offer}_{max} : tiempo límite para generar ofertas.

Los participantes al recibir esta petición, pueden optar a una de las siguientes acciones:

1. Indicar que no entiende la propuesta, por lo tanto ya no continúa participando de la subasta.

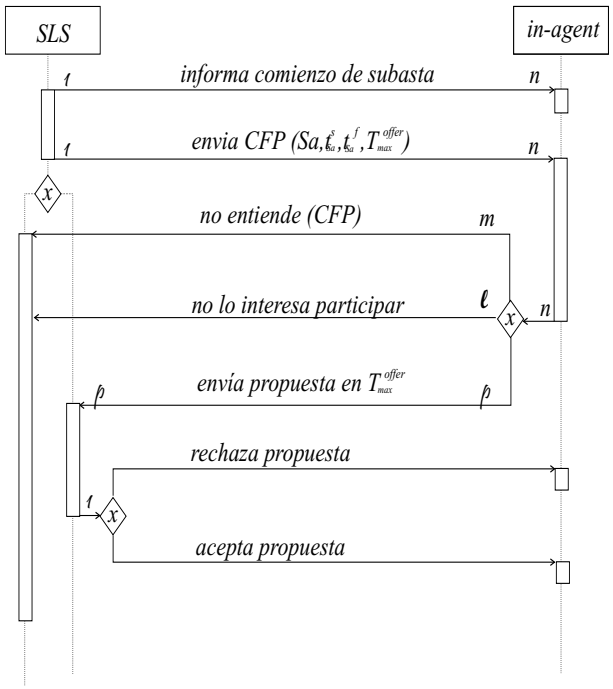


Figura 5.8: Protocolo de la Subasta del Sobre Cerrado

2. Indicar que no le interesa participar de las subastas.

3. Hacer una oferta (contra-oferta) por el tiempo de slack que se está subastado.

La acción que tome dependerá de:

- Que no se hayan ejecutado ninguno de sus niveles.
- $mcet \leq Sa$: El tiempo de ejecución medio ($mcet$) sea menor o igual al tiempo subastado por el *SLS*.
- $t_d < t^f_{Sa}$: El instante de vencimiento de su próximo deadline, t_d, sea menor que el instante de donde termina el espacio subastado.

La oferta que finalmente generará el *in-agent* está reflejada en la Ecuación (5.1).

$$Off(i) = \left[\frac{Q_i^{off}}{QE_{SLS}} * \frac{L_i^{ex}}{Levels_i} * \frac{1}{(D_i - mcet_i)} * Imp_i \right] \tag{5.1}$$

donde:

- QE_{SLS} : Calidad esperada por el *SLS*.
- Q_i^{off} : Calidad ofrecida por el *in-agent* i (al ejecutar L_i^{ex} del total de sus niveles).
- L_i^{ex} : Número de niveles que debe ejecutar el *in-agent* i para obtener la calidad ofrecida (Q_i^{off})
- $Levels_i$: Número total de niveles que tiene el *in-agent* i
- D_i : Próximo deadline del *in-agent* i
- $mcet_i$: Tiempo medio que necesita el *in-agent* i para ejecutar L_i^{ex}niveles

- Imp_i : Importancia del *in-agent* i

El *SLS* recibe todas las ofertas desde los *in-agent participantes* y selecciona la mejor.

5.5.1.2. Protocolo para *in-agents* con MKS de Métodos Múltiples

El protocolo utilizado por lo *in-agents* con este tipo de mks será básicamente el mismo que el utilizado por los *in-agents* con MKS de refinamientos sucesivos.

La diferencia radica en que este *in-agent* pujará por la ejecución de uno de sus niveles. En concreto por aquel que ofrezca la mejor calidad en la solución al problema del agente $\mathcal{ARTIS}$.

De esta forma solo variará la ecuación que genera la puja del *in-agent*, quedando como se muestra en la ecuación (5.2).

$$Off(i,L) = \left[\frac{Q_i^{off}}{QE_{SLS}} * \frac{1}{(D_i - mcet_i)} * Imp_i \right] \tag{5.2}$$

donde:

- QE_{SLS} : Calidad esperada por el *SLS*.
- Q_i^{off} : Calidad ofrecida por el *in-agent* i (al ejecutar el nivel L).
- D_i : Próximo deadline del *in-agent* i
- $mcet_i$: Tiempo medio que necesita el *in-agent* i para ejecutarel nivel L.
- Imp_i : Importancia del *in-agent* i

5.5.1.3. Implementación de la Subasta del Sobre Cerrado

Para implementar esta subasta hemos incorporado unas variantes para adaptarla al funcionamiento y arquitectura interna de un agente $\mathcal{ARTIS}$. Estas son:

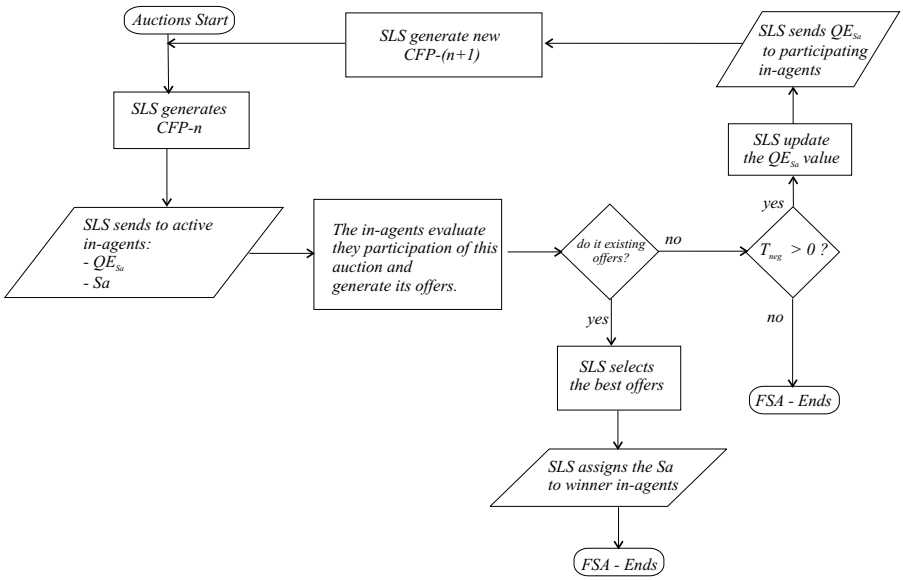


Figura 5.9: Diagrama de Subasta del Sobre Cerrado en $\mathcal{ARTIS}$

1. Como en este tipo de subastas el vendedor llama a ofertas una única vez, si no hay *in-agents* interesados, el *SLS* actualiza los parámetros (tiempo de slack disponible, tiempo para generar ofertas, etc.) y llama nuevamente a propuestas (*CFP*).

2. Si se han recibido propuestas desde los *in-agents* participantes, por utilizar el slack disponible, el *SLS* adjudica el slack disponible a la mejor oferta recibida. Si sobrase slack, asigna el restante a la segunda mejor oferta recibida. Así sucesivamente hasta que el slack disponible se haya asignado por completo o no hayan mas ofertas.

La implementación de esta subasta se muestra en la Figura 5.9.

Las funciones y parámetros que aparecen en el diagrama ya se explicaron en la sección anterior (protocolo para la subasta del sobre cerrado).

5.5.2. Subasta Inglesa

En este tipo de Subastas, el *vendedor* (*SLS*) abre la subasta con una propuesta mínima (*precio reservado* capítulo 3.1) y sobre esta propuesta los *participantes* (*in-agents*) efectuarán sus ofertas. Esta subasta termina cuando no existan más ofertas ó bien cuando el tiempo destinado a la subasta (T_{neg}) se agote, o no quede tiempo de slack disponible para subastar. Este tipo de subasta se conoce también como "*subasta al alza*".

5.5.2.1. Protocolo para *in-agents* con MKS de Refinamientos Sucesivos

El protocolo utilizado para la subasta inglesa se muestra en la Figura *5.10* y está basado en el protocolo propuesto por FIPA en FIPA00031 (FIPA).

El Planificador de Segundo Nivel (*SLS*) envía una *señal de comienzo* de la subasta a todos los participantes y *llama a propuestas* (*CFP*) de ofertas indicando:

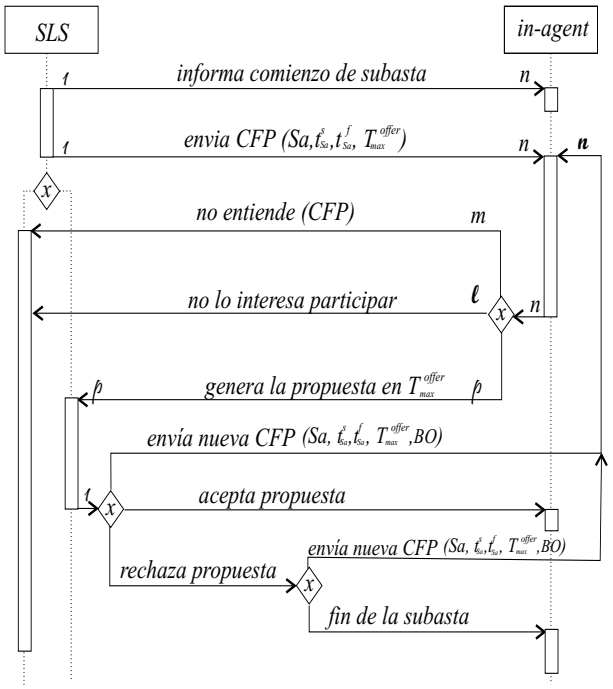


Figura 5.10: Protocolo para subastas inglesa en $\mathcal{ARTIS}$

- S_a : duración del espacio a subastar.
- t^s_{Sa} : instante de tiempo en que comenzará dicho espacio.
- t^f_{Sa} : instante de tiempo en que terminará dicho espacio.
- T^{offer}_{max} : tiempo límite para generar ofertas.
- B^t_k : en la primera llamada es la oferta mínima con que se abre la subasta. En las siguientes llamadas (k) pasa a ser la mejor oferta recibida hasta el momento t.
- QE_{SLS} : Calidad mínima que espera recibir el *SLS* en las respuestas de los *in-agents* participantes.

Los participantes al recibir la llamada, pueden optar por una de las siguientes acciones:

1. Indicar que no entiende la propuesta, por lo tanto ya no continúa participando de esta llamada.
2. Indicar que no le interesa participar en esta llamada a subasta.
3. Hacer una oferta (contra-oferta) por el tiempo que se está subastado (slack disponible).

La acción que tome el *in-agent participante* dependerá de:

- Que aún no se ejecuten algunos o todos de sus niveles.
- $mcet \leq Sa$: el tiempo de ejecución medio ($mcet$) sea menor o igual al slack disponible ofrecido por el *SLS*.
- $t_d < t^f_{Sa}$: el instante de vencimiento de su próximo deadline, t_d, sea menor que el instante de donde termina el espacio subastado.

La oferta que generará el *in-agent* la obtiene con la Ecuación (5.3).

$$Off(i) = \left[\frac{Q_i^{off}}{QE_{SLS}} * \frac{L_i^{ex}}{Levels_i} * \frac{1}{(D_i - mcet_i)} * Imp_i \right] + Off(i)^{(t-1)} \quad (5.3)$$

donde:

- QE_{SLS} : Calidad esperada por el *SLS*.
- Q_i^{off} : Calidad ofrecida por el *in-agent* i (al ejecutar L_i^{ex} niveles)
- L_i^{ex} : Número de niveles que debe ejecutar el *in-agent* i para obtener la calidad ofrecida (Q_i^{off})
- $Levels_i$: Número total de niveles que tiene el *in-agent* i
- D_i : Próximo deadline del *in-agent* i
- $mcet_i$: Tiempo medio que necesita el *in-agent* para ejecutar L_i^{ex}niveles.
- Imp_i : Importancia del *in-agent* i
- $Off(i)^{(t-1)}$: Anterior oferta del *in-agent* i.

Luego el *in-agent* deberá comparar la oferta que generó con la mejor oferta que ha recibido el *SLS* hasta este instante (BO_k^t). Aquí se producen dos situaciones:

1. $Off(i) \geq BO_k^t$, la oferta generada por el *in-agent* es mayor que la mejor recibida por el *SLS*, por lo tanto la oferta final con la que participará el *in-agent* es: $Offer_k^t = Off(i)$.
2. $Off(i) < BO_k^t$, la oferta generada por el *in-agent* es menor que la mejor recibida por el *SLS*. En este caso, el *in-agent* no descarta inmediatamente su participación sino que evalúa su situación para decidir el continuar o retirarse. Para esto considera:

a) La calidad que ofrece con la calidad que espera el *SLS*.

b) La proximidad del vencimiento de su próximo deadline.

c) Si estas características le son favorables, su oferta final se refleja en la ecuación (5.4).

$$Offer_k^t = Off(i) + [(BO_k^t - Off(i)) * (Q_i^{off} - QE_{SLS})] \qquad (5.4)$$

El *SLS* recibe todas las ofertas y selecciona la mejor, si aún queda tiempo para interactuar con los participantes llamará nuevamente a propuestas indicando en cada llamada la mejor oferta BO_k^t, recibida hasta el momento t y la calidad que espera recibir QE_{SLS}[5] en la siguiente llamada (*CFP-2*).

La subasta terminará cuando ya no hayan más ofertas o bien cuando el tiempo destinado para la subasta, T_{neg}, se agote. En ambos casos el *SLS* seleccionará la mejor oferta recibida.

5.5.2.2. Protocolo para *in-agents* con MKS de Métodos Múltiples

El protocolo utilizado para la subasta inglesa por *in-agents* con MKS de métodos múltiples será el mismo que el utilizado por los *in-agents* con MKS de refinamientos sucesivos.

La diferencia radica en que este *in-agent* pujará por la ejecución de uno de sus niveles. En concreto por aquel que ofrezca la mejor calidad en la solución al problema del agente *ARTIS*.

De esta forma solo variará la ecuación que genera la puja del *in-agent*, quedando como se muestra en la ecuación (5.5).

$$Off(i, L) = \left[\frac{Q_i^{off}}{QE_{SLS}} * \frac{1}{(D_i - mcet_i)} * Imp_i \right] + Off(i)^{(t-1)} \qquad (5.5)$$

donde:

[5] Será igual a la calidad asociada al *in-agent* que envió la mejor oferta en la llamada k.

- QE_{SLS} : Calidad esperada por el *SLS*.
- Q_i^{off} : Calidad ofrecida por el *in-agent* i (al ejecutar el nivel L)
- D_i : Próximo deadline del *in-agent* i
- $mcet_i$: Tiempo medio que necesita el *in-agent* para ejecutar el nivel que ofrece la calidad Q_i^{off}.
- Imp_i : Importancia del *in-agent* i
- $Off(i)^{(t-1)}$: Anterior oferta del *in-agent* i.

Luego el *in-agent* deberá comparar la oferta que generó con la mejor oferta que ha recibido el *SLS* hasta este instante (BO_k^t). Aquí se producen dos situaciones:

1. $Off(i) \geq BO_k^t$, la oferta generada por el *in-agent* es mayor que la mejor recibida por el *SLS*, por lo tanto la oferta final con la que participará el *in-agent* es: $Offer_k^t = Off(i)$.

2. $Off(i) < BO_k^t$, la oferta generada por el *in-agent* es menor que la mejor recibida por el *SLS*. En este caso, el *in-agent* no descarta inmediatamente su participación sino que evalúa su situación para decidir el continuar o retirarse. Para esto considera:

   a) La calidad que ofrece con la calidad que espera el *SLS*.

   b) La proximidad del vencimiento de su próximo deadline.

   c) Si estas características le son favorables, su oferta final se refleja en la ecución (5.6).

$$Offer_k^t = Off(i) + [(BO_k^t - Off(i)) * (Q_i^{off} - QE_{SLS})] \qquad (5.6)$$

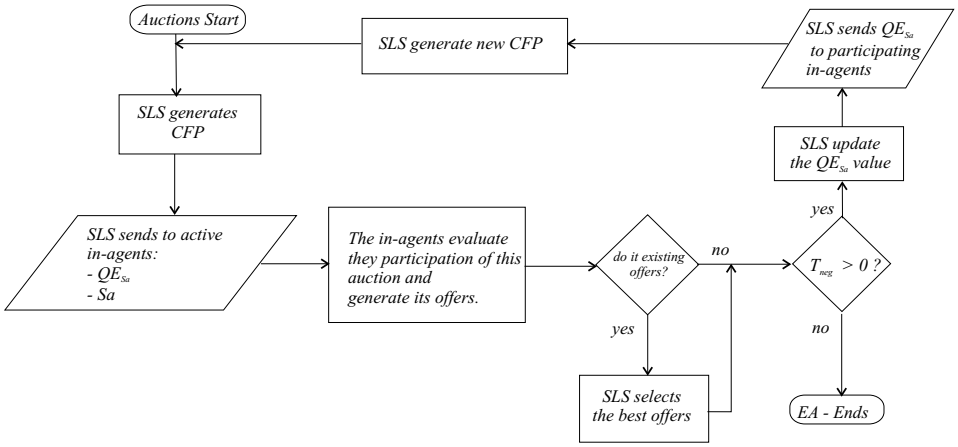


Figura 5.11: Diagrama de Subasta Inglesa en $\mathcal{ARTIS}$

5.5.2.3. Implementación de la Subasta Inglesa

Para la implementación de este tipo de subastas también consideramos una modificación en su protocolo original propuesto por FIPA. Este es, que en cada llamada *CFP-n* el *SLS* compara el *BO* recibido con su *precio máximo reservado (Raiffa, 1982)* y si éste es cercano en un *90 %* asigna el slack disponible al *in-agent* ganador. Si luego de esta acción sobrase slack, llama nuevamente a propuestas, de lo contrario se considera como un factor para el término de la subasta.

Una vez que se han terminado las negociaciones, el *SLS* procede a ejecutar los *in-agents* ganadores.

El diagrama de la figura 5.11 muestra la implementación y funcionamiento de esta subasta en el agente $\mathcal{ARTIS}$.

5.5.3. Subasta Alemana

En la *Subasta Holandesa* o *Subasta Alemana*, el vendedor (*SLS*) comienza la subasta enviando a los participantes (*in-agents* activos y vigentes) una oferta

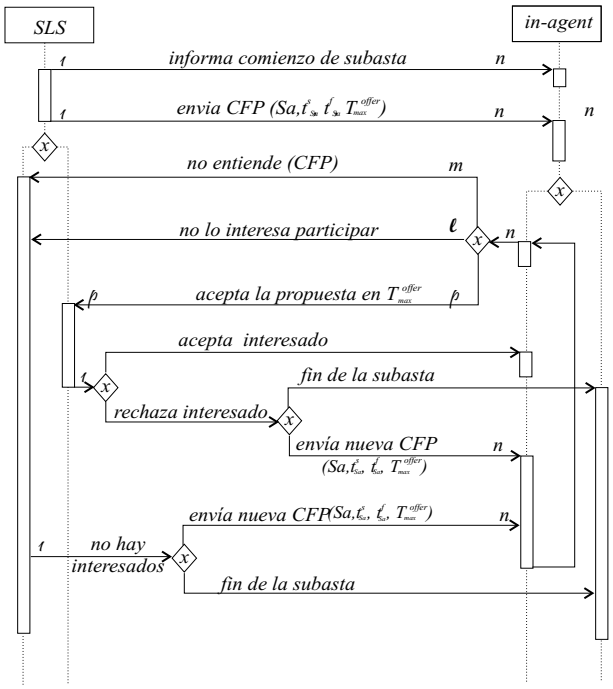


Figura 5.12: Subasta Alemana

por el recurso que está vendiendo (tiempo de CPU). Esta oferta comienza con un valor inicial elevado y lo va reduciendo progresivamente hasta que algún participante lo acepte.

5.5.3.1. Protocolo para *in-agents* con MKS de Refinamientos Sucesivos

El protocolo de esta subasta se muestra en la figura 5.12. Está basado en el propuesto por FIPA para este tipo de subastas (FIPA00032). La subasta Alemana también es conocida como *Subasta Holandesa* o *Subasta a la baja*.

El comienzo de la subasta lo hace *SLS*, quien al recibir el tamaño del *slack disponible* para la ejecución de partes opcionales de los *in-agents* envía un

mensaje a los *in-agent* vigentes y activos proponiéndoles una oferta para que se ejecuten en este tiempo de slack. Para esto, el *SLS* deberá "*encontrar*" el valor de su oferta tal que sea atractiva a los *in-agent participantes*. Así, este valor dependerá de:

- Tamaño del *slack disponible* recibido, Sa.
- Calidad que espera recibir en la respuesta al problema en cuestión, QE_{SLS}. Esta calidad comenzará en *95 %* y disminuirá en cada nueva llamada de la subasta hasta llegar a un *"precio reservado"* o "*calidad mínima"* del *20 %*. Esta disminución será de dos formas:
  - Cuando ya se ha adjudicado una parte del slack a un *in-agent*, la calidad que este ofrecía pasa a ser la calidad mínima que espera recibir el *SLS* en la siguiente llamada.
  - Si no hay interesados en la oferta del *SLS*, este asignará el valor obtenido con la ecuación 5.7 a la calidad esperada y recalculará el peso con este nuevo valor.
- Tiempo destinado a la negociación, T_{neg}.

La función que determinará el *peso* que finalmente el *SLS* asocie al *slack disponible* y que envíe a los *in-agents*, será del tipo exponencial con pendiente negativa de tal forma que la disminución de la oferta sea gradual y no alcance a cero. Esta oferta minina se conoce como precio mínimo reservado (Raiffa, 1982), de tal forma que si la oferta del *SLS* estuviera por debajo de este valor no le convendría vender. Luego, la fórmula que calculará la el peso asociado a la calidad que espera recibir está descrito por la ecuación 5.7.

$$WeightSLS(t) = e^{(-ln(t+Sa))} * QE_{SLS} \tag{5.7}$$

Donde

- t: tiempo actual donde se calcula este peso.
- Sa: tamaño del *slack disponible* para ejecución de las partes opcionales de los *in-agent participantes*.
- QE_{SLS}: calidad que espera recibir el *SLS* desde los *in-agent participantes* por la solución del problema.

Una vez que se ha calculado el peso, $WeightSLS$, el *SLS* procede a enviar este valor a los *in-agent activos y vigentes*, enviando además:

- Sa : tamaño del slack disponible.
- QE_{SLS} : calidad que espera recibir.
- T_{max}^{offer} : tiempo máximo que tienen los *in-agent* participantes para responder.
- t_{Sa}^{s} : instante de tiempo en que comenzará dicho espacio.
- t_{Sa}^{f} : instante de tiempo en que terminará dicho espacio.

Los *in-agent participantes*, al recibir esta información, proceden a evaluarla considerando sus creencias. En esta evaluación los *in-agent* considerarán los siguientes puntos:

- Q_i^{off} : calidad que pueden ofrecer para solucionar el problema tal que cumple con: $Q_i^{off} \approx QE_{SLS}$. Y se calcula como:
  - $Q_i^{off} = \sum_{j=1}^{L_i^{ex}} Q_j$
  - donde L_i^{ex} son los niveles que necesita ejecutar para ofrecer la calidad Q_i^{off}.
- $mcet_i$: tiempo medio de ejecución que necesita el *in-agent* para ofrecer la calidad Q_i^{off}, es decir en ejecutar L_i^{ex} de sus niveles.

Así la evaluación que realizan los *in-agent participantes* por la oferta recibida desde el *SLS* se regirá por la ecuación 5.9.

$$\left(\frac{mcet_i}{Sa}\right) + \left(\frac{Q_i^{off}}{QE_{SLS}}\right) \leq 1,5 \tag{5.8}$$

Finalmente, si la desigualdad planteada en 5.9 se cumple, el *in-agent participante* estará en posición de aceptar la oferta del *SLS*. Una vez que todas los *in-agent participantes* han enviado sus respuestas al *SLS*, procede a la adjudicación del *slack disponible* al primer *in-agent* que haya aceptado. Si luego de la adjudicación se cumple que $Sa > 0$, el *SLS* procederá llamar nuevamente a propuestas.

5.5.3.2. Protocolo para *in-agents* con MKS de Métodos Múltiples

Al igual que en las anteriores subastas, el protocolo utilizado por lo *in-agents* para este caso será mismo que itulizan los *in-agents* con MKS de refinamientos sucesivos, nuevamente la diferencia radica en la forma que tendrán los *in-agents* para evaluar la oferta del *SLS*. Esta se refleja en la desigualdad planteada en (5.9).

$$\left(\frac{mcet_i}{Sa}\right) + \left(\frac{Q_i^{off}}{QE_{SLS}}\right) \leq 1,5 \tag{5.9}$$

- Q_i^{off} : calidad que pueden ofrecer para solucionar el problema tal que cumple con: $Q_i^{off} \approx QE_{SLS}$.
- $mcet_i$: tiempo medio de ejecución que necesita el *in-agent* para ofrecer la calidad Q_i^{off} (ejecutar el nivel que la ofrece).

5.5.3.3. Implementación de la subasta alemana en el agente $\mathcal{ARTIS}$

Para este tipo de subastas también consideramos una modificación en su protocolo. Este es, que en cada envío de la oferta del *SLS*, éste compara la

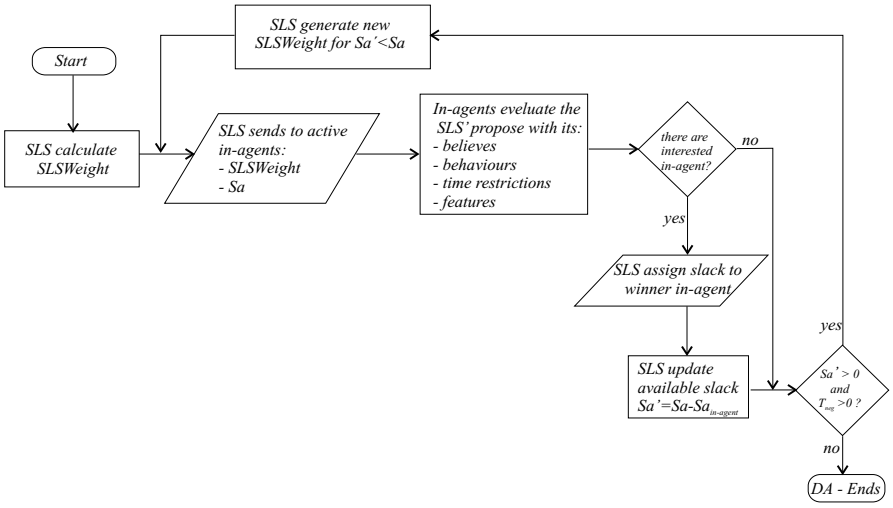


Figura 5.13: Diagrama de Subasta Alemana.

calidad que le dará el *in-agent* ganador i (Q_i^{off}), con la calidad que espera recibir (QE_{SLS}), si el porcentaje de diferencia está en un *90 %* asigna el slack a ese *in-agent*. Si sobrase slack envía una nueva oferta por él a los demás *in-agent*, de lo contrario se considera un factor para el término de la subasta.

Una vez que se han terminado las negociaciones, el *SLS* procede a ejecutar los *in-agents* ganadores.

El diagrama de la figura 5.13 muestra los pasos en la inserción de la subasta Alemana en la arquitectura del agente $\mathcal{ARTIS}$.

Capítulo 6

Pruebas Aplicadas en el Agente *ARTIS*

En este capítulo expondremos las pruebas de nuestras subastas que hemos realizado en el agente *ARTIS*. Estas pruebas tiene la finalidad de comparar nuestras técnicas de negociación con las demás heurísticas o políticas de planificación implementadas en el agente *ARTIS* (capítulo 4.5), sometiéndolas a diversos escenarios.

Estas pruebas se realizaron de dos formas:

1. *Pruebas teóricas o simuladas.* Para estos tests, utilizamos un Simulador desarrollado especialmente para crear, probar y ejecutar agentes *ARTIS* antes de implementarlos sobre una plataforma real.

2. *Pruebas experimentales o reales.* Para estos tests, implementamos las baterías de pruebas generadas para la simulación, sobre una plataforma

real.

Comenzaremos este capítulo explicando como se obtuvieron los distintos valores temporales para los *in-agents*, utilizados luego en la generación las baterías de pruebas que fueron sometidas a las simulaciones y a las pruebas reales.

Indicaremos también los criterios utilizados para las comparaciones entre las distintas heurísticas con nuestras subastas. Justificando también el motivo de la selección de estos criterios.

Finalmente mostraremos los resultados obtenidos en las ejecuciones de estas baterías de pruebas tanto en las simulaciones como en las experimentaciones.

6.1. Generación de baterías de Pruebas

Para las tareas basadas en sistemas de tiempo-real como lo son los *in-agents* del agente $\mathcal{ARTIS}$, los tiempos involucrados en las diversas etapas de su ejecución deben estar debidamente acotadas y controladas. Para esto hemos utilizado las distribuciones propuestas por Campos-López (2004) en su tesis doctoral. Con estas distribuciones generaremos las restricciones temporales de los *in-agents* de los agentes $\mathcal{ARTIS}$, tales como: deadlines, períodos, tiempos de ejecución, etc.

Comenzamos determinando el porcentaje de cargas que tendrá el sistema donde se ejecutará el agente. Distinguimos dos tipos de cargas:

1. *Carga crítica.* Esta carga representa el porcentaje del tiempo total de ejecución del sistema que es consumido por los componentes críticos del agente.

2. *Saturación o carga opcional.* Esta carga representa el porcentaje de tiempo que sería necesario para ejecutar todos los niveles opcionales del agente frente al tiempo total de ejecución del sistema. Es una relación entre el tiempo total necesario para ejecutar las partes opcionales de las tareas activas en el sistema (entre ellas las partes opcionales de los *in-agents*),

y el tiempo real disponible para ello. Este porcentaje podría ser mayor del *100 %*, lo cual se utiliza para representar que no se dispone de tiempo suficiente para ejecutar todos los niveles opcionales del agente. El que este porcentaje de saturación sea mayor del *100 %*, está considerado el escenario más común, ya que por lo general en el sistema donde se están ejecutando tareas críticas y opcionales, el tiempo destinado para la ejecución de las últimas (tareas opcionales) está muy limitado y debe ser compartido por las demás tareas acríticas del sistema.
No obstante, lo anterior no implica que los niveles más altos de los *in-agents* no tengan la posibilidad de ejecutarse. Por el contrario, sus ejecuciones están también asociadas a sus *deadlines* y *períodos* de activación. Es decir, sus ejecuciones dependerán también de la cantidad de tareas o *in-agents* activos en el sistema, en cada instante. Para esto, consideramos dos escenarios probables:

a) Cuando los deadlines de los *in-agents* (tareas) son iguales a sus períodos de activación, con lo cual en todo momento existen *in-agents* activos.

b) Cuando los deadlines de los *in-agents* son menores o iguales que sus períodos de activación, lo que significa que el sistema será más restrictivo en cuanto al tiempo, pudiendo existir intervalos sin *in-agents* activos.

La *carga crítica* del sistema la obtendremos con la ecuación (6.1).

$$\rho = \sum_{i=1}^{n} \bar{C}_i * f_i \tag{6.1}$$

Donde $\bar{C}_i$ es el tiempo de cómputo del *in-agent*$_i$ (*tarea*$_i$) y f_i serán sus frecuencias medias de activación.

Por otro lado, la *saturación* presente en el sistema la determinaremos con la ecuación (6.2)

$$\rho(t) = max(\rho_i(t)),\ \forall i \in [1 \ldots total\_de\_tareas\_activas] \tag{6.2}$$

Además utilizaremos la desigualdad de Tchebycheff[1] (Ecuación (6.3)) con media y varianza muestral $\hat{\mu_i}$, $\hat{\sigma}_i$ respectivamente, para obtener los tiempos de cómputos, C_i, para los distintos *in-agents* cuyas funciones de densidad de probabilidad son desconocidas.

$$P(\hat{\mu_i} - k\hat{\sigma}_i \leq C_i \leq \hat{\mu_i} + k\hat{\sigma}_i) \geq 1 - \frac{1}{k^2} \tag{6.3}$$

Además, si consideramos $k \in \Re, k \geq 1$ medidas son finitas, los valores de C_i estarán en el intervalo de centro $\hat{\mu_i}$ y radio $\hat{\sigma}_i$.

El siguiente teorema muestra que, sólo con el extremo derecho de esta desigualdad, se pueden obtener los C_i necesarios para nuestras simulaciones.

Teorema.
Sean una variable aleatoria, C_i, con media y varianzas finitas, μ_i y σ_i^2, que modela el tiempo de computación de una tarea inteligente, τ_i, inserta en un STR con probabilidad de fallo admisible α.
Sean $f_i(t)$ la función de densidad de probabilidad de C_i y $C_{i,\alpha} \in \Re$ tal que:

$$P(C_i \leq C_{i,\alpha}) = \int_{-\infty}^{C_{i,\alpha}} f_i(t)dt = 1 - \alpha \tag{6.4}$$

Entonces siempre se verificará que

$$P(C_i \leq \mu_i + \alpha^{-1/2}\sigma_i) \geq P(C_i \leq C_{i,\alpha}) \tag{6.5}$$

***Prueba**:*

[1] Propuestas por Campos-López (2004).

Usando la desigualdad de Tchebychef y las propiedades de $C_{i,\alpha}$ puede asegurarse que,

$$P(\mu_i + \alpha^{-1/2}\hat{\sigma}_i \leq C_i \leq \mu_i + \alpha^{-1/2}\hat{\sigma}_i) \geq 1 - \alpha \tag{6.6}$$

$$P(C_i \leq C_{i,\alpha}) = \int_{-\infty}^{C_{i,\alpha}} f_i(t)dt = 1 - \alpha \tag{6.7}$$

Es decir, que siempre es cierto que

$$P(\mu_i + \alpha^{-1/2}\hat{\sigma}_i \leq C_i \leq \mu_i + \alpha^{-1/2}\hat{\sigma}_i) \geq P(C_i \leq C_{i,\alpha}) \tag{6.8}$$

Usando 6.7 y suponiendo que 6.5 es falsa, se puede demstrar el teorema pro reducción al absurdo. Así, si 6.5 es falsa se verificará que:

$$P(C_i \leq \mu_i + \alpha^{-1/2}\sigma_i) \geq P(C_i \leq C_{i,\alpha}) \tag{6.9}$$

Usando la función de densidad de probabilidad de C_i, $f_i(t)$, a partir de 6.9 se obtiene inmediatamente que

$$\int_{-\infty}^{\mu_i+\alpha^{-1/2}\sigma_i} f_i(t)dt < \int_{-\infty}^{C_{i,\alpha}} f_i(t)dt \tag{6.10}$$

Puesto que, por ser $f_i(t)$ una función de densidad de probabilidad, $f_i(t) \geq 0 \forall t \in \Re$, para que se cumpla 6.10 la función de densidad de probabilidad $f_i(t)$ tomará valores no nulos en el intervalo $[\mu_i + \alpha^{-1/2}\sigma_i, C_{i,\alpha}]$, y además

$$\mu_i + \alpha^{-1/2}\sigma_i < C_{i,\alpha} \tag{6.11}$$

Pero si se verifica 6.11, al expresión 6.8 es equivalente a

$$P(C_i \leq \hat{\mu}_i+\alpha^{-1/2}\hat{\sigma}_i)-P(C_i \leq \hat{\mu}_i-\alpha^{-1/2}\hat{\sigma}_i) \geq P(C_i \leq \hat{\mu}_i+\alpha^{-1/2}\hat{\sigma}_i)+P(\hat{\mu}_i-\alpha^{-1/2}\hat{\sigma}_i \leq C_i \leq C_{i,\alpha}) \tag{6.12}$$

Expresión que simplificada resulta finalmente

$$P(C_i \leq \hat{\mu}_i + \alpha^{-1/2}\hat{\sigma}_i) + P(\hat{\mu}_i + \alpha^{-1/2}\sigma_i \leq C_i \leq C_{i,\alpha}) \leq 0 \tag{6.13}$$

Puesto que las probabilidades no pueden ser menores que cero, para que se verifique la expresión anterior ambos sumandos deben ser cero. Pero para que el segundo sea cero debe cumplirse que

$$\mu_i + \alpha^{-1/2}\sigma_i = C_{i,\alpha}\, Contradiccion! \tag{6.14}$$

Lo que contradice 6.11, obtenida tras suponer que 6.5 era falsa.□

Los deadlines de cada *in-agent* se obtendrán a partir de sus tiempos de cómputos (ecuación 6.15).

$$D_i = \bar{C}_i + 4 \cdot \bar{C}_i \tag{6.15}$$

En cuanto a sus períodos, se obtendrán desde los deadlines, utilizando para someter nuestras baterías de pruebas a diferentes condiciones (deadlines iguales a los períodos y deadlines menores o iguales a los períodos), esto se explica con mayor detalle en las siguientes secciones.

6.2. Criterios de Comparación

Para comparar nuestras subastas con las demás heurísticas implementadas en el agente $\mathcal{ARTIS}$, nos centramos en el objetivo final de nuestras subastas el cual era obtener la mejor calidad posible en la solución al problema del agente $\mathcal{ARTIS}$. Por lo tanto nuestro principal parámetro de comparación será la calidad que obtienen los *in-agents* al momento de dar solución a un problema específico del agente.

La duración de nuestras mediciones fueron hasta el ***hiper-período***, el cual

es calculado como el mínimo común múltiplo del conjunto de períodos de todos los *in-agents* del agente $\mathcal{ARTIS}$. Se hicieron así porqué cuando los *in-agents* llegan hasta este límite, la relación entre sus activaciones se repite produciendo las mismas situaciones, es decir, sus comportamientos regresan al estado inicial del sistema.

6.2.1. Definiciones

Antes de continuar con las descripción de las medidas utilizadas para las comparaciones entre las heurísticas actuales de $\mathcal{ARTIS}$ y nuestras subastas, procederemos a definir algunos términos que utilizaremos en ellas. Estos son:

- N : número de *in-agents*
- MR_i : número de *MKS* con método de refinamientos sucesivos
- MM_i : número de *MKS* con métodos múltiples
- ACT_i : número de activaciones del *in-agent*
- L_i^m : número de niveles del *MKS* m del *in-agent* i
- $q_{m,i}^l$: calidad ofrecida por el nivel l del *MKS* m del *in-agent* i
- q_{max}^m : calidad máxima ofrecida de entre todos los niveles del *MKS* m del tipo métodos múltiples
- ne_i^m : número de niveles ejecutados del *MKS* m del *in-agent* i
- $e_{m,i}^l$: número de ejecuciones del nivel l del *MKS* m del *in-agent* i
- $ef_{m,i}^l$: número de ejecuciones efectivas del nivel l del *MKS* m del *in-agent* i. Para *MKS* con métodos múltiples solo se considerará efectiva aquel que proporcionó la mejor calidad.
- I_i^m : importancia del *MKS* m del *in-agent* i.

Los criterios para calcular las distintas calidades obtenidas en cada uno de los test realizados, se basaron en las propuestas en (Hernandez-López, 2003), las cuales procederemos a explicar a continuación.

Calidad Real Absoluta (*CRA*) Esta calidad se obtiene como la suma de las calidades proporcionadas por cada nivel del *in-agent* multiplicada por la importancia de sus *MKS*s asociados (ecuación 6.16).

Para *in-agents* con *MKS*s de refinamiento progresivo, se consideran todas las importancias asociadas a cada uno de ellos.

Para los *MKS*s de métodos múltiples, se consideran solo las importancias de aquellas *MKS*s que han sido ejecutadas.

$$CRA = \sum_{i=1}^{N} \left\{ \sum_{m=1}^{MR_i} \left[I_m^i * \sum_{l=1}^{L_m^i} \left(q_{m,i}^l * e_{m,i}^l \right) \right] + \sum_{m=1}^{MM_i} \left[I_m^i * \sum_{l=1}^{L_m^i} \left(q_{m,i}^l * ef_{m,i}^l \right) \right] \right\} \tag{6.16}$$

Calidad Ideal (*CI*) Esta calidad se calcula como la calidad total que se puede obtener si de ejecutasen todos los niveles de los *in-agents* (Ecuación 6.17).

$$CI = \sum_{i=1}^{N} \left\{ ACT_i * \left[\sum_{m=1}^{MR_i} \left(I_m^i * \sum_{l=1}^{L_m^i} q_{m,i}^l \right) + \sum_{m=1}^{MM_i} \left(I_m^i * q_{max}^m \right) \right] \right\} \tag{6.17}$$

Calidad Real Relativa (*CRR*) Los valores obtenidos para la *CRA* no nos servirán como medidas uniformes para todas nuestras pruebas, ya que en algunas puede darnos mejores resultados que en otras, dependiendo de las restricciones temporales del *in-agent* y de la máxima calidad que este pueda ofrecer en un instante dado.

Así utilizaremos la *Calidad Real Relativa* (*CRR*), la cual se calcula como un cuociente entre la *CRA* y la *CI* (Ecuación 6.18). Siendo esta medida más representativa de los resultados que obtengamos permitiéndonos compararlos con mayor fiabilidad.

$$CRR = \frac{CRA}{CI} \tag{6.18}$$

Dado que en nuestras subastas los principales participantes son los *in-agents* del agente $\mathcal{ARTIS}$, y en particular, ellos subastan *tiempo de CPU* para ejecutar sus respectivos niveles, utilizaremos esta característica para compara las distintas políticas.

Así, otra medida que utilizaremos para nuestras comparaciones serán la cantidad de niveles que se ejecutan del agente $\mathcal{ARTIS}$ en cada batería a la que fue sometido.

Porcentaje Niveles Ejecutados (ecuación (6.21)) Este porcentaje es el cuociente entre los niveles efectivamente ejecutados del agente $\mathcal{ARTIS}$ (Ecuación 6.19) y el total de niveles del mismo (Ecuación 6.20).

$$Niveles\_Ejecutados = \sum_{i=1}^{N} \left[\sum_{j=1}^{MM_i} \left(\sum_{h=1}^{ne_h^j} h \right) + \sum_{j=1}^{MR_i} \left(\sum_{h=1}^{ne_h^j} h \right) \right] \tag{6.19}$$

$$Total\_Niveles = \sum_{i=1}^{N} \left[\sum_{j=1}^{MM_i} \left(\sum_{h=1}^{L_h^j} h \right) + \sum_{j=1}^{MR_i} \left(\sum_{h=1}^{L_h^j} h \right) \right] \tag{6.20}$$

$$\%niveles = \frac{Niveles\_Ejecutados}{Total\_Niveles} \tag{6.21}$$

6.3. Pruebas Simuladas

Para el diseño de agentes $\mathcal{ARTIS}$ existe una herramienta llamada InSiDe (Julián et al., 2000). InSiDe facilita el diseño e implementación del agente $\mathcal{ARTIS}$ traduciendo el modelo de alto nivel o *modelo de usuario* ingresado por el diseñador, a un modelo de bajo nivel o *modelo de sistema*, el cual es entendido por el sistema donde se ejecutará finalmente.

Además InSiDe ofrece, como parte de sus herramientas, un módulo para la simulación de la ejecución del agente $\mathcal{ARTIS}$. El cual utilizaremos en primera instancia para ejecutar nuestras baterías de pruebas.

La razón para comenzar utilizando esta herramienta de simulación es la gran cantidad de pruebas que se realizarán para obtener conclusiones significativas para ser luego representadas en una plataforma real.

Además, en el simulador ofrecido por InSiDe, podremos generar y comparar diferentes escenarios, pudiendo luego determinar cuales de todas las baterías ejecutadas implementaremos en el sistema real.

Para generar y ejecutar agentes $\mathcal{ARTIS}$ en el simulador, se deben ingresar todos los datos concernientes al sistema donde se ejecutará finalmente tales como *carga crítica* y *saturación* (para simulación del sistema final real), y las características de los componentes de su *modelo de usuario*, es decir:

- Número de *in-agents* del agente
- Períodos y deadlines de cada *in-agent*
- Número de MKS de cada *in-agent*
- Número de niveles opcionales e importancia de cada MKS. La importancia del MKS refleja su aporte a la calidad final ofrecida por el *in-agent*.
- Tiempo de ejecución y calidad asociada a los niveles de cada *in-agent*.

6.3.1. Tests Preliminares

Inicialmente, las variantes en nuestros tests (baterías de pruebas), fueron orientadas a ir eliminando variables que no sean significativas para las comparaciones entre los métodos.

De esta forma, como nuestras subastas y las políticas ya implementadas en el agente $\mathcal{ARTIS}$, se ejecutan en el *Servidor Deliberativo* esto implica que solo les afectaría directamente los niveles de saturación presentes en el sistema e indirectamente los niveles de carga crítica en el mismo.

Así que comenzaremos con pruebas generales que detecten la influencia que ejerce la carga crítica sobre nuestras subastas y heurísticas implementadas en el agente $\mathcal{ARTIS}$.

Para esto ejecutamos los primeros escenarios de pruebas de la siguiente forma:

- Ejecutar *800* pruebas con *12 in-agents* cuyos deadlines son iguales a sus períodos y con las especificaciones de la tabla 6.1.

- Ejecutar *800* pruebas con *12 in-agents* cuyos deadlines son menores a sus períodos hasta un 70 % y con las especificaciones de la tabla 6.1.

Tabla 6.1: Parámetros para Generación de Pruebas Simuladas.

Dato	Rango de Selección	Forma de Selección
$\#MKS$	$[1, 2, 3, 4]$	Uno de los valores
#Niveles por *MKS*	$[1, 2, 3, 4]$	Uno de los valores
Importancia del *MKS*	$[1, 2, 3]$	Uno de los valores
Período de c/*in-agent*	$[1000 - 1000000]$(mseg)	Un valor del intervalo
Deadline de c/*in-agent*[1]	$[1000 - 1000000]$(mseg)	Un valor del intervalo
Deadline de c/*in-agent*[2]	$(70\,\%, 80\,\%, 90\,\%, 100\,\%)$	Uno de los Valores

[1]: para *in-agents* con deadlines iguales al período

[2]: para *in-agents* con deadlines menores que el período

Los resultados obtenidos en estas pruebas se muestran en las gráficas de las figuras 6.1 y 6.2.

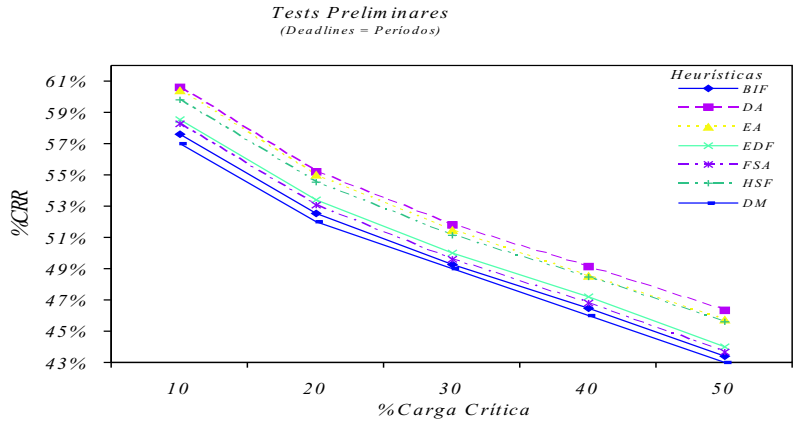


Figura 6.1: Pruebas Simuladas variando la Carga Crítica, para *deadlines* = *períodos*.

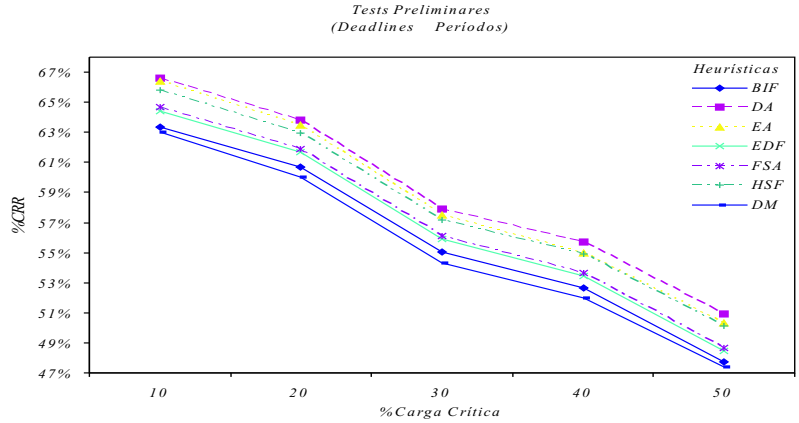


Figura 6.2: Pruebas Simuladas variando la Carga Crítica, para *deadlines* $\leq$*períodos*.

Como se puede observar de las gráficas en las figuras 6.1 y 6.2, las heurísticas

se comportan de forma similar para las distintas cargas opcionales. También se puede observar que todas las heurísticas que disminuyen su rendimiento proporcionalmente al aumento de la carga crítica.

En vista de los resultados obtenidos en estas primeras pruebas decidimos mantener la carga opcional fija en *30 %* para las siguientes simulaciones, ya que nuestros métodos están relacionados directamente con la cantidad de carga opcional o saturación presente en el sistema donde se están ejecutando.

En esta primera etapa, pudimos apreciar también, que la heurística que entrega las peores calidades es la *Deadline Monotonic* (*DM*). Por lo tanto, quisimos comprobar cual era su comportamiento cuando la saturación del sistema variaba.

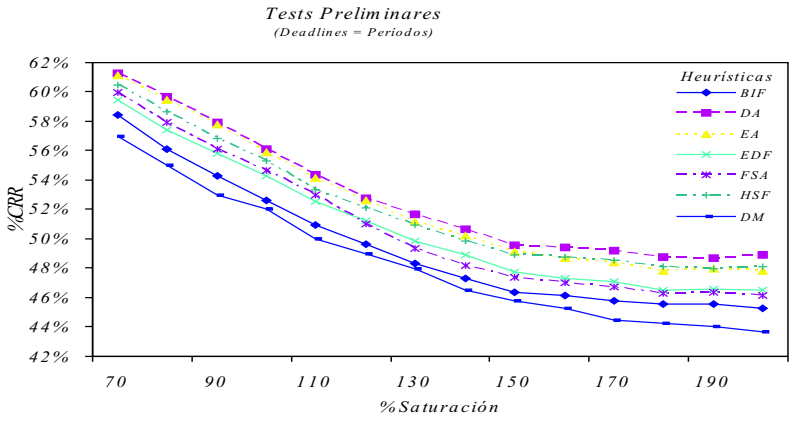


Figura 6.3: Pruebas Simuladas variando la Saturación, para *deadlines = períodos*.

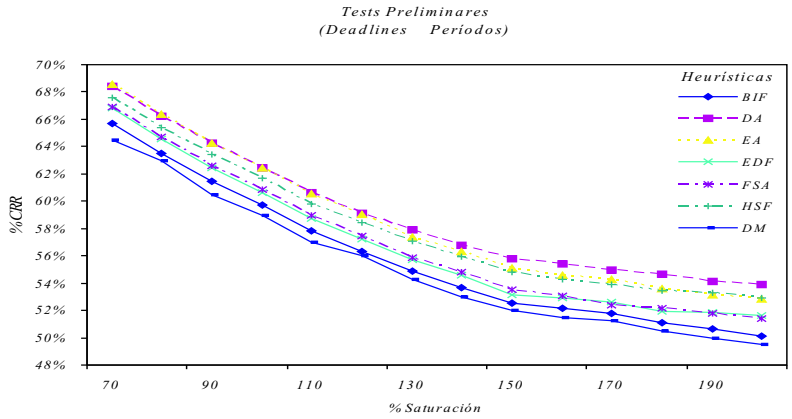


Figura 6.4: Pruebas Simuladas variando la Saturación, para *deadlines* $\leq$*períodos*.

Las gráficas de las figuras 6.3 y 6.4 muestran que, cuando la saturación del sistema aumenta, la heurística *DM* continua entregando los peores resultados. Sin embargo, las demás heurísticas tienden a un comportamiento uniforme, pero este no será concluyente pues en ambas gráfica (Figuras 6.3 y 6.4) existen diferencias en las calidades entregadas por las demás heurísticas.

Según estos comportamientos, decidimos no considerar la heurística *DM* dado los resultados que entrega y que no serán de relevancia para las evaluaciones de nuestras subastas.

La siguiente cuestión que quisimos dilucidar es el tipo de comportamiento que tendrán nuestras subastas cuando variemos la saturación del sistema manteniendo constante la carga crítica en *30 %*.

6.3.2. Baterías de Pruebas Exhaustivas

En las siguientes pruebas variamos la saturación opcional entre el *80 %* al *200 %* manteniendo constante la carga crítica del sistema en 30 %.

En estas baterías de pruebas nuevamente consideramos dos casos:

1. Cuando los deadlines son iguales a los períodos (Tabla 6.2).

2. Cuando los deadlines con menores o iguales a los períodos (Tabla 6.3).

Además, para cada caso consideramos las siguientes variaciones:

- La cantidad de *in-agent* participantes.

- El límite máximo del período.

De esta forma generamos 500 pruebas para cada uno de los escenarios que se muestran en las tablas 6.2 y 6.3.

Los resultados obtenidos se muestran en las gráficas de las figuras 6.5 a la 6.24 inclusive.

Tabla 6.2: Parámetros para Pruebas Simuladas con Deadlines = Períodos

N°	Rango del Período (mseg)	#in-agents*	#mks*	#niveles por mks*	#Imp.*	Saturación*
1	1000-20000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)	[80 % – 200 %]
2	1000-40000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)	[80 % – 200 %]
3	1000-80000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)	[80 % – 200 %]
4	1000-160000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)	[80 % – 200 %]
5	1000-2560000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)	[80 % – 200 %]

*: selecciona un valor para cada prueba

Tabla 6.3: Parámetros para Pruebas Simuladas con Deadlines $\leq$ Períodos.

N°	Rango del Período (mseg)	#in-agents*	#mks*	#niveles por mks*	#Imp.*	Saturación*
6	1000-20000	$(3, 6, 9, 12)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$[80\,\% - 200\,\%]$
7	1000-40000	$(3, 6, 9, 12)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$[80\,\% - 200\,\%]$
8	1000-80000	$(3, 6, 9, 12)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$[80\,\% - 200\,\%]$
9	1000-160000	$(3, 6, 9, 12)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$[80\,\% - 200\,\%]$
10	1000-2560000	$(3, 6, 9, 12)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$(1, 2, 3, 4)$	$[80\,\% - 200\,\%]$

*: selecciona un valor para cada prueba

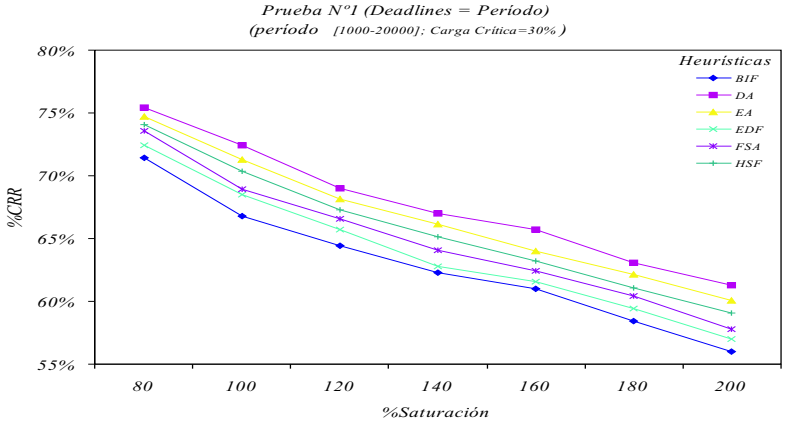


Figura 6.5: Prueba N° 1 - *Calidad Real Relativa.*

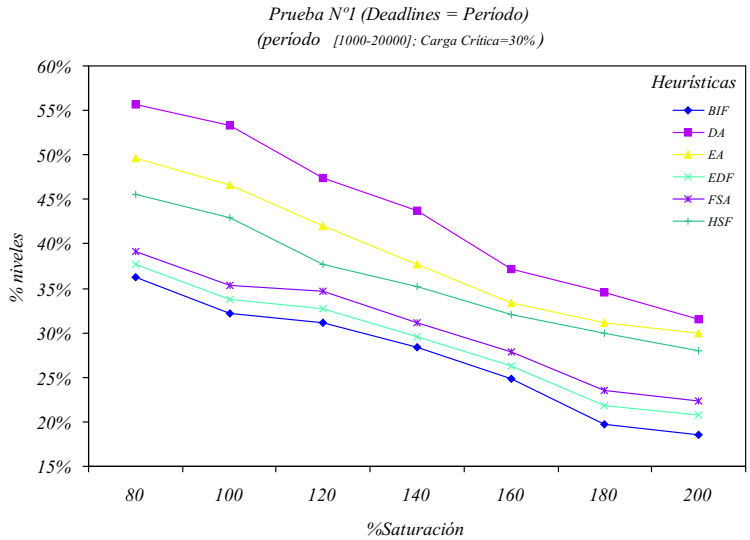


Figura 6.6: Prueba N° 1 - *%Niveles Ejecutados.*

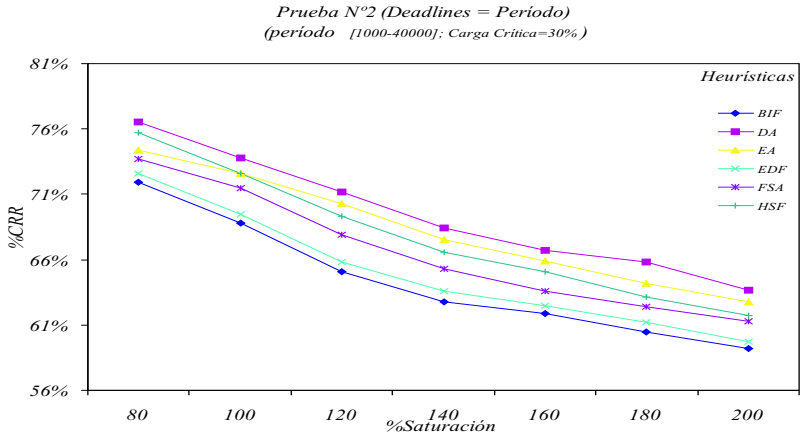


Figura 6.7: Prueba N°2 - *Calidad Real Relativa.*

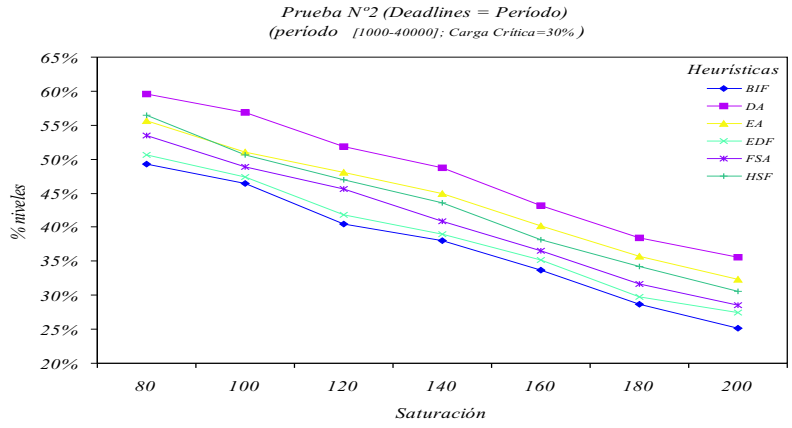


Figura 6.8: Prueba N°2 - *%Niveles Ejecutados.*

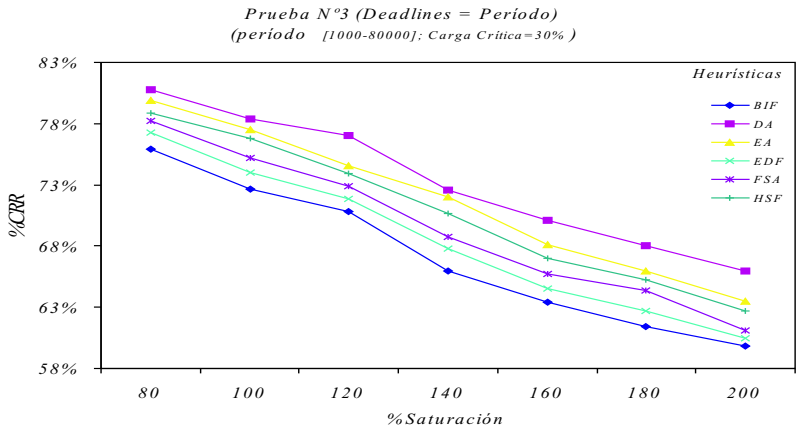


Figura 6.9: Prueba N°3 - *Calidad Real Relativa.*

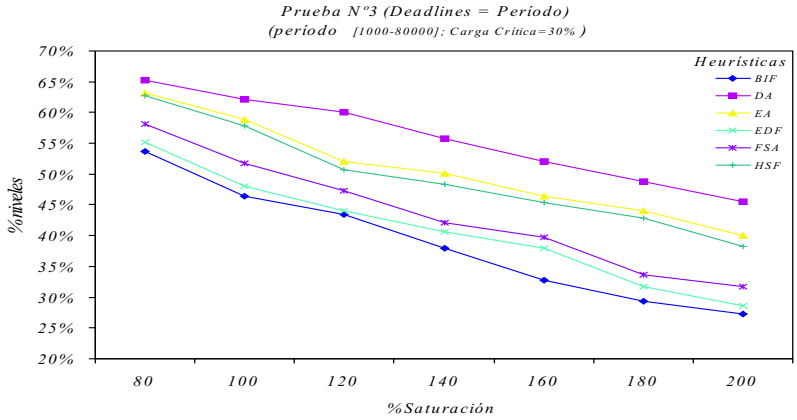


Figura 6.10: Prueba N°3 - *%Niveles Ejecutados.*

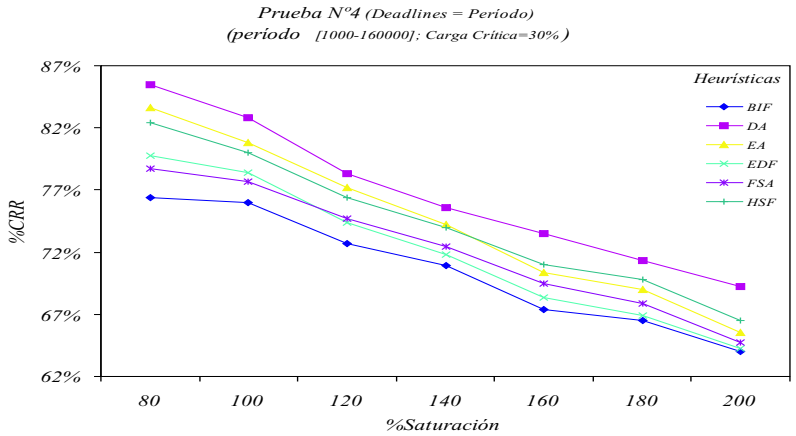


Figura 6.11: Prueba N°4 *Calidad Real Relativa* (*deadlines* = *períodos*$\in$ [1000 − 160000]).

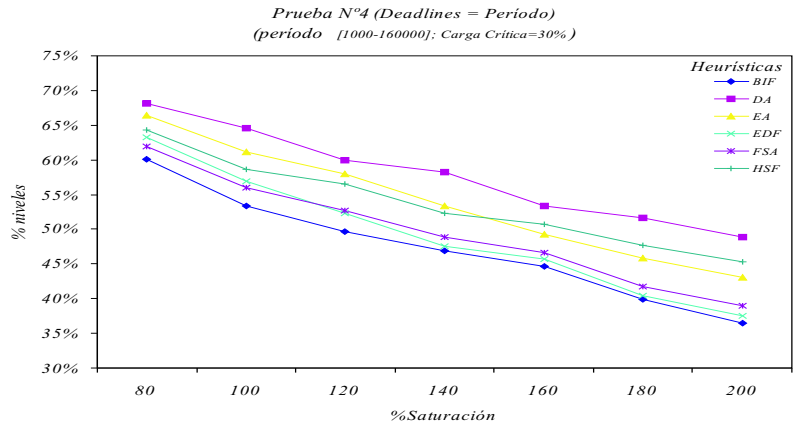


Figura 6.12: Prueba N°4 - *%Niveles Ejecutados.*

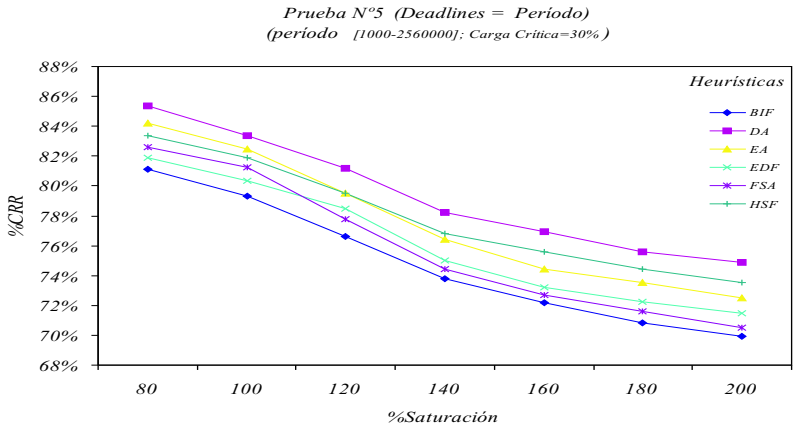


Figura 6.13: Prueba N°5 - *Calidad Real Relativa.*

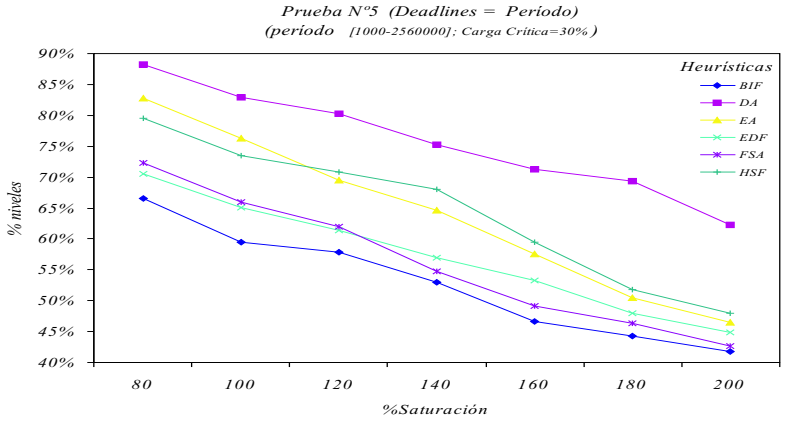


Figura 6.14: Prueba N°5 - *%Niveles Ejecutados.*

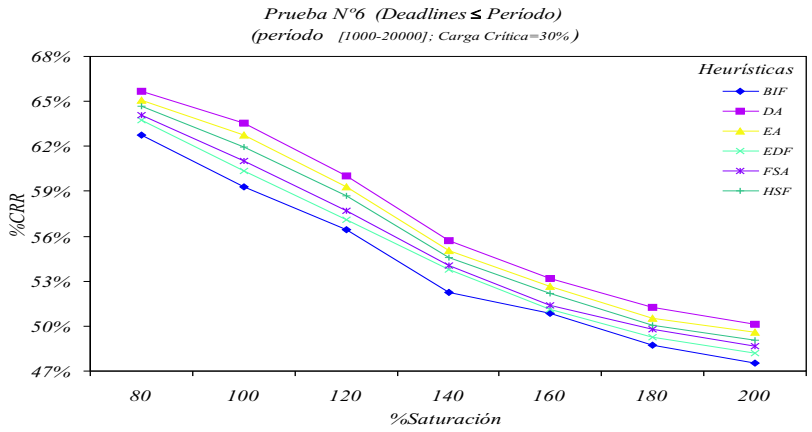


Figura 6.15: Prueba N°6 - *Calidad Real Relativa.*

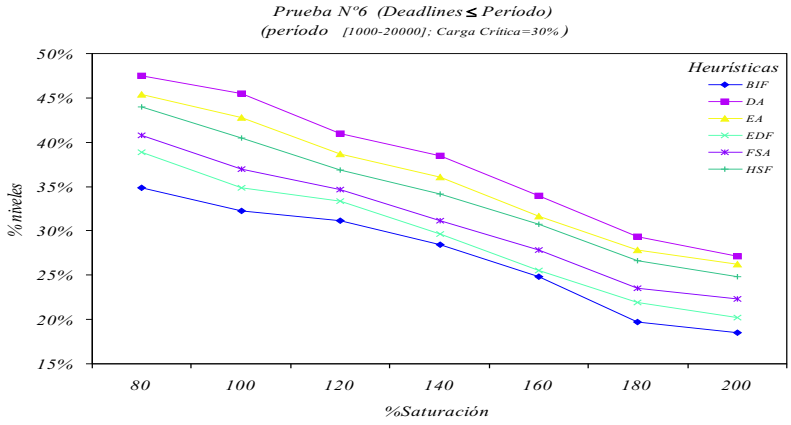


Figura 6.16: Prueba N°6 - *%Niveles Ejecutados.*

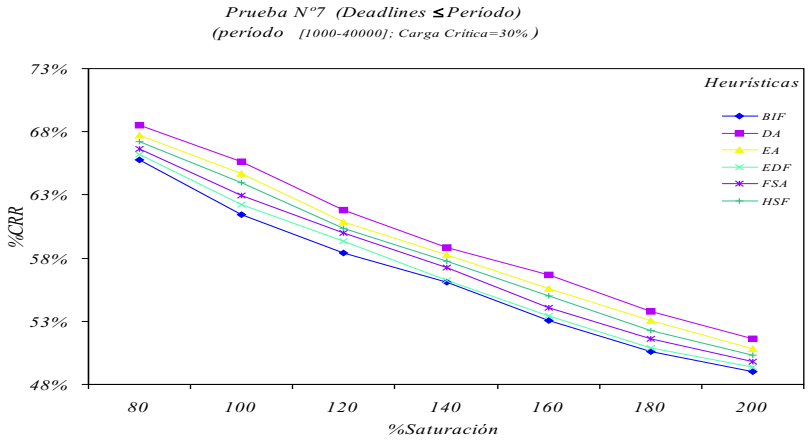


Figura 6.17: Prueba N°7 - *Calidad Real Relativa.*

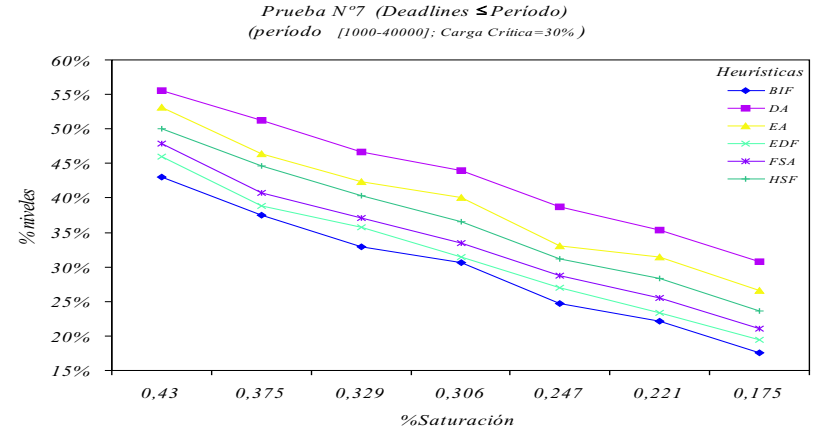


Figura 6.18: Prueba N°7 - *%Niveles Ejecutados.*

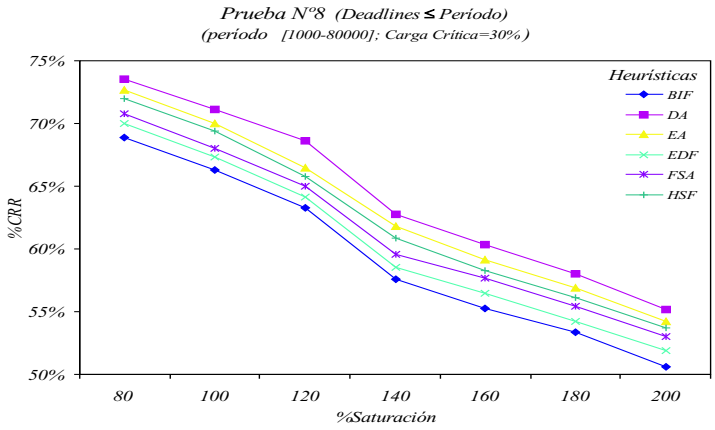


Figura 6.19: Prueba N°8 - *Calidad Real Relativa.*

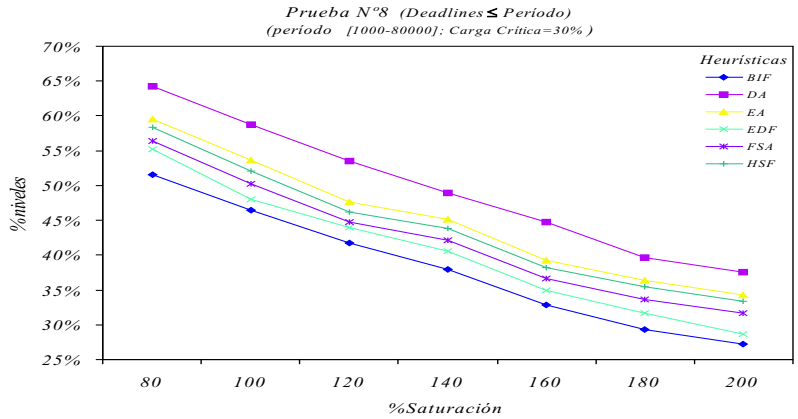


Figura 6.20: Prueba N°8 - *%Niveles Ejecutados.*

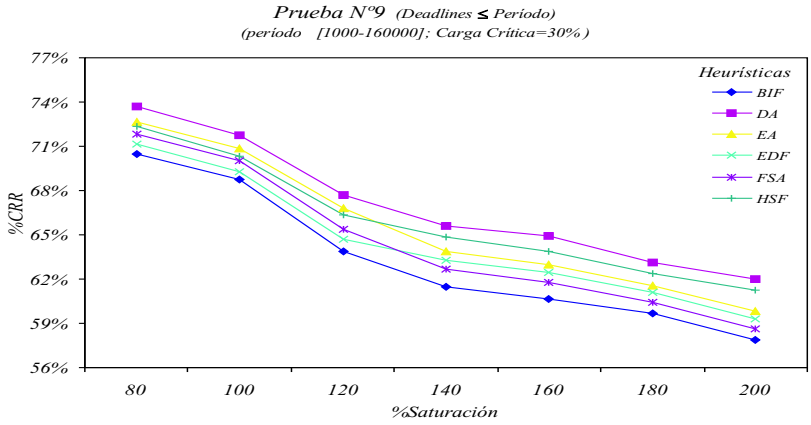


Figura 6.21: Prueba N°9 - *Calidad Real Relativa.*

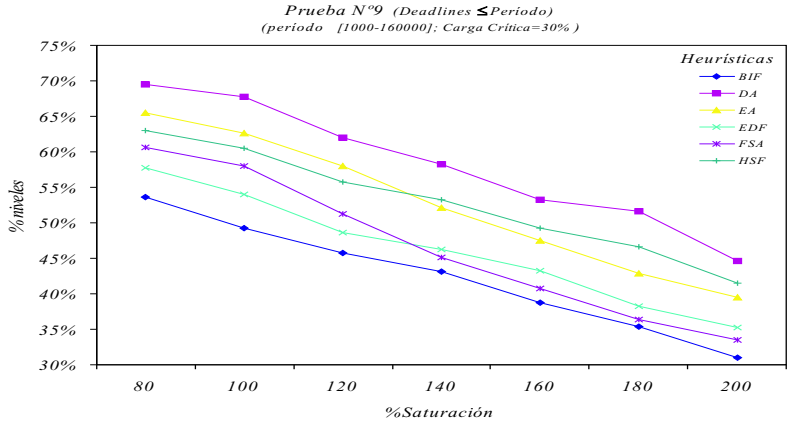


Figura 6.22: Prueba N°9 - *%Niveles Ejecutados.*

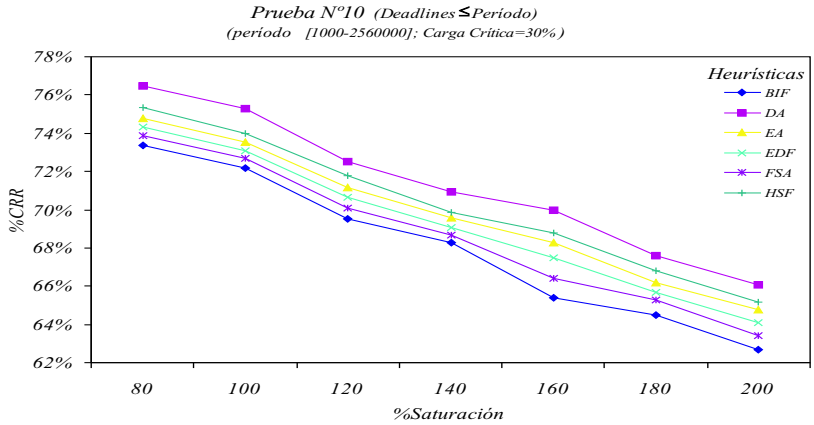


Figura 6.23: Prueba N°10 - *Calidad Real Relativa.*

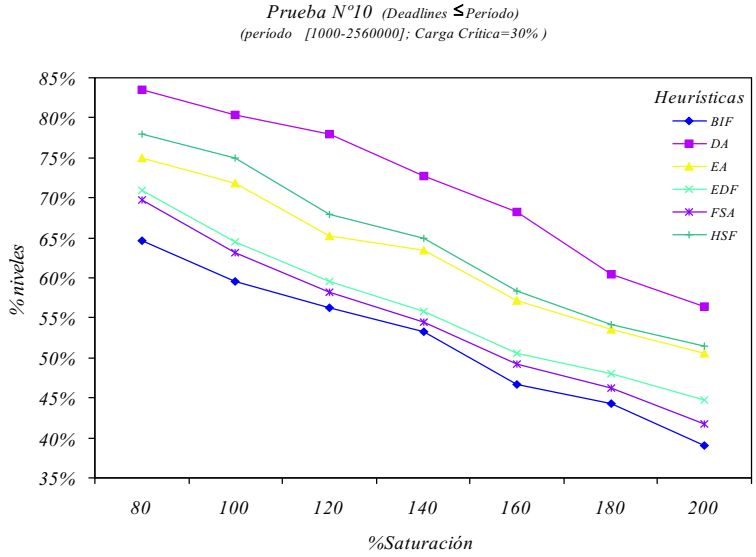


Figura 6.24: Prueba N°10 - *%Niveles Ejecutados.*

En las gráficas se puede apreciar que:

- Nuestras subastas obtienen calidades iguales y superiores que las heurísticas ya implementadas.
- En todos los escenarios la *Subasta Holandesa* (*DA*) obtiene las mejores calidades, al igual que es la que mayor cantidad de niveles del agente *ARTIS* ejecuta.
- Se puede observar que, en general, todas las heurísticas obtienen mejores resultados (en calidad obtenida y porcentaje de niveles ejecutados) cuando los deadlines son iguales al período (Escenarios 1 al 5 – Figuras 6.5 al 6.14) que cuando son menores.
- La *Subasta Inglesa* (*EA*) es la segunda heurística que ejecuta más niveles, solo la supera la heurística *High Slope First* (*HSF*) cuando los períodos son mayores que *160000(mseg)* (Figuras 6.12, 6.14, 6.22 y 6.24). Así mismo, calidades obtenidas por la *EA* son mejores cuando los períodos son pequeños (menores de *80000(mseg)*, Figuras 6.5, 6.7, 6.9, 6.15, 6.17 y 6.19) siendo superada después por la heurística *HSF*.

Dados los óptimos resultados obtenidos en las simulaciones, decidimos implementar estas las baterías de pruebas en una plataforma real.

6.4. Pruebas Experimentales

Para las pruebas experimentales utilizamos las mismas características de las baterías de pruebas utilizadas en las simulaciones. Así, en las tablas 6.4 y 6.5 muestran el resumen de los distintos escenarios a ser ejecutados. Aquí también consideramos dos tipos de pruebas:

- Baterías de pruebas con *in-agents* que tienen los deadlines iguales a sus períodos.

- Baterías de pruebas con *in-agents* que tienen los deadlines menores o iguales a sus períodos.

De esta forma las gráficas de las figuras 6.25, 6.26, 6.27 y 6.28 resumen los resultados obtenidos.

Tabla 6.4: Parámetros Pruebas Experimentales N°1 con Deadlines = Períodos

Rango del Período (mseg)	#in-agents*	#mks*	#niveles por mks*	#Imp.*
1000-20000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-40000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-80000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-160000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-2560000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)

*: selecciona un valor para cada prueba

Tabla 6.5: Parámetros Pruebas Experimentales N°2 con Deadlines $\leq$ Períodos

Rango del Período (mseg)	#in-agents*	#mks*	#niveles por mks*	#Imp.*
1000-20000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-40000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-80000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-160000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)
1000-2560000	(3, 6, 9, 12)	(1, 2, 3, 4)	(1, 2, 3, 4)	(1, 2, 3, 4)

*: selecciona un valor para cada prueba

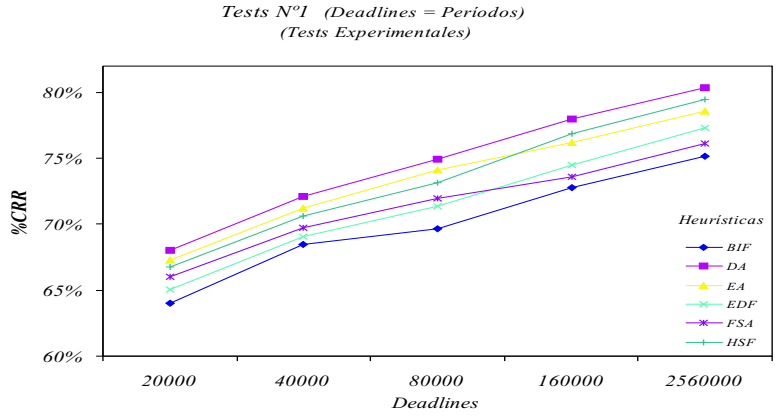


Figura 6.25: Pruebas Experimentales N°1 - *Calidad Real Relativa.*

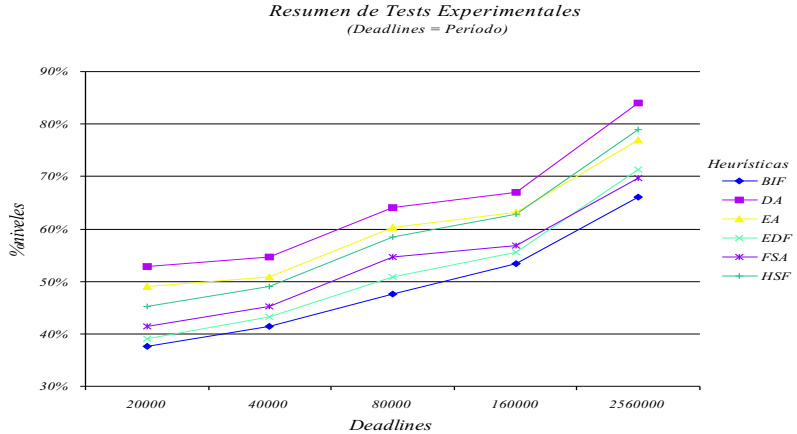


Figura 6.26: Pruebas Experimentales N°1 - *%Niveles Ejecutados.*

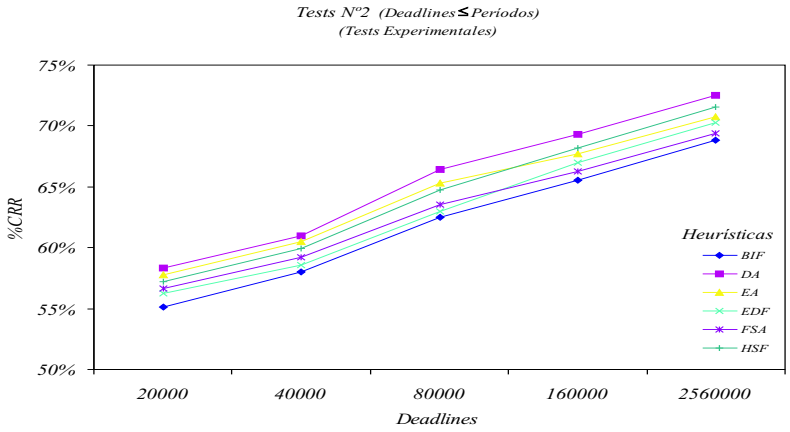


Figura 6.27: Pruebas Experimentales N°2 - *Calidad Real Relativa.*

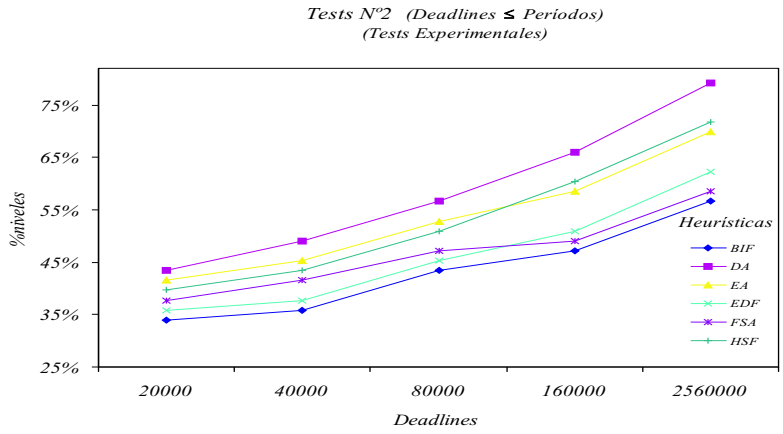


Figura 6.28: Pruebas Experimentales N°2 - *%Niveles Ejecutados.*

Nuevamente observamos la *Subasta Holandesa* (*DA*) obtiene los mejores resultados en todos los escenarios. Sin embargo, de las demás subastas, sólo la *Subasta Inglesa* (*EA*) alcanza resultados prometedores.

Podemos ver que para deadlines menores que los períodos, y siendo los períodos menores de *80000 (mseg)*, nuevamente la subasta *EA* ejecuta una cantidad de niveles mayores que la heurística *High Slope First* (*HSF*), y por lo tanto sus calidades son mayores también.

Podemos decir entonces que los comportamientos obtenidos en las simulaciones se acercan a los obtenidos en el sistema real.

6.5. Resumen de las Pruebas

Para facilitar la lectura y interpretación de las gráficas anteriores, presentamos unas gráficas con el resumen de los resultados obtenidos en las distintas pruebas.

Estas gráficas-resumen están ordenadas por el tipo de deadline escogido para la prueba:

- Deadlines iguales a los períodos (Figuras 6.29 a la 6.32).

- Deadlines menores o iguales a los períodos (Figuras 6.33 a la 6.36).

Con estas gráficas se pretende mostrar una visión general los resultados obtenidos al utilizar las distintas heurísticas en cada caso en comparación con nuestras subastas.

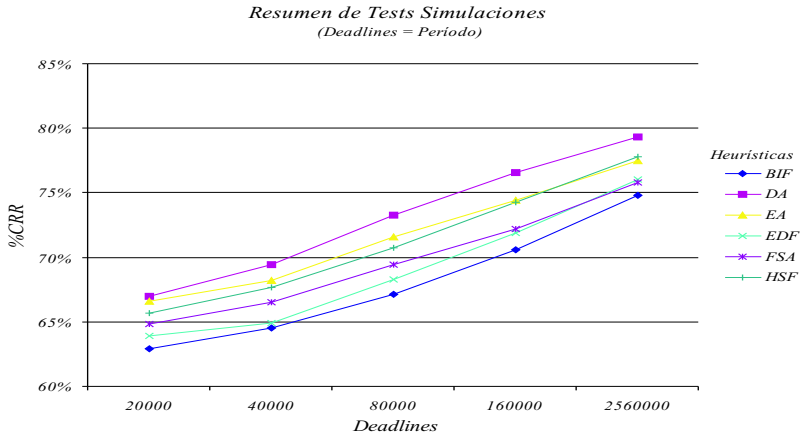


Figura 6.29: Resumen Pruebas Simuladas *Calidad Real Relativa*, con *deadlines = períodos*.

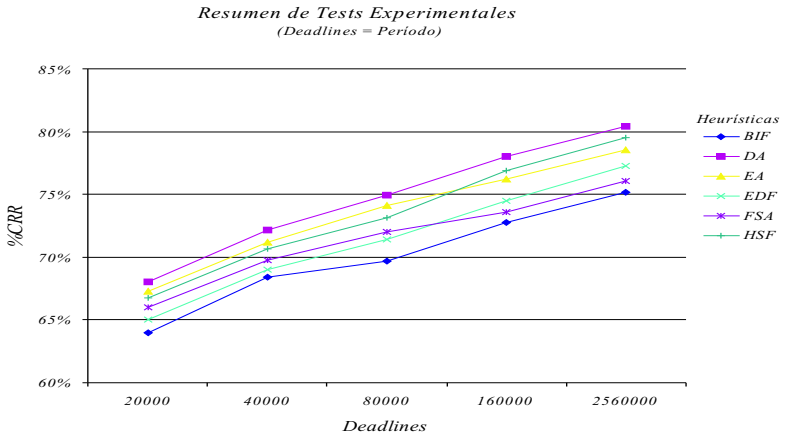


Figura 6.30: Resumen Pruebas Experimentales *Calidad Real Relativa*, con *deadlines = períodos*.

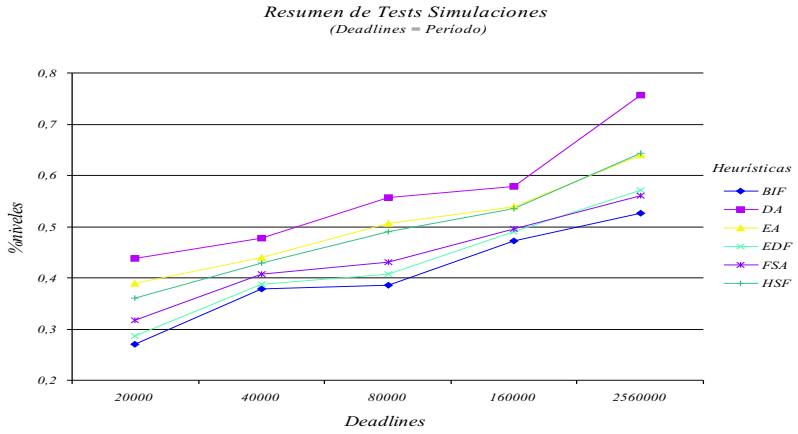


Figura 6.31: Resumen Pruebas Simuladas *%Niveles Ejecutados*, con *deadlines = períodos.*

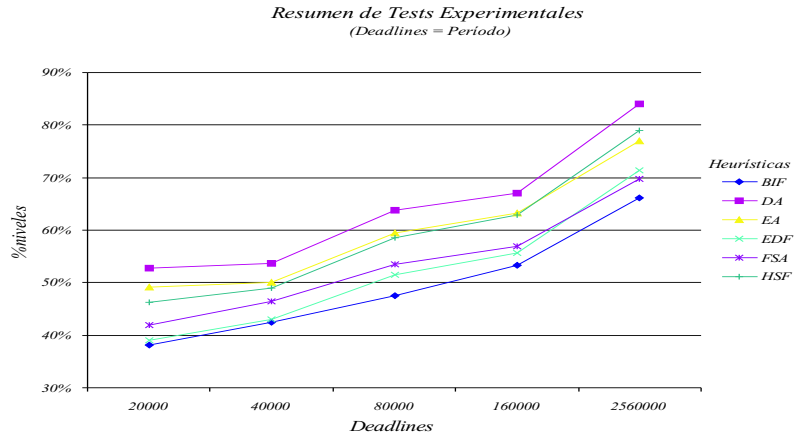


Figura 6.32: Resumen Pruebas Experimentales *%Niveles Ejecutados*, con *deadlines = períodos.*

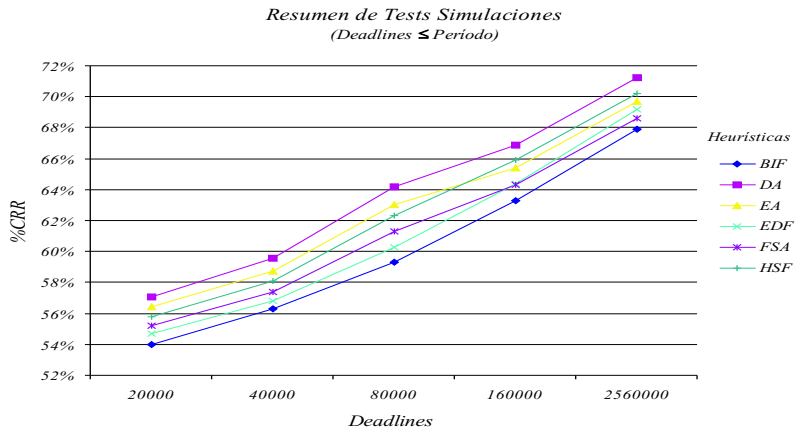


Figura 6.33: Resumen Pruebas Simuladas *Calidad Real Relativa*, con *deadlines* $\leq$ *períodos*.

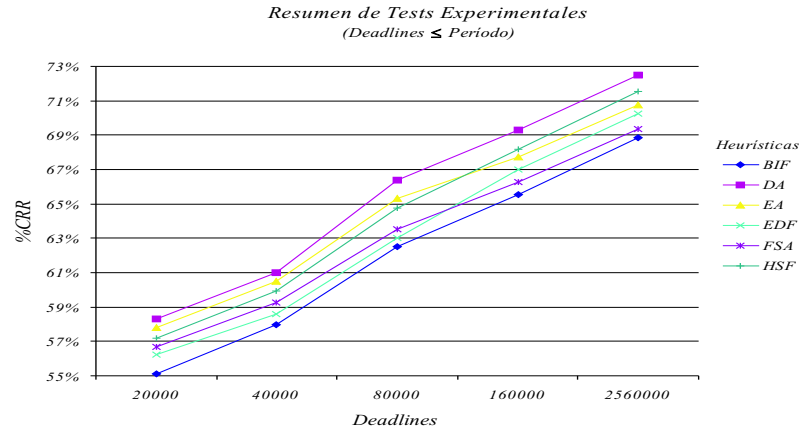


Figura 6.34: Resumen Pruebas Experimentales *Calidad Real Relativa*, con *deadlines* $\leq$ *períodos*.

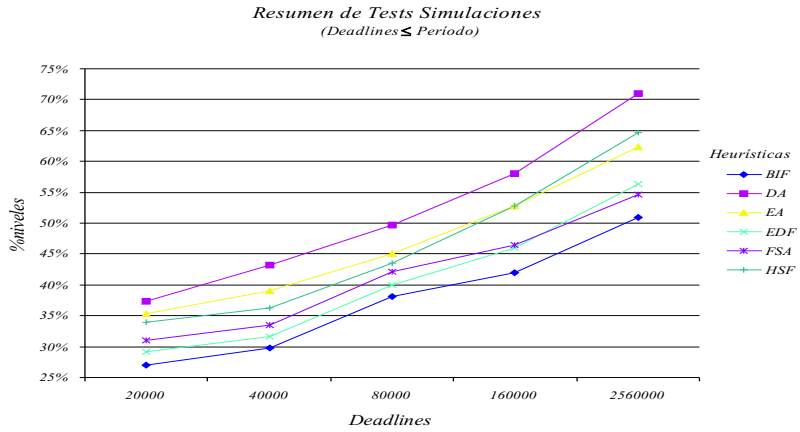


Figura 6.35: Resumen de Pruebas Simuladas *%Niveles Ejecutados*, con *deadlines* $\leq$ *períodos.*

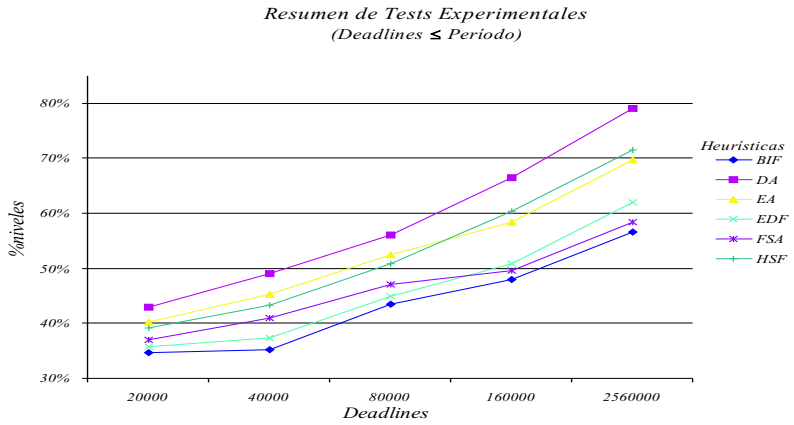


Figura 6.36: Resumen de Pruebas Experimentales *%Niveles Ejecutados*, con *deadlines* $\leq$ *períodos.*

Finalmente algunas conclusiones que se pueden obtener de estas pruebas

(simulada y experimentales) son:

- Se obtienen mejores resultados, tanto en la calidad obtenida como en el número de niveles ejecutados, cuando los deadlines de los *in-agents* del agente $\mathcal{ARTIS}(\mathcal{AA})$ son iguales a sus períodos de activación (Figuras 6.29 a la 6.32).

- Tanto en las pruebas simuladas como en las experimentales, la *Subasta Holandesa* (*DA*) obtiene los mejores resultados en calidad y en porcentaje de niveles ejecutados.

- Cuando los deadlines son menores de *80000* (*mseg*), la Subasta Inglesa obtienen los segundos mejores resultados (en calidad obtenida y porcentaje de niveles ejecutados). No obstante, cuando los deadlines son superiores a esta cantidad, la heurística *High Slope First* (*HSF*) obtiene pasa a la segunda posición.

- Cuando los deadlines son menores a sus períodos, la calidad obtenida disminuye hasta en un *10 %* con respecto a las calidades obtenidas cuando los deadlines son iguales a los períodos. El mismo comportamiento se puede observar con el procentaje de niveles ejecutados.

- Estos resultados demuestran la utilidad de usar las subastas como métodos de planificación de ejecución entre los *in-agents* bajo determinadas circunstancias. Con lo cual, el diseñador de un agente podrá decir que heurística utilizar dependiendo de las características del problema que debe resolver el $\mathcal{AA}$, y las restricciones temporales de sus *in-agents*.

- Así mismo, para las siguientes versiones de $\mathcal{ARTIS}$ sería conveniente utilizar estos resultados, ya que la planificación de la ejecución de los *in-agents* del $\mathcal{AA}$ podría ser mas deliverativa y así, su *Servidor Deliberativo* pueda decidir en tiempo de ejecución que heurística le conviene ejecutar dependiendo de las características de su entorno en cada instante.

Capítulo 7

Conclusiones

Los procesos de negociación han sido y serán un método efectivo para resolver algunos problemas y conflictos en las sociedades humanas, en especial cuando los conflictos entre las partes involucradas implican la utilización de recursos o habilidades que una de ellas no posee.

Sin embargo, para aplicar estos métodos se han establecido ciertas reglas básicas por las cuales se deben regir los participantes. No obstante, estas reglas se pueden ampliar dependiendo del tipo de negociación, entorno donde se produzcan, tipo de participantes, etc.

Al igual que en nuestras sociedades, los investigadores hemos aprovechado estos estudios y los hemos extendido a los sistemas de agentes. En este ámbito, los agentes involucrados en las negociaciones deben contar con las suficientes herramientas como para sostenerla el tiempo suficiente como para llegar a un desenlace satisfactorio para ambas partes.

Así se han establecido una serie de normas, protocolos y estrategias que

regulan estas interacciones. Nuevamente se presenta el patrón de poder ampliar o extender estas normas, protocolos, etc. dependiendo del entorno, tipo de negociaciones escogidas, tipo de participantes, etc.

Siguiendo esta línea, el objetivo principal de esta tesis doctoral era el presentar los métodos de negociación como una herramienta a utilizar por el agente $\mathcal{ARTIS}$ para que pueda obtener las mejores respuestas a sus problemas.

Para lograr esto se llevó a cabo una exhaustiva investigación sobre el estado del arte del los actuales modelos de negociaciones que se aplican en los sistemas multi-agentes, así como también de los modelos que se aplican en las sociedades.

También se estudiaron las mejores formas de aplicar esta negociación en los agentes $\mathcal{ARTIS}$, teniendo presente sus restricciones temporales. Esto nos llevó a la conclusión que el mejor método de negociación aplicable al agente $\mathcal{ARTIS}$ serían las subastas, ya que:

1. Las subastas son métodos de negociación de resolución rápida y eficaz. Condición necesaria dada la naturaleza de los agentes $\mathcal{ARTIS}$.

2. El fin de las subastas es que, tanto comprador como vendedor, tengan una utilidad máxima.
   Como las entidades involucradas en las negociaciones son parte del agente $\mathcal{ARTIS}$, este motivo fue fundamental ya que ambas partes (comprador y vendedor) trabajan en conjunto para obtener la mejor solución al problema del agente. Luego, el que ambos alcancen utilidades máximas en sus respectivas transacciones implica, inevitablemente, la máxima utilidad para el agente al que pertenecen.

3. En toda negociación es necesaria una moneda común tal que las pujas realizadas por los participantes puedan ser valoradas rápidamente.
   En nuestro caso la moneda común era la calidad que ofrecen los in-agent del agente $\mathcal{ARTIS}$ al *Servidor de Segundo Nivel* (*SLS*) por ejecutar sus respectivas respuestas (partes opcionales) de esta forma las valoraciones

de ambas partes se basan en esta moneda, calculando rápidamente la utilidad que les significa la oferta propuesta.

4. Tanto los *in-agents* como el *SLS* del agente $\mathcal{ARTIS}$ tienen a su disposición información sobre las posibles estrategias a utilizar por sus oponentes para valorar sus pujas, lo cual proporciona un medio de negociación con información completa. Es decir, las funciones de ganancias de los participantes son de dominio público y vienen parametrizadas por las máximas valoraciones que estén dispuestos a dar los participantes. Sin embargo, la exactitud de este último factor (máxima valoración) es conocido solo por el participante en sí, con lo cual el vendedor (*SLS*) desconoce la valoración exacta que hará el participante por su participación en el sistema.
   No obstante lo anterior, como los *in-agents* trabajan están en un entorno cooperativo, sus objetivos son compatibles con los objetivos del sistema al que pertenecen. Lo que implica que sea cual fuese el resultado de estas subastas, siempre se obtendrá una mejora en la respuesta. La cuestión principal para el *Servidor Deliberativo* y en particular para el *Servidor de Segundo Nivel* (*SLS*) como parte vendedora, es elevar esta mejora a su límite máximo utilizando las subastas como medio de planificación de sus actividades.

Para integrar estos métodos, se tuvo que modificar parte de la estructura interna del agente $\mathcal{ARTIS}$ considerando los siguientes puntos:

1. Detectar donde se producen las interacciones que dan mejora a las soluciones encontradas por las partes críticas del agente $\mathcal{ARTIS}$. Es decir *dónde* y *cómo* funcionan las partes opcionales del agente.

2. Insertar, en forma de heurísticas, las subastas en el agente $\mathcal{ARTIS}$. Para esto se modificó parte del razonamiento deliberativo del agente insertando el proceso de subastas, como parte integral del mismo para solucionar un problema específico: *¿qué partes opcionales ejecutar en el tiempo de*

slack disponible?.
Esto se logró modificando el *Servidor Deliberativo* del agente $\mathcal{ARTIS}$ para incluir los protocolos de las subastas. El resultado de las mismas sería la distribución eficiente del hueco de slack disponible en el sistema para la ejecución de las partes opcionales de los in-agents activos del agente.

3. Compatibilizar la inserción de las subastas con el correcto funcionamiento del agente $\mathcal{ARTIS}$. Para esto se tuvo que adaptar los protocolos de las subastas, tales que sus resultados se puedan aprovechar de la mejor forma.
   La adaptación fue orientada a mejorar la ejecutabilidad de las partes opcionales de los in-agents involucrados en las subastas.

4. El canal de comunicación entre *in-agent* y el *SLS* fuera expedito, lo cual era proporcionado por los protocolos internos de comunicación del agente $\mathcal{ARTIS}$.

Una vez que insertadas con éxito las subastas en el agente $\mathcal{ARTIS}$, debíamos compararlas con las demás políticas implementadas actualmente. Para esto se sometieron a pruebas masivas nuestras subastas junto con las políticas actuales.

Las primeras pruebas fueron hechas en el simulador que ofrece la herramienta de diseño de agente $\mathcal{ARTIS}$, InSiDe. Se sometieron a mas de *8000* pruebas cada una de las heurísticas, incluidas nuestras subastas.

Los resultados obtenidos con estas primeras pruebas fueron alentadores, ya que mostraban la eficiencia de nuestras subastas por sobre las políticas actuales. Llegando a tener nuestras subastas un aumento de la calidad del *10 %* con respecto a las demás políticas.

En vista de estos resultados, procedimos a implementar físicamente nuestras subastas en un agente $\mathcal{ARTIS}$ real. Sin embargo, para darle validez a esta implementación sometimos al agente nuevamente a pruebas masivas. Esta vez sobre la arquitectura real de $\mathcal{ARTIS}$.

Los resultados obtenidos en las pruebas reales mantienen el patrón obtenido en las pruebas simuladas. Mostrando un leve aumento en las calidades finales, esto debido a que el sistema simulado fue sobrecargado para reflejar fehacientemente al sistema real final.

7.1. Trabajos Futuros

Como trabajos futuros a esta investigación se propone seguir con la línea de investigación de aplicar las subastas a los agentes que trabajan en entornos de tiempo real, específicamente en los siguientes puntos:

- Ampliar las técnicas de negociaciones entre las entidades del agente $\mathcal{ARTIS}$ hacia unas que utilicen todo el tiempo de slack disponible durante la ejecución del agente. Es decir, que se pueda subastar o negociar el hueco de slack total al principio de la ejecución del agente, de esta forma se podría mejorar la eficacia del las planificaciones.

- Dados los resultados obtenidos de las pruebas simuladas y experimentales efectuadas en al agente $\mathcal{ARTIS}$, se propone orientar estas subastas hacia una reacción mas deliberativa del agente. Con esto se quiere decir que, con base en los resultados aquí expuestos, se puede orientar el razonamiento del agente a que utilice uno u otro tipo de heurística, dependiendo de las características de su entorno en ese instante. Por ejemplo, para deadlines mayores de *80000 (mseg)* es mejor utilizar la política de planificación *High Slope First* (*HSF*), sin embargo, para deadlines menores se recomienda la utilización de los métodos de subastas presentados aquí.

- Compatibilizar los métodos de negociación presentados aquí, con protocolos de subastas entre agentes $\mathcal{ARTIS}$, resolviendo aquellas dificultades que se presenten al utilizar uno u otro tipo de negociación.

- Aplicar estos métodos de negociaciones entre agentes $\mathcal{ARTIS}$ ejecutándose en la plataforma SIMBA. Para ello se deberán establecer los

protocolos necesarios para estas interacciones. Además se extienden este trabajo a observar los distintos comportamientos del agente en las negociaciones, utilizando como heurísticas internas nuestras subastas versus las demás políticas implementadas.

- Finalmente, se propone la ampliación de la herramienta gráfica InSiDe, tal que en ella se pueda ingresar todos los datos necesarios para llevar a cabo las subastas entre los in-agents, así como también las subastas entre agentes. Esta información sería por ejemplo: precios reservados, precio inicial, función de valoración para las pujas, lenguaje a utilizar en las negociaciones, etc.
  Con esta ampliación se facilitaría el ingreso de estos datos al diseñador del agente $\mathcal{ARTIS}$.

7.2. Trabajos

Reporte Técnico:

1. Patricia Maldonado; *Comunicación y Coordinación en Sistemas Multi-Agentes: Aplicación sobre una Arquitectura de Tiempo Real*; Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia; N° DSIC-II/03/03; Feb – 2003.

Artículos Publicados:

1. Patricia Maldonado, Vicente Botti; *Negociación en el control de sistemas Inteligentes de Tiempo Real*; IX Conferencia de la Asociación Española para la Inteligencia Artificial, CAEPIA'2001; Gijón – España; 2001.

2. Patricia Maldonado, Pedro Alberti, Vicente Botti; *Negociación en Sistemas Inteligentes de Tiempo-Real*; XI Jornadas Chilenas de Computación; Chillán – Chile; 2003.

3. Patricia Maldonado, Carlos Carrascosa and Vicente Botti; *Negotiation in Real-Time Multi-Agent Systems*; IADIS International Conference – Applied Computing 2005; Algarve – Portugal; 2004.

Artículos Enviados:

1. Patricia Maldonado, Carlos Carrascosa and Vicente Botti; *Auction for Real-Time Agent Scheduling*; Innovative Applications of Artificial Intelligence (IAAI'05).

2. Patricia Maldonado, Carlos Carrascosa and Vicente Botti; *Applied Auctions in Hard Real-Time Systems*; 17th Euromicro Conference on Real-Time Systems (ECRTS'05).

3. Patricia Maldonado, Carlos Carrascosa and Vicente Botti; *Using Negotiation Techniques as Real-Time Scheduling Policies on Multi-Agent Systems*; Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05).

Bibliografía

Technical report.

Enciclonet. http://www.enciclonet.com, 2005. Proyecto subvencionado por el Ministerio de Ciencia y Tecnología.

Martin R. Andersson and Tuomas W. Sandholm. Contract types for satisficing task allocation: Ii experimental results. In *AAAI Spring Symposium Series: Satisficing Models*, pages 1–7, Stanford University, CA, Mar 1998.

N.C. Audsley, A. Burns, R.I. Davis, K.W. Tindell, and A.J. Wellings. Fixed priority pre-emptive scheduling: An historical perspective. *Real-Time Systems*, (8):173–198, 1995.

N.C. Audsley, A. Burns, M.F. Richardson, and A.J. Welling. Hard Real-Time Scheduling: The Deadline Monotonic Approach. In *VIII IEEE Workshop on Real-Time Operating Systems and Software*, Atlanta, Ga, USA, May 1991.

Robert Axelrod. *The Complexity of Cooperation: Agent-Based Models of Competition and Collaboration*. Princeton University, 1997.

Federico Barber, Vicente Botti, Eva Onaindía, and Alfons Crespo. Temporal reasonig in reakt: An environment for real-time knowledge-based systems. In *AICOMM*, number 7, pages 175–202, 2003.

Mihai Barbuceanu. Negotiation as interactive exchange of constraints about

agent behavior. In *The International Workshop on MultiAgent Systems*, Oct 1999.

Mihai Barbuceanu and Mark Fox. The design of a coordination language for multi-agent systems. In M.J.Wooldridge J.P.Muller and N.R.Jennings, editors, *Intelligent Agents III: Agent Theories, Architectures, and Languages*, pages 341–355. Springer Verlag Lecture Notes inArtificial Intelligence, 1996.

Mihai Barbuceanu and Mark S. Fox. Capturing and modelling coordination knowledge in multi-agent systems. *International Journal on Cooperative Information Systems*, 5:275–314, 1966.

David P. Barnes, Robert A. Ghanea-Hercock, Ruth Aylett, and Alex M. Coddington. Many hands make light work? an investigation into behaviourally controlled co-operant autonomous mobile robots. In *International Conference on Autonomous Agents*, pages 413–420, 1997.

Mark S. Boddy. Anytime problem solving using dynamic programming. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 738–743. AAAI Press/MIT Press, 1991.

V. Botti, C. Carrascosa, V. Julian, and J. Soler. Modelling agents in hard real-time environments. In *MAAMAW'99 Proceedings*, volume 1647 of *LNAI*, pages 63–76. Springer-Verlag, 1999.

Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3–15, jun 1990.

S. Bura, F. Guerin-Pace, H. Mathian, D. Pumain, and L. Sanders. Multi-agent systems and the dynamics of a settlement system. In *Simulating Societies Symposium, Siena*. C.Castelfranchi, 1993.

Antonio Cabrales. Nash and game theory. It will appear, in Greek translation, in a festscrift in Nash's honor edited by Demetrios Christodoulou, 2001. This article discusses the contributions of John Nash to Game Theory.

Luis M. Camarinha-Matos and Hamideh Afsarmanesh. Virtual enterprise modeling and support infraestructures: Applying multi-agent system approaches. In O.Stepankova-R.Trappl M.Luck, V.Marik, editor, *Multi-Agent Systems and Aplications*, number 2086 in LNAI, pages 335–364. Springer, EC-CAI, ACAI'2001, EASSS'2001, 2001.

C. Cambier. SIMDELTA: un système multi-agent pour simuler la pêche sus le delta central du níger. Thèse de l'université París, 1994.

Antonio Campos-López. *Una Arquitectura para Sistemas Expertos de Tiempo Real Basada en Técnicas de Planificación Dinámica.* PhD thesis, Departamento de Informática – Universidad de Oviedo – España, 2004.

Carlos Carrascosa, Vicente Julián, Ana García-Fornes, and Agustin Espinosa. Un lenguaje para el desarrollo y prototipado rápido de sistemas de tiempo real inteligentes. In *CAEPIA*, 1997.

Carlos Carrascosa, Miguel Rebollo, Vicente Julián, and Vicente Botti. Deliberative server for real-time agents. In Vladimir Marik, Jörg Müller, and Michal Pechoucek, editors, *The 3rd International/Central and Eastern European Conference on Multi-Agent Systems, CEEMAS 2003*, volume 2691 of *LNAI*. Multi-Agent Systems and Applications III, Springer-Verlag, 2003a.

Carlos Carrascosa, Miguel Rebollo, José Soler, Vicente Julián, and Vicente Botti. SIMBA architecture for social real-time domains. In *EUMASS 2003: The First European Workshop on Multi-Agent System*, 2003b.

Cristiano Castelfranchi, Rosaria Conte, and Mario Paolucci. Normative reputation and costs of compliance. *Artificial Societies and Social Simulation*, 1 (3), 1998.

Cristiano Castelfranchi, Fiorella de Rosis, and Floriana Grasso. Deception and suspect in medical interactions: Towards a simulation of believable dialogues.

In In Y. Wilks, editor, *International Series in Engineering and Computer Science*, volume 511. Machine Conversations, 1999.

Anthony Chavez and Pattie Maes. Kasbah: An agent marketplace for buying and selling goods. In *The first international conference on the practical Application of Intelligent Agents and Multi Agents Technology*, pages 75–90, Apr 1996.

Alfons Crespo, Vicente Botti, Federico Barber, D. Gallardo, and Eva Onaindía. A temporal balckboard for real-time process control. *Engineering Applications of Artificial Intelligence*, pages 225–256, 1994.

Edmund H. Durfee and Thomas A. Montgomery. MICE: A flexible testbed for intelligent coordination experiments. In *Proceedings of the 1989 Distributed Artificial*, pages 25–40. Intelligence Workshop, 1989.

E-Bay. Ebay - your personal trading community. http://www.ebay.com/aw/, 2001.

& J.Stark Engelbrecht-Wigganns, M.Shubik, editor. *Auctions, Bidding, and Market: An Historical Sketch*, 1983.

Peyman Faratin. *Automated Service Negotiation Between Autonomous Computational Agents.* PhD thesis, University of London, Queen Mary College, Department of Electronic Engineering, 2000.

Jacques Ferber. *Multi-Agent Systems, An introduction to distributed Artificial Intelligence.* Addison Wesley Longman Inc., 1999.

FIPA00029. Fipa contract net interaction protocol specification. Foundation for Intelligence Phisycal Agents, 2000. http://www.fipa.org/specs/fipa00029/, FIPA.

FIPA00031. Fipa english auction interaction protocol specification. Foundation for Intelligence Phisycal Agents, 2000. http://www.fipa.org/specs/fipa00031/, FIPA.

FIPA00032. Fipa dutch auction interaction protocol. Foundation for Intelligence Phisycal Agents, 2000. http://www.fipa.org/specs/fipa00032/, FIPA.

Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Tehories, Architectures, and Languages, Springer-Verlag*, 1996.

E. Giménez Funes, L. Godo, J.A. Rodríguez Aguilar, and P. Garcia Calvés. Designing bidding strategies for trading agents in electronic auctions. In *ICMAS98*, pages 136–143, Paris, 1998.

Ana García-Fornes. *ARTIS: Un modelo y una arquitectura para sistemas de tiempo real inteligentes.* PhD thesis, Universidad Politénica de Valencia, Valencia – España, 1996.

Ana García-Fornes, Andrés Terrasa, Vicente Botti, and Alfons Crespo. Analizing the shedulability of hard real-time artificial intelligent systems. In *Engineering Applications of Artificial Intelligence*, pages 369–377. Pergamon Press Ltd., 1997.

Weiss Gerard. *Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence.* The MIT Press: Cambridge, 1999.

Robert H. Guttman and Pattie Maes. Cooperative vs. competitive multi-agent negotiations in retail electronic commerce. In *Cooperative Information Agents*, pages 135–147, 1998.

Vu Ha and David J. Musliner. Toward decision-theoretic circa with application to real-time computer security control. In *Working notes of the AAAI 2002 Workshop on Real-Time Decision Support and Diagnostics Systems*, 2002.

Kwun Han and Manuela Veloso. Reactive visual control of multiple non-holonomic robotic agents. In *Proceedings of the International Conference on Robotics and Automation*, Leuven; Belgium, 1998.

Barbara Hayes-Roth. An architecture for adaptive intelligent systems. *Artificial Intelligence*, 72:329–365, 1995.

Luis Hernandez-López. *Heurísticas para el Control Deliberativo en una Arquitectura de Agentes Inteligentes de Tiempo Real*. PhD thesis, Universidad Politénica de Valencia, Valencia – España, 2003.

Bryan Horling, Victor Lesser, Regis Vincent, and Thomas Wagner. The soft real-time agent control architecture. In *Proceedings of the AAAI/KDD/UAI-2002 Joint Workshop on Real-Time Decision Support and Diagnosis Systems*, July 2002. http://mas.cs.umass.edu/paper/221.

Nick Jakobi. The minimal simulation approach to evolutionary robotics, 1998.

Nick R. Jennings, Peyman Faratin, Lomuscio A.R., Simon Parsons, Carles Sierra, and Michael Wooldridge. Automated negotiation: Prospects, methods and challenges. In *Pacific Rim International Conference on Artificial Intelligence*, 2000.

Vicente Julián and Vicente Botti. Agentes inteligentes: El siguiente paso en la inteligencia artificial. *Novática*, (145):95–99, May 2000.

Vicente Julián, Carlos Carrascosa, and Vicente Botti. Formalización y traducción a un modelo ejecutable de las entidades de un agente artis. In *CAEPIA-TTIA*, Murcia – España, 1999.

Vicente Julián, Mario González, Miguel Rebollo, Carlos Carrascosa, and Vicente Botti. Inside: Una herramienta para el desarrollo de agentes artis. In *Actas SEID 2000*, pages 79–88, Ourense, España, 2000.

Gal A. Kaminka and Milind Tambe. Robust agent teams via socially-attentive monitoring. *Artificial Intelligence Research*, 7:83–124, Dec 1999.

Reinhold Kloos, Rolf Reinema, and Michael Schroeder. An adaptive trading framework based on agents supporting a geographically distributed team.

In Krautwurmova H. Briot J. Marik V., Stepankova O., editor, *Multi-Agent Systems and Applications. ACAI-2001&EASSS-2001. Student Sessions*, pages 80–88, Prague-Czech Republic, Jul 2001.

Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2):79–97, 1997. Special Issue on Economic Principles of Multi-Agent Systems.

Sarit Kraus. Automated negotiation and decision making in multiagent environments. pages 150–172, 2001a.

Sarit Kraus. *Strategic Negotiation in Multiagent Environments*. MIT, 2001b.

Sarit Kraus and Daniel Lehmann. Designing and building a negotiating automated agent. *Computational Intelligence*, 11(1):132–171, 1995.

Martín Krause. *La teoría de los Juegos y el Origen de las Instituciones*. Number 31. Libretas Eseade, 2002.

Susan E. Lander and Victor R. Lesser. Customizing distributed search among agents with heterogeneous knowledge. In *Proceedings of the First International Conference on Information and Knowledge Management*, pages 335–344, Baltimore, Maryland, 1992.

Victor Lesser, editor. *Negotiation through argumentation - a preliminary report*, 1995. MIT.

Victor R. Lesser, Jasmina Pavlin, and Edmund Durfee. Approximate Processing in Real-Time Problem Solving. *Artificial Intelligence*, 9(1):49–61, 1988.

Alessio Lomuscio, Michael Wooldridge, and Nicholas R. Jennings. A classification scheme for negotiation in electronic commerce. In *Agent Mediated Electronic Commerce, The European AgentLink Perspective.*, pages 19–33. Springer-Verlag, 2001. ISBN 3-540-41671-4.

John-Jules Ch. Meyer. Agent languages and their relationship to other programming paradigms. In M. P. Singh J. P. Muller and A. S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 309–316, Berlin, 1998. Springer Verlag.

T.A. Moehlman, V.R. Lesser, and B.L. Buteau. Decentralized negotiation: An approach to the distributed planning problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 1992. Special Issue on Control, Planning and Scheduling.

David J. Musliner, Edmund H. Durfee, and Kang G. Shin. Circa: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1561–1574, 1993.

David John Musliner, James Hendler, Ashok Agrawala, Edmund Durfee, Jay Strosnider, and C.J. Paul. The challenges of real-time ai. *IEEE Computer*, 28(1), 1995.

H. Penny Nii. Blackboard systems: Blackboard application systems, blackboard systems from a knowledge engineering perspective. *Artificial Intelligence*, pages 82–106, 1986a.

H. Penny Nii. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *The AI Magazine*, pages 38–53, Summer 1986b.

Hyacinth Nwana, Lyndon Lee, and Michael Wooldridge. Coordination in software agent systems. *The British Telecom Technical Journal*, 14(4):79–88, 1996.

Hyacinth S. Nwana. Software Agents: An Overview. *Knowledge Engineering Review Journal*, 11(3):1–40, Nov 1996.

Simon Parsons, Carles Sierra, and Nick R. Jennings. Agents that reason and negotiate by arguing. *Logic and Computation*, 8(3):261–292, 1998.

H.Van-Dyke Parunak and James Odell. Engineering artifacts for multi-agent systems. ERIM CEC., 1999.

Luigi Portinale and Pietro Torasso. Diagnosis as a variable assignment problem: A case study in space robot fault diagnosis. In *16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 1087–1093, Aug 1999.

Howard Raiffa. *The Art and Science of Negotiation*. Harvard University, Cambridge, USA, 1982.

Anita Raja and Victor Lesser. Real-time meta-level control in multi-agent systems. In *Multi-Agent Systems and Applications - ACAI 2001 & EASSS 2001 Student Sessions*, Prague, 2001.

Anand Rao and Michael Georgeff. Bdi agents: From theory to practice. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, pages 312–319, 1995.

W. Scott Reilly and Joseph Bates. Natural negotiation for believable agents. Technical Report CS-95-164, 1995.

Jeffrey S. Rosenschein and Gilad Zlotkin. Designing conventions for automated negotiation. *Artificial Intelligence*, 15(3):29–46, 1994.

Stuart Russell and Peter Norving. *Inteligencia Artificial: Un Enfoque Moderno*. Prentice Hall Hispanoamericana, S.A., 1996.

Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 295–308, Hidden Valley, Pennsylvania, 1993.

Tuomas W. Sandholm and Victor Lesser. Issues in automated negotiation and electronic commerce: Extending the contract net framework. In *First International Conference on Multiagent Systems*, pages 328–335, San Fransisco, 1995. ICMAS-95.

Sandip Sen and Edmund H. Durfee. The role of commitment in cooperative negotiation. *International Journal on Intelligent and Cooperative Information Systems*, 3(1):67–81, 1994.

Carles Sierra, Peyman Faratin, and Nick R. Jennings. A service-oriented negotiation model between autonomous agents. In *MAAMAW'97*, pages 17–35, 1997.

Carles Sierra, Nick R. Jennings, Pablo Noriega, and Simon Parsons. Negociación Mediante Argumentación en Sistemas Multi-Agentes. *Revista Iberoamericana de Inteligencia Artificial*, (6):36–45, 1998.

Jose Soler, Vicente Julián, and Carlos Carrascosa. Applying the ARTIS Agent Architecture to Mobile Robot Control. In *Iberamia*, Atibaia, Brazil, 2000. No published yet.

Jose Soler-Bayona. *SIMBA, una Plataforma para el Desarrollo de Sistemas Multiagentes en entornos de Tiempo Real.* PhD thesis, Universidad Politénica de Valencia, Valencia – España, 2003.

John Stankovic. Distributed real-time computing: The next generation. *Society of Instrument and Control Engineers of Japan*, 1992.

John Stankovic and Krithi Ramamritham. What is predictability for real time systems. Technical report, Dept. of Computer and Information Science. Univ. of Massachusetts, 1993.

John A. Stankovic. Misconceptions about real-time computing. *IEEE Computer*, 12(10):10–19, 1988.

Peter Stone, Manuela Veloso, and Patrick Riley. The cmunited-98 champion simulator team. In Minoru Asada and Hiroaki Kintano, editors, *Robocup-98: Robot Soccer World Cup II*, Springer Verlag, Berlin, 1999.

Katia Sycara. *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods.* PhD thesis, Information and Computer Science–Georgia Institute of Technology, 1987.

Katia Sycara. Persuasive argumentation in negotiation. *Theory and Decision*, 28(3):203–242, May 1990.

Andrés Terrasa. *Flexible Real-Time Linux: A new environment for Flexible Hard Real-Time Systems.* PhD thesis, Departamento de Sistemas Informáticos y Computación – Universidad Politécnica de Valencia – España, 2000.

Andrés Terrasa, Ana García-Fornes, and Vicente Botti. Flexible real-time linux. *Real-Time Systems Journal*, (2):149–170, 2002.

Maksim Tsvetovatyy, Maria Gini, Bamshad Mobasher, and Zbigniew Wieckowski. Magma: an agent based virtual market for electronic commerce. In *Applied Articial Intelligence*, 1997.

Manuela Veloso and Peter Stone. Individual and collaborative behaviors in a team of homogennus robotic soccer agents. In *Proceedings of the Third International Conference on Multi-Agent Systems*, pages 309–316, Paris, 1998.

Barry Werger and Maja Mataric. Broadcast of local eligibility: Behavior-based control for strongly cooperative robot teams. In *Autonomous Agents*, pages 21–22, Barcelona-Spain, Jun 2000.

Michael Wooldridge and Nick R. Jennings. Intelligent agents: Theory and practice. In *The Knowledge Engineering Review*, pages 115–152, 1995.

Michael Wooldridge and Simon Parsons. Languages for Negotiation. In *Fourteenth European Conference on Artificial Intelligence (ECAI-2000)*, Berlin, Germany, 2000.

P. Wurman, W. Walsh, and M. Wellman. Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems*, 24:17–27, 1998.

Dajun Zeng and Katia Sycara. Bayesian learning in negotiation. In Sandip Sen, editor, *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, pages 99–104, Stanford University, CA, 1996.

Gilad Zlotkin and Jeffrey S. Rosenschein. A domain theory for task oriented negotiation. In *Proceeding of the American Association of Artificial Intelligence*, pages 148–153, San Jose, CA, 1992.

Gilad Zlotkin and Jeffrey S. Rosenschein. Mechanisms for automated negotiation in state oriented domains. *Artificial Intelligence Research*, (5):163–238, 1996.

Índice alfabético


UNIVERSIDAD DE MAGALLANES
SCIENTIAE IN AUSTRO
CHILE