La presente Tesis Doctoral ha sido financiada de forma conjunta por el Ministerio de Educación y Ciencia de España y por los fondos FEDER de la Comisión Europea bajo los proyectos TIN2006-15516-C04-04 y Consolider Ingenio-2010 CSD2006-00046, respectivamente.




UNIVERSITAT DE VALÈNCIA

**Departament d'Informàtica**

TESIS DOCTORAL

VERSIÓN EN CASTELLANO

# INTEGRACIÓN DE HABILIDADES SOCIALES EN LA ANIMACIÓN COMPORTAMENTAL DE ACTORES SINTÉTICOS

*Presentada por:*

***D. Francisco Grimaldo Moreno***

*València, mayo de 2008*

Trabajo dirigido por:

Dr. D. Miguel Lozano Ibáñez

*A la meua família*

*...sols En nós dos, aMor se manIfesta,*
*e nós vivents, no Li fallIrà casA...*
*Ausiàs March*

# Agradecimientos

Al llarg del procés kafkià que suposa realitzar una tesi doctoral, l'autor necessita la col·laboració de tot un seguit de persones per poder arribar a l'ansiejat final. Aquestes breus paraules queden lluny de l'agraïment que realment mereixen totes aquelles persones, sense l'ajuda de les quals, aquest treball hagués estat directament impossible.

En primer lloc, vull agrair profusament el Dr. Miguel Lozano per l'esforç que m'ha abocat al llarg de tots aquests anys. Del seu consell, sempre d'amic, no només floreix aquesta tesi sinó també una forma honorable d'entendre la investigació i la vida que, per a mi, és tot un exemple a seguir. Regracie així mateix la seua família, Loles i el nouvingut Lucas, per obrir-me les portes de sa casa i per permetre'm furtar tant de temps del dia a dia de Miguel. En segon lloc, desitge agrair l'ajut incondicional de Fernando Barber com a company fidel de *viatge*. Les seues bones maneres esquitxen cadascuna de les aportacions d'aquesta tesi, la qual sense elles, mancaria del formalisme i funcionalitat escaient. Vull reconèixer especialment la confiança dipositada pel Dr. Juan Manuel Orduña vers la recerca desenvolupada en aquesta tesi i en els membres que la duen a terme. Gràcies pel recolzament econòmic i per convidar-nos a formar part d'un grup d'investigació amb tantes ambicions. Agraïsc l'afany de servei de Vicente Cavero, ja que he tingut el privilegi de comptar amb ell davant qualsevol problema tècnic. De Guillermo Vigueras són les últimes animacions 3D i la meua gratitud pel seu bon treball.

Envie un afectuós agraïment al Dr. Joan Pelechano i al Dr. Jesús Albert per tota la dedicació que desinteressadament han tingut amb mi des de sempre. També a la resta de companys del Departament d'Informàtica i de l'Institut de Robòtica de la Universitat de València per la companyonia (especialment a Rafa Lucía pel seu suport en els moments inicials). Vull expressar la meua gratitud cap a la Dra. Núria Pelechano i el Dr. Norman I. Badler per totes les atencions rebudes durant l'estada realitzada al *Centre for Human Modeling and Simulation* (University of Pennsylvania - EUA).

Tocant a la vessant lingüística d'aquesta tesi, cal agrair l'altruisme de dos grans amics: Óscar Díaz i Ivan Carbonell. Sense el primer, cap publicació en llengua anglesa haguera assolit el nivell necessari. Del mestre Carbonell agraïsc les seues correccions i la seua devoció dominical per mostrar-me un camí que m'està donant tant.

Finalment, però no menys important, agraïsc tot el temps que la meua família ha sacrificat per mi; sense ells, de ben segur que no hagués arribat a fer aquesta tesi. D'Emi vénen tots els ànims que he necessitat per concloure aquest treball i el desig de no despertar mai del nostre somni, que a poc a poc, es va fent realitat...

# Resumen

En los inicios del siglo XXI, el mundo físico pierde distancia en la carrera por la realidad que los enfrenta a los mundos virtuales. Día tras día, los mundos simulados arrebatan un palmo de terreno e incluyen nuevos campos de aplicación que no sólo complementan los servicios que se encuentran en el mundo real, sino que pueden llegar a substituirlos en un futuro posiblemente no muy lejano.

La simulación de mundos virtuales habitados por personajes 3D es un problema complejo que necesita la integración de áreas diversas como por ejemplo los gráficos por computador y la inteligencia artificial. De primera se puede obtener la credibilidad gráfica de la escena, de la segunda, la autonomía y la interactividad propias de la animación comportamental de los así conocidos como 3DIVA (*3D Intelligent Virtual Agents*). Normalmente, la interactividad de los 3DIVA se centra en el entorno (p. ej. la capacidad de ejecutar tareas sobre los objetos 3D) o en el usuario (p. ej. la capacidad de comunicación con el usuario humano, la reproducción de emociones o actitudes expresivas, etc.). No obstante, hace falta realizar un paso adelante cuando se trabaja con sociedades artificiales como las que aparecen, por ejemplo, dentro de los mundos virtuales de los juegos de entretenimiento, de los simuladores para el aprendizaje de tareas civiles o militares, o de ciberespacios como el famoso SecondLife. A pesar del realismo gráfico alcanzado por las técnicas de animación, es necesario profundizar en la interacción entre estos actores sintéticos autónomos de cara a hacerlos evolucionar hacia una inteligencia social que esté más cerca de la sociabilidad humana.

**El objetivo principal de esta tesis es la integración de habilidades sociales en la animación comportamental de humanoides autónomos situados dentro de mundos virtuales 3D.** Por tanto, en primer lugar hacemos una revisión de los mundos virtuales habitados, así como de los diferentes mecanismos de animación comportamental utilizados normalmente en actores sintéticos. En segundo lugar, repasamos los modelos de sociabilidad más comunes que se pueden encontrar dentro del estado del arte de los agentes socialmente inteligentes.

La toma de decisiones que han de llevar a cabo los actores sintéticos con capacidades sociales para interaccionar con el resto de agentes supone realizar procesos cognitivos complejos y requiere de un conocimiento abstracto de los elementos del entorno. En los

últimos tiempos, se ha propuesto la **inclusión de información semántica dentro de los mundos virtuales**, conocidos bajo el término anglosajón de *Semantic Virtual Environments* (SVE). De acuerdo con esto, en este trabajo proponemos el uso de **ontologías** para conseguir tres objetivos: mejorar la sensorización de escenas complejas, definir operativas generales que los agentes puedan reutilizar en situaciones diversas y definir las relaciones sociales establecidas entre los miembros de una sociedad artificial.

La naturaleza proactiva de los actores sintéticos ha hecho que las técnicas de animación comportamental más comunes, basadas en la selección dinámica de acciones, se dividan fundamentalmente en dos grupos: la planificación de tareas (p. ej. las basadas en el modelo *STRIPS*) y los sistemas basados en reglas (p. ej. los modelos *Belief-Desire-Intention*). Las aportaciones principales de esta tesis se encuentran en la inclusión de habilidades sociales en ambos paradigmas, ya que los consideramos aproximaciones válidas, extensamente utilizadas y muchas veces complementarias. En primer lugar, presentamos como conseguir **comportamientos colaborativos en grupos de agentes basados en planificadores heurísticos**. Hacemos servir la comunicación de acciones para la coordinación entre las actividades de los humanoides y la comunicación de objetivos para la cooperación entre los personajes 3D. Así pues, hemos desarrollado un mecanismo de sopesado heurístico que permite asignar un peso a cada objetivo, de manera que refleje su importancia social. Sin embargo, la eficiencia no es la única vía de expresar la sociabilidad. Por el contrario, la toma de decisiones humana a menudo tiene en cuenta más de un punto de vista y no todos persiguen la consecución rápida de las tareas encargadas. Por ello, presentamos **MADeM (Multi-modal Agent Decision Making)**, un formalismo de toma de decisiones de tipo social que puede ser utilizado por agentes BDI para la toma de decisiones socialmente aceptables. MADeM es capaz de evaluar diferentes soluciones a un problema mediante técnicas sociales basadas en la economía de mercados, donde se utilizan las subastas como mecanismo para que los agentes expresen su preferencia

Finalmente, exponemos un conjunto de experimentos que se han realizado para comprobar el impacto de las distintas técnicas sociales desarrolladas. Para evaluar objetivamente los resultados, no sólo se analizan las trazas de comportamiento de los personajes 3D sino que también se muestran diversas métricas de rendimiento y de animación que miden aspectos como: la coordinación, la especialización, el altruismo, la reciprocidad, etc. Por último, exponemos las conclusiones más significativas de este trabajo así como las líneas de investigación futuras.

**Palabras clave:** Animación comportamental, entornos virtuales semánticos, simulación social.

# ÍNDICE

# ÍNDICE DE TABLAS

# ÍNDICE DE FIGURAS

# PARTE I

# MARCO DE TRABAJO

**CAPÍTULO 1**

# INTRODUCCIÓN

*El hombre, ese ser tan débil, ha recibido de la naturaleza dos cosas que deberían hacer de él el más fuerte de los animales: la razón y la sociabilidad.*

Lucio Anneo Séneca (4 a.C, 65 d.C.).

## 1.1. INTRODUCCIÓN Y MOTIVACIONES

El grado de avance alcanzado por la tecnología en los últimos tiempos ha permitido acercar los mundos virtuales al mundo físico hasta un punto en que podemos ver ciertos aspectos de los ciberespacios definidos en la ciencia-ficción [Gibson, 1984, Stephenson, 1992] como un hito alcanzable. Con la ayuda de un soporte *hardware* cada vez más potente, las técnicas de animación gráfica son capaces de reproducir escenarios tridimensionales en tiempo real con mucha fidelidad. Paralelamente a la evolución de Internet, este hecho ha provocado la aparición de **mundos simulados** donde usuarios de alrededor del mundo pueden interactuar con diferentes elementos virtuales (como, por ejemplo, objetos 3D o representaciones gráficas de otros usuarios conocidas como *avatares*). Algunos ejemplos típicos de aplicaciones que utilizan estos mundos 3D son los juegos de entretenimiento, los simuladores para el aprendizaje de tareas civiles o militares, o ciberespacios como el famoso SecondLife (ver la figura 1.1).

Día tras día, los mundos simulados arrebatan un palmo de terreno e incluyen nue-


*Figura 1.1: Ejemplos de mundos virtuales 3D: (a) ciudad habitada de Stormwind en el juego Warcarft, (b) simulador militar del Departamento de Defensa de los E.E.U.U. i (c) ciudad de Amsterdam en SecondLife.*

vos servicios que no sólo complementan aquellos que se encuentran en el mundo real, sino que contra todo pronóstico podrían llegar a substituirlos parcialmente en un futuro no muy lejano. Esta funcionalidad en exclusiva podría llegar a comprometer la hegemonía del mundo tangible y producir, por tanto, el surgimiento de una *interrealidad* [van Kokswijk, 2007] poblada de **sociedades artificiales de humanoides virtuales potencialmente autónomos**.

Un aspecto al que se enfrentan muchos de estos mundos es al reto de habitar el entorno con criaturas virtuales autónomas (sean humanoides o animales) que no sólo muestren una buena calidad gráfica sino también una cierta **credibilidad de comportamiento**. El objetivo final de la **animación comportamental** es la construcción de un sistema inteligente capaz de integrar las técnicas necesarias para la simulación realista del comportamiento de los seres virtuales: la percepción, el control motor, la toma de decisiones, la ejecución de acciones, la comunicación entre los individuos, la interacción con el entorno, etc. Esta

necesidad ha promovido la inclusión de técnicas provenientes del campo de la inteligencia artificial con el objetivo de conseguir actores virtuales inteligentes (IVA, *Intelligent Virtual Agents*) con **propiedades centradas en el agente** como por ejemplo: **la autonomía, la reactividad y la proactividad**. En esta línea se encuentra el trabajo seminal de esta tesis, realizado en el Grupo de Caracteres Inteligentes (ICG) del Departament d'Informàtica de la Universitat de València, que consistía en implementar un sistema de animación comportamental de personajes 3D basado en técnicas de planificación heurística [Lozano, 2005].

Sin embargo, los actores virtuales inteligentes todavía carecen de una cualidad que los asemeje a sus referentes originales: **la inteligencia colectiva**. La inteligencia colectiva surge como el resultado de la interacción entre los personajes y ésta no puede olvidar las características sociales de la animación comportamental [1]. Esta necesidad es todavía más adecuada cuando hablamos de humanoides virtuales o actores sintéticos, ya que la sociabilidad que un observador externo espera de un humano simulado es más compleja que la mostrada por otros animales, con menos capacidades de razonamiento.

La investigación llevada a cabo en el área de la interacción de los humanos 3D se ha centrado mayoritariamente en la mejora de la comunicación entre un carácter autónomo y un usuario humano. Como resultado de estos trabajos, se pueden encontrar actores sintéticos con los cuales mantener conversaciones más o menos realistas como por ejemplo presentadores, doctores o guías virtuales (ver el capítulo 2). Para incrementar su fidelidad, estos personajes a menudo incluyen mecanismos de expresividad gestual y corporal con los que exteriorizar emociones o enfatizar el significado del discurso. A pesar de ello, los **actores socialmente inteligentes** también deberían ser capaces de dirigirse a otros individuos autónomos para conseguir sus objetivos, sin la intervención de un usuario humano. Normalmente, los mundos artificiales están poblados de diversos caracteres que comparten un número finito de recursos. Por tanto, los conflictos aparecen rápidamente a causa de la voluntad de usar un mismo recurso de manera simultánea. Por ejemplo, un mismo martillo 3D podría ser deseado por dos carpinteros virtuales; uno para clavar un clavo y otro para encajar dos piezas. Otro ejemplo de recursos compartidos es un pedido hecho por un cliente en un bar virtual; el cual puede ser atendido por diversos camareros. Por tanto, el uso de técnicas de **coordinación y cooperación** es necesario porque los personajes autointeresados pronto caerán en conflictos y producirán simulaciones de

[1] Según la teoría de agentes, las cuatro propiedades que definen un comportamiento inteligente son: la autonomía, la reactividad, la proactividad y la sociabilidad [Wooldridge and Jennings, 1995].

baja calidad. Por ejemplo, los camareros comenzarán a obstruirse si pretenden atender el mismo pedido.

Además, los humanoides virtuales habitualmente representan un rol en el escenario (p. ej. carpintero, camarero, etc.) que establece relaciones diversas entre los miembros de la sociedad artificial. Estos enlaces definen una **red social** que debería ser considerada a la hora de tomar **decisiones socialmente aceptables** y evitar de esta manera comportamientos robóticos. Los actores inteligentes necesitan evaluar el impacto social de sus acciones y decidir su actuación de acuerdo con la sociedad simulada. Uno espera, por ejemplo, que dos actores y amigos virtuales intercambien información, se agrupen y conversen entre ellos esporádicamente.

En resumen, hace falta profundizar en la interacción entre los actores sintéticos autónomos de cara a hacerlos evolucionar hacia una inteligencia social capaz de emular la toma de decisiones humana, siendo esta la **principal motivación** de esta tesis. Esta tarea conlleva hacer una sinergia entre múltiples disciplinas como por ejemplo: los gráficos por computador, la inteligencia artificial, la psicología y la sociología. De manera particular, este documento estudia el uso de técnicas provenientes de la teoría de la decisión social (*Social Choice Theory*) y de la economía del bienestar (*Welfare Economics*).

## 1.2. OBJETIVOS

*El objetivo principal de esta tesis es la integración de habilidades sociales en la animación comportamental de humanoides autónomos situados dentro de mundos virtuales 3D.*

La generación de comportamientos sociales animales (p. ej. el movimiento de una bandada de pájaros) se puede conseguir con el uso de técnicas de razonamiento reactivas que no necesiten de una representación del entorno muy sofisticada. Sin embargo, la toma de decisiones que han de llevar a cabo los humanos virtuales para interaccionar tanto con los objetos como con el resto de agentes lleva conlleva procesos cognitivos complejos y requiere de un conocimiento abstracto de los elementos del entorno. En los últimos tiempos, se ha propuesto la **inclusión de información semántica dentro de los entornos virtuales**, conocidos bajo el término anglosajón de *Semantic Virtual Environments* (SVE).

De acuerdo con esto, el primer objetivo de esta tesis será la búsqueda de una solución genérica para la representación simbólica de los mundos virtuales 3D.

La naturaleza proactiva de los actores sintéticos ha hecho que las técnicas de animación comportamental más comunes, basadas en la selección dinámica de las acciones, se dividan fundamentalmente en dos grupos: la planificación de tareas (p. ej. las basadas en el modelo *STRIPS*) y los sistemas basados en reglas (p. ej. los modelos *Belief-Desire-Intention*). Consideramos ambas aproximaciones como válidas, extensamente utilizadas y muchas veces complementarias. Por ejemplo, es posible combinar la planificación para la secuenciación de tareas en el bajo nivel y el modelo BDI para el mantenimiento de los objetivos del más alto nivel. En consecuencia, el objetivo principal de esta tesis ha sido la inclusión de habilidades sociales en ambos paradigmas. Más concretamente, se han abordado los siguiente subobjetivos:

- **Colaboración en planificadores heurísticos:** A pesar de que las simulaciones de humanos virtuales no requieren normalmente la ejecución de planes óptimos, es evidente que la credibilidad de un conjunto de personajes depende en gran medida de la coherencia de sus acciones dentro del contexto del grupo. Esta coherencia puede emerger de comportamientos próximos a la situación más eficientes; como por ejemplo el cambio y la adaptación de las tareas que realiza un carácter de acuerdo con aquellas realizadas por el resto de compañeros. El alto dinamismo de los entornos 3D hace desaconsejable la creación centralizada de un único plan común para todos los agentes, ya que éste será inválido en poco tiempo. Bien al contrario, mediante la coordinación y la cooperación de planes individuales, los agentes pueden evitar interferencias y sacar beneficio en la consecución de sus objetivos personales.

- **Toma de decisiones multimodal:** Las decisiones humanas a menudo atienden a más de un punto de vista y no todos persiguen un comportamiento coordinado. Por ejemplo, un actor virtual vago podría escoger el descanso en vez de trabajar conjuntamente con un compañero para conseguir los objetivos más rápidamente. Asimismo, un actor virtual calculador podría decidir ayudar a otro dependiendo del balance en el intercambio de favores realizado previamente. El objetivo final de este trabajo ha sido el desarrollo de un módulo de razonamiento general capaz de tener en cuenta preferencias diferentes de los agentes y que pueda ser utilizado por agentes BDI a la hora de tomar decisiones socialmente aceptables.

## 1.3. APORTACIONES

De conformidad con los objetivos definidos en el punto anterior, las aportaciones principales de esta tesis se pueden resumir en los siguiente puntos:

- Proponemos un modelo general de **entorno virtual semántico (SVE) basado en ontologías** con el que se pueden conseguir tres objetivos básicos para la animación de actores sintéticos: la mejora de la sensorización de escenas complejas, la definición de operativas generales que los personajes puedan reutilizar en situaciones diversas y la definición de las relaciones sociales y de organización establecidas entre los miembros de una sociedad artificial.

- Proponemos una nueva **técnica para conseguir comportamientos colaborativos en grupos de agentes basados en planificadores heurísticos**. Utilizamos la comunicación de acciones para reducir las interferencias entre las actividades de los humanoides y la comunicación de objetivos para la cooperación entre los personajes 3D. De acuerdo con esto, hemos desarrollado un mecanismo de sopesado independiente del planificador y de la función heurística que permite asignar un peso a cada objetivo, de manera que refleje su importancia social. Así pues, los agentes pueden adoptar dinámicamente objetivos externos (de menos importancia que los propios) y conseguir comportamientos coordinados sin aumentar la complejidad de sus decisiones.

- Presentamos **MADeM (*Multi-modal Agent Decision Making*)**, un proceso de toma de decisiones de tipo social que puede ser utilizado por agentes BDI a la hora de tomar decisiones socialmente aceptables. MADeM es capaz de evaluar diferentes soluciones a un problema mediante una técnica social basada en el mercado que utiliza las subastas como el mecanismo para que los agentes expresen sus preferencias. Además, MADeM permite definir diferentes órdenes sociales en una sociedad (p. ej. elitista, utilitario, etc.) así como las actitudes personales de sus miembros (p. ej. egoísmo, altruismo, etc.).

## 1.4. ORGANIZACIÓN DEL TRABAJO

La estructura de este documento se divide en los siguiente capítulos:

- El capítulo 2 hace un repaso a los entornos virtuales habitados por actores sintéticos. En este capítulo se estudia la inclusión de información semántica en los mundos simulados. Asimismo, se analiza el espectro actual de los actores virtuales inteligentes 3D (3DIVA) y se describen los mecanismos más comunes de selección dinámica de acciones en agentes racionales.

- En el capítulo 3 se examinan los ejemplos más representativos de animación comportamental social en personajes 3D. A continuación, hacemos un análisis de los modelos de sociabilidad más comunes que se pueden encontrar dentro del estado del arte de los sistemas multiagente. Posteriormente, se definen algunos conceptos a menudo borrosos como por ejemplo: coordinación, cooperación, tipos de redes sociales, etc.

- El capítulo 4 presenta el marco de simulación multiagente diseñado para la integración de actores sintéticos con habilidades sociales en entornos virtuales 3D. Primeramente, describimos nuestro modelo de entorno virtual semántico (SVE) basado en ontologías OWL (Web Ontology Language) para la representación semántica del mundo. En segundo lugar, proponemos una arquitectura modular para el control de la actuación de los actores socialmente inteligentes.

- El capítulo 5 contiene las aportaciones principales de esta tesis en materia de toma de decisiones sociales: a) la técnica de coordinación de tareas y de sopesado de objetivos externos para conseguir comportamientos colaborativos en planificadores heurísticos; y b) MADeM, el proceso de toma de decisiones multimodal y social basado en el reparto de recursos en sistemas multiagente (*Multi-Agent Resouce Allocation, MARA*).

- En el capítulo 6 examinamos los resultados obtenidos en tres ejemplos de complejidad creciente: el mundo de bloques clásico, el mundo de bloques 3D multipersonaje y la animación de un bar universitario virtual. Demostramos el impacto de las distintas técnicas sociales desarrolladas mediante el análisis de las trazas de comportamiento y la comparación de diversas técnicas de rendimiento y de animación que miden aspectos como: la coordinación, la especialización, el altruismo, la reciprocidad, etc.

- Finalmente, el capítulo 7 recoge las conclusiones generales del trabajo realizado además de las líneas de investigación que se derivan y que todavía permanecen abiertas. De acuerdo con éstas, se indicarán los trabajos de investigación futuros así como los ya publicados.

# PARTE II

# ESTADO DEL ARTE

CAPÍTULO 2

# ENTORNOS VIRTUALES 3D HABITADOS

En este capítulo analizamos los entornos virtuales 3D habitados por actores sintéticos autónomos como evolución de los escenarios despoblados de la realidad virtual clásica. En estos entornos, el grado de realismo alcanzado por el comportamiento de los personajes depende fundamentalmente de dos aspectos. El primero hace referencia a la **gestión semántica** de la información de la escena; la cual puede favorecer la interacción tanto con los objetos del mundo como con el resto de individuos. El segundo, apunta al mecanismo de **toma de decisiones** del personaje, el cual gobierna su animación comportamental.

## 2.1. POBLACIÓN DE LOS ENTORNOS VIRTUALES 3D

Los entornos virtuales tridimensionales (EV3D) han experimentado una evolución muy significativa a causa de la gran cantidad de avances tecnológicos que se han producido desde que comenzó su desarrollo. Aunque fueron concebidos originariamente con el propósito de reproducir entornos protencialmente peligrosos para el entrenamiento militar (p. ej. los simuladores de vuelo), la popularización de la tecnología de visualización gráfica en tiempo real para el uso civil significó el surgimiento del concepto más amplio de **realidad virtual (RV)** [Stanney, 2002]. Al amparo de este termino tan laxo fueron apareciendo todo un conjunto de EV3D en **campos de aplicación** tan diversos como: la ingeniería, la ergonomía, la medicina, el entrenamiento, la educación, la visualización científica, el ocio, etc.

El continuo aumento de la potencia del *hardware* y la mejora del *software* gráfico 3D al alcance de los usuarios particulares creó nuevas expectativas de realismo en los

EV3D. Frente a la visión clásica de inmersión presencial perseguida por los dispositivos de entrada y salida como los cascos de RV o los guante de datos, los EV3D pretenden alcanzar una **inmersión mental del usuario**. Es decir, aquella a la que se llega cuando el usuario se siente integrado en una realidad más allá de la física; como la que somete al espectador de una representación teatral o al lector de un libro. Para conseguirla, se hizo necesario substituir las escenas estáticas y despobladas de los inicios por mundos poblados de objetos y de personajes cuyo comportamiento recordara a sus equivalentes en el mundo real.

De esta manera surgen los **entornos virtuales inteligentes** (*Intelligent Virtual Environments* o IVE) [Luck and Aylett, 2000], combinación de la inteligencia artificial (IA) y las técnicas de vida artificial con los entornos virtuales. Esta sinergia ha dado sus frutos en la construcción de EV3D dinámicos [Aylett and Cavazza, 2001] y en su interacción con el usuario [Calderón et al., 2003]. No obstante, el aspecto en el que se produce una mayor convergencia es aquel que versa sobre los personajes 3D que habitan los mundos virtuales. Se puede concebir un personaje 3D bajo el paradigma de agente autónomo proveniente de la IA, ya que comparten propiedades importantes. Básicamente, ambos necesitan un sistema de control con capacidades de percepción, de toma de decisiones y de actuación para comportarse de forma correcta en su entorno. Por tanto, los actores sintéticos heredan el problema clásico del comportamiento autónomo al que se enfrentan, por ejemplo, los agentes *software* o los agentes robóticos. El término científico que aúna este marco con los requerimientos de los entornos y de los agentes virtuales es el de **actores virtuales inteligentes 3D** (3DIVA).

El desarrollo de 3DIVA conlleva la resolución de multitud de problemas complejos (p. ej. la navegación, la planificación, los modelos de personalidad, etc.) y su integración en entornos 3D [Badler et al., 1993]. Este hecho ha provocado que los grupos de investigación hayan centrado sus esfuerzos en asuntos concretos. Las aportaciones más representativas las encontramos en los campos del entretenimiento, de la pedagogía y de la comunicación y socialización en comunidades virtuales por Internet [Laird and Duchi, 2000, Elliott et al., 1999, Pelachaud, 2005]. Como resultado de todo ello, los **entornos virtuales habitados** se encuentran salpicados de agentes 3D que se especializan en conseguir comportamientos creíbles a distintos niveles: físico, racional, social...(ver apartado 2.3).

La animación del comportamiento de uno o más agentes en un entorno 3D propor-

ciona la experiencia visual para poder validar su grado de credibilidad o realismo comportamental alcanzado. En este sentido, el término credibilidad aparece como una norma escasamente definida [Badler and Allbeck, 2000] que tendrán que cumplir estos personajes 3D y que influirá decisivamente en el diseño de sus arquitecturas. Se ha distinguido la **investigación en agentes autónomos** de la **investigación en sistemas multiagente** [Luck and Aylett, 2000]. La primera, que trataremos en el punto 2.3 de este capítulo, trata normalmente sobre la generación de actores sintéticos que se comportan de manera egocéntrica y realiza especial hincapié en la coherencia de comportamiento en el uso de los objetos de entorno y en la expresión gestual y emocional. Sin embargo, el comportamiento de los 3DIVA autointeresados pronto se ve comprometido en contextos poblados por diversos agentes (p. ej. con la aparición de conflictos en el acceso a los recursos compartido). Por consiguiente, la inteligencia colectiva es una necesidad para muchos actores sintéticos y especialmente para los humanoides virtuales, ya que la sociabilidad que uno espera de un humano simulado es más compleja que la mostrada por otros animales. Así pues, hace falta incorporar mecanismos de interacción entre los distintos agentes autónomos que forman una sociedad artificial. Este aspecto se estudia en la investigación en sistemas multiagente y será tratado en el capítulo 3 de esta tesis.

La toma de decisiones que llevan a cabo los humanos virtuales para interaccionar tanto con los objetos como con el resto de agente conlleva procesos cognitivos complejos y requiere un conocimiento abstracto de los elementos del entorno. En consecuencia, antes de repasar los mecanismos de animación de comportamientos que se han aplicado comúnmente sobre los 3DIVA, analizamos con más detalle las soluciones adoptadas para la representación de la información contenida en los entornos virtuales 3D.

## 2.2. INFORMACIÓN SEMÁNTICA EN ENTORNOS VIRTUALES

A la hora de generar comportamientos complejos en los personajes 3D, la preocupación principal ha sido generalmente el proceso de toma de decisiones, ya que es el responsable de las acciones que finalmente son animadas. Sin embargo, hay una propiedad muy característica de los entornos virtuales que demanda una atención especial previamente: la **gran cantidad de información** que desprenden. Imaginemos, por ejemplo, la cantidad de información que contienen las estanterías llenas de libros de una biblioteca virtual. Si toda esta información se hubiera de tener en cuenta, la toma de decisiones sería del todo

intratable. A pesar de este hecho, todavía no existe un formalismo comúnmente aceptado de gestión de la gran cantidad de información asociada a los entornos virtuales 3D.

Una primera aproximación para organizar la información de una escena tridimensional son los sistemas basados en grafos de escena como por ejemplo Java3D [JAVA3D, 2008] o OpenSceneGraph [OSG, 2008]. A pesar de que pueden incluir sencillas reglas de animación y sistemas muy básicos de inteligencia artificial destinados al *software* de visualización gráfica [Rodríguez-García, 2004], estas estructuras de datos se utilizan fundamentalmente para el recorte automático de partes no visibles (*culling*). Por tanto, las soluciones basadas en grafos de escena ofrecen normalmente una interacción con los objetos del entorno nula o muy limitada.

Por el contrario, los motores de juegos (p. ej. Unreal Tournament [Epic games, 2008]) permiten la creación de sistemas muy interactivos [Lewis and Jacobson, 2002]. Los motores de juegos suelen utilizar un modelo de eventos para notificar los cambios que se producen en un entorno dinámico y un lenguaje de *script* para la definición de las modificaciones que provoca una acción sobre el estado del mundo. En este contexto, sin embargo, el código fuente que implemente la interacción es difícilmente reutilizable. Por ejemplo, mientras que un barman de la vida real tiene la habilidad general de servir bebidas a los clientes, un barman virtual normalmente tendrá definido un operador para cada tarea que pueda realizar. Así pues, podríamos encontrarnos con un barman virtual capaz de servir un zumo de naranja y una copa de vino pero sin la menor idea de como preparar una taza de té. Esto sucederá siempre que los agentes virtuales no tengan un conocimiento general sino particular de las situaciones a las que se enfrentan.

En los últimos tiempos, se ha propuesto la inclusión de modelos de representación semánticos para gestionar la información de los entornos virtuales, conocidos entonces como **Entornos Virtuales Semánticos (SVE)**. La definición de una base de conocimiento semántico puede beneficiar la producción de los entornos virtuales tridimensionales (sección 2.2.1). De manera especial si están poblados por personajes inteligentes, los cuales necesitarán interaccionar tanto con los objetos como con el resto de agentes de la escena. Por un lado, el uso de la semántica puede favorecer a la interacción agente-objeto, a través de esquemas de acción más generales (sección 2.2.2). Por otro lado, la interacción agente-agente puede mejorar gracias a la definición de las relaciones que ligan a los agentes y que restringen sus posibles actuaciones (sección 2.2.3).

Desde el punto de vista de los entornos virtuales habitados, podemos clasificar las aportaciones semánticas en tres categorías, que detallaremos a continuación: la semántica del mundo o representación del entorno; la semántica de las acciones o representación de las tareas y la semántica de la sociedad o representación de las relaciones entre los agentes.

### 2.2.1. Semántica del mundo: Representación del entorno

La comunidad de los gráficos por ordenador ha comenzado a incluir en los últimos tiempos el uso de información semántica en la construcción de los mundos virtuales. Por ejemplo, el sistema VR-Wise [Pellens et al., 2005] facilita el diseño de entornos simulados a los usuarios no especializados en realidad virtual. Para hacer esto, este *software* utiliza un modelo conceptual de mundo definido por medio de un conjunto de ontologías expresada en el lenguaje DAML+OIL [Joint US/EU ad hoc Agent Markup Language Committee, 2008]. La idea es, por tanto, que el diseñador seleccione los conceptos de la ontología de definición del mundo y, posteriormente, que el sistema se encargue automáticamente de transformarlos en las representaciones gráficas necesarias. Los lenguajes utilizados en esta aproximación han sido VRML y X3D para el renderizado de la escena en entornos web.

El concepto abstracto de entidad virtual ha sido desarrollado por Mario Gutiérrez en su trabajo de tesis sobre SVE [Gutierrez, 2006]. Este concepto incluye la geometría y la interfaz de usuario de los objetos 3D y se usa principalmente para la visualización y la interacción con los objetos en diferentes contextos o dispositivos. Este trabajo también incorpora una versión preliminar de una ontología para humanos virtuales (implementada en XML) que tiene como objetivo: la modelización del cuerpo humano, su animación y la interacción de los humanos virtuales con unos objetos especiales llamados *Smart Objects* [Kallmann and Thalmann, 1999]. Los *Smart Objects* son el resultado de extender los datos geométricos de un objeto 3D con información semántica para su correcta manipulación (p. ej. las posiciones para cogerlo). Desgraciadamente, la interacción personaje-*Smart Object* está normalmente gobernada por un conjunto de animaciones predefinidas (*scripts*), ya que la idea era reducir la complejidad de las tareas mediante la transferencia de información orientada a la animación al entorno virtual.

Las ciudades virtuales habitadas por actores sintéticos han utilizado, asimismo, la información semántica para definir los perfiles de la población y las topologías de los entor-

nos urbanos complejos [Farenc et al., 1999, Thomas and Donikian, 2000, Costa de Paiva et al., 2005] (ver la figura 2.1). Estos entornos informados definen particiones espaciales (p. ej. edificios, calles, cruces, etc) y los objetos que podemos encontrar dentro de éstas (p. ej. pasos de peatones, paradas de autobús, etc.). Esta información estructurada es utilizada por agentes inteligentes como por ejemplo peatones o conductores de coches virtuales, básicamente con propósitos de navegación.

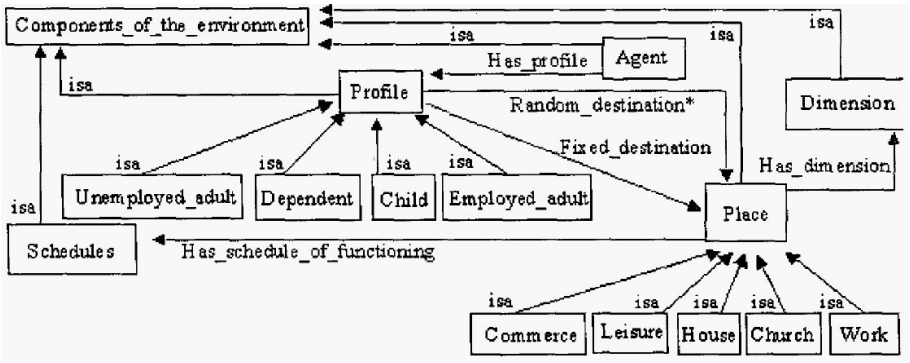


*Figura 2.1: Ontología para representar entornos urbanos [Costa de Paiva et al., 2005].*

### 2.2.2. Semántica de las acciones: Representación de las tareas

La interacción agente-objeto puede ser enriquecida gracias a la información semántica en esquemas de acción complejos [Gil, 2005]. A continuación mostramos un conjunto de ejemplos representativos. Primeramente, el razonador específico de objetos OSR (*Object Specific Reasoner*) [Levison, 1996] era capaz de clasificar los objetos en taxonomías y de decidir qué se podía hacer con ellos dependiendo de esta información (p. ej. los objetos contenedores pueden ser abiertos con las manos). En segundo lugar, la representación parametrizada de las acciones [Badler et al., 2000] permitía definir tareas no instanciadas (llamadas UPARs) que no especificaban ni el actor virtual ni los objetos involucrados en la acción. De esta manera, todas juntas representaban cualquier acción posible en el mundo. A la hora de ejecutar una acción en un momento particular, las UPARs se instanciaban con las entidades virtuales afectadas. En tercer lugar tenemos el lenguaje de alto nivel *Task Definition Language* [Vosinakis and Panayotopoulos, 2003], que persigue el objetivo de definir tareas independientes del contexto. La ventaja principal del lenguaje propuesto es el hecho de que permite que las tareas sean construidas fácilmente y que puedan ser utilizadas por agentes diferentes en entornos diversos. Finalmente, la separación de acciones y consecuencias en [Soto and Allongue, 2002] muestra otro uso interesante de la

semántica para conseguir la interoperabilidad y la reusabilidad de las entidades virtuales. Desgraciadamente, los autores no definen una representación que pueda ser utilizada en la construcción y gestión de los mundos virtuales.

Una de las primeras soluciones para la representación de la información consiste en insertar conocimiento en el mundo mediante anotaciones (*Knowledge in the World*) [Doyle, 2002]. Esta aproximación propone la codificación dentro del mundo de aquel conocimiento y de aquellas habilidades de un caracter que son específicas de un entorno particular; mientras que las habilidades invariables permanecen como parte del agente. De esta manera, se favorece la reusabilidad de los actores en dominios diferentes. Recientemente pero de una manera similar, el sistema SeVEn [Otto, 2005] ha tratado de conseguir que el *software* pueda ser reutilizado sobre diferentes entornos virtuales en un contexto web. Este sistema admite la definición de SVEs mediante el uso de RDF (*W3C Resource Description Framework*) y categoriza los objetos mediante un campo explícito de *tipo de objeto*. De acuerdo con este campo, los agentes pueden extraer información relevante para ejecutar tareas sobre el objeto en cuestión. Sin embargo, las anotaciones estáticas no son suficiente cuando se trabaja con entornos dinámicos como los entornos virtuales 3D.

Des de la comunidad de agentes, una aproximación semánticamente mejorada pretende conseguir la anotación del entorno de forma dinámica por medio de una capa conceptual basada en ontologías [Chang et al., 2005]. El uso de una capa conceptual puede ayudar a los agentes inteligentes a planificar su comportamiento mediante la inferencia ontológica (ver la figura 2.2). No obstante, el esquema de acción adoptado en esta solución tiene diversas limitaciones de expresividad; el efecto de una acción se restringe a un único objeto y siempre supone el cambio de éste a una instancia de otro concepto cualitativo diferente. Sin embargo, a menudo las acciones pueden afectar múltiples objetos o cambiar una propiedad de un objeto sin modificar ningún concepto. Un ejemplo sería el de rellenar un vaso con el líquido de una botella de vino. A pesar de que la cantidad de líquido variará, ambos objetos se mantendrán como instancias de los mismos conceptos.

La representación de los objetos y de la operatividad de los agentes puede juntarse en una base de conocimiento común mantenida por el entorno. El lenguaje de descripción de entornos para simulaciones multiagente ELMS (*Environment Description Language for Multi-Agent Simulation*) [Okuyama et al., 2005] permite definir tanto las propiedades de los objetos como los *cuerpos* de los agentes: percepción y actuación. Es decir, el diseñador

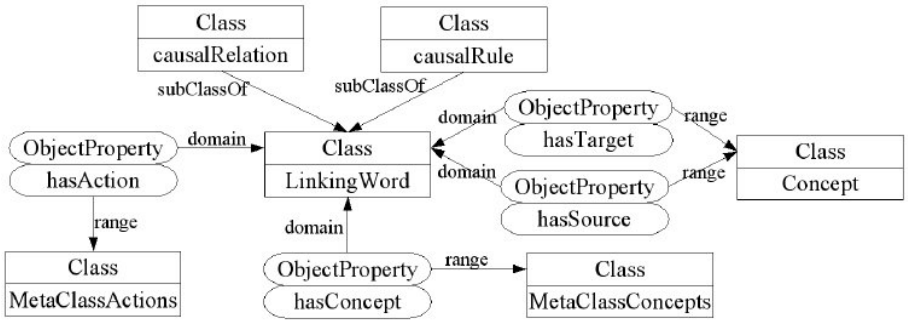


*Figura 2.2: Ontología para la representación de las tareas [Chang et al., 2005].*

puede controlar qué propiedades son accesibles per las *mentes* de los agentes mediante modos de percepción diversos. Asimismo, y de conformidad con el modelo STRIPS, una acción se define como la secuencia de cambios que produce en las propiedades de los recursos afectados; junto con las precondiciones que se han de satisfacer para que la acción pueda ser ejecutada.

### 2.2.3. Semántica de las sociedades: Representación de las relaciones

La búsqueda de personajes creíbles con frecuencia implica la simulación de múltiples caracteres, los cuales forman una especie de sociedad artificial. Así como no se puede concebir un ser humano real sin ningún tipo de relación social con otros, los humanos virtuales que carezcan de enlaces sociales dentro de las sociedades virtuales verán reducida su credibilidad. El uso de información semántica puede mejorar la representación de las relaciones entre los actores sintéticos.

La figura 2.3 muestra un ejemplo reducido de sociedad feudal aparecido en [Kao et al., 2005]. Este contexto social, expresado en OWL (*Web Ontology Language*) [W3C, 2004], admite la representación de: roles (círculos), compromisos sociales (hexágonos) e intenciones sociales (octágono). Mediante estas relaciones y un mecanismo de inferencia se pueden detectar conflictos sociales que tendrán que ser resueltos por la toma de decisiones del agente. Por ejemplo, el personaje *Sot* de la figura necesita cambiar su estatus social de bandido a caballero para poder casarse con *Julio* a causa del compromiso social que fuerza a los nobles a emparejarse con otros de su mismo rango.

Los agentes cognitivos autónomos pueden aumentar su eficiencia y adaptabilidad a las

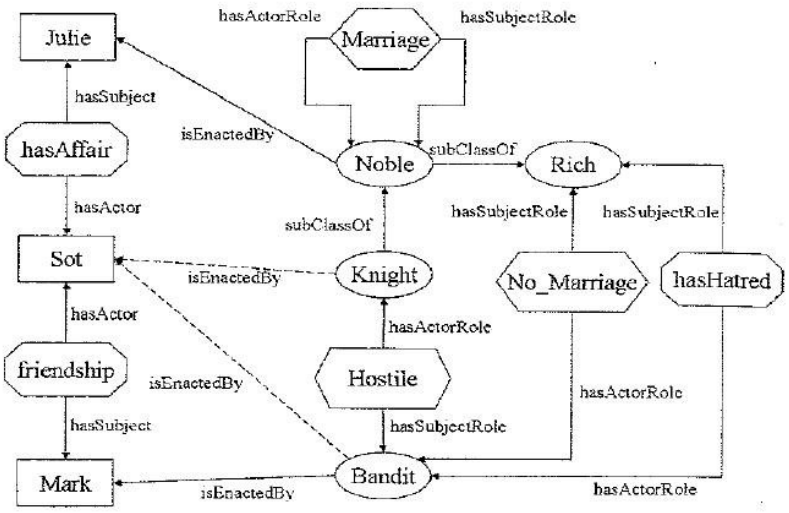


*Figura 2.3: Representación ontológica de una pequeña sociedad feudal [Kao et al., 2005].*

alteraciones del entorno si son capaces de representar explícitamente y de explorar, a través de mecanismos de razonamiento adecuados, las capacidades de otros agentes y de las organizaciones donde están inmersos. Se puede concebir una organización como el conjunto de restricciones en el comportamiento de los agentes con el objetivo de conducirlos a una finalidad comun. Para describir una organización, se han utilizado modelos funcionales (TÆMS [Decker, 1998]), estructurales (AALAADIN [Ferber and Gutknecht, 1998]) y mixtos (TOVE [Fox et al., 1998]). Más contemporáneo, el modelo Moise+ [Giménez-Lugo et al., 2005] permite la descripción de una organización en tres dimensiones: estructural, funcional y deontológica. El modelo estructural define los grupos y los roles que pueden interpretar los agentes. También incluirá diversas relaciones entre ellos como por ejemplo las relaciones de autoridad, de responsabilidad, etc. El modelo funcional define el objetivo final del grupo, las misiones necesarias para conseguirlo y las tareas en las que se descompone. Finalmente, el modelo deontológico asocia los roles responsables de cada una de las tareas, es decir, las obligaciones de los agentes de la organización.

Un último ejemplo de información semántica alojada en el entorno virtual y diseñada para ser usada por parte de los agentes que lo habitan son los *coordination artifacts* [Viroli et al., 2006]. Estos *artefactos* son unas entidades no racionales que ofrecen protocolos de coordinación para que los agentes colaboren dentro de una sociedad. Esto es, en vez de enviar y recibir mensajes directamente, los agentes ejecutarán acciones sobre los *coordination artifacts*. Por consiguiente, esta técnica supone una alternativa interesante

de interacción estructurada frente a los lenguajes de comunicación de agentes (ACLs).

En la primera parte de este capítulo hemos analizado el problema de la representación semántica de los entornos virtuales habitados. La segunda parte tratará sobre los mecanismos de toma de decisiones utilizados más a menudo en la animación comportamental de los 3DIVA.

## 2.3. ANIMACIÓN COMPORTAMENTAL EN 3DIVA

La animación comportamental de agentes virtuales 3D tiene como objetivo la construcción de sistemas inteligentes capaces de integrar diversas técnicas con el fin de conseguir la simulación verosímil de sus comportamientos. Entre ellos podemos incluir la percepción, el control motor, la selección de objetivos, la ejecución de acciones, la comunicación entre los agentes, etc. [Iglesias and Luengo, 2004]. La visión clásica de la animación comportamental, defendida por Demetri Terzopoulos desde la *Universidad de Toronto*, muestra en forma de jerarquía piramidal la sinergia entre los gráficos por computador, las técnicas de animación y la inteligencia artificial (ver la figura 2.4a). Esta visión concibe el problema como una evolución desde los modelos geométricos hasta las técnicas de modelado cognitivo [Tu and Terzopoulos, 1994, Funge et al., 1999].

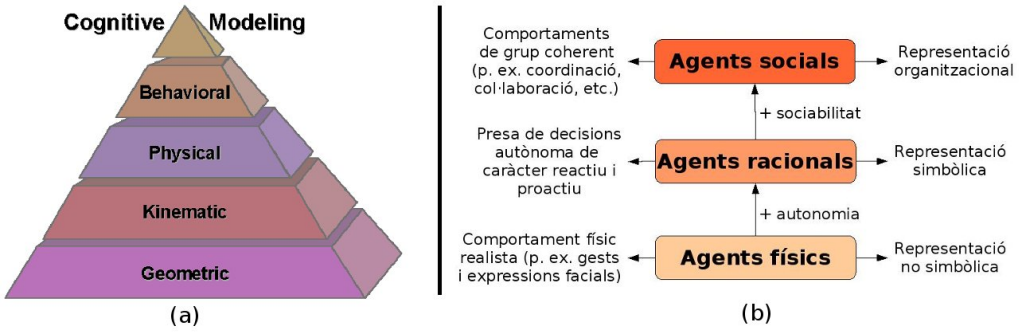


*Figura 2.4: (a) Visión piramidal clásica de la animación comportamental [Funge et al., 1999] (b) Espectro actual de actores 3D.*

El trabajo desarrollado en esta tesis se emplaza en la parte alta de esta pirámide. En este contexto, el estado del arte de los 3DIVA puede ser analizado mediante un espectro de agentes como el que aparece en la figura 2.4b. Se distinguen tres categorías principales: los agentes físicos, los agentes racionales y los agentes sociales. Por un lado, los **agentes**

**físicos** ponen énfasis en las habilidades motoras para conseguir comportamientos realistas a nivel físico (p. ej. interacción física con el entorno, expresiones faciales y gestuales, etc.). Estos agentes normalmente actúan de manera guiada y perciben el EV3D mediante sensores que trabajan a un nivel no simbólico. Por otro lado, los **agentes racionales** trabajan el comportamiento con una perspectiva operativa, es decir, como el problema de animar un conjunto de acciones coherentes con los objetivos definidos para el agente y el estado del entorno. Al contrario de los físicos, se caracterizan por una toma de decisiones autónoma, que necesitará a menudo información simbólica del EV3D para poder razonar. Por último encontramos los **agentes sociales**, los cuales tienen como objetivo enriquecer la toma de decisiones con comportamientos de grupo verosímiles (p. ej. comunicación entre los agentes autónomos, coordinación de sus tareas, etc.). Mientras que los agentes racionales deciden en función de puntos de vista personales o egocéntricos, los agentes sociales tienen en cuenta también al resto de agentes en su proceso de toma de decisiones. Por tanto, estos agentes necesitarán, además, de información organizacional sobre las relaciones que se establecen en la sociedad artificial.

Idealmente, los humanoides 3D deberían mostrar una interacción física con el resto del mundo totalmente realista así como también deberían poseer habilidades racionales y sociales similares a las de los humanos. Sin embargo, todos estos aspectos conllevan la resolución de problemas complejos, de manera que los investigadores han tendido a poner el énfasis en una u otra parte del espectro. A continuación, hacemos un repaso más detallado a los agentes físicos y racionales definidos previamente. Como ya se ha comentado, dejamos la discusión alrededor de los agentes sociales para el capítulo 3 de este documento.

### 2.3.1. Agentes físicos

El propósito de los agentes físicos es aparentar **comportamientos realistas a nivel físico**. Normalmente, estos personajes 3D carecen de sensores y de mecanismos de toma de decisiones, cosa que hace de su sistema motor el principal encargado de proporcionar los signos de expresividad necesarios. A causa de este énfasis en la manifestación física del actor en su entorno, estos agentes son llamados habitualmente *embodied agents*.

La aproximación más sencilla a la hora de animar un actor de este tipo consiste en describirlo completamente a lo largo de la simulación mediante un guión, de manera similar

a como un director de cine dirige a sus actores. Los **presentadores 3D** son el ejemplo más claro de **actores 3D guiados**. La figura 2.5a muestra a *Cléo*, uno de los primeros actores sintéticos en tiempo real desarrollado en el MEDIALAB de París por Canal+ Francia. El control corporal de *Cléo* se realizaba mediante la captura de los movimientos interpretados por un actor real mientras que el control gestual utilizaba un guante de datos. El uso de guiones programados o *scripts* ha sido otra manera muy común de realizar el guiado automático de un actor virtual. Un ejemplo representativo es *Jack Presenter* [Noma and Badler, 1997], un presentador virtual que utiliza el humanoide cuerpo entero *Jack* [Badler, 1997], desarrollado originariamente en la Universidad de Pennsylvania (ver la figura 2.5b). Los datos de entrada al presentador virtual incluyen típicamente comportamientos verbales, los cuales son procesados por un generador sintético del habla, así como comandos de control referidos al lenguaje corporal (p. ej. los gestos de las manos para dirigir la atención del público hacia una determinada información existente en la pantalla).

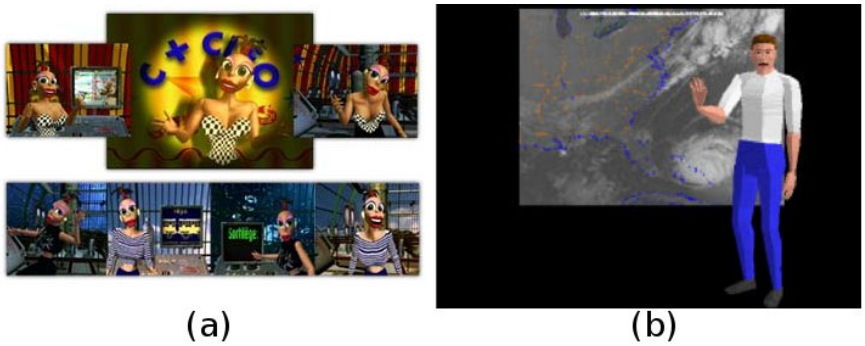


*Figura 2.5: Ejemplos de presentadores 3D: (a) Cléo y (b) Jack.*

A menudo la animación de un actor 3D se ve reducida únicamente a la cabeza, razón por la que se habla muchas veces de **cabezas hablantes** o *talking heads*. Estos personajes han centrado su atención en el movimiento sincronizado de los labios con el discurso y en la expresión facial de emociones. Estas funcionalidades son muy beneficiosas para el desarrollo de agentes que actúan como interfaz entre una persona y un ordenador (p. ej. asistentes de un programa de ordenador, presentadores de información por Internet, sistemas de videoconferencia 3D, etc.). El realismo expresivo de la cara es un campo de investigación con una gran tradición en la animación de humanos 3D [Thalmann et al., 1998] y todavía hoy en día genera mucha actividad [Pelachaud, 2005, Niewiadomski and Pelachaud, 2007]. Un ejemplo son los trabajos realizados por el grupo de Catherine Pelachaud en la Universidad de París para proporcionar a *Greta*, su agente conversacional, con las capacidades de realimentación conversacional y expresión de

emociones (ver la figura 2.6).

*Figura 2.6: Diferentes gestos y expresiones faciales reproducidos por* Greta *[Pelachaud, 2005].*

Las características referidas anteriormente no sólo son interesantes para los actores completamente guiados sino también para los personajes 3D semiautónomos, como los que representan los usuarios humanos en los entornos tridimensionales multiusuario basados en la red (p. ej. SecondLife [Linden Lab, 2008], Active Worlds [Activeworlds Corporation, 2008], etc.). Estas comunidades virtuales, conocidas genéricamente como entornos virtuales distribuidos (*Distributed Virtual Environments* o DVE)[1] nacen con la idea de ofrecer un espacio común de esparcimiento representado mediante un EV3D, donde los usuarios serán representados por **actores semiautónomos** o *avatares* que les permitirán percibir el entorno, reconocer al resto de usuarios (por su avatar), comunicarse y socializarse con ellos. Sin embargo, ciberespacios tan famosos como SecondLife todavía carecen de humanoides totalmente autónomos que puedan poblar el mundo sin que haya un usuario humano controlándolo por detrás. De hecho, éste es un punto alrededor del cual todavía hay abierta una gran discusión moral y legal [van Kokswijk, 2007].

### 2.3.2. Agentes racionales

Al contrario de los agentes físicos, los agentes racionales destacan por su **capacidad de tomar decisiones** de una manera autónoma. Según la inteligencia artificial, un agente es racional si hace aquello correcto de acuerdo con la información de que dispone. Por tanto, los comportamientos simulados, así como el grado de racionalidad alcanzado, dependerán entonces del modelo o formalismo de IA que se utilice en cada momento.

[1] O bien como MMORPG (*Massive Multiplayer On-line Role-Playing Games*). En los últimos tiempos, también han sido acuñadas como MMORLGs (*Massive Multi-player Online Real-Life Games*), ya que el usuario puede modificar su avatar e interpretar un rol más dinámico o incluso múltiples roles.

Podemos entender la toma de decisiones como el proceso de búsqueda de las acciones a realizar por un personaje. Atendiendo a la cantidad de búsqueda realizada, uno puede ver los agentes racionales como una gradación de agentes: desde los que no realizan ningún tipo de búsqueda sino que únicamente reaccionan a estímulos externos (**reactividad**) hasta los que incluyen algún mecanismo de búsqueda de acciones futuras para conseguir objetivos propios (**proactividad**).

Por definición, un agente es reactivo si percibe su entorno y reacciona ante sus cambios. El sistema de toma de decisiones del actor será, pues, el encargado de reaccionar a los cambios que afectan a su comportamiento y de filtrar aquellos que le son irrelevantes. El formalismo computacional más básico, utilizado ampliamente en el control autónomo de los actores 3D es el de las **máquinas de estados finitos**. Estos autómatas codifiquen la actuación del agente como un grafo de estados junto con las condiciones que provocan el cambio de un estado a otro. El sistema multiagente IMPROV [Perlin and Goldberg, 1996] es un ejemplo clásico de sistema de animación comportamental de personajes 3D que sigue esta filosofía. Asimismo, algunos humanos virtuales desarrollados por el matrimonio Thalmann, en los laboratorios *MIRALab* de la *Universidad de Gènova* y *VRLAB* de la *Universidad de Laussane*, han utilizado las máquinas de estados para controlar su comportamiento reactivo [Molet et al., 1997]. Los videojuegos han utilizado el término *Non-Player Character* (NPC) para referirse a las entidades que interpretan algún rol en el mundo ficticio del juego pero no que no están controladas por el jugador. Tradicionalmente, el comportamiento reactivo de los NPCs ha sido preprogramado en máquinas de estados. Por ejemplo, la figura 2.7 muestra el plan de ataque de un *bot* en Quake II. La principal ventaja de las máquinas de estados es la sencillez de desarrollo y la eficiencia de ejecución, ya que todos los planes de actuación están especificados de antemano. Sin embargo, su uso supone ciertas limitaciones de flexibilidad y escalabilidad. Por una parte, como el comportamiento del agente se encuentra imbricado en los estados de la máquina, la introducción de cambios de comportamiento produce, frecuentemente, que el grafo de estados tenga que ser remozado completamente. Por otra parte, a medida que crece la complejidad de los comportamientos, las máquinas de estados se hacen grandes y son más difíciles de mantener.

Una aproximación muy relacionada, utilizada en ocasiones para representar autómatas de estados finitos, son los **sistemas basados en reglas** (*Rule Based Systems* o RBS). Este paradigma de programación define el comportamiento de un agente a través de un conjunto de reglas. En este contexto, los cambios percibidos en el estado del mundo su-

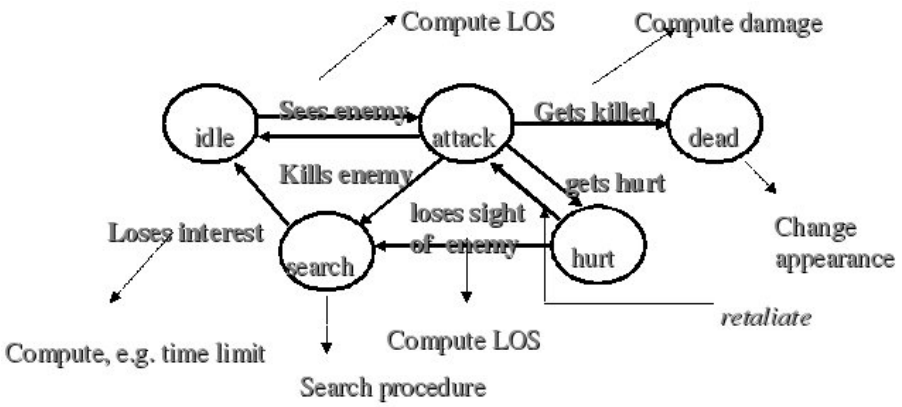


*Figura 2.7: Máquina de estados finitos que define el plan de ataque de un NPC en Quake II.*

ponen la activación de aquellas reglas que son aplicables. De esta manera, la animación del comportamiento consiste en decidir qué regla lanzar (resolución de conflictos entre reglas) y en gestionar las nuevas reglas que se activarán como resultado. Así pues, los RBS permiten la manipulación flexible de un gran número de reglas y, por tanto, la definición de comportamientos más sofisticados que los conseguidos con los autómatas de estados finitos. Los agentes reactivos basados en reglas que han sido referencia dentro de los mundos simulados son los *Boids*; introducidos por Reynolds para la generación de comportamientos de navegación propios de especies animales como los bancos de peces [Reynolds, 1987, Reynolds, 1999] (ver la figura 2.8b). Este sistema se basa en el uso de reglas numéricas para gobernar fundamentalmente reacciones motoras de atracción, de repulsión y de alineamiento con el resto del grupo (ver la figura 2.8a). No obstante, mediante el uso de reglas simbólicas, los RBS pueden mostrar una riqueza de comportamiento más cercana al razonamiento proactivo de los humanos [Laird et al., 1987]. Para una discusión más profunda alrededor del uso de los sistemas basados en reglas simbólicas para la animación comportamental de agentes 3D proactivos ver la sección 2.4.2.

Los mundos virtuales 3D habitados son escenarios altamente variables en los que las propiedades de cualquier objeto pueden cambiar de forma impredecible, a causa de la actuación de otros personajes o por su propia naturaleza dinámica. Todo y que los sistemas reactivos han sido utilizados en la generación de comportamientos en entornos dinámicos, no suelen incorporar ningún mecanismo de búsqueda de acciones futuras. Por contra, el comportamiento proactivo de un personaje autónomo es aquel que busca cuál es la mejor secuencia de acciones para conseguir unos objetivos propios. A causa del dinamismo de los entornos virtuales, la toma de decisiones de los agentes proactivos se realiza mediante

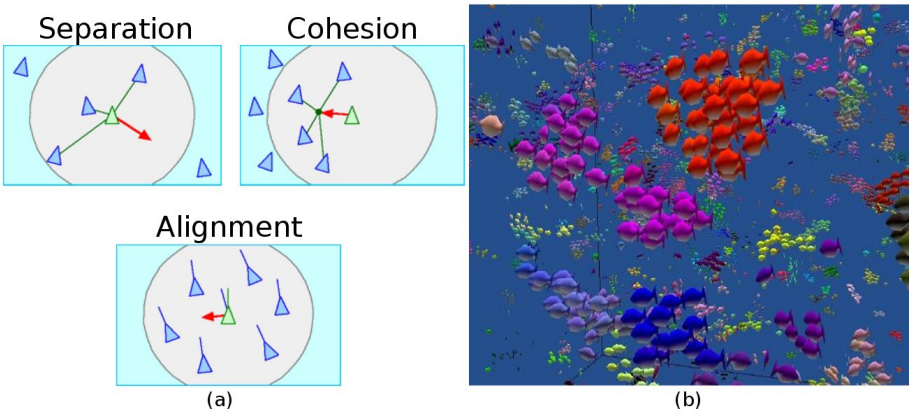


*Figura 2.8: Boids: (a) Modelo básico [Reynolds, 1987] y (b) aplicación a un banco de peces virtuales.*

sistemas de **selección dinámica de acciones**. En el siguiente punto estudiamos con detalle los formalismos de selección dinámica de acciones utilizados con más frecuencia por los 3DIVA proactivos.

## 2.4. MECANISMOS DE SELECCIÓN DINÁMICA DE ACCIONES

Las técnicas de animación comportamental más comunes basadas en la selección dinámica de acciones, se dividen fundamentalmente en dos grupos: la planificación de tareas y los sistemas basados en reglas. Por un lado, el primero trata la construcción de un plan o secuencia de tareas que conduzca al agente desde un estado inicial hacia un estado objetivo. Por otro lado, el segundo tiene los planes precompilados en un conjunto de reglas y se preocupa de su lanzamiento con el objetivo de convertir el estado del mundo actual en un estado deseado. Consideramos ambas aproximaciones válidas, extensamente utilizadas y a menudo complementarias. Por ejemplo, es posible combinar la planificación para la secuenciación de tareas en el bajo nivel y el modelo BDI para el mantenimiento de los objetivos de más alto nivel.

### 2.4.1. Planificación de tareas

La representación clásica de la planificación de tareas describe un problema en términos de predicados, los cuales representan aseveraciones sobre el estado del mundo

[2]. Las acciones posibles se definen mediante *operadores*, que hacen servir las nociones de *precondiciones* y *efectos*; que describen respectivamente cuándo es factible realizarlas y cuáles serán sus consecuencias sobre el estado del mundo. Así pues, el objetivo de la planificación es encontrar una secuencia de acciones que transforme el estado inicial del sistema en un estado objetivo deseado. Esta definición proviene de los años setenta, cuando Fikes y Nilsson describieron el **sistema de planificación STRIPS** [Fikes and Nilsson, 1971]. Des de entonces, muchos planificadores han estados basados en el modelo STRIPS, que ha sido extendido para permitir cuantificadores, efectos condicionales, aritmética básica, incertidumbre, planificación temporal y muchas otras funcionalidades. Asimismo, diversas familias de algoritmos se han desarrollado, entre las más notables: Graphplan [Blum and Furst, 1997], FastForward [Hoffmann and Nebel, 2001], *Heuristic Search Planning* [Bonet and Geffner, 2001], etc.

La planificación de tareas ha sido un asunto muy estudiado desde el campo de la inteligencia artificial y se ha utilizado con éxito en robótica, control de procesos industriales, control aeronáutico, etc. No obstante, su aplicación a los EV3D es muy escasa. Con seguridad, una causa de ello la encontramos en su **alta complejidad**, la cual aumenta de forma exponencial con la **cantidad de información** que se ha de procesar durante el cálculo del plan. De cara a reducir la carga del planificador, han habido intentos de **simplificación** del problema. Por ejemplo, es posible eliminar los hechos no relevantes para el problema del que se busca solución en una fase de pre-procesamiento [Nebel et al., 1997]. Por otra parte, debido a la alta **variabilidad del entorno**, la creación *off-line* de un plan completo de actuación para un actor sintético es a menudo de poca utilidad; ya que este plan no será aplicable en poco tiempo. En consecuencia, se ha propuesto el intercalado de las fases de percepción-planificación-acción; modelo que suele adjetivarse como *continous planning*, *situated planning* o *on line planning* [Avradinis et al., 2005].

Una buena aproximación para reducir la complejidad comentada previamente es el uso de sistemas de **planificación jerárquica**. Este modelo genera planes con un grado de especificación general, que son refinados con los detalles de ejecución por los agentes responsables de las tareas. *SodaJack* [Geib et al., 1994] (ver la figura 2.9) es un ejemplo de uso de la planificación jerárquica en 3DIVA. Fue desarrollado por el grupo dirigido por Norman I. Badler en la Universidad de Pennsylvania y aplicado a en un entorno simple en que el humanoide *Jack* sirve productos a los visitantes. *Jack* es dirigido por un sistema

[2]Con frecuencia, de acuerdo con la presunción de mundo cerrado, todo aquello que no se hace explícito con un predicado se considera falso [Nilsson, 2001]

de planificación jerárquica que utiliza tres planificadores diferentes: *ItPlans* como planificador de alto nivel, un planificador (*OSR*) especializado en la localización de objetos y otro planificador específico para la manipulación de los objetos. *ItPlans* trabaja de una manera iterativa, es decir, produce y expande los planes dependiendo de los resultados de las acciones previas y del estado actual del entorno. Por ejemplo, si *Jack* ha de servir un helado, primero tiene que conseguir una tarrina. Supongamos que las tarrinas se pueden encontrar en diferentes lugares: el armario o el mostrado. Entonces, *ItPlans* genera primeramente la tarea de buscar la tarrina en el armario. Si hay tarrinas, el plan avanza hacia el objetivo final. Pero si no hay, se generará una nueva búsqueda en el mostrador. La ventaja es que el sistema puede responder rápidamente, aspecto vital para la animación de comportamientos en tiempo real. Ahora bien, la principal desventaja es que el plan separa el objetivo final en subobjetivos independientes y los resuelve uno tras de otro, cosa que provoca comportamientos pobres. Por ejemplo, el agente busca la tarrina en el armario pero después se da cuenta de que también necesita una cucharilla. Entonces, vuelve a comenzar la búsqueda secuencial de nuevo en vez de haber extraído todos los objetos que necesita al mismo tiempo.

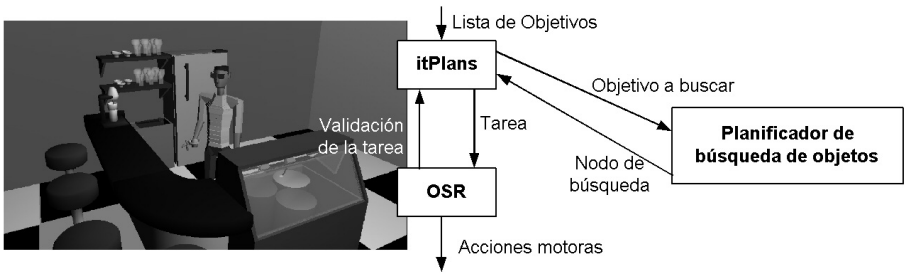


*Figura 2.9: Planificación jerárquic utilizada en el entorno SodaJack [Geib et al., 1994].*

Otro tipo de planificación jerárquica lo encontramos en los generadores de historias interactivas (*interactive storytelling*). Estos sistemas están designados a generar historias dentro de EV3D a partir del comportamiento del usuario (que interacciona de forma impredecible con los objetos 3D o pasa información a los personajes) y de los personajes autónomos. Diversos proyectos han utilizado diferentes formalismos de la IA para mejorar la riqueza y variabilidad de los hilos narrativos, por ejemplo: The Oz Project [Bates, 1992], Mimesis [Young, 2001] y Virtual Storytelling [Cavazza et al., 2002]. Éste último ha utilizado una técnica conocida como **redes jerárquicas de tareas** (*Hierarchical Task Networks* o HTN). Las HTN son un mecanismo por medio del cual el creador de la historia define todas las posibilidades que se encuentran al alcance de los personajes 3D durante una primera fase conocida como *authoring*. Como resultado de esta fase, queda

definido el conjunto de planes (parcialmente ordenados) que los personajes han de llevar a cabo (mediante árboles jerárquicos de tareas como el de la figura 2.10). De esta manera, en tiempo de ejecución los agentes se dedicarán a extraer uno de los posibles planes para monitorizarlo posteriormente. Sin embargo, la simplificación de la actuación de un agente en tareas independientes con un orden parcial predefinido limita la toma de decisiones de los personales y resta variabilidad y flexibilidad a sus comportamientos.

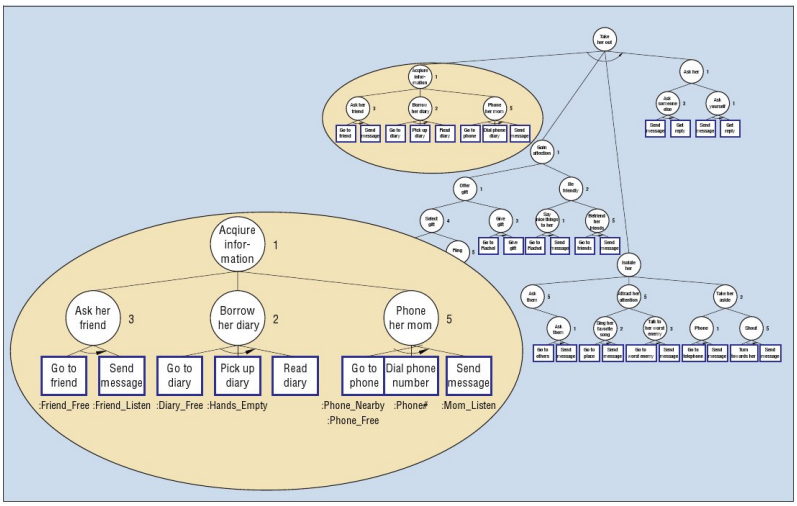


*Figura 2.10: Descomposición de las tareas que describen completamente la actuación de un actor mediante HTN [Cavazza et al., 2002].*

Un ejemplo similar de simplificación del proceso de planificación, basado en el cálculo de situaciones, es el propuesto por el **lenguaje de modelado cognitivo** (*Cognitive Modelling Language* o CML) [Funge et al., 1999]. CML ayuda el diseñador de una animación a dar instrucciones a los actores por medio de guiones prediseñados, además, permite describir el conocimiento que el carácter tiene sobre el mundo mediante las acciones que puede realizar (ver la figura 2.11a). Este esquema se represente típicamente en forma de árbol de planificación donde los nodos representan situaciones (ver la figura 2.11b). La raíz es la situación inicial y cada camino representa una secuencia de acciones. A pesar de que las precondiciones de las acciones pueden hacer que ciertas secuencias no sean posibles (nodos negros del árbol), la búsqueda de una secuencia para llegar a una situación o nodo objetivo todavía será exponencial. Para agilizar el proceso, CML no utiliza algoritmos sofisticados de búsqueda como A* sino que hace una poda de esta árbol mediante *acciones complejas*. En este contexto, las acciones complejas son simplemente reglas sobre ejecuciones de acciones no permitidas (nodos marrones del árbol). Esta técnica fue usada para la obtención de comportamientos cognitivos dentro de los mundos submarinos de Tu y Terzopoulos [Tu and Terzopoulos, 1994], que ya habían integrado

con éxito otros comportamientos físicos y reactivos.

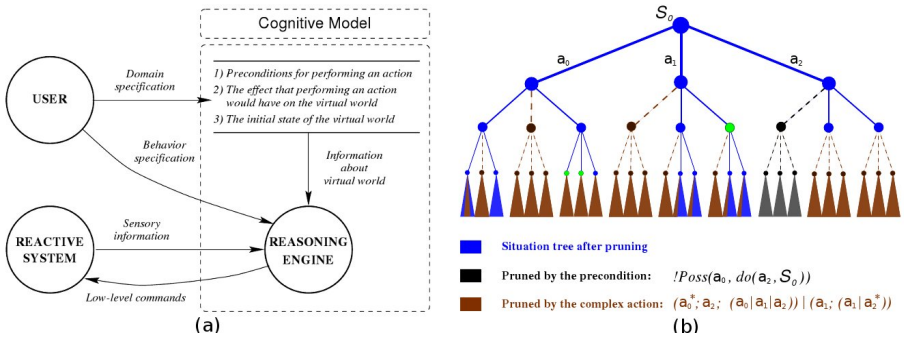


*Figura 2.11: Modelado cognitivo: (a) esquema general del modelo y (b) árbol de planificación [Funge et al., 1999].*

En los últimos años, las **técnicas de planificación heurística** (*Heuristic Search Planning* o HSP) [Bonet and Geffner, 2001] también han sido propuestas como aproximaciones muy interesantes a la hora de animar comportamientos de personajes 3D en entornos altamente cambiantes [Lozano, 2005, Ciger, 2005]. Por oposición a los sistemas donde el autor define el perfil argumental de la historia (conocidos como *top-down*, en la animación con HSP el actor construye su plan de forma proactiva; cosa que hace que se les caracterice como sistemas *bottom-up*. Este hecho, junto con el intercalado de las fases de decisión y de ejecución de acciones, hace que estas soluciones se adapten mejor a los cambios que se producen en el entorno. Asimismo, la generación automática del plan hace que los actores puedan tratar situaciones que, a pesar de tener importancia narrativa, no fueron contempladas por el autor en su guión original (p. ej. cuando dos actores sintéticos que son amigos se encuentran de manera inesperada en un espacio virtual). De acuerdo con esto, dado un conjunto de operadores que definen cada una de las posibles acciones que un carácter puede realizar (figura 2.12a), la toma de decisiones consiste en buscar la secuencia de acciones que permite llegar a un estado objetivo. Por ejemplo, la figura 2.12b muestra el árbol con las posibilidades al alcance de un personaje 3D que tiene el propósito de preparar una cita: envolver un regalo, limpiar la casa, darse una ducha, hacer la cena... Las decisiones adoptadas corresponderán con el plan de longitud mínima, que evite la repetición (en cierta manera inverosímil) de acciones innecesarias como el hecho de lavarse las manos cada vez que realiza una acción. Para guiar el algoritmo de búsqueda a lo largo de este árbol y garantizar una solución aceptable en un tiempo reducido, los HSP utilizan las funciones heurísticas. El uso de heurísticas independientes del dominio proporciona robustez a estos métodos, que pueden funcionar correctamente en distintos escenarios 3D y para diversos roles (p. ej. un vendedor virtual, un camarero, etc.) [Lozano, 2005].

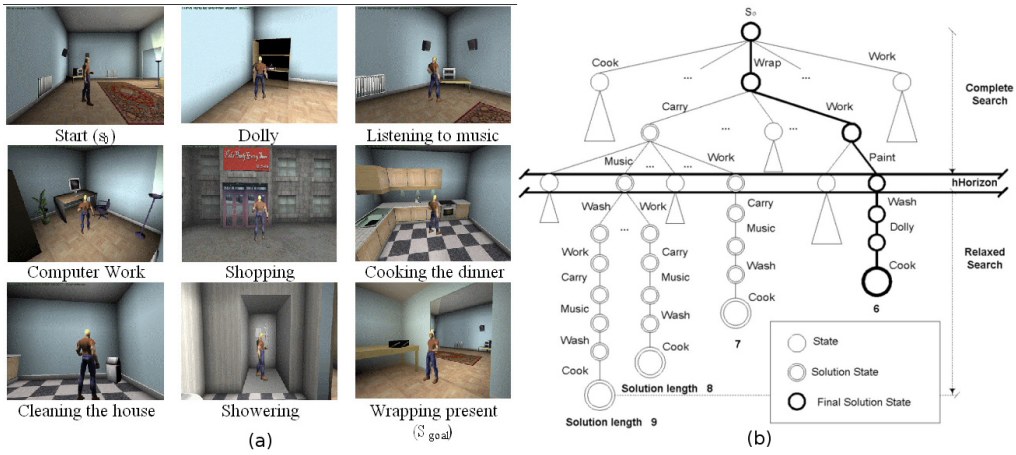


*Figura 2.12: Técnica de planificación heurística: (a) Acciones posibles para a un personaje 3D, (b) Árbol de decisión HSP [Lozano, 2005].*

En el capítulo 3 retomaremos la discusión sobre este tipo de métodos de animación comportamental para la toma de decisiones en entornos multiagente. Como parte de las aportaciones de esta tesis, en el capítulo 5 propondremos una técnica para conseguir comportamientos colaborativos en grupos de agentes basados en planificadores heurísticos.

### 2.4.2. Sistemas basados en reglas

Los sistemas basados en reglas, o sistemas de producción, son un paradigma de programación adecuado para el control de sistemas que requieren declaratividad, modularidad, y técnicas de exploración; como es el caso de los personajes 3D inteligentes. Como ya se comentó en la sección 2.3.2, estos sistemas utilizan las reglas como mecanismo de representación del conocimiento. De manera general, una regla de producción tiene el siguiente formato: **SI** *antecedente* **ENTONCES** *consecuente*. El antecedente es una condición que se ha de cumplir para que la regla sea aplicable y el consecuente puede ser interpretado como la conclusión a la que se llega o la acción que se debe realizar. En este contexto, el razonamiento, llevado a cabo por un motor de inferencia, puede ser de dos tipos: hacia adelante (*forward chaining*) o hacia atrás (*backward chaining*). El razonamiento hacia adelante parte de los hechos que definen el estado actual y explora las reglas para extraer las conclusiones, siendo un ejemplo de razonamiento desde los datos hacia los objetivos. Por el contrario, el razonamiento hacia atrás va desde los objetivos hacia los datos, es decir, a partir de las conclusiones que se buscan, busca los hechos que las producen. Un aspecto que han de resolver ambos tipos de razonamiento son los conflictos

entre las reglas, producidos bien porque los consecuentes son contradictorios o porque hay más de una regla que se puede aplicar. Las soluciones van desde la asociación de un valor numérico a cada regla, que exprese su preferencia o probabilidad, hasta la definición de meta-reglas que especifiquen el orden que se debe aplicar cuando hay diversas reglas posibles.

Un ejemplo bien conocido de sistema basado en reglas es SOAR [Laird et al., 1987], desarrollado por Laird en la Universidad de Michigan. SOAR permite el encadenamiento bidireccional de reglas (*forward/backward chaining*) y la resolución de conflictos mediante la definición de meta-reglas. Es capaz de razonar sobre las reglas (acciones a ejecutar) con el fin de identificar la más adecuada y es capaz de generar automáticamente subobjetivos para resolver objetivos más grandes que no se pueden resolver directamente. SOAR se ha utilizado en el campo de los videojuegos para simular NPC controlados por el ordenador. Por ejemplo, la figura 2.13a muestra el esquema propuesto por el SOAR-Bot [Laird and Duchi, 2000], probado sobre Quake II, Unreal Tournament i Descent III, y que implementa aproximadamente 800 reglas de comportamiento dedicadas a proporcionar inteligencia táctica. El actor pedagógico Steve [Elliott et al., 1999] también utiliza SOAR para monitorizar de manera reactiva las tareas del alumno (ver la figura 2.13b).

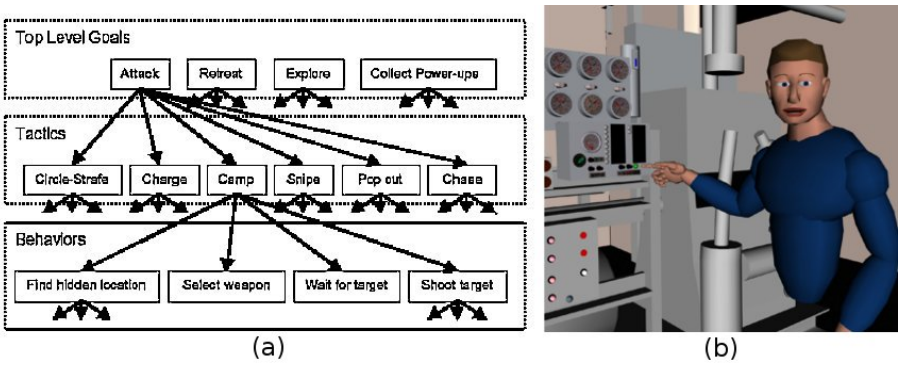


*Figura 2.13: Ejemplos que utilizan SOAR: (a) Esquema de operadores utilizado por el SOARBot y (b) Steve [Elliott et al., 1999].*

Un modelo para agentes inteligentes que utiliza, típicamente, los sistemas basados en reglas es el **modelo BDI**; introducido originariamente por Bratman el año 1978 [Bratman, 1987]. Este paradigma postula que el comportamiento de un agente autónomo puede ser controlado atendiendo a tres conceptos: sus creencias (*Beliefs*), sus deseos (*Desires*) y sus intenciones (*Intentions*). Las creencias de un agente expresan su conocimiento sobre el mundo y sobre su estado interno. Las creencias pueden también incluir reglas de inferencia que permitan la creación de nuevas creencias mediante razonamiento hacia adelante

(p. ej. si un objeto está ardiendo entonces quema). Los deseos representan las motivaciones del agente. Son los objetivos de alto nivel que el agente persigue, como por ejemplo el deseo de no tener hambre o de ir a una fiesta. Finalmente, las intenciones representan el estado deliberativo del agente, es decir, aquello que el agente ha decidido realizar. Esto es, las intenciones son deseos para la consecución de los cuales el agente ha comenzado a ejecutar algún plan. Un plan es una secuencia de acciones que el agente puede ejecutar con vistas a conseguir una o más de sus intenciones. Sucesivamente, los planes pueden estar formados por otros planes; por ejemplo, un plan para dar una vuelta en coche puede incluir un subplan para buscar las llaves del coche.

Normalmente, el modelo BDI usa una librería de planes creados previamente para que un agente pueda enfrentarse a las diversas situaciones que le pueden aparecer a lo largo de su vida. Agentes BDI que usen la planificación de tareas en vez de librerías de planes precompilados también son posibles [Meneguzzi et al., 2004], pero no son comunes. La desventaja de las técnicas que usan **librerías de planes preconstruidos** es su fuerte dependencia con el dominio para el cual han sido desarrolladas. Sin embargo, su sencillez de desarrollo y su rapidez de decisión han hecho del modelo BDI una aproximación que ha sido utilizada ampliamente a la hora de implementar 3DIVA. Algunos ejemplos se encuentran en los contrincantes reactivos de los videojuegos, como el agente *KGBot* [Kim, 2003] de *Unreal Tournament* o los NPCs del juego *Black & White* [Molyneux, 2001]. El modelo BDI ha inspirado desde las arquitecturas de agentes individuales racionales [Caicedo and Thalmann, 2000] (ver la figura 2.14) hasta el movimiento de muchedumbres de agentes (*crowds*) [Raupp and Thalmann, 2001].

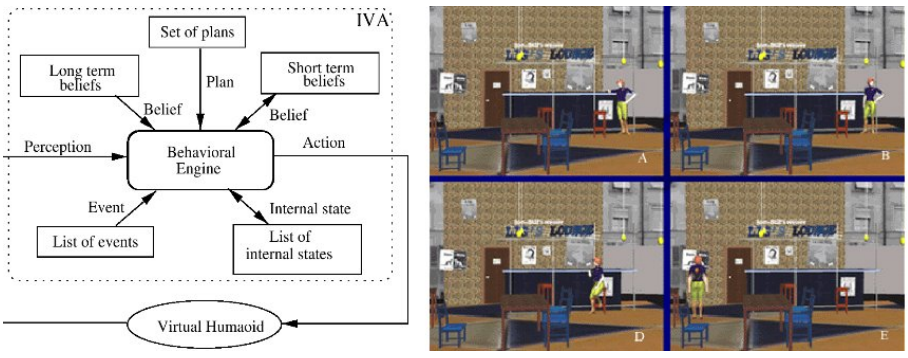


*Figura 2.14: Modelo BDI utilizado en [Caicedo and Thalmann, 2000] para la animación de un cliente en un bar virtual.*

El modelo BDI se ha utilizado extensamente como el paradigma base para el diseño de sistemas multiagente de propósito general (p. ej. Jason [Bordini and Hübner, 2007] o Ja-

dex [Distributed Systems and Information Systems Group - University of Hamburg, 2008]). En el capítulo 5 presentaremos nuestra aportación social en este contexto: **MADeM (Multi-modal Agent Decision Making)**. MADeM es un proceso de toma de decisiones de tipo social que puede ser utilizado por agentes BDI para tomar decisiones socialmente aceptables en entornos virtuales 3D.

## 2.5. CONCLUSIONES

La tarea de poblar los mundos virtuales 3D con personajes autónomos es un requisito para alcanzar la deseada inmersión visual y mental del usuario. Sin embargo, este objetivo se ha convertido en una tarea demasiado compleja debido a la gran cantidad de aspectos que se han de tener en cuenta. De una parte, un aspecto que ha demandado una especial atención ha sido el de la representación de la información, en unos entornos tan informados como son los EV3D. La toma de decisiones de los humanos virtuales para interaccionar tanto con los objetos como con el resto de agentes conlleva procesos cognitivos complejos que requieren un conocimiento abstracto o simbólico de los elementos del entorno. De acuerdo con esto, el uso de información semántica ha demostrado ser beneficioso para la representación del entorno, de las tareas y de las relaciones entre los agentes. De otra parte, el uso de técnicas provenientes de la IA ha permitido la evolución desde los actores con una buena fidelidad locomotriz a los agentes que incorporan comportamientos racionales. Tal vez uno de los aspectos más deseados haya sido la reproducción de una toma de decisiones coherente con el estado del entorno y con los deseos del personaje. Por ello, los actores 3D han utilizado un amplio abanico de mecanismos para la reproducción de comportamientos reactivos y proactivos.

En el siguiente capítulo centramos nuestro discurso en los **agentes sociales**, agentes que han sido escasamente tratados en la literatura de los 3DIVA. Las dificultades a las que se ha enfrentado la animación comportamental de un actor individual han hecho que los personajes 3D hayan dejado de lado a menudo su comportamiento social. Sin embargo, de la misma manera que la IA ha permitido evolucionar desde los agentes guiados a los 3DIVA, conviene incorporar el conocimiento proveniente del campo de los sistemas multiagente y de la sociología para mejorar el comportamiento de las sociedades artificiales de actores sintéticos.

CAPÍTULO 3

# SOCIEDADES ARTIFICIALES

En este capítulo estudiamos el comportamiento social de los actores sintéticos y centramos la atención en la **interacción** entre los personajes autónomos. La inclusión de habilidades sociales en la toma de decisiones de los caracteres es un aspecto de vital importancia en los entornos virtuales habitados, ya que los humanoides conforman una especie de sociedad artificial. En este contexto, los sistemas multiagente proporcionan un marco de trabajo elegante y formal para el diseño de comportamientos sociales para personajes 3D. En consecuencia, repasamos los mecanismos de interacción multiagente utilizados más a menudo para satisfacer los objetivos de un individuo en contextos poblados por otras entidades autónomas. Analizamos asimismo las redes sociales, ya que son soluciones que van más allá de la racionalidad individual y incluyen modelos de organización que son interesantes para alcanzar los objetivos globales de una sociedad.

## 3.1. COMPORTAMIENTO SOCIAL EN PERSONAJES 3D

Con frecuencia, los actores sintéticos no están solos en el entorno, sino que comparten un espacio común de actuación con otros personajes con los que forman una especie de sociedad artificial. Entonces, sus comportamientos pueden beneficiarse de la **inclusión de habilidades sociales**, cosa que hará que los llamemos **agentes sociales**. La adición de comportamiento social no es una tarea sencilla ya que afecta a distintos niveles de actuación (p. ej. desde la capacidad de conversar con realismo a la modificación de la actuación para atender órdenes externas). A continuación, mostramos un conjunto representativo de comportamientos sociales que han sido aplicados sobre personajes 3D. Primeramente, introducimos los caracteres dotados de la capacidad de conversar con un usuario humano.

En segundo lugar, hablamos del establecimiento de diálogos y la interacción entre dos caracteres autónomos. En tercer lugar, mostramos algunas soluciones propuestas para conseguir comportamientos sociales en grupos de personajes. Finalmente, comentaremos como el comportamiento social se ha escalado en la generación de movimientos verosímiles de aglomeraciones de actores virtuales.

una de las expresiones sociales más característica de los seres humanos es la **capacidad de conversar**. Así pues, se han realizado muchos esfuerzos para dotar a los humanoides virtuales de esta habilidad[1]. Tal vez uno de los primeros ejemplos fue el trabajo en caracteres interactivos inteligentes (*Smart Interactive Characters*) que se realizó en la *Universidad de Stanford* bajo la supervisión de Barbara Hayes-Roth [Hayes-Roth, 2008]. Este trabajo tenía como objetivo la construcción de *mentes* para caracteres virtuales con diferentes cualidades humanas como por ejemplo: identidad y personalidad; afecto y empatía; conocimiento y experiencia; relaciones sociales y evolución de las mismas; etc. Los personajes implementados fueron principalmente asistentes y guías virtuales capaces de mantener una conversación en lenguaje natural con un usuario humano para que este dirigiera su comportamiento. En esta línea se encuentran también los humanos virtuales desarrollados por el laboratorio *MIRALab* de la *Universidad de Génova* [Pandzic et al., 1998]. Estos humanoides se comunican a través de una interfaz de texto y un analizador de mensajes. No obstante, no muestran una interacción autónoma entre diferentes actores virtuales, ya que los diálogos sólo se establecen entre un actor sintético y el usuario humano. La animación de diálogos entre individuos autónomos también ha sido tratada por los agentes conversacionales [Cassell et al., 1994, Bickmore and Cassell, 2001]. Sin embargo, estos actores dedican especial atención a la simulación gestual, mientras que la conversación se genera automáticamente por un planificador simple y normalmente incapaz de razonar sobre la información que está siendo comunicada.

El campo de aplicación del drama interactivo (*interactive drama*) ha utilizado acciones predefinidas para incorporar comportamientos sociales realistas que sí atienden a la información intercambiada. La metodología de W. Scott Neal [Reilly, 1996] para la creación de comportamiento social defiende que la personalidad de un actor influye de forma tan determinante en su comportamiento que los comportamientos universales no son posibles de crear. Por tanto, proporciona un conjunto de mecanismos para ayudar al artista en la creación de comportamientos sociales particulares para caracteres específicos en entornos concretos. De manera similar, los generadores de historias interactivas (*interactive*

[1]Persiguiendo, así, un objetivo similar al formulado por el antiguo test de Turing.

*storytelling*) animan las interacciones entre los personajes de la escena a raíz de la trama (*storyline*) global definida por el autor de la historia [Cavazza et al., 2002].

Los proyectos comentados previamente incluían la animación de pocos individuos (normalmente, uno o dos), sin embargo, el comportamiento social es todavía más adecuado cuando se habla de **grupos de agentes**. La etología ha inspirado algunos sistemas de animación de grupos de animales artificiales [Tomlinson and Blumberg, 2002, Delgado-Mata and Aylett, 2004]. En este contexto, la transmisión de emociones mediante un modelo de feromonas ha sido un mecanismo probado con éxito para influenciar el comportamiento del resto de animales del rebaño. La búsqueda de humanoides virtuales también ha desarrollado modelos para conseguir comportamientos verosímiles a nivel de grupo. Por ejemplo el sistema *Avatar Arena* [Schmitt and Rist, 2003] define un modelo de dinámica de grupo para el establecimiento de negociaciones en grupos reducidos. En este modelo, cada uno de los agentes tiene una personalidad individual y unas relaciones de atracción social hacia los otros. Estas características guían las interacciones de los agentes y la generación de los diálogos de negociación entre caracteres.

En los últimos tiempos, los mundos virtuales se han convertido en sistemas donde múltiples usuarios y personajes autónomos comparten un entorno de actuación común [Bioware, 2003, Linden Lab, 2008]. Estos escenarios interactivos a menudo presentan tareas que han de ser resueltas por los participantes de manera colaborativa. El modelo SGD (*Synthetic Group Dynamics*) [Prada and Paiva, 2005] es un ejemplo, basado en teorías socio-psicológicas, que ha sido implementado en grupos de actores sintéticos que colaboran con el usuario en la resolución de ciertas tareas dentro de un entorno virtual (ver figura 3.1a). Por lo que respecta a la colaboración entre personajes autónomos, el trabajo doctoral de Jan Ciger [Ciger, 2005] utiliza un tipo especial de acciones llamadas acciones delegadas (*delegated actions*). Por medio de estas acciones, un humano virtual puede pasar la ejecución de una acción a otro agente, siempre que él no la pueda realizar. Por ejemplo, en la figura 3.1b un agente abre la puerta a otro, que está ocupado arrastrando una caja.

El comportamiento social también ha sido incluido en la animación de **muchedumbres** de agentes virtuales (*crowds*). La primera aproximación en este campo fueron los *Boids* desarrollados por Reynolds [Reynolds, 1987]. Estos agentes percibían sus alrededores y reaccionaban a un conjunto de reglas sencillas. Como resultado, surgía un movimiento de grupo que reproducía patrones sociales verosímiles (p. ej. el vuelo de una


*Figura 3.1: Personajes 3D que colaboran (a) con el usuario humano [Prada and Paiva, 2005] y (b) con otro carácter autónomo [Ciger, 2005].*

bandada de pájaros). Benford [Benford et al., 1997] propuso un modelo que utilizaba la comunicación con un objeto especial llamado *Third Party Object* y permitía, entonces, la creación y la separación de grupos de personajes virtuales. Por último, el modelo local de fuerzas sociales introducido por Helbing [Helbing et al., 2005] ha inspirado sistemas como HiDAC [Pelechano et al., 2007], con el que se pueden obtener comportamientos emergentes como la formación de filas de agentes o atascos en zonas de mucha densidad. Sin embargo, los personajes que encontramos generalmente en las *crowds* no tienen una concepción social profunda y carecen de las capacidades de construcción de relaciones sociales. Es decir, son adecuados para la simulación de movimientos realistas a nivel macroscópico pero su comportamiento social a nivel de individuo será muy limitado.

Como se desprende de los ejemplos anteriores, la inclusión de comportamientos sociales en personajes 3D no es un problema nuevo. El reto actual, al que nos enfrentamos en esta tesis, consiste en juntar los modelos sociales con los modelos de toma de decisiones racionales (presentados en 2.4), de manera que se puedan obtener actores sintéticos autónomos con ambas capacidades: racionalidad y sociabilidad. Este tipo de personajes son requeridos por los entornos virtuales habitados como por ejemplo los simuladores civiles o militares, los juegos por ordenador y los mundos virtuales en red de última generación.

El comportamiento racional y social ha sido un tema tratado con gran detalle por la comunidad de agentes. Conviene analizar la investigación llevada a cabo en este campo para estimar qué soluciones pueden ser interesantes a la hora de generar **actores sintéticos socialmente inteligentes**. De forma general, los agentes socialmente inteligentes se han

definido como aquellos entes capaces de resolver problemas de forma autónoma y de alcanzar sus objetivos, si es necesario, por medio de la interacción con otras entidades igualmente autónomas [Hogg and Jennings, 2001]. Los sistemas multiagente, referidos a menudo como *sociedades de agentes*, proporcionan un marco de trabajo elegante y formal para el diseño de comportamientos racionales y sociales para personajes 3D.

## 3.2. SISTEMAS MULTIAGENTE

Un agente es una entidad física o virtual capaz de actuar, de percibir el entorno y de comunicarse con otros agentes. Es autónomo, tiene habilidades para alcanzar sus objetivos y se encuentra localizado a menudo dentro de un sistema multiagente (*Multi-Agent System* o MAS) [Ferber, 1999]. Se ha definido un MAS como una red débilmente acoplada de agentes, los cuales interaccionan para resolver problemas que se encuentran más allá de sus capacidades o de su conocimiento individuales [Sycara, 1998]. De una manera més formal, podemos describir un MAS con la *tupla* $< E, O, A, R, Op, L >$, donde:

- $E$ - es un entorno dimensional.
- $O$ - es un conjunto de objetos situados en $E$.
- $A$ - es un conjunto de agentes que cumple que $A \subseteq O$.
- $R$ - es un conjunto de relaciones que enlazan los objetos y los agentes de $O$.
- $Op$ - es un conjunto de operaciones que permiten a los agentes percibir, producir, consumir, transformar y manipular los objetos.
- $L$ - es un conjunto de leyes universales que representan el efecto de las operaciones en el mundo.

La figura 3.2a muestra una visión general de la estructura de un sistema multiagente. En la parte inferior encontramos el entorno compartido que los agentes ocupan. Cada uno de los agentes tiene una esfera de influencia en este entorno, esto es, una porción que puede controlar total o parcialmente. Cuando el entorno se controla de forma conjunta con otros agentes, las esferas se solapan. Esta intersección complica la vida de los agentes, ya que para conseguir el estado del mundo deseado, los agentes han de tener en cuenta

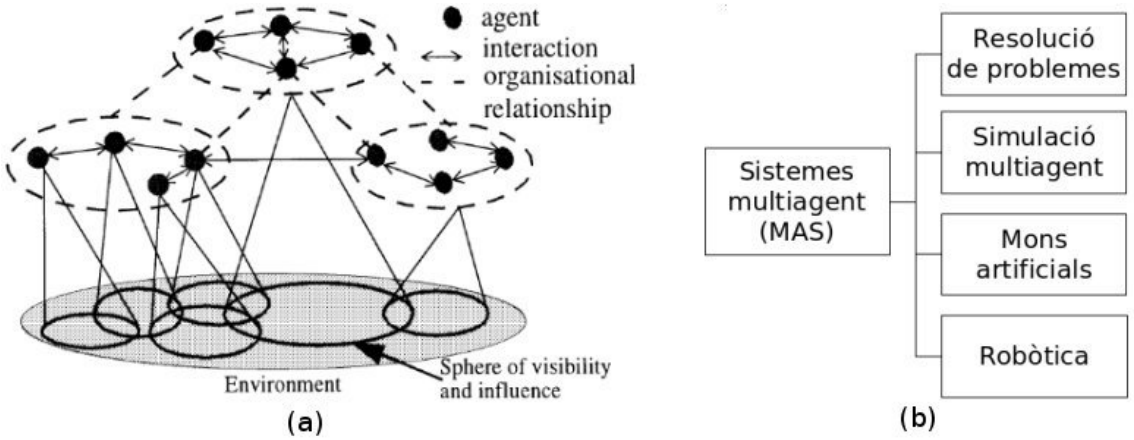


*Figura 3.2: Sistemas multiagente: (a) visión general y (b) áreas de aplicación [Ferber, 1999].*

también la actuación de otros agentes. La parte superior de la figura muestra las interacciones que se producen entre los agentes así como las interacciones organizacionales que los ligan (p. ej. un agente puede se compañero de otro o tener algún tipo de autoridad sobre otro). De acuerdo con la figura 3.2b, la aplicación de los sistemas multiagente abarca principalmente cuatro áreas:

**Resolución de problemas:** El uso de un MAS puede ser una buena alternativa a la resolución centralizada de problemas; bien porque la naturaleza del problema sea distribuida o bien porque la distribución entre un conjunto de agentes suponga una forma más eficiente de organizar la resolución. Algunos ejemplos son la monitorización distribuida de una red, las aplicaciones de diseño industrial o los problemas basados en restricciones.

**Simulación multiagente:** Los MAS permiten crear dominios artificiales que actúan como pequeños laboratorios en los que probar teorías sobre comportamientos locales. La simulación multiagente se utiliza hoy en día para mejorar el conocimiento en campos diversos como las ciencias sociales, la educación, la biología...

**Mundos artificiales:** Los cuales pueden ser utilizados para describir mecanismos de interacción específicos y analizar su impacto en el sistema a nivel global. De manera análoga al juego de la vida (*Game of Life*) [Gardner, 1970], el objetivo de esa área es la construcción de sociedades de agentes y el estudio de su evolución.

**Robótica:** Un robot puede ser concebido como un MAS en el que cada agente se encar-

ga de un objetivo específico. La consecución conjunta de las tareas más sencillas y concretas hará pues que el objetivo global se complete. Asimismo, los MAS pueden ser usados para la coordinación de diversos robots móviles que se encuentren situados dentro de un espacio común.

Los entornos virtuales 3D habitados encajan, pues, dentro del área de actuación de los MAS. En estos mundos simulados, un conjunto de entidades autónomas tendrán que trabajar de forma conjunta para poder obtener como resultado una especie de comportamiento inteligente a nivel colectivo. De acuerdo con esto, algunos investigadores han optado por desarrollar su propio sistema multiagente, como sucede con los sistemas *mVITAL* [Anastassakis et al., 2001] y *SimHuman* [Vosinakis and Panayiotopoulos, 2001]. En los últimos tiempos, los MAS de propósito general (p. ej. *Jason* [Bordini and Hübner, 2007]) han sido propuestos también como buenas aproximaciones para la animación comportamental de humanos autónomos en entornos tridimensionales [Torres et al., 2003].

La **inteligencia colectiva** que uno desea obtener no puede ser generada con las técnicas tradicionales de la Inteligencia Artificial. Éstas centran la atención en los procesos cognitivos del agente y los tratan normalmente como sistemas independientes; ya que siguen la premisa de que la inteligencia es una propiedad del agente. Sin embargo, los MAS son estudiados por la Inteligencia Artificial Distribuida (*Distributed Artificial Intelligence* o DAI). La DAI considera que, sin dejar de lado las capacidades cognitivas individuales, la inteligencia surge como el resultado de la **interacción** entre los agentes; los cuales forman parte de un sistema interconectado a través de procesos sociales. En consecuencia, la investigación en MAS se ha centrado en la interacción, como base para la comprensión del comportamiento y la evolución de un sistema multiagente. En el siguiente punto identificamos diferentes tipos de interacción y definimos los modelos operacionales a que conducen.

## 3.3. INTERACCIÓN EN SISTEMAS MULTIAGENTE

La investigación alrededor de las interacciones en sistemas multiagente a menudo ha acuñado el comportamiento con adjetivos como *coherente*, *colaborativo*, *cooperativo*, *competitivo* o *coordinado*. Diversas aproximaciones han tratado de organizar y relacionar todos estos términos, a la hora de categorizar con precisión cómo interaccionan los

| | | Relación entre los agentes | |
|---|---|---|---|
| | | Centralizada | Descentralizada |
| Flujo de información | Directo | Orden | Conversación |
| | Indirecto | Restricción | Competición |

*Tabla 3.1: Categorías de comunicación.*

agentes. Tal vez una de las más brillantes de todas es la taxonomía que considera todos estos términos como especializaciones de un concepto abstracto llamado **correlación** [Parunak et al., 2002].

La manera más general de medir como un conjunto de agentes hacen cosas juntos es mediante su información conjunta, también conocida como entropía conjunta o de correlación. La correlación es un término teórico que corresponde con la no-independencia estadística entre los comportamientos de los agentes, sin atender a la estructura interna ni a la organización externa. Fundamentalmente, un agente está correlacionado cuando sus acciones son estadísticamente dependientes de las acciones de los otros. Esta dependencia provendrá de un flujo de información entre un agente individual y el exterior; es decir, el estado del entorno y el resto de los agentes. Este hecho hace que el análisis de la comunicación sea de capital importancia en el estudio de la interacción.

### 3.3.1. Comunicación

El flujo de información entre un agente y el entorno que lo rodea puede transmitirse de dos maneras: directa o indirectamente. Mientras que la primera lleva asociado un intercambio de mensajes entre los agentes, la segunda aprovecha la capacidad comunicativa del mismo entorno, que puede almacenar información para que sea percibida por otros agentes posteriormente. En lo que respecta a la relación entre los agentes, distinguimos dos tipos de comunicación: centralizada y descentralizada. En la primera categoría encontramos las relaciones jerárquicas en las que un agente *jefe* manda sobre un agente *subordinado*. De manera inversa, en la comunicación descentralizada todos los agentes se hablan de igual a igual, en inglés *peer-to-peer*. La tabla 3.1 clasifica diversos **tipos de correlación** dependiendo del modo de comunicación de la información y de las relaciones entre los agentes.

La correlación en una topología centralizada puede ser ejercida por un agente supervisor utilizando órdenes dirigidas a sus subordinados o bien modificando ciertas propiedades del entorno para restringir de forma indirecta su actuación. Un ejemplo de **orden** lo encontramos en un albañil virtual que da órdenes a sus peones en una escena 3D. Por lo que respecta a la **restricción**, el albañil podría dotar a sus peones de un conjunto de herramientas o recursos determinados para que realizaran sus tareas. La comunicación directa en topologías descentralizadas, también llamada **conversación**, ha sido tratada extensamente desde el campo de la investigación en negociación entre agentes inteligentes así como por los actores virtuales conversacionales (ver el punto 3.1). La comunicación indirecta descentralizada se produce cuando los agentes perciben los cambios del entorno a causa de las acciones de los otros agentes. El problema más común dentro de esta categoría es la **competición**, que sucede cuando los agentes quieren acceder a unos recursos limitados y compartidos. La competición, una situación muy habitual en los entornos 3D habitados, demanda el uso de técnicas de coordinación para evitar la interferencia entre las acciones realizadas por los agentes (recordemos los ejemplos del carpintero y del camarero introducidos en el punto 1.1).

### 3.3.2. Coordinación

Uno de los tipos de interacción utilizado más a menudo por la comunidad de los sistemas multiagente es la **coordinación**. Se puede concebir la coordinación como un caso específico de correlación en el que hay se produce un proceso de **comunicación** entre el agente y el contexto que lo rodea. De acuerdo con Bergenty y Ricci [Bergenty and Ricci, 2002], la coordinación en MAS ha utilizado diferentes formalismos que podemos clasificar en las siguientes categorías: centros de tuplas, protocolos de interacción, semántica de los lenguajes de comunicación de agentes. Todas tienen sus ventajas, pero también reciben ciertas críticas.

Los **centros de tuplas** siguen la filosofía de los modelos de pizarra, en los que no se requiere contacto físico ni temporal entre los agentes. Su uso supone una solución no acoplada para la coordinación, ya que la comunicación y el procesado se realiza en los centros de tuplas; es decir, fuera de los agentes. En consecuencia, estos modelos son adecuados en entornos dinámicos y abiertos. Sin embargo, sus críticos argumentan que son sistemas limitados a causa de la centralización y del hecho que frecuentemente sólo trabajan sobre el comportamiento observable de los agentes y no realizan ninguna suposición sobre las

actitudes mentales.

Los **protocolos de interacción** se describen mediante una máquina de estados finitos. Los estados de un protocolo identifiquen los estadios en los que los agentes pueden estar a lo largo de la comunicación y las transiciones representan los mensajes intercambiados entre un emisor y un receptor. Diversos lenguajes de comunicación de agentes (*Agent Communication Languages* o ACL) han sido desarrollados. Los más destacados son KQML [Finin et al., 1994] y FIPA ACL [FIPA (Foundation for Intelligent Physical Agents), 2008] (ver la figura 3.3). Ambos basan la comunicación en un intercambio asíncrono de mensajes. Los mensajes se estructuran siguiendo la teoría conocida como actos del habla (del inglés *speech-acts*), que utiliza fundamentalmente dos campos: el verbo correspondiente al tipo de mensaje (llamado *performative*) y el contenido intercambiado. Los protocolos de interacción han sido reprobados por su escasa descripción formal (p. ej. las máquinas de estados no suelen especificar formalmente el contenido de los mensajes). Otras críticas han venido del alto nivel de acoplamiento de esta solución, que muchas veces hace que una parte útil del código del agente se encuentre mezclada con código exclusivo para la comunicación.

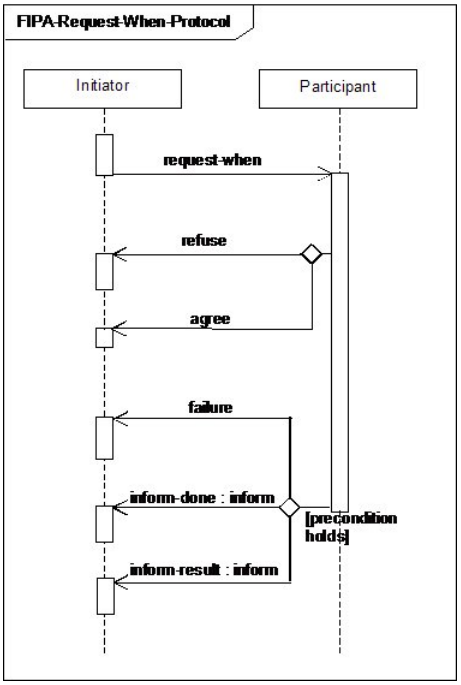


*Figura 3.3: Protocolo de interacción* Request when *en FIPA.*

La **semántica de los ACL** apuesta por la introducción de las acciones comunicativas en el proceso de planificación del agente, junto con el resto de acciones posibles. En es-

te contexto, las leyes de coordinación emergen de la interacción entre los objetivos del agente y las restricciones en la consecución de los susodichos objetivos. Esto es, son el resultado de los efectos y las precondiciones de los mensajes y de las acciones realizadas por el agente. Así pues, la interacción no está diseñada de antemano sino que surge como resultado del proceso de planificación, una misión demasiado compleja para un planificador cuando el mundo deja de ser trivial.

Los actores sintéticos que pueblan los mundos virtuales necesitan utilizar la coordinación para evitar el caos y ahorrarse interferencias innecesarias que pueden surgir si cada carácter sólo mira por su propio beneficio inmediato. A pesar de que los humanos virtuales no requieren en principio la animación de planes óptimos, está claro que la verosimilitud de un grupo de individuos depende en gran medida de la coherencia de las acciones con respecto a los otros. Esta coherencia puede ser conseguida con agentes que ejecuten planes coordinados cercanos al óptimo y que se adapten a cambios impredecibles.

### *Coordinación de planes*

El problema de la coordinación de planes en entornos multiagente (*Multiagent Plan Coordination Problem* o MPCP) aparece cuando un conjunto de agentes ha de planificar su actuación de manera individual pero pueden sacar provecho con la coordinación de sus planes. Diferentes perspectivas han sido estudiadas dentro del MPCP [Durfee, 1999, de Weerdt, 2003]. La coordinación con **agentes especializados** es un ejemplo donde los individuos están dotados con la capacidad exclusiva de realizar ciertas acciones [Kambhampati et al., 1991] o de manipular ciertos objetos [Brenner, 2003]. Esta aproximación reduce la complejidad del problema de coordinación pero crea agentes predecibles. Sin embargo, los personajes 3D orientados a realizar tareas en un mundo virtual generalmente comparten las capacidades de actuación; cosa que hace muy importante la consideración del grado de dependencia entre las actividades y los objetivos de los agentes.

Cox i Durfee [Cox and Durfee, 2005] han introducido recientemente una técnica eficiente de **post-planificación** para producir planes óptimos y coordinados en agentes casi independientes. Existe todo un conjunto de investigaciones en coordinación que tratan sobre cómo sincronizar de forma eficiente los planes después de que estos ya se hayan construido [Georgeff, 1983, Ephrati and Rosenschein, 1994, Yang, 1997]. Un ejemplo muy representativo es *Generalized Partial Global Planning* (GPGP) [Decker and Lesser, 1997].

En esta aproximación los planes con las acciones que quieren realizar los agentes son intercambiados. Posteriormente, los agentes mezclan estos planes y ordenan las acciones utilizando unas relaciones que informan sobre cuáles pueden facilitar la ejecución de otra (ver la figura 3.4). El criterio es maximizar una función de utilidad global que normalmente está asociada con la rapidez en la ejecución del plan. Con todo, el uso de planes completos que surgen como resultado de una fase de post-planificación no es recomendable en los entornos virtuales. El alto dinamismo de estos entornos, donde los usuario y los personajes actúan de manera asíncrona, hará que los planes sean inaplicables en poco tiempo y tengan que ser abortados. La coordinación en entornos asíncronos ha sido tratada con agentes que intercalan planificación y acción [Durfee, 1999]. En unos casos, la coordinación se ha conseguido mediante planificadores STRIPS con nuevas precondiciones; que representan restricciones que se aplican en tiempo de planificación [Boutilier and Brafman, 2001]. En otros casos, es una fase de **pre-planificación** la que aplica algún tipo de restricción, como por ejemplo ciertas leyes sociales [Shoham and Tennenholtz, 1995].

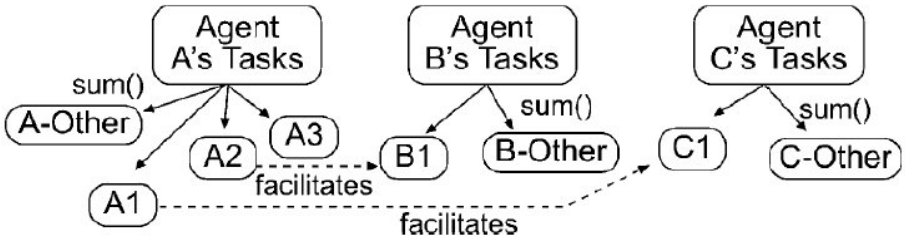


*Figura 3.4: Ejemplo de ordenación de las tareas de tres agentes en GPGP.*

Como ya hemos comentado, la correlación mostrada por una población de agentes es un termino abstracto que no hace referencia ni a la organización externa de la sociedad ni a la estructura interna del agente. El estudio de los procesos de comunicación y de las interrelaciones entre los agentes nos ha llevado al análisis de la coordinación. En el siguiente punto analizamos la colaboración como una expresión de la lógica interna de un agente.

### 3.3.3. Colaboración

Una sociedad artificial puede mostrar correlación sin que sus miembros tengan ninguna intencionalidad hacia los otros. Por ejemplo, en la competición por un recurso compartido, unos agentes podrían coordinarse mientras permanecen libres de malicia o de bondad. Sin embargo, se pueden identificar dos tipos especiales de correlación que apa-

recen ligados a las intenciones internas de un agente: la **cooperación** y la **contención**. La cooperación requiere la existencia de unas intenciones de trabajo conjunto con el resto de agentes. Por el contrario, la contención implica la determinación de frustrar las intenciones de otro. Así pues, es fácil comprender porqué estos calificativos sólo deberían ser atribuidos a los agentes cognitivos; los cuales tratan de imitar las representaciones y los procesos cognitivos humanos. La cooperación no lleva asociada que la comunicación entre los agentes tenga que ser directa o indirecta. Un caso que merece especial atención es aquel que se produce cuando los agentes cooperan y conversan directamente, situación reconocida como **colaboración**.

Una pieza clave en el estudio del trabajo en equipo (*teamwork*) ha sido la **teoría de las intenciones conjuntas** (*joint intention theory*) [Cohen and Levesque, 1991]. Según esta teoría, la formación de un equipo necesita: a) que todos los agentes implicados hayan adquirido el compromiso de conseguir un objetivo común y b) que reciban este mismo compromiso del resto de miembros del equipo. Wooldridge y Jennings desarrollaron un modelo formal de resolución colaborativa de problemas basado en el uso de compromisos conjuntos así como de convenciones o normas para definir la admisibilidad de un compromiso [Wooldridge and Jennings, 1994]. Este paradigma ha inspirado diversas arquitecturas de simulación multiagente. Destacados en este ámbito son los trabajos realizados por Milind Tambe sobre su sistema STEAM [Tambe, 1997]. STEAM utiliza un lider centralizado para la formación de equipos y la resolución de conflictos. Este agente divide los objetivos globales del equipo en subobjetivos que tendrán que ser conseguidos por roles diferentes. Una vez asignados los roles, se supone que los miembros del equipo pueden completar sus tareas de manera independiente (ver la figura 3.5). Desgraciadamente, esta situación ideal es difícil de conseguir en los mundos virtuales, donde diferentes actores pueden interferir cuando ejecutan sus acciones.

Los **planes compartidos** (*SharedPlans*) [Grosz et al., 1999] han tratado la colaboración como el problema de elaborar un plan conjunto que tenga en cuenta el estado mental y los objetivos de múltiples participantes. Esta aproximación necesita que los agentes conozcan desde un inicio el modelo del equipo, es decir, todas las actuaciones posibles para conseguir el objetivo (aunque de algunas no tengan detalles específicos porque no las realizarán ellos). Asimismo, deberán acordar ciertos procedimientos de decisión, como por ejemplo la asignación de agentes a acciones, de manera que puedan completar el plan global. *SharedPlans* se ha aplicado con éxito en diferentes dominios: desde el comercio electrónico a interfaces colaborativas en sistemas de aprendizaje a distancia.

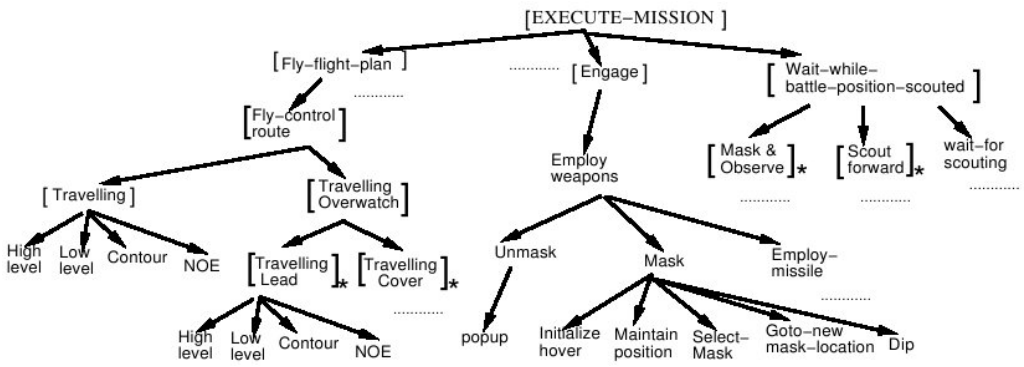


*Figura 3.5: Descomposición de una misión de ataque militar [Tambe, 1997]. Los operadores marcados con * corresponden a operadores de equipo.*

Finalmente, la **delegación** de tareas también se ha utilizado para obtener colaboración en sociedades con un modelo jerárquico de responsabilidad [Ioeger and Johnson, 2001]. Dentro de estas sociedades, algunos agentes líderes pueden encargar la ejecución de ciertas tareas directamente a otros agentes subordinados. Un sistema influyente en esta área es *Open Agent Architecture* (OAAA) [Martin et al., 1999]. Este sistema utiliza un *facilitador* de servicios como por ejemplo: la comunicación entre agentes distribuidos o la subdivisión de tareas complejas en trozos más pequeños que puedan ser ejecutados independientemente por múltiples agentes. En vez de usar un agente facilitador central, el sistema RETSINA [Giampapa and Sycara, 2002] utiliza la técnica de las páginas amarillas (directorios de agentes que ofrecen ciertas capacidades). RETSINA intercala planificación con ejecución y da soporte a la colaboración mediante un planificador que modela las tareas a través de redes jerárquicas de tareas (HTN). Por tanto, los actos comunicativos se encuentran predefinidos en el árbol HTN. De esta manera, el planificador en tiempo de ejecución puede solicitar información así como la ejecución de acciones a otro agente y suspender esa línea de actuación hasta que llegue el resultado.

La investigación en MAS ha estudiado los diferentes tipos de correlación como un mecanismo para desarrollar sistemas más productivos. Sin embargo, la racionalidad no es el único camino para conseguir la **congruencia**, definida como el grado en el que las interacciones entre los agentes satisfacen las expectativas generales del sistema. Por ejemplo, los personajes 3D a menudo operan en entornos compartidos con recursos limitados, cosa que apunta al uso de técnicas de coordinación y colaboración. Ahora bien, los actores sintéticos normalmente representan roles humanos, como por ejemplo un camarero o un cliente en un bar virtual. Entonces, uno espera que estas relaciones sociales también se manifiesten en su comportamiento. El razonamiento social ha sido estudiado

profundamente en la comunidad de los MAS con vistas a incorporar **elementos sociales** en los agentes cognitivos [Conte and Castelfranchi, 1995, Conte, 1999]. Como resultado, los modelos de interacción entre los agentes han evolucionado hacia las redes sociales, las cuales tratan de imitar las estructuras sociales que se pueden encontrar en la realidad [Hexmoor, 2001].

## 3.4. REDES SOCIALES

La componente social de la interacción ha recibido una atención considerable en el campo de los sistemas multiagente [Jennings and Campos, 1997, Castelfranchi, 1998]. La búsqueda de elementos sociales que influyan en la toma de decisiones de los miembros de una sociedad artificial ha tratado sobre temas como: los **roles**, las **relaciones** de poder o confianza, las **normas**, las **preferencias** de los agentes, etc. De manera general, podemos afirmar que todo agente inmerso en un grupo juega un papel o rol; que determina sus responsabilidades. Asimismo, los roles de una sociedad definen a menudo relaciones entre los agentes (p. ej. la relación de poder que existe entre un agente líder y un subordinado). Estas relaciones pueden hacerse más complejas y llegar a construir redes que representen la estructura de una organización. En dicho caso, la actuación de los agentes se verá restringida, ya que han de obedecer a una serie de reglas o normas dentro de la organización. En este contexto, los agentes pueden cambiar de un grupo a otro cuando sus preferencias internas dejen de coincidir con los requerimientos de la organización. Existe un gran abanico de trabajos que han tenido como asunto principal las normas y los contratos de compromiso para formalizar las restricciones que definen los comportamientos aceptables dentro de una organización [Shoham and Tennenholtz, 1995, Agotnes et al., 2007]. El análisis exhaustivo de este problema queda fuera del alcance de esta tesis.

Las organizaciones y las **redes sociales** humanas han sido tema de estudio durante mucho tiempo [Scott, 1991]. El objetivo del análisis de las redes sociales es comprender los enlaces entre los individuos (como los individuos se relacionan los unos con los otros) así como extraer ciertas propiedades asociadas al grupo entero. La forma más común de representar las relaciones sociales de un grupo es mediante un **sociograma**. Un sociograma es un grafo donde los nodos suelen representar agentes o predicados mientras que los arcos que los unen corresponden a las relaciones. Identificamos tres categorías principales de redes sociales dependiendo del significado de estos arcos: a) las redes de dependencia,

b) las redes de confianza y c) las redes de preferencia.

### 3.4.1. Redes de dependencia

La autonomía de los agentes se encuentra en ocasiones limitada por redes de interdependencia, donde los agentes dependen los unos de los otros para conseguir sus propios objetivos. La **teoría de la dependencia**, propuesta por diversos autores [Castelfranchi et al., 1992, Sichman et al., 1994], considera que un agente $i$ es dependiente de otro agente $j$ con respecto a un objetivo $p$ cuando: $i$ carece al menos de una de las acciones o recursos necesarios para conseguir $p$, mientras que $j$ puede realizar la susodicha acción o está en posesión del recurso adecuado. Inversamente, se dice que el agente $j$ tiene poder sobre el agente $i$. Por ejemplo, un trabajo conjunto en equipo puede ser representado como una agregación donde cada agente depende del resto para llegar a alcanzar un objetivo compartido (ver la red completa de la figura 3.6a). Los entornos de mercado también pueden ser vistos como redes de dependencia, aunque menos cohesionadas (ver la figura 3.6b). En estos entornos, el deseo común de que el mercado genere beneficios hace que las empresas implicada hagan tratos y tengan socios de negocios [Conte, 1999].

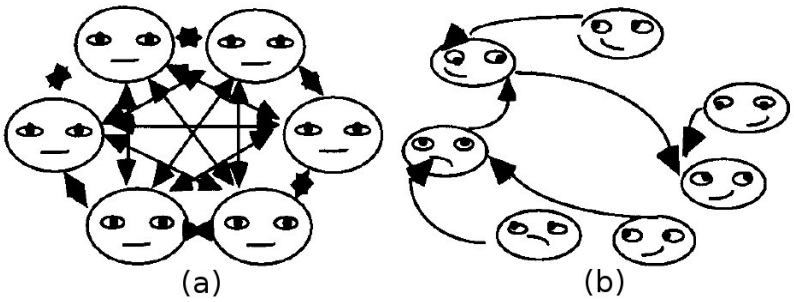


*Figura 3.6: Ejemplos de redes de dependencia: a) trabajo conjunto en equipo y b) entorno de mercado.*

La teoría de la dependencia da respuesta, así, a las dos preguntas fundamentales en una sociedad de agentes autónomos. Un agente consigue que su objetivo personal se convierta en social, esto es, que sea adoptado por otros agentes, gracias a su poder social sobre éstos. De manera análoga, un agente autónomo forma parte de una interacción social a causa de la dependencia social que lo liga a otro agente.

El **razonamiento social** [Sichman and Demazeau, 2001] se basa en la teoría de la dependencia para implementar la formación dinámica de coaliciones. El diseño de la es-

tructura de una organización con una red de dependencia es interesante. Sin embargo, el modelo de la organización no puede ser concebido en tiempos de diseño en los mundos abiertos, donde los agentes pueden unirse y dejar la sociedad en cualquier momento. Como este tipo de agentes normalmente no dispone de una representación correcta y completa del resto, esta aproximación incorpora un procedimiento de revisión de creencias. Finalmente, el modelo distingue dos tipos de interacciones sociales de acuerdo con las coaliciones aceptadas: la cooperación (diversos agentes que trabajan de forma conjunta para alcanzar un objetivo común) y el intercambio social (diversos agentes que trabajan de forma conjunta para conseguir diferentes objetivos individuales).

### 3.4.2. Redes de confianza

Los agentes a menudo son considerados entidades autónomas en las que no es posible ejercer control sobre su comportamiento interno. Dentro de un MAS, los agentes se ven involucrados en la realización de tareas colectivas que requieren la cooperación y la confianza en otros agentes. La posibilidad de que un agente no se comporte de manera correcta, intencionadamente o no, produce riesgos de fiabilidad y robustez a nivel del sistema, especialmente en el caso de sistemas abiertos. Se ha propuesto como solución el uso de **modelos de confianza** (*trust models*). Uno de los primeros trabajos en modelos de confianza fue el realizado por Steve Marsh en el Reino Unido [Marsh, 1994]. Su modelo distingue tres niveles de confianza: básico, general y situacional. La confianza básica (*Basic Trust*) es la actitud general del agente a la hora de confiar en los otros. La confianza general (*General Trust*) corresponde con la confianza genérica depositada sobre un agente concreto. Finalmente, la confianza situacional (*Situational Trust*) es la actitud de confianza hacia los otros en circunstancias específicas.

Los modelos de confianza dotan a los agentes de modelos de otros agentes para que puedan decidir si confían en ellos o no. Habitualmente, la fuente de información que se utiliza en la toma de decisiones confiada es la **reputación** [Castelfranchi and Falcone, 1998]. Paradójicamente, no existe una definición comúnmente aceptada para este término. Inicialmente, la reputación fue considerada un valor único, asociado a un agente y manipulado por el sistema, que podía hacer referencia a la diligencia. Las redes sociales han permitido que dos agentes tengan opiniones diferentes sobre la reputación de un tercero [Sabater and Sierra, 2002]. En última instancia, se ha propuesto una ontología funcional de reputación para permitir la interoperabilidad entre diferentes modelos de confianza

[Vercouter et al., 2007].

No obstante, a la hora de confiar o delegar tareas en una red de confianza, otras fuentes de información también pueden ser tenidas en cuenta. Por ejemplo, el trabajo de Rino Falcone [Falcone et al., 2004] utiliza como estrategia de delegación, las habilidades para ejecutar ciertas tareas (*ability*) o la predisposición para adoptar tareas externas (*willingness*). Además, incorpora un mecanismo para alterar estos valores en función del estado del entorno (ver la figura 3.7).

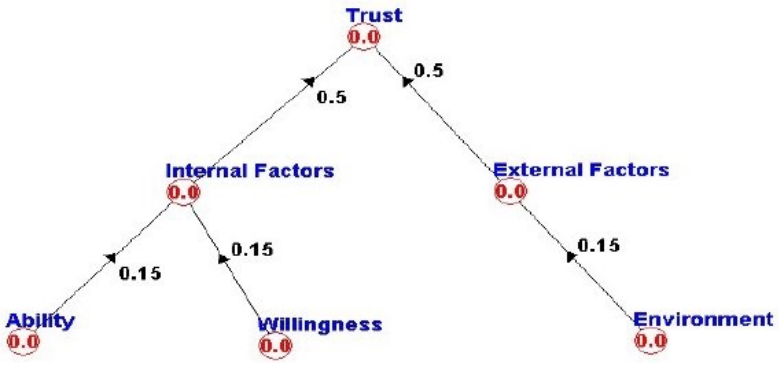


*Figura 3.7: Estrategia de delegación de tareas con diversos criterios de confianza definida en [Falcone et al., 2004].*

### 3.4.3. Redes de preferencia

Se puede entender la toma de decisiones como el proceso cognitivo que permite seleccionar una alternativa de entre un conjunto de posibilidades. En entornos colectivos, como lo son los sistemas multiagente, estas decisiones deberían tener en cuenta también las opiniones externas. De acuerdo con esto, las redes de preferencia permiten que los agentes expresen sus **preferencias individuales** mediante una relación de orden sobre el conjunto de alternativas. Mientras que este conjunto sea reducido, los problemas son computacionalmente sencillos. Sin embargo, muchos problemas presentan un conjunto de alternativas con estructura combinatoria. Por ello, la elección del lenguaje de preferencia ha sido un problema tratado con profundidad en el campo de la selección social (*social choice*) [Lang, 2005]. En la práctica, los agentes normalmente expresan sus preferencias utilizando **funciones de utilidad**. El propósito de una función de utilidad es asignar un valor numérico que exprese la opinión del agente sobre cada una de las soluciones a un problema dado. Una antigua cuestión que afecta a las funciones de utilidad hace referencia

a la dificultad de expresar una decisión humana en términos numéricos. Esta dificultad, apuntada ya por Jacques Ferber [Ferber, 1999], todavía está por resolver y necesitará la aportación de otras áreas de conocimiento como la filosofía y la psicología.

La teoría de la selección social no determina un mecanismo concreto para la obtención de estas preferencias. Dependiendo de la naturaleza del problema, se han utilizado diversos **protocolos** [Sandholm, 1998] como por ejemplo: las técnicas de votación, las subastas, la negociación bilateral, la negociación distribuida, etc. Desde un punto de vista estratégico, la teoría de juegos (*game theory*) [Osborne, 2004] ha sido la rama de las ciencias económicas encargada de estudiar la toma de decisiones en estos contextos en que los agentes no tienen porqué ser necesariamente veraces.

Una vez se han obtenido las preferencias de los agentes de la sociedad, se computa cuál es la solución socialmente más adecuada. Este problema se ha tratado bajo el formalismo de la repartición de recursos en sistemas multiagente (*Multi-Agent Resource Allocation* o MARA) [Chevaleyre et al., 2006]. A la hora de tomar esta decisión, MARA utiliza el concepto de bienestar social (*social welfare*) proveniente de la economía del bienestar (*Welfare Economics*). Así pues, las funciones de bienestar social se usan para definir diferentes órdenes sociales por medio de la agregación de las preferencias individuales. En el caso habitual en que las preferencias se expresan mediante funciones de utilidad, el **orden social** se define a través de **funciones de utilidad colectivas** (*Collective Utility Functions* o CUF) [Sandholm and Suri, 2001].

Para entender el funcionamiento de las redes de preferencia, imaginemos el siguiente juego mental: *.ºriol y Marta desean salir una noche y se plantean dos posibilidades: ir al cine o ir al teatro. Supongamos que a Oriol le gusta más el teatro y que le asigna un valor de 8, mientras que la utilidad que da a ir al cine es de 3. Sin embargo, las preferencias de Marta son contrarias y menos vehementes, 4 para ir al cine y 2 para ir al teatro"*. La tabla 3.2 muestra los diferentes órdenes sociales. Una sociedad utilitaria escogerá aquella solución que maximice la utilidad total del grupo (sumatorio de utilidades), mientras que una sociedad igualitaria escogerá aquella solución que beneficie al más débil del grupo (mínimo de las utilidades). Por último, una sociedad elitista seleccionará aquella solución que sigua las preferencia de aquel que más puede ganar con la resolución del problema (máximo de las utilidades).

En las redes de preferencia, la capacidad de un agente de modificar con un diferen-

| Soluciones/CUFs | Utilitaria ($\sum$) | Igualitaria (min) | Elitista (max) |
|---|---|---|---|
| Ir al cine | 7 | **3** | 4 |
| Ir al teatro | **10** | 2 | **8** |

*Tabla 3.2: Diferentes bienestares sociales expresados con CUFs.*

cial de importancia las preferencias recibidas del resto de agentes permite la definición de **actitudes sociales** [Brainov and Sandholm, 1999]. Por ejemplo, un agente egoísta reduciría las utilidades recibida fortaleciendo de esta manera su opinión. Al contrario, un humanoide altruista incrementaría las utilidades recibidas en detrimento de su preferencia. El capítulo 5 de esta tesis analiza con más detalle la expresión de preferencias sociales con funciones de utilidad y la aplicación de diferenciales para modelar actitudes como la indiferencia, la reciprocidad, el egoísmo y el altruismo.

Las aproximaciones basadas en MARA han sido aplicadas en un rango muy amplio de aplicaciones reales [Chevaleyre et al., 2006]: gestión de tráfico aéreo, logística, provisionamiento industrial, explotación conjunta de satélites de observación de la Tierra, asignación de recursos en arquitecturas malladas (*grid computing*), etc. En esta tesis propondremos su uso para la toma de decisiones sociales de actores sintéticos que habitan mundos virtuales.

## 3.5. CONCLUSIONES

Con frecuencia, los actores sintéticos no están solos en el entorno, sino que comparten un espacio común de actuación con otros personajes; con los que forman una especie de sociedad artificial. Sin embargo, la inclusión de comportamientos sociales en personajes 3D se ha limitado normalmente a la interacción verosímil con el usuario, mientras que las interacciones con otros caracteres autónomos han sido escasas. El reto actual consiste en incorporar el conocimiento que proviene del área de los sistemas multiagente, de manera que se puedan obtener actores sintéticos autónomos dotados de ambas capacidades: racionalidad y sociabilidad. De acuerdo con esto, en este capítulo se han repasado los diferentes modelos de interacción que se pueden encontrar en un conjunto de agentes que trabajan de manera conjunta. Cabe destacar la colaboración, ya que corresponde a la situación en que los agentes cooperan y conversan directamente. Por lo que respecta a los

elementos sociales de la interacción, las redes sociales han sido presentadas como modelo de reproducción de las relaciones humanas reales. Entre ellas, las redes de preferencia han demostrado ser un mecanismo muy flexible a la hora de tomar decisiones en un entorno social.

La tercera parte de este trabajo (capítulos 4, 5, y 6) contiene las principales aportaciones al estado del arte que ara concluye. En primer lugar, definimos un modelo general de entorno virtual semántico basado en ontologías para mejorar la sensorización de escenas complejas, la interacción agente-objeto y la definición de relaciones entre los agentes de una sociedad artificial (capítulo 4). En segundo lugar, describimos dos mecanismos para la inclusión de habilidades sociales en dos de los paradigmas utilizados de manera más habitual por los agentes racionales. Concretamente, en el capítulo 5 presentaremos: a) una técnica de sopesado de objetivos para conseguir comportamientos colaborativos en grupos de agentes basados en planificadores heurísticos; y b) MADeM (*Multi-modal Agent Decision Making*), un proceso de toma de decisiones de tipo social que utiliza las preferencias de otros agentes para tomar decisiones socialmente aceptables. Por último, en el capítulo 6 mostraremos los resultados obtenidos y el grado de comportamiento social alcanzado por los personajes 3D que utilizan estos mecanismos sociales.

# PARTE III

# APORTACIONES

CAPÍTULO 4

# MARCO DE SIMULACIÓN

En este capítulo, presentamos las aportaciones menores de esta tesis, recogidas en el **marco de simulación multiagente** diseñado para la integración de actores sintéticos con habilidades sociales en entornos virtuales 3D. Fundamentalmente, este marco se compone de: a) un modelo de **entorno virtual semántico** para la construcción y mantenimiento del mundo virtual; y b) una arquitectura base de agentes en la que poder integrar técnicas de razonamiento social. El entorno virtual semántico incorpora el uso de **ontologías** para la representación de la base de conocimiento del mundo y de las relaciones entre los agentes 3D. Este conocimiento semántico beneficia la producción, la percepción y la interacción de los personajes en escenarios 3D complejos. Referente a la **arquitectura de agentes socialmente inteligentes**, repasamos los módulos principales requeridos para controlar el comportamiento autónomo de los personajes 3D. En este contexto, tratamos el problema de la actuación sobre los objetos del entorno (interacción agente-objeto) y de la comunicación entre los personajes (interacción agente-agente). La toma de decisiones de tipo social, principal aportación de esta tesis, será tratada en el capítulo 5.

## 4.1. INTRODUCCIÓN

Como se desprende del capítulo 2, la evolución de la animación comportamental de los personajes 3D ha ido de la mano del nivel de representación del estado del mundo. En cierta manera, la interacción es directamente proporcional a la cantidad de conocimiento que los agentes perciben. Los entornos virtuales 3D clásicos han utilizado diversos mecanismos de representación; desde los grafos de escena a las tautologías de objetos. En estos entornos, las capacidades de interacción de sus habitantes se reducían normalmente

a un conjunto de objetos o de protocolos de comunicación predefinidos entre los caracteres. Sin embargo, la incorporación de técnicas de razonamiento cada vez más sofisticadas, como es el caso del comportamiento social, ha promovido la separación entre el **modelo semántico**, el modelo geométrico y el modelo comportamental del mundo. Por lo que respecta al modelo semántico, uno de los retos actuales se encuentra en la búsqueda de una solución genérica para la representación semántica de los mundos virtuales 3D; de manera que se favorezca la interacción entre los objetos y los agentes que lo pueblan.

De acuerdo con este razonamiento, en el punto 4.2 describimos el **marco de simulación multiagente** diseñado para la generación de entornos virtuales 3D habitados por actores sintéticos con habilidades sociales. El modelo general de **entorno virtual semántico** (SVE), incluido en este sistema, se trata en el punto 4.3. Proponemos las ontologías como una solución extensible y reutilizable para la representación del entorno y de las relaciones entre los actores (sección 4.3.1). La construcción de escenarios complejos, con numerosos objetos interactivos, así como la percepción de la gran cantidad de información que estos entornos producen son cuestiones que se tratan en la sección 4.3.2.

La base semántica del mundo no incluye los esquemas de acción de los personajes que lo pueblan, ya que no consideramos que estos sean necesariamente compartidos. Al contrario, consideramos los agentes como entidades autónomas potencialmente heterogéneas que han de mantener su propia operativa; la cual podrá ser intercambiada, si procede, con otros agentes. En el punto 4.4 revisamos los módulos principales de nuestra **arquitectura de agente socialmente inteligente**, donde integraremos los mecanismos de toma de decisiones sociales del capítulo 5. En la sección 5.4.2 exponemos como la semántica del mundo definida en la ontología puede ser utilizada internamente por los agentes para la definición de operativas generales. Por último, la sección 4.4.2 versa sobre el **sistema de comunicación**; un requisito básico para la toma de decisiones social, tratada en el capítulo 5.

## 4.2. MARCO DE SIMULACIÓN MULTIAGENTE

La simulación de entornos virtuales habitados es una tarea altamente compleja que se enfrenta a la resolución de problemas provenientes del campo de la animación gráfica y de la inteligencia artificial. Este hecho aconseja una descomposición modular del **sistema de**

**simulación** de manera que se puedan atender por separado el modelo semántico, el modelo comportamental y el modelo geométrico [Lozano and Calderón, 2004]. De acuerdo con esto, el marco de simulación multiagente, que proponemos en ls figura 4.1, está formado por; a) un **entorno virtual semántico**, que crea y mantiene el estado del mundo; b) un conjunto de **agentes socialmente inteligentes**, que genera el comportamiento de los personajes; y c) un **motor gráfico 3D**, encargado de la visualización

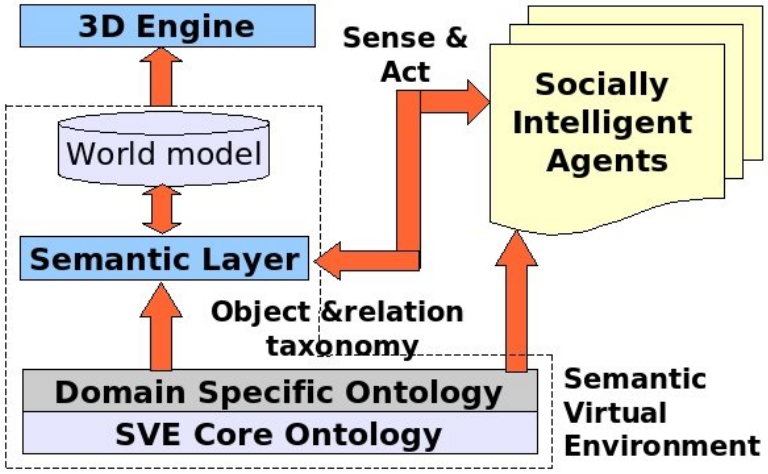


*Figura 4.1: Marco de simulación multiagente propuesto.*

El entorno virtual semántico propuesto se basa en el uso de **ontologías** para definir la base de conocimiento del mundo así como el conjunto de todas las relaciones posibles entre los agentes; los cuales forman una especie de sociedad artificial. Distinguimos dos niveles de representación ontológica (sección 4.3.1). El modelo base está formado por la *SVE Core Ontology*, que es una ontología única que contiene las clases básica para definir cualquier entorno virtual de manera general, como por ejemplo: los objetos contenedores, los objetos móviles, etc. A la hora de modelar el conocimiento específico asociado a ciertas escenas o situaciones, esta ontología base se extenderá por medio de las ontologías específicas del dominio adecuadas (*Domain Specific Ontologies*). Por ejemplo, la producción de una cocina virtual necesitará la creación de diversos tipos de objetos que son habituales: los armarios, las botellas, los platos, los vasos... En este contexto semántico, la construcción de un entorno consiste en crear los objetos 3D de la escena como instancias de las clases definidas en estas ontologías (ver el bar virtual construido en la sección 6.4).

El modelo del mundo (*World model*), esto es, la base de datos con la información de las entidades que lo componen, se gestiona a través de la **capa semántica** (*Semantic Layer*)

(sección 4.3.2). Esta capa actua como interfaz entre el agente y el estado del mundo y se encarga, fundamentalmente, de dos aspectos: la sensorización y la actuación. De un lado, la capa semántica utiliza las relaciones de la ontología para reducir el flujo de información que provoca la percepción de objetos ricos en información. Un ejemplo, en la anterior cocina virtual, sería un armario que almacenara diferentes tipos de platos, vasos y cubiertos. De otro lado, la capa semántica se hace cargo de la ejecución de las acciones solicitadas por los agentes y del mantenimiento de sus efectos sobre el modelo del mundo. Por ejemplo, cuando un actor sintético abre la puerta de un ascensor virtual, este permanece en el estado *ocupado* mientras que la puerta no sea cerrada. El intercambio de información entre el entorno virtual semántico y los agentes inteligentes se hace mediante cadenas XML, las cuales siguen un formato de verbo y contenido inspirado en los *speech-acts* de KQML y FIPA ACL [Finin et al., 1994, FIPA (Foundation for Intelligent Physical Agents), 2008].

El modelo comportamental es aquel que hace referencia a la generación del comportamiento de los personajes autónomos que pueblan un entorno virtual. En nuestro marco de simulación, el control principal de cada humanoide se encuentra situado en un agente diferente. El grado de realismo comportamental alcanzado dependerá, pues, de sus capacidades racionales y sociales; razón por la que los adjetivamos como **agentes socialmente inteligentes** (*Socially Intelligent Agents* en la figura 4.1). Estos agentes reciben la información sensorial de la capa semántica y están dotados de algún formalismo de IA para calcular la secuencia apropiada de acciones que los lleve a conseguir sus objetivos. En este contexto, la taxonomía de objetos definida por la ontología se utiliza para definir operadores generales (p. ej. abrir cualquier tipo de contenedor), los cuales incrementarán la reusabilidad de los agentes en diferentes escenarios (5.4.2). Como veremos en el capítulo 5, la inclusión de técnicas sociales en este proceso de toma de decisiones adornará el comportamiento con la selección de acciones socialmente aceptables. Asimismo, la arquitectura de estos agentes también contemplará procesos de: percepción, control motor, navegación, comunicación, etc. (punto 4.4). La naturaleza distribuida de esta solución, en la que cada agente puede utilizar un mecanismo de decisión diferente, así como ejecutarse en máquinas diversas, responde a las fuertes necesidades de heterogeneidad y escalabilidad de los sistemas de simulación 3D.

Finalmente, el modelo geométrico del escenario 3D así como el sistema de animación (p. ej. tablas de animación, etc.) se encuentran localizados en el **motor gráfico 3D** (ver *3D Engine* en la figura 4.1). Este módulo se encarga de la visualización gráfica de la escena. La formalización semántica del modelo del mundo junto con la separación de

la visualización permite intercambiar el módulo de dibujado de una manera sencilla. A modo de ejemplo, el capítulo 6 muestra los resultados obtenidos con dos motores gráficos 3D diferentes: el motor del juego *Unreal Tournament* [Epic games, 2008] y la librería gráfica *OpenSceneGraph* [OSG, 2008].

De acuerdo con el marco de simulación presentado en la primera parte de este capítulo, en el siguiente punto describimos el modelo propuesto de entorno virtual semántico. Seguidamente, analizaremos la arquitectura de los agentes inteligentes que nos sirve de base para la integración de habilidades sociales.

## 4.3. ENTORNO VIRTUAL SEMÁNTICO

En el contexto actual de los 3DIVA se acepta comúnmente que los personajes 3D, que aspiran a reproducir comportamientos inteligentes, han de manipular información a distintos niveles de representación. Por una parte, hace falta un nivel bajo de representación numérica para tratar la información relativa a la posición, velocidad y otras variables geométricas de las entidades presentes en el entorno. Por otra parte, un nivel alto de representación simbólica permitirá a los agentes conocer el estado del escenario así como las relaciones entre los objetos o agentes, y posibilitará la resolución de situaciones diversas surgidas en el mismo. El modelo de entorno virtual semántico propuesto en esta tesis incluye ambos niveles de representación mediante el uso de **ontologías**.

El término ontología, originario de la filosofía, se refiere a la ciencia que describe los tipos de entidades presentes en el mundo y como se relacionan. A diferencia de un formato de mensaje (p. ej. XML o XML Schema), una ontología se caracteriza por se una representación semántica del conocimiento sobre la cual se puede razonar. La semántica formal de una ontología especifica cómo derivar consecuencias lógicas; esto es, hecho que no están literalmente presentes en la ontología pero que pueden ser derivados por la semántica.

El lenguaje adoptado para la implementación de las ontologías ha sido el ***Web Ontology Language* (OWL)** [W3C, 2004], recomendación del W3C (*World Wide Web Consortium*) que se prevé que se convertirá en el estándar *de facto* para la creación de ontologías en la Web Semántica. OWL ha sido desarrollado como una extensión de RDF y se deriva

del lenguaje ontológico DAML+OIL [Joint US/EU ad hoc Agent Markup Language Committee, 2008]. Una ontología en OWL incluye, fundamentalmente, descripciones de clases, propiedades y sus instancias. Una clase representa un conjunto de entidades individuales o instancias, las cuales también son casos particulares de cualquier superclase de esta clase. Una propiedad especifica la relación que se establece entre una instancia de una clase origen y una instancia de una clase destino. En nuestro entorno virtual semántico, las ontologías definen el modelo abstracto del mundo, esto es, la jerarquía de clases de objetos/agentes, sus propiedades y las posibles interrelaciones entre ellos (ver la figura 4.2).

El lenguaje OWL tiene tres sublenguajes llamados, en orden ascendente de expresividad: *OWL Lite*, *OWL DL* y *OWL Full*. *OWL Lite* permite la descripción de clasificaciones jerárquicas con restricciones fuertes (p. ej. en las restricciones de cardinalidad sólo permite los valores 0 ó 1). Por tanto, es la aproximación más sencilla para realizar una migración rápida desde un *tesauro* u otras taxonomías a una representación ontológica. *OWL DL* recibe su nombre de la lógica descriptiva (*Description Logics*), un campo de investigación que estudia aquella parte de la lógica de primer orden que es decidible. *OWL DL* incluye casi todas las construcciones posibles del lenguaje OWL, pero impone algunas restricciones, como la separación de tipos (una clase no puede ser tratada además como un individuo o una propiedad, y viceversa). De esta manera, *OWL DL* proporciona una **expresividad** máxima pero garantiza que los sistemas de razonamiento ontológico pueden computar todas las inferencias en un tiempo finito. Finalmente *OWL Full* está orientado a aquellos usuarios que quieren expresividad máxima gracias a una libertad sintáctica total. Por ejemplo, una clase en *OWL Full* puede ser tratada simultáneamente como una colección de individuos y como un individuo de pleno derecho. Desgraciadamente, no se espera que ningún *software* de razonamiento sea capaz de admitir todas las características de *OWL Full*. Por tanto, el sublenguaje utilizado en nuestro marco de simulación es *OWL DL*.

La elección de OWL como lenguaje ontológico es muy interesante ya que, debido a su grado de madurez, se dispone de un conjunto de herramientas para trabajar con ontologías en OWL, como por ejemplo: editores que facilitan el desarrollo (p. ej. Protégé [Stanford Medical Informatics, 2006]), librerías para su manipulación (p. ex. Jena [Jena, 2008]) y razonadores semánticos (p. ej. RACER [Racer Systems, 2008]). Asimismo, la utilización de un lenguaje proveniente de la Web Semántica es un aspecto clave para la extensibilidad de los modelos de los entornos virtuales. La **reutilización** de las ontologías puede llevarse a cabo, de manera sencilla, a través de especificaciones OWL

ya existente. Este hecho puede aligerar la creación de nuevos mundos y la integración de múltiples mundos ya existente. Además, este tipo de solución facilitará la **movilidad** de los humanoides virtuales a través de mundos creados por diferentes autores. La base de conocimiento mantenida por el SVE desacopla los agentes de los entornos, de manera que ambos pueden ser desarrollados de manera independiente.

En la sección siguiente, analizamos las ontologías que se han desarrollado para la representación semántica de los entornos virtuales. Estas ontologías se encuentran al alcance de la comunidad científica en la *Protege Ontology Library* [Protégé Wiki, 2008], junto con un conjunto de ontologías interesantes realizadas en OWL. La sección 4.3.2 trata sobre la capa semántica, la otra pieza clave de nuestro entorno virtual semántico.

### 4.3.1. SVE Core Ontology y Domain Specific Ontologies

El entorno virtual semántico propuesto en esta tesis utiliza las ontologías a dos niveles de abstracción: la ontología base (*SVE Core Ontology*) y las ontologías específicas del dominio (*Domain Specific Ontologies*). La *SVE Core Ontology* define las **clases básicas** que son necesarias para poder crear cualquier entorno virtual. De acuerdo con esto, la figura 4.2 muestra un fragmento de la ontología base en el que, mediante la herencia, se define una taxonomía de objetos básica (clases con prefijo *core*). La ontología completa *SVE Core Ontology* en lenguaje OWL[1] puede ser consultada en el anexo A.1.

La clase *GraphicalEntity* modela el conjunto de todas aquellas entidades que disponen de una representación gráfica propia en el entorno virtual; esto es, tanto los agente como los diferentes objetos 3D de la escena. Dentro de este conjunto, las clases *MovableObject* y *BaseObject* hacen referencia, respectivamente, a aquellos objetos que se pueden mover y a aquellos donde poder depositar objetos encima. Los diferentes tipos de objetos contenedores que se pueden encontrar en un mundo simulado están representados por la clase *Container*. En este contexto, distinguimos dos categorías principales: a) los contenedores de sustancias incontables (*UncountableContainer*), como por ejemplo un bote con sal; y b) los contenedores de elementos contables (*CountableContainers*), sobre los que hacemos una clasificación más detallada. De un lado, la clase *ServiceContainer* representa aquellos objetos que proporcionan una cierta cantidad de servicios asociados a entidades

[1] Para el desarrollo de las ontologías de nuestro entorno virtual semántico hemos utilizado el programa Protégé [Stanford Medical Informatics, 2006]

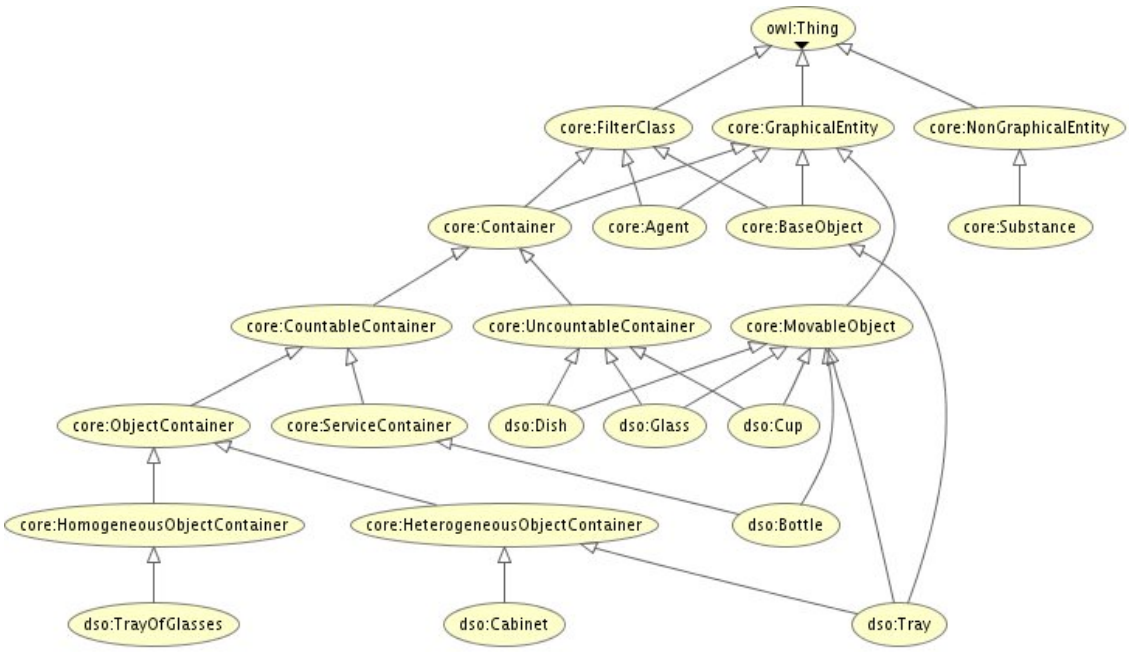


*Figura 4.2: Taxonomía de clases para la creación de un bar virtual.*

no contables (p. ej. una máquina que sirve cafés). De otro lado, los contenedores de objetos (*ObjectContainers*) contendrán objetos con representación gráfica propia. Un caso concreto de contenedor de objetos son los *HeterogeneousObjectContainer*, que podrán mantener objetos de tipos diferentes (p. ej. una estanteria con libros, figuritas de porcelana, etc.). Alternativamente, los contenedores de tipo *HomogeneousObjectContainer* sólo contendrán objetos indistinguibles; como por ejemplo una bandeja con vasos vacíos. La clase *NonGraphicalEntity* simboliza todos aquellos objetos físicos que no tienen una representación gráfica propia en el mundo virtual, sino que si visibilidad depende de otros objetos. Por ejemplo, las sustancias (*Substance*) serán dibujadas por los objetos que las contienen: un vaso de agua, un azucarero con azúcar, etc. Por último, la clase *FilterClass* es una clase especial de filtrado que será utilizada por la capa semántica para reducir la cantidad de información percibida por los 3DIVA en entornos virtuales con muchos objetos interactivos. Su uso será analizado en la sección 4.3.2).

Acabamos de ver como la *SVE Core Ontology* proporciona una clasificación general de alto nivel. Sin embargo, a la hora de crear un entorno virtual particular, se necesitará crear objetos dotados de ciertas características especiales. Por consiguiente, la ontología base puede ser extendida mediante las **ontologías específicas del dominio** adecuadas. Por ejemplo, las clases con prefijo *dso* de la figura 4.2 han sido heredadas de las clases básicas para crear objetos necesarios en la construcción de un bar virtual (p. ej. platos,

vasos, botellas...). En este tipo de escenario, una bandeja (*Tray*) será representada como un contenedor de objetos heterogéneos (*HeterogeneousObjectContainer*) que se puede apilar (*BaseObject*) y mover *MovableObject*. La ontología OWL correspondiente a estas extensiones puede ser consultada en el anexo A.2.

Junto con la definición de la taxonomía de clases, las ontologías definen las **propiedades** que las caracterizan. Por ejemplo, la figura 4.3 muestra el conjunto de propiedades que comparten todos los objetos contenedores (case *Container*). Podemos identificar dos tipos de propiedades: las cuantitativas y las cualitativas. Las propiedades cuantitativas representan valores numéricos que corresponden a menudo a propiedades físicas del objeto; por ejemplo, la posición (*posx*,*posy*,*posz*) o las dimensiones espaciales (*rx*,*ry*). Por el contrario, las propiedades cualitativas normalmente representan diferentes variables de estado del objeto; como por ejemplo si un contenedor está lleno (*full*) o vacío (*empty*). Las propiedades de una clase también definen las **relaciones** que se pueden establecer entre los objetos de la susodicha clase y otros objetos de clases diferentes. Por ejemplo, las botellas (*Bottle*) se relacionan con las sustancias (*Substance*) que contienen, mediante la relación *containsSubstance*. Otra relación es *on*, que enlaza un objeto móvil (*MovableObject*) con el objeto base (*BaseObject*) donde está depositado.

Las **relaciones sociales** entre los agentes de una sociedad artificial también pueden ser representadas ontológicamente en forma de clases y relaciones. La figura 4.4 muestra las clases y las relaciones definidas en la ontología base para la creación de relaciones organizacionales en los entornos virtuales habitados. Distinguimos tres niveles de relación: el nivel individual, el nivel institucional y el nivel interinstitucional. En **nivel individual** representa las relaciones 'uno a uno' (*one-to-one*) y está modelado a través de la clase *agentSocialRelation*. Siempre que un agente se relacione con otro agente de forma individual, ambos estarán ligados por una instancia de esta clase. Como sucedía anteriormente, diferentes dominios de aplicación pueden necesitar relaciones específicas. En ese caso, las ontologías específicas del dominio pueden heredar de las clases base definidas en la ontología básica e incluir la semántica requerida para cada situación particular. Por ejemplo, la relación *WorkmateRelation* de la figura 4.4 lleva la cuenta del número de favores intercambiado por dos agentes que son compañeros de trabajo. Otros ejemplos de relaciones individuales son las relaciones familiares, como por ejemplo la relación de parentesco (*ParentOf*) o de matrimonio (*MarriedWith*).

El **nivel institucional** representa las relaciones 'uno a muchos' (*one-to-many*) y la

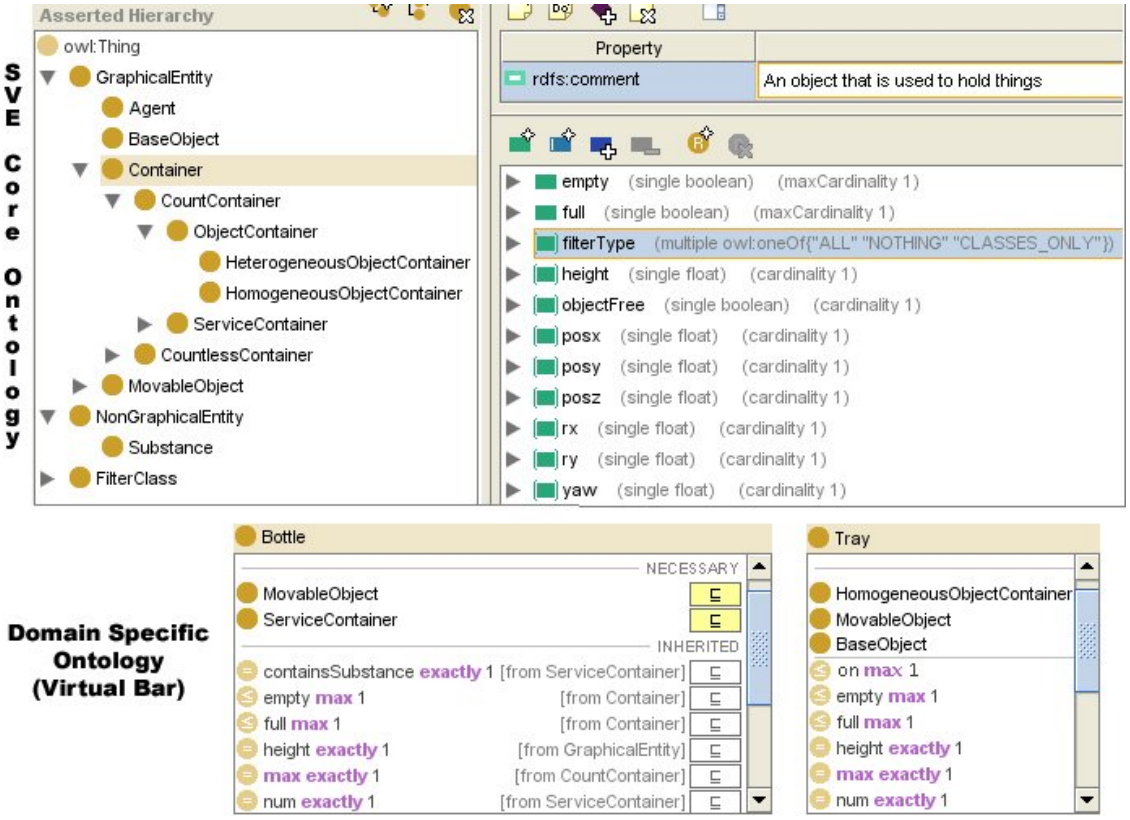


*Figura 4.3: Propiedades de las clases definidas en la ontología.*

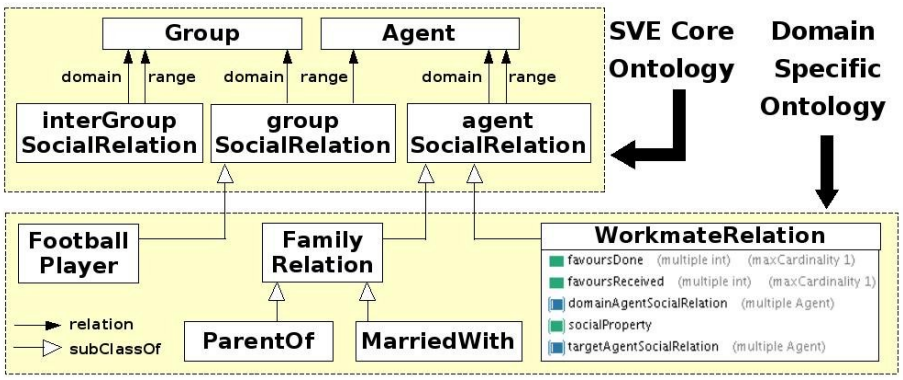


*Figura 4.4: Relaciones sociales definidas en la ontología.*

clase que lo simboliza es *groupSocialRelation*. Cuando un agente pertenezca a un grupo, una instancia de esta clase los enlazará. La red social creada por este tipo de relaciones podrá, por ejemplo, ser explorada en el caso de que el agente quiera extraer el resto de miembros del grupo. La clase *Group* es una abstracción de cualquier tipo de agregación. Por tanto, esta aproximación permite modelar desde relaciones con grupos físicos (p. ej. la clase *FootballPlayer* relacionará a los jugadores de fútbol con su equipo) hasta agregaciones mentales más sofisticadas (p. ej. los individuos de una clase social o los seguidores de una ideología religiosa). Finalmente el **nivel interinstitucional** representa las relaciones 'muchos a muchos' (*many-to-many*) y servirá para relacionar dos grupos, como hemos dicho, de índole diversa. Por ejemplo, dos ciudades virtuales hermanadas utilizarían este tipo de relación social entre grupos.

En conclusión, las ontologías permiten definir las clases de entidades que uno se puede encontrar en un mundo simulado, así como las posibles interrelaciones entre ellas. En este contexto, la construcción de un escenario particular consistirá en: a) crear los objetos como instancias de las clases definidas en la ontología; b) dar valores iniciales a las propiedades como por ejemplo la posición, las dimensiones o el estado inicial; y c) relacionar las instancias adecuadamente. Por ejemplo, una leja de un bar virtual con bebidas alcohólicas se puede representar como un objeto de tipo *HeterogeneousObjectContainer* que se relaciona a través de la relación *in* con las botellas que contiene. A su vez, cada botella será una instancia de la clase *Bottle* que tendrá un número determinado de servicios (valor de la propiedad *num*) y se relacionará con la bebida que contiene como por ejemplo el whisky o el ron (ambas instancias de la clase *Substance*). Para una descripción completa de un bar virtual habitado por camareros y clientes con relaciones sociales ver el anexo A.3, que corresponde al ejemplo presentado en la sección de resultados 6.4.

Cuanto más estructurada esté la información, mayor capacidad tendrán los agentes de percepción (sección 4.3.2) y de representación de tareas (sección 5.4.2). Una vez que hemos estudiado la representación ontológica de nuestro entorno virtual semántico, pasamos a describir el funcionamiento de la capa semántica, la cual hace de interfaz entre el mundo simulado y los agentes que lo pueblan.

### 4.3.2. La capa semántica

La capa semántica es el módulo del entorno virtual semántico que se encarga de gestionar la base de datos del mundo a lo largo de la simulación. Fundamentalmente, esta capa realiza tareas de percepción y de actuación; dos aspectos de vital importancia en la interacción entre un actor sintético y el resto de entidades de la escena. Como veremos en esta sección, la capa semántica utiliza el modelo del mundo definido en las ontologías para la consecución satisfactoria de su misión.

La **percepción** de entornos virtuales dotados de muchos objetos con los que interaccionar suele ser una tarea difícil, debido a la gran cantidad de información que contienen. Por ejemplo, en una librería virtual 3D, la percepción completa de todos los libros supondría demasiada información para que un agente inteligente pudiese realizar una búsqueda de las acciones como las que vimos en la sección 2.4. Por consiguiente, uno de los primeros objetivos consiste en reducir la cantidad de información de los escenarios 3D a un subconjunto abarcable pero suficientemente expresivo. Para conseguir esto, la capa semántica construye un **árbol de dependencia sensorial** como el que aparece en la figura 4.5. Este árbol organiza los objetos de la escena en una disposición jerárquica, en la que unos objetos delegan su percepción en otros, atendiendo a su relación de dependencia sensorial. La dependencia sensorial se ha modelado en la ontología base (*SVE Core Ontology*) mediante una *superpropiedad* OWL [2] llamada *senseDepends*. Algunas relaciones concretas que establecen, a parte de su significado semántico, una dependencia sensorial para la capa semántica son: *in*, *on* y *pickedBy*. Por ejemplo, *in* relaciona los objetos móviles (*MovableObject*) con los contenedores (*Container*), así pues, los objetos contenedores gestionarán qué información se puede percibir de los objetos que contienen.

A la hora de sensorizar el estado de una escena, la capa semántica hace un recorrido del árbol de dependencia sensorial, de manera que los objetos con responsabilidad sensorial (nodos padres del árbol) puedan filtrar la información de los objetos subordinados (nodos hijos del árbol). Como se aprecia en la definición de *senseDepends*, y por tanto de las relaciones derivadas, los objetos con responsabilidad sensorial corresponden a instancias de la clase *FilterClass*, definida en la ontología base. Esta clase permite la definición de diferentes tipos de **filtrado**, por medio de una propiedad llamada *filterType* (ver la figura 4.3). Hasta el momento actual, hemos implementado tres tipos de filtros

[2] Las propiedades OWL utilizan la misma metodología que los objetos y utilizan la herencia para clasificar los tipos de propiedades.

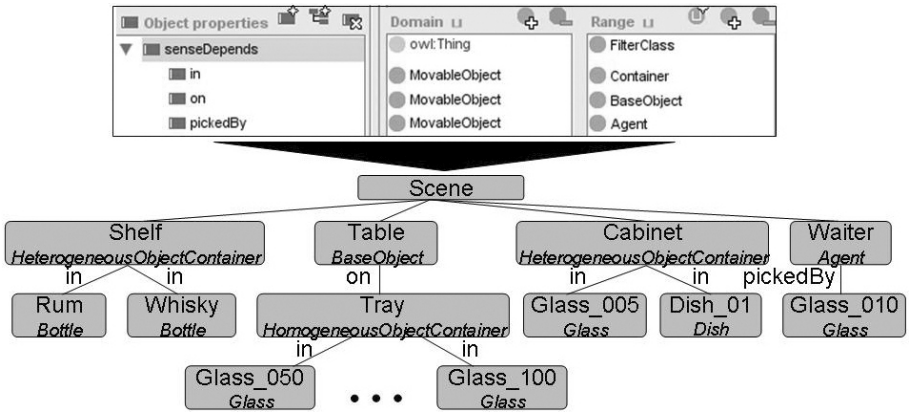


*Figura 4.5: Árbol de dependencia sensorial.*

referidos como: *ALL*, *NOTHING* y *CLASSES-ONLY*. Mientras que el filtro *ALL* deja pasar cualquier información, el filtro *NOTHING* hace imperceptible toda la información de los objetos subordinados. El tipo de filtrado es una propiedad que tendrá un valor inicial pero que puede ser cambiada a lo largo de la simulación. Este hecho permite la definición de niveles de detalle semántico en la percepción de objetos complejos. Por ejemplo, se puede modelar una vitrina como un objeto de tipo *Cabinet* que sólo publica las clases de los objetos contenidos mientras está cerrada (filtro *CLASSES-ONLY*) y que publica toda la información cuando se abre (filtro *ALL*).

Un aspecto para reducir la cantidad de información de los entornos virtuales con muchos objetos de interacción es **evitar la multiplicidad**. Hemos definido las instancias de tipo *HomogeneousObjectContainer* como aquellos contenedores que incluyen una cierta cantidad de objetos indistinguibles. Para evitar información redundante, la capa semántica permite un tipo especial de filtrado sobre esta clase de objetos. La propiedad *numBaseObjects* (figura 4.3) permite publicar la información de un número más reducido de objetos interactivos. Esta propiedad se puede ajustar dinámicamente, como ya sucedía con la propiedad *filterType*. De esta manera, la interacción se garantiza la interacción en todo momento mientras que la cantidad de información se reduce. Por ejemplo, supongamos que en un bar virtual tenemos una bandeja con 50 vasos vacíos y sólo 3 camareros que pueden cogerlos. En tal caso, tiene sentido publicar sólo el número mayor de vasos con los que los camareros pueden interactuar (p. ej. una cantidad de 6, uno por cada mano de camarero), ya que la interacción con el resto no supondría ninguna diferencia. A medida que los vasos se utilizan, y mientras que no se agoten, la bandeja publicará nuevos vasos interactivos al alcance de los camareros.

En lo que respecta a la **actuación**, la capa semántica proporciona tres servicios fundamentales: el desplazamiento de los agentes por el escenario, la interacción agente-objeto y la comunicación agente-agente. Referente al **desplazamiento**, los agentes inteligentes le pedirán el movimiento de los avatares que los representan para llegar a los puntos deseados de interacción. Relacionado con el movimiento, el mundo notificará las colisiones en caso de que se produzcan. Referente a la **interacción agente-objeto**, la capa semántica es la encargada de ejecutar las acciones solicitadas por los agentes inteligentes. En consecuencia, sobre ella recae la responsabilidad final de modificar la base de datos que almacena el estado del mundo. Para hacer esto, la capa semántica contiene los procedimientos que le permiten comprobar la aplicabilidad, el éxito o el fracaso de toda acción permitida. Cuando una acción se realiza con éxito, la capa semántica aplica los efectos adecuados sobre los objetos involucrados. Por ejemplo, si continuamos con la bandeja introducida previamente, la acción en la que un camarero coge un vaso reduce en una unidad el número de vasos contenidos en la bandeja. Además, si se coge el último vaso disponible, la propiedad del objeto que marca su cualidad de estar vacío (*empty*) será cambiada al valor verdadero (*true*). Si, por el contrario, una acción no se puede aplicar o se produce un fallo durante su ejecución, la capa semántica lo notificará al agente solicitante para que éste reaccione a la situación.

El papel que juega la capa semántica en la **comunicación agente-agente** es el de un simple canal de transmisión, de manera similar al aire en el mundo real. A pesar de que los entornos virtuales pueden estar poblados por agentes con objetivos y habilidades diversas, podemos considerarlos sistemas cerrados a nivel de comunicación; ya que los personajes comparten el conocimiento sobre cómo han de comunicarse. No estamos, por tanto, ante un escenario como Internet; en el que interaccionan una mezcla de agentes totalmente heterogéneos y se justifica el uso de intermediarios [Benford et al., 1997] o de modelos de comunicación de tipo pizarra [Omicini and Zambonelli, 1999]. Por el contrario, la comunicación de igual a igual será más adecuada en los mundos simulados. En segundo lugar, hace falta cuestionarse si esta se ha de realizar de forma directa o a través del entorno. Consideramos que la segunda opción es más realista y permite reproducir situaciones interesantes desde el punto de vista de la animación (p. ej. un actor que cotillea la conversación de otros que están hablando en un punto cercano). Por tanto, en nuestro marco de simulación la comunicación se realiza a través del entorno virtual semántico. En este contexto, la capa semántica recibe los mensajes de los agentes y los dirige a los destinatarios correspondientes, de manera que es el agente el único encargado de producir y gestionar la información intercambiada (ver la sección 4.4.2).

Después de haber analizado el entorno virtual semántico, presentamos la arquitectura de agentes socialmente inteligentes que proponemos en nuestro marco de simulación. A continuación, analizamos los módulos principales de la arquitectura para animar comportamientos autónomos inteligentes. En el capítulo siguiente, profundizaremos en las habilidades sociales de los actores sintéticos.

## 4.4. ARQUITECTURA DE AGENTES SOCIALMENTE INTELIGENTES

Como vimos a lo largo del estado del arte, la animación comportamental de los personajes 3D lleva asociada la resolución de problemas complejos de naturaleza diversa, como por ejemplo: la percepción, el control motor, la toma de decisiones, la comunicación, etc. Este hecho ha conducido al desarrollo de sistemas en los que diferentes módulo, especializados en tareas específicas, se acoplan para conseguir un comportamiento global verosímil. Las **arquitecturas modulares** son una buena solución para el diseño de 3DIVA, ya que dividen la complicada animación comportamental en trozos más pequeños y abordables. Además, estas soluciones ofrecen la posibilidad de intercambiar módulos y experimentar con diferentes técnicas de manera más sencilla. La figura 4.6 muestra un esquema general de la arquitectura modular utilizada por los agentes socialmente inteligentes de nuestro marco de simulación. Cada uno de estos agentes recibe la información sensorial de la capa semántica y calcula la secuencia de acciones apropiada para que el personaje 3D que gobierna consiga sus objetivos. Para poder hacer todo esto, contará con la ayuda de un conjunto de subsistemas que detallaremos a continuación.

El **módulo de percepción** (*Perception module*) tiene como tarea principal el procesamiento de los mensajes de percepción recibidos desde el entorno virtual semántico. Se pueden distinguir tres modos de envío de mensajes: el envío a una frecuencia fija (*on time*), el envío bajo demanda del agente (*on request*) y el envío asíncrono cuando se produce un cambio sensorial (*on property change*). A causa de la **reactividad**, demandada por el comportamiento verosímil de los personajes 3D, hemos escogido la tercera opción en nuestro marco de simulación. La percepción asíncrona informará los agentes de los cambios producidos en otros agentes y objetos en el momento en el que éstos se produzcan. Entonces, el módulo de percepción compilará estas percepciones y actualizará la base de creencias del agente en consecuencia. Por ejemplo, en base a las observaciones

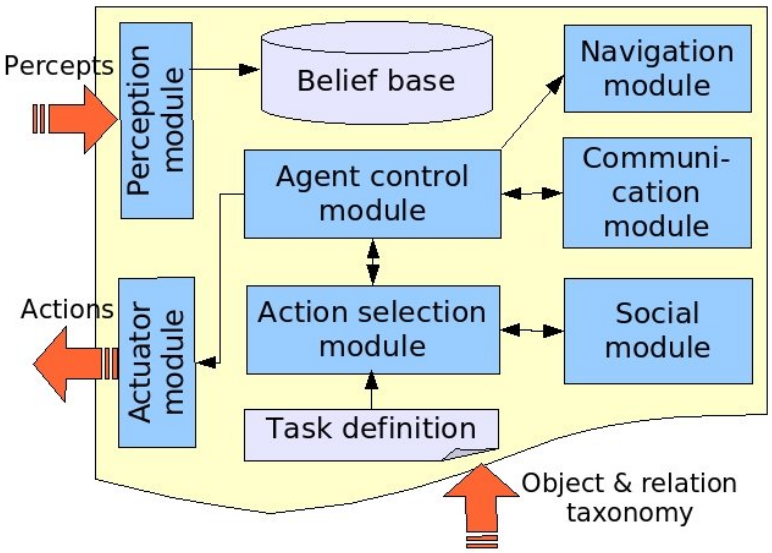


*Figura 4.6: Arquitectura de nuestros agentes socialmente inteligentes.*

de entrada y salida de un objeto en el campo de visión, el módulo de percepción puede simular fácilmente mecanismos de olvido y recuerdo, muy interesantes desde el punto de vista comportamental [Kuffner and Latombe, 1999]. En la sección 5.3.2 analizaremos el papel clave de este módulo en el mantenimiento de las creencias para la coordinación de planes en entornos multiagente.

La **base de creencias** (*Belief base*), que corresponde con la memoria del agente, almacena el estado del mundo percibido, el estado interno del agente y una representación del estado interno de otros agentes. Se trata de un contenedor dinámico de objetos que recibe y actualiza asíncronamente su información a través del módulo de percepción. La base de creencias está estructurada de acuerdo con la definición ontológica del modelo del mundo, proporcionada por el entorno virtual semántico. Esencialmente, aloja dos niveles de representación: información numérica de bajo nivel (p. ej. posición, dimensiones...) y datos simbólicos que describen las propiedades o relaciones de los objetos y agentes (p. ej. la sustancia contenida por una botella en un bar virtual). La memoria del agente será utilizada por el resto de módulos a la hora de generar la actuación del actor sintético.

El **módulo de actuación** (*Actuator module*) juega un papel de interfaz en nuestros agentes socialmente inteligentes, ya que las acciones se ejecutan en el entorno virtual semántico (ver sección 4.3.2). En consecuencia, este módulo ofrece a los agentes los mecanismos para enviar de forma asíncrona sus peticiones de actuación hacia la capa se-

mántica. Una vez realizada la petición, el agente permanecerá a la espera del resultado de la acción (*feedback* del entorno) mientras la capa semántica ejecuta el código comportamental asociado a la acción. Si ésta puede realizarse con éxito, provocará los cambios correspondientes en los objetos afectados y devolverá éxito. En caso contrario, la capa semántica retornará al agente un fallo de la acción. Independientemente del resultado de la acción, el agente detectará los cambios producidos en el mundo a través de su módulo de percepción, aunque éstos sean producto de su propia actuación.

El **módulo controlador del agente** (*Agent control module*) gobierna la actividad del actor en todo momento y decide aquello que ha de hacer dependiendo de su estado interno y del estado del mundo que lo rodea. De forma básica, está formado por una máquina de estados que atiende los aspectos reactivos del personaje 3D y que define su comportamiento proactivo esencial (p. ej. ver el controlador de la figura 5.2). De esta manera, el módulo controlador se apoya en otros módulos para realizar tareas específicas como la navegación, la toma de decisiones y la comunicación.

Además de las habilidades de actuación comportamental e interacción propias del actor que representan, los actores sintéticos deben combinar habilidades de navegación similares a las estudiadas en la robótica; esto es, la capacidad de encontrar rutas libres de obstáculos. Sin embargo, las facilidades de navegación ofrecidas por los motores gráficos (*pathnodes* o mapas de carriles libres de obstáculos introducidos *off-line*) plantean problemas serios de autonomía y realismo en las rutas recorridas, que serán siempre las mismas [Cavazza et al., 2002]. Por tanto, el **módulo de navegación** aplicado habitualmente a personajes autónomos 3D es un sistema híbrido, que combina un planificador de caminos 2D (p. ej. dla familia de algoritmos $A^*$) [Kuffner, 1999] con algún sistema local basado en: reglas [Reynolds, 1999], campos potenciales [Mamei et al., 2004], redes neuronales [Lozano, 2005], etc. La combinación de sistemas globales y locales es una solución comúnmente aceptada para los personajes 3D [Pelechano et al., 2007, Shao and Terzopoulos, 2005], ya que el sistema local aportará la reactividad necesaria, mientras que el sistema global solucionará los problemas (mínimo locales) derivados de la complejidad espacial del escenario virtual.

El **módulo de selección de acciones** (*Action selection module*) es el responsable de la toma de decisiones y, por tanto, de las acciones que finalmente serán animadas sobre el personaje 3D. Normalmente, este módulo está formado por un proceso de búsqueda de la mejor opción entre un conjunto de tareas posibles. Esta búsqueda se apoya en un **esquema**

**de acción** para seleccionar qué tareas ha de realizar un agente para conseguir sus objetivos (ver la sección 5.4.2). En este contexto, la inteligencia de los actores sintéticos proviene a menudo del comportamiento racional. Según el criterio de la racionalidad, la idoneidad de una acción corresponde con la medida en que su ejecución acerca el estado del mundo al estado deseado por el agente.

Sin embargo, el grado de verosimilitud de los actores sintéticos se verá limitado sin el comportamiento social, encargado de cumplir con los frecuentes y difícilmente definibles objetivos sociales (p. ej. coordinación, cooperación, etc.). Por una parte, es necesario que loa agentes interaccionen entre ellos. De acuerdo con esto, nuestros agentes socialmente inteligentes incorporan el **módulo de comunicación** (*Communication module*), encargado de mantener los protocolos de interacción para la comunicación agente-agente. Este módulo genera los mensajes que se enviarán al entorno para que los reparta a los destinatarios correspondientes y procesa los mensajes obtenidos como respuesta (ver la sección 4.4.2). Por otra parte, hace falta extender la selección de acciones con un **módulo social** (*Social module*), que introduzca criterios sociales en el mecanismo de toma de decisiones; aspecto que estudiaremos en el capítulo 5.

En las siguientes secciones analizamos los esquemas de acción y el módulo de comunicación incorporados en la arquitectura de agentes socialmente inteligentes, ya que los consideramos de vital importancia para la inclusión de habilidades sociales en el comportamiento de los actores sintéticos. La toma de decisiones de tipo social, donde se encuentran las aportaciones principales de esta tesis, será tratada en el capítulo 5.

### 4.4.1. Esquemas de acción

En la arquitectura de agentes propuesta, una tarea se define como la secuencia de acciones que hace falta solicitar al entorno virtual semántico para animarla completamente. De conformidad con los mecanismos de selección dinámica de acciones introducidos en la sección 2.4, hemos probado dos aproximaciones para el módulo de selección de acciones, cada una con su esquema de acción:

- Selección mediante **planificación heurística**: Dentro de esta categoría hemos utilizado el planificador *miniMin-HSP* desarrollado por Miguel Lozano para la animación comportamental de personajes inteligentes 3D [Lozano, 2005]. En esta apro-

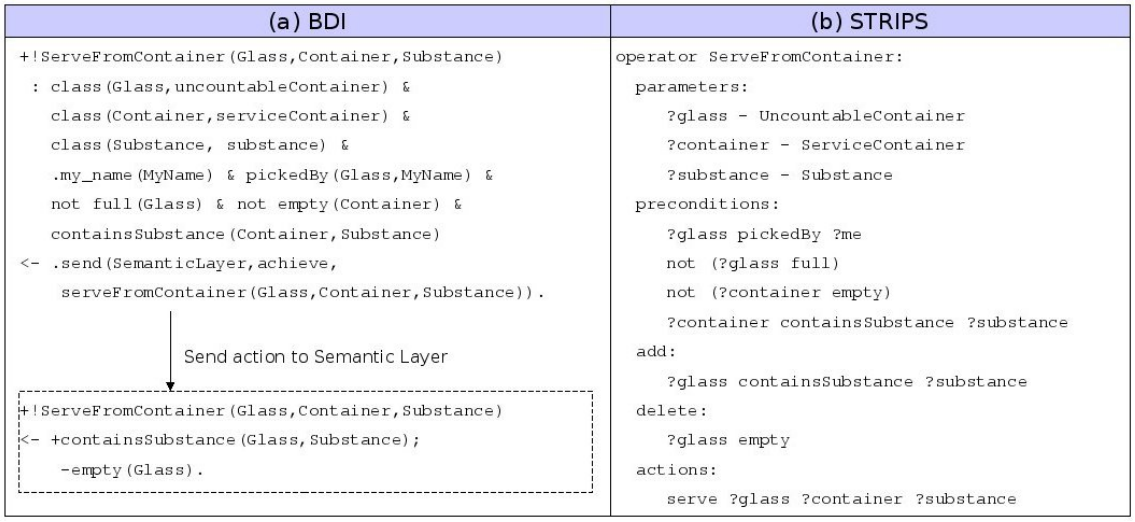


*Figura 4.7: Ejemplo de operador genérico expresado mediante la parametrización de: (a) un plan BDI o (b) un operador basado en el modelo STRIPS.*

ximación, las tareas disponibles se definen a través de operadores inspirados en el lenguaje de acción STRIPS (ver la figura 4.7b). Esto es, la descripción de una tarea se compone de unas listas de precondiciones (*preconditions*) y efectos (positivos *add* y negativos *delete*) que expresan, respectivamente, su aplicabilidad y los cambios que produce su ejecución sobre el estado del mundo y del agente.

- Selección basada en el **modelo BDI**: Para el desarrollo de estos agentes hemos utilizado *Jason* [Bordini and Hübner, 2007], una implementación del lenguaje de programación *AgentSpeak* [Rao, 1996] para sistemas multiagente. En *AgentSpeak*, los agentes gestionan su operatividad a través de un conjunto de planes que describen la secuencia de acciones a realizar para conseguir las intenciones que aparecen durante el tiempo de vida del agente. La figura 4.7a muestra un ejemplo de un plan en *Jason* para servir una sustancia en un vaso. El plan tiene el siguiente formato *evento : contexto ->cuerpo* donde: a) el *evento* define el objetivo reactivo o proactivo que provoca su ejecución; b) el *contexto* describe la situación en la que el plan es aplicable; y c) el *cuerpo* establece las acciones a realizar.

Los esquemas de acción arriba comentados no tienen a menudo en cuenta un aspecto muy importante de los entornos con muchos objetos de interacción: la posibilidad de **reutilizar los operadores** en contextos diferentes. Por ejemplo, los personajes 3D normalmente definen un operador para cada una de las tareas que pueden llevar a cabo. Entonces, uno se podría encontrar un barman virtual capaz de servir un zumo de naranja y una copa

de vino pero sin la más remota idea de cómo preparar una taza de té. No obstante, tal y como demostraron L. Levison [Levison, 1996] y N.I. Badler [Badler et al., 2000] desde la *Universidad de Pennsylvania*, la clasificación de los objetos en categorías puede ser utilizada para definir operadores genéricos que puedan ser aplicados sobre objetos diferentes. De acuerdo con esto, la definición de las tareas de nuestros agentes inteligentes ha sido revisada para incorporar la información semántica de la ontología, que representa el modelo abstracto del mundo.

En nuestro marco de simulación, la taxonomía de objetos definida en las ontologías del entorno virtual semántico es usada por los agentes socialmente inteligentes para definir **operadores parametrizados**. La inclusión de parámetros en el esquema de acción permite definir operadores que sean reusables, independientemente del formalismo de toma de decisiones. Por ejemplo, la figura 4.7 muestra un operador genérico expresado mediante un plan BDI así como un operador basado en el modelo STRIPS. Este operador permite que un actor sintético, como por ejemplo un camarero virtual, pueda servir cualquier sustancia (instancia de la clase *Substance*) que esté contenida en un contenedor de tipo *ServiceContainer* dentro de un contenedor de objetos incontables (*UncountableContainer*).

Por un lado, el plan BDI de la figura 4.7a incluye, en las condiciones de contexto, las clases a las que deben pertenecer los objetos para poder ejecutar las acciones sobre ellos. Por ejemplo, el plan *ServeFromContainer* se puede aplicar tanto para servir un vaso de vino de una botella como una taza de café de una cafetera, siempre que estos objetos sean instancias de las clases antes referidas. La taxonomía de objetos definida en la ontología se utiliza, así pues, para clasificar los objetos interactivos y para obtener sus propiedades y relaciones. Posteriormente, la capa semántica será quien ejecute las acciones y aplique los efectos sobre los objetos involucrados. Por otro lado, la figura 4.7b muestra la definición del operador *ServeFromContainer* basada en el modelo STRIPS. De manera similar a PDDL [pdd, 2008], hemos incluido una sección separada de parámetros (*parameters*), que relaciona los objetos con las clases adecuadas de la ontología. El resto de secciones (*preconditions*, *add*, *delete*) son las clásicas del modelo STRIPS. Finalmente, la sección con nombre *actions* contiene las acciones que serán solicitadas al entorno.

La parametrización, que fundamentalmente introduce variables y tipos que hacen referencia a la jerarquía de clases de la ontología, ahorra la definición de un operador para cada objeto de interacción, cosa que en última instancia se ve reflejada en un grado más

alto de interacción. Además, la interoperabilidad del agente entre escenarios diferentes se ve mejorada, ya que los agentes serán capaces de manipular cualquier objeto del mundo, siempre que éste sea una instancia de una clase definida en la ontología.

Tanto la planificación heurística como el modelo BDI son aproximaciones que se han utilizado en la animación de comportamientos racionales en personajes 3D (ver la sección 2.4). En consecuencia, las aportaciones principales de esta tesis han sido la inclusión de habilidades sociales en ambos paradigmas (capítulo 5). Como cierre de este capítulo explicamos brevemente el módulo de comunicación, componente en el que se ayudan estas técnicas sociales.

### 4.4.2. Módulo de comunicación

El módulo de comunicación es el componente que se encarga de guiar a los personajes durante las interacciones con el resto de actores sintéticos. En un entorno virtual poblado por diversos agentes, la necesidad de interaccionar puede provenir de un comportamiento reactivo o ser el resultado de un toma de decisiones proactiva. Como ejemplo reactivo, cuando un agente se dirige a coger un objeto 3D y otro se lo arrebata, el primero podría preguntar al segundo se se lo deja o cuándo lo liberará. Como ejemplo proactivo, diversos agentes emplazados en un espacio común podrían comunicar sus objetivos con vistas a colaborar en la consecución de ciertas tareas comunes.

En este contexto, la comunicación agente-agente se modela mediante **protocolos de interacción**. Fundamentalmente, estos protocolos son máquinas de estados finitos que identifican los estadios en los que los agentes pueden estar a lo largo de la comunicación, así como los mensajes intercambiados entre los interlocutores. Así pues, el módulo de comunicación almacena los protocolos de interacción y se encarga de la dinámica. Es decir, genera los mensajes (actos del habla o *speech-acts*) correspondientes al estado de la comunicación y recibe las respuestas que hacen avanzar el protocolo de interacción.

Los protocolos incluidos en el módulo de comunicación han de ser adecuados para la comunicación entre humanos virtuales. No estamos ante la comunicación entre agentes telemáticos, donde la importancia recae sobre el intercambio de información y la concisión del protocolo. Por el contrario, los agentes socialmente inteligentes están interesados en animar conversaciones y situaciones verosímiles entre personajes 3D. Esto hace que

los protocolos se definan a menudo como la combinación de otros protocolos básicos de interacción definidos por FIPA [FIPA (Foundation for Intelligent Physical Agents), 2008] (p. ej. *Request*, *Propose*, *Query*, etc).

Por ejemplo, la figura 4.8 contiene dos protocolos utilizados en la coordinación y la colaboración de agentes basados en planificador heurístico, que describiremos en el apartado 5.3. La figura 4.8a describe la conversación que se produce, de forma reactiva, cuando un agente interrumpe la tarea de otro. En este diálogo, el agente afectado pregunta al primero qué está haciendo y usa esta información para evitar colisiones futuras. Como se puede ver, este protocolo se puede descomponer en dos protocolos FIPA básicos: el protocolo de propuesta (*Propose*) y el protocolo de pregunta (*Query*). La figura 4.8b muestra el protocolo que utilizan los agentes para publicar sus objetivos y establecer puntos de colaboración. En ambos ejemplos, los primeros mensajes de saludo son, en principio, superfluos. Sin embargo, son mensajes que aportan realismo a la interacción y, por tanto, a la animación.

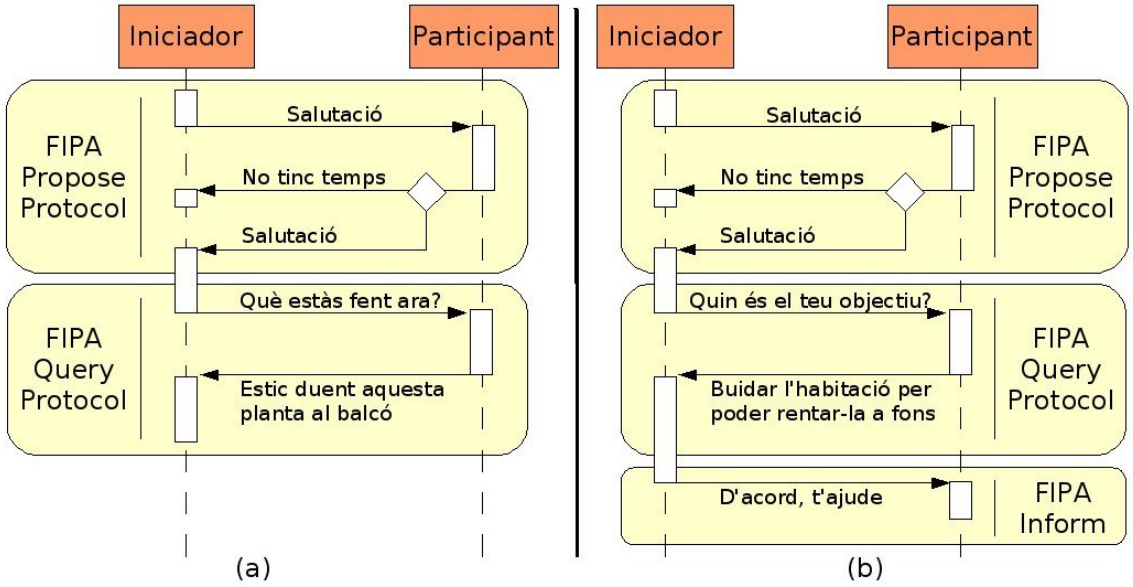


*Figura 4.8: Ejemplos de conversaciones entre dos agentes para: (a) conocer la tarea actual de otro agente, (b) conocer los objetivos de otro agente.*

Finalmente, el módulo de comunicación también incluye protocolos más complejos con diversos participantes. Por ejemplo, la toma de decisiones MADeM, que presentaremos en el apartado 5.4, utiliza el protocolo *FIPA-Contract Net* para realizar subastas entre diferentes grupos de agentes. Asimismo, el módulo de comunicación aporta dife-

rentes protocolos de conversación entre los agentes para animar acuerdos y desacuerdos que se hayan producido como resultado de la subasta (ver la sección 6.4.4).

## 4.5. CONCLUSIONES

En este capítulo, hemos presentado las aportaciones menores de esta tesis, recogidas en el **marco de simulación multiagente** diseñado para el desarrollo de entornos virtuales 3D habitados por actores sintéticos con habilidades sociales. Por un lado, hemos analizado el modelo general de entorno virtual semántico incluido en este sistema. Esta aproximación utiliza las **ontologías** como una solución extensible y reutilizable para la representación del entorno y de las relaciones entre los actores. Por tanto, el **conocimiento semántico** beneficia la **producción**, la **percepción** y la **interacción** de los personajes en escenarios con numerosos objetos interactivos. Por otro lado, hemos presentado una arquitectura de agente capaz de gobernar el comportamiento de personajes 3D socialmente inteligentes (p. ej. percepción, control, navegación, etc.). En este contexto hemos estudiado el uso de información semántica para definir **esquemas de acción generales** para la interacción agente-objeto, así como los **protocolos de comunicación** para modelar la interacción agente-agente.

En resumen, hemos analizado el conjunto de aportaciones colaterales que sostienen las aportaciones principales de esta tesis en materia de toma de decisiones sociales, que son tratadas en el siguiente capítulo.

CAPÍTULO 5

# TOMA DE DECISIONES SOCIALES

Este capítulo contiene las aportaciones principales de esta tesis en lo que se refiere a la integración de habilidades sociales en el comportamiento de los actores sintéticos. Primeramente, analizamos los requerimientos sociales de los actores virtuales inteligentes 3D. En segundo lugar, presentamos dos aproximaciones para la toma de decisiones de tipo social: una a nivel *micro* (centrada en el agente) y otra a nivel *macro* (centrada en la sociedad). A nivel *micro*, proponemos una técnica de **colaboración con planificadores heurísticos**. Mediante la detección de conflictos y sinergias con otros agentes, esta técnica dota a los actores con la habilidad de adaptarse a los entornos con recursos compartidos por diversos caracteres autónomos. Con una perspectiva *macro*, presentamos **MADeM (*Multi-modal Agent Decision Making*)**, un proceso de toma de decisiones de tipo social que puede ser utilizado por agentes de tipo BDI a la hora de tomar **decisiones socialmente aceptables**. Así pues, MADeM permite a los agentes reproducir comportamientos deseados a nivel de grupo como por ejemplo: el bienestar social, las actitudes respecto a los otros, etc.

## 5.1. INTRODUCCIÓN

La toma de decisiones llevada a cabo por los personajes 3D, a la hora de animar su comportamiento, a menudo ha seguido criterios racionales autointeresados. Este hecho se debe a que la verosimilitud de un actor sintético en un contexto aislado (sin la interferencia de otros actores externos) depende, en gran medida, e la eficiencia en la consecución de sus objetivos. Sin embargo, en los entornos virtuales habitados, los actores necesitan incluir criterios adicionales que representen sus capacidades sociales.

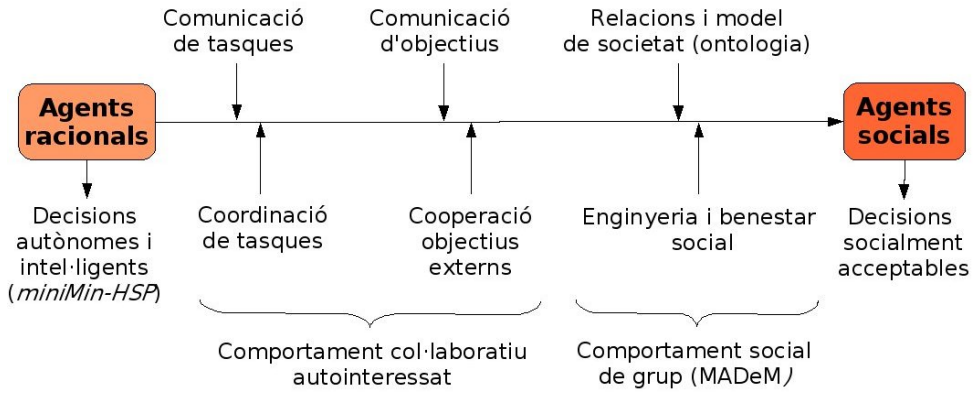


*Figura 5.1: Espectro de decisiones desde los agentes racionales a los agentes sociales.*

La figura 5.1 muestra el espectro de decisiones entre los agentes racionales y los agentes sociales. El comportamiento inteligente de los agentes racionales es insuficiente en los entornos con recursos compartidos, en los que los conflictos pueden aparecer rápidamente a causa de la voluntad de interactuar con los mismos recursos de manera simultánea (ver los ejemplos del punto 1.1). Por consiguiente, el intercambio de información alrededor de las tareas y de los objetivos de los agentes externos puede servir para incorporar **comportamientos de grupo orientados a la eficiencia** como por ejemplo la **coordinación** o la **cooperación**. Por otro lado, los humanoides establecen relaciones con otros miembros de la sociedad artificial, no necesariamente orientadas a la eficiencia pero que pueden aportar realismo a la animación (p. ej. una pandilla de amigos virtuales que prefieren hablar que trabajar), y que uno espera ver reflejadas en su comportamiento. Por tanto, los actores inteligentes necesitan evaluar también el impacto social de sus acciones y decidir su actuación teniendo en cuenta las relaciones establecidas, cosa que los conducirá hacia una toma de **decisiones socialmente aceptables**.

De acuerdo con esto, en el apartado 5.2 analizamos los **requerimientos sociales de los actores virtuales inteligentes 3D**. Por un lado, los agentes han de ser capaces de detectar las dependencias que se producen entre sus acciones y las del resto de cohabitantes (ver la sección 5.2.1). Por otro lado, hace falta que los agentes realicen una toma de decisiones de tipo social. Es decir, que incluya las acciones y los intereses de los otros agentes en la selección de las acciones a realizar (ver la sección 5.2.2). El marco de simulación descrito en el capítulo 4 proporciona los módulos necesarios para que ambos aspectos puedan ser tratados.

En este capítulo, presentamos dos aproximaciones para la incorporación de criterios de tipo social en dos mecanismos bien conocidos para la selección dinámica de acciones. En primer lugar, el apartado 5.3 define una técnica de **colaboración basada en planificadores heurísticos**, para que los actores se adapten de forma eficiente en entornos con recursos compartidos. Como muestra la figura 5.1, esta técnica utiliza la comunicación directa entre los actores (sección 5.3.1), la coordinación el el uso de los recursos (sección 5.3.2) y la cooperación para conseguir objetivos externos (sección 5.3.3). En segundo lugar, en el apartado 5.4 presentamos **MADeM (*Multi-modal Agent Decision Making*)**. MADeM es un proceso de toma de decisiones que permite evaluar diversos puntos de vista (*multimodalidad*), así como consultar las preferencias de otros agentes alrededor de las diferentes acciones a realizar (secciones 5.4.2 y 5.4.3). Este *feedback* social nutre un mecanismo de selección de decisiones socialmente aceptables (sección 5.4.4), que permite reproducir comportamientos deseados a nivel de grupo como por ejemplo: el bienestar social, las actitudes hacia los otros, etc. (ver la figura 5.1).

Las dos aportaciones sociales arriba comentadas han sido probadas de manera independiente. Como ya se dijo en el capítulo 2, consideramos tanto la planificación heurística como el modelo BDI soluciones adecuadas para la toma de decisiones en personajes 3D. De hecho, pueden ser complementarias. Por ejemplo, se puede pensar en combinar la planificación para la secuenciación de tareas en el bajo nivel y el modelo BDI para el mantenimiento de los objetivos de más alto nivel. Desgraciadamente, la integración de ambas técnicas queda fuera del alcance de este documento y queda como trabajo futuro de esta tesis (ver el punto 7.2).

## 5.2. REQUERIMIENTOS SOCIALES DE LOS 3DIVA

A pesar de que algunos entornos virtuales tienden a simular actores sintéticos que realizan tareas de una manera independiente, esta suposición reduce su verosimilitud porque constituye una simplificación muy fuerte. Bien al contrario, las tareas llevadas a cabo por los personajes 3D suelen ser dependientes, cosa que hace que su ejecución afecte la operativa de otros individuos y produzca problemas de **coordinación**.

En los inicios de su estudio, la *Teoría de la Organización* definió la coordinación como la gestión de dependencias entre actividades (p. ej. el flujo de trabajo de una empresa para

manufacturar un producto). Más recientemente, D'Inverno y Luck [D'Inverno and Luck, 2004] han planteado una formalización de las diferentes relaciones que se pueden dar en un sistema multiagente. Dentro de este modelo, el proceso de coordinación consiste en dos tareas fundamentales que analizaremos a continuación: la **detección de las dependencias** y la **toma de decisiones** sobre la acción de coordinación que conviene utilizar.

### 5.2.1. Detección de dependencias

En un sistema multiagente, las dependencias se manifiestan en forma de interacciones entre dos o más agentes, los cuales establecen una especie de relación dinámica a través de un conjunto de acciones recíprocas. En tal caso, las interacciones producirán una serie de acciones que influirán en el comportamiento futuro del agente.

Se pueden identificar diversos **tipos de interacción**; en función de los objetivos de los agentes (compatibles o incompatibles), de sus habilidades y de los recursos disponibles (suficientes o insuficientes) [Ferber, 1999]. La tabla 5.1 muestra esta clasificación, en la que dos objetivos son incompatibles si conseguir uno de ellos conlleva que el otro no podrá ser alcanzado y son compatibles si no se cumple esta condición. Cuando los objetivos son compatibles estamos delante de una situación de *cooperación*, mientras que si son incompatibles aparece el *antagonismo*.

La *Independencia* no supone realmente un problema de coordinación. Por ejemplo, 'a priori', no es necesaria la interacción entre personajes 3D que se encuentran en habitaciones distintas o que trabajan en problemas diferentes en un entorno sin carencia de recursos. La *Colaboración simple* surge cuando los agentes necesitan añadir las habilidades de otros para complementar su insuficiencia, como es el caso de los actores que se ayudan para mover un objeto pesado de una estancia a otra (ver la figura 3.1b). Sucede una *Obstrucción* cuando un agente se interpone en el camino de otro, a pesar de que no se necesiten para cumplir con sus tareas. Un ejemplo es la disputa de dos carpinteros virtuales por un mismo martillo expuesta en el punto 1.1, que requerirá el uso de técnicas de coordinación de acciones. Cuando tanto los recursos como las habilidades son insuficientes, hace falta una *Colaboración coordinada*. Esto es, cooperaciones más complejas en las que se lleve a cabo tanto la coordinación de acciones como el reparto de tareas.

Si los agentes tienen suficientes habilidades y recursos pero sus objetivos son incom-

| **Categoría** | **Objetivos** | **Recursos** | **Habilidades** | **Tipos de situación** |
|---|---|---|---|---|
| Indiferencia | Compatibles | Suficientes | Suficientes | Independencia |
| Cooperación | Compatibles | Suficientes | Insuficientes | Colaboración simple |
| | Compatibles | Insuficientes | Suficientes | Obstrucción |
| | Compatibles | Insuficientes | Insuficientes | Colaboración coordinada |
| Antagonismo | Incompatibles | Suficientes | Suficientes | Competición individual pura |
| | Incompatibles | Suficientes | Insuficientes | Competición colectiva pura |
| | Incompatibles | Insuficientes | Suficientes | Conflicto individual de recursos |
| | Incompatibles | Insuficientes | Insuficientes | Conflicto colectiu de recursos |

*Tabla 5.1: Tipos de interacción en los sistemas multiagente [Ferber, 1999].*

patibles, la situación se reconoce como una *Competición individual pura*. Entonces los agentes han de negociar para superar la incompatibilidad de objetivos. Un ejemplo podría ser una competición deportiva, en la que todos los participantes compiten para ganar. La situación en la que los agentes no disponen de todas las habilidades necesarias se llama *Competición colectiva pura*. En este contexto, los agentes deberán agruparse y colaborar para enfrentarse con otros grupos; como por ejemplo en una carrera de relevos. Cuando la carencia se encuentra en los recursos, estamos ante un caso de *Conflicto individual de recursos*. Por ejemplo, en el juego del *Risk* el recurso insuficiente es el territorio. Finalmente, si ambos, habilidades y recursos, son insuficientes aparece un *Conflicto colectivo de recursos*. En el caso de una guerra virtual, cada agrupación necesita una colaboración coordinada dentro del grupo para que las coaliciones luchen para obtener los recursos.

Las condiciones que ha de cumplir un sistema multiagente para permitir las interacciones arriba descritas se resumen en los siguientes puntos:

- Presencia de agentes capaces de actuar y de comunicarse.
- Existencia de situaciones en las que los agentes interactúen (p. ej. movimiento de agentes, disputa por recursos limitados, etc.).

- Elementos que permitan el desarrollo de relaciones (p. ej. canales de comunicación, acceso o denegación de acceso a recursos, etc.).
- Autonomía de los agentes para iniciar, mantener y acabar las susodichas relaciones.

Como se puede apreciar, el marco de simulación multiagente presentado en el capítulo 4 incluye todas estas características. Por tanto, los entornos virtuales habitados creados con él pueden reproducir todas las situaciones de interacción que hemos definido anteriormente. No obstante, el estado actual de los 3DIVA todavía se encuentra lejos de poder tratarlas. Para hacerlo, los actores necesitan incorporar mecanismos de coordinación de grupos a diferentes niveles (p. ej. navegación, toma de decisiones, etc.).

De acuerdo con esto, la investigación llevada a cabo en esta tesis se ha centrado en aquellas situaciones en las que los objetivos son compatibles y los actores sintéticos son benevolentes. Esta no es una suposición muy inadecuada en los mundos virtuales, ya que la compatibilidad y la benevolencia se pueden ver como el resultado de una negociación previa entre personajes con objetivos incompatibles (p. ej. mediante un proceso de regateo). Sin embargo, a pesar de que los agentes estén de acuerdo en lo que respecta a unos objetivos comunes, es necesario especificar todavía los detalles de realización: qué tareas realiza cada uno, a quién informan cuando se consigan, etc. Estas cuestiones deberán ser resueltas por los agentes virtuales inteligentes haciendo uso de una toma de decisiones coordinada.

### 5.2.2. Toma de decisiones coordinada

Una vez que los agentes son conscientes de la necesidad de adaptar sus planes a la presencia de otros individuos, han de decidir cómo coordinar su actuación de acuerdo con las dependencias detectadas. Se puede concebir la coordinación a diferentes niveles de granularidad: el nivel *micro* y el nivel *macro* [ANA MAS, 2005].

La coordinación con una perspectiva *micro* se centra en el agente y se entiende como una forma de adaptación al entorno. Es decir, cada agente monitoriza las dependencias (potenciales) con los objetivos, planes o acciones del resto de agentes. Entonces, determina qué acciones ha de realizar para llegar al estado objetivo. Por ejemplo, consideremos un agente que utiliza un planificador STRIPS para resolver un problema en el mundo de

bloques [Nilsson, 2001]. En un contexto multiagente, este agente deberá acomodar sus planes para aprovechar las posibles sinergias (p. ej. otros agentes pueden colocar bloques por él). La toma de decisiones coordinada en términos de dependencias entre las acciones de los agentes recibe el nombre de **razonamiento social**.

La perspectiva *macro* enfoca el problema de la toma de decisiones coordinada desde el punto de vista de un observador externo. La cuestión esencial es cómo modificar el comportamiento para que se produzcan características deseadas en el comportamiento del sistema y/o de sus agentes (p. ej. bienestar social, eficiencia, comportamiento estratégico, etc.). La **ingeniería social** estudia las diferentes maneras de influir en las decisiones de los agentes para reproducir los patrones deseados por los diseñadores (p. ej. normas o leyes sociales [Shoham and Tennenholtz, 1995]). Es importante darse cuenta que influir en las decisiones no significa determinar totalmente el comportamiento de un agente, sino sólo predisponerlo en una dirección deseada. Por tanto, este tipo de aproximaciones sociales no se oponen de ninguna de las maneras a la noción fundamental de autonomía de un agente [Wooldridge and Jennings, 1995].

Como hemos dicho a lo largo de este documento, los criterios racionales autointeresados no son suficientes a la hora de simular sociedades de actores sintéticos con un comportamiento verosímil. Por consiguiente, la arquitectura de los agentes socialmente inteligentes descrita en el capítulo 4 incluye el **módulo social** (ver la figura 4.6). Este módulo es el encargado de incorporar criterios de tipo social o coordinados en la toma de decisiones de los personajes 3D. En lo que queda de capítulo explicamos las dos aportaciones fundamentales de esta tesis para la toma de decisiones sociales. A continuación, presentamos una técnica a nivel *micro* para conseguir comportamientos colaborativos en agentes basados en planificadores heurísticos. En el apartado 5.4, exponemos un proceso de nivel *macro* para la toma de decisiones socialmente aceptables llamado MADeM (*Multi-modal Agent Decision Making*).

## 5.3. COLABORACIÓN CON PLANIFICADORES HEURÍSTICOS

En este apartado, presentamos una técnica para conseguir comportamientos colaborativos en aquellos actores sintéticos que utilizan los planificadores heurísticos [Bonet et al., 1997] como el mecanismo de selección dinámica de acciones.

Según el análisis realizado en la sección 3.3.3, se puede afirmar que un conjunto de agentes colabora cuando los individuos se comunican de manera directa y cooperan con el resto de agentes para alcanzar sus objetivos. De acuerdo con esto, en la sección 5.3.1, exponemos cómo los personajes 3D intercambian información mediante **protocolos de comunicación directa** para construir una representación de los agentes externos; formada por un conjunto de creencias sobre sus acciones y sus objetivos. Esta información sirve para que los agentes coordinen sus acciones, se agrupen de forma dinámica en equipos de trabajo y cooperen para cumplir con sus objetivos.

En este contexto, explicamos dos mecanismos para incluir criterios sociales cooperativos en la toma de decisiones de los agentes virtuales inteligentes. Ambos son independientes tanto del planificador como de la función heurística y han sido implementados sobre el planificador *miniMin-HSP*, desarrollado por Miguel Lozano para la animación comportamental de personajes inteligentes 3D [Lozano, 2005]. En primer lugar, la sección 5.3.2 trata sobre la coordinación de tareas basada en una fase de ***pre-planificación***, que evita los conflictos potenciales que se puedan producir en el uso de los recursos compartidos. En segundo lugar, en la sección 5.3.3 proponemos un mecanismo de cooperación basado en el **sopesado de la función heurística**. Fundamentalmente, esta aproximación consiste en dividir el objetivo global en subobjetivos y asignar un peso a cada uno de ellos, de manera que refleje su importancia social. Así pues, los agentes pueden adoptar dinámicamente objetivos externos (de menos importancia que los propios) y conseguir comportamientos cooperativos sin aumentar la complejidad de sus decisiones.

### 5.3.1. Comunicación directa y agrupación entre actores

Un conjunto de actores sintéticos, que realice tareas en un entorno virtual compartido, necesita trabajar en equipo para aumentar su eficiencia; asociada a menudo con la verosimilitud de sus comportamientos. El trabajo en equipo (*teamwork*) ha sido estudiado profundamente a raíz de la *teoría de las intenciones conjuntas* [Cohen and Levesque, 1991] (ver la sección 3.3.3). Esta teoría afirma que la creación de un equipo necesita que cada miembro se comprometa con un objetivo común y que reciba el compromiso del resto de individuos.

Los sistemas que hacen trabajo en equipo normalmente establecen los equipos de antemano y proporcionan a los agentes la consciencia de formar parte de un equipo

[Grosz et al., 1999, Giampapa and Sycara, 2002]. La definición persistente de los equipos, sin embargo, no es adecuada en los mundos virtuales con un cierto grado de dinamismo. En estos contextos, los caracteres pueden cambiar sus objetivos así como hacer y deshacer **asociaciones temporales** para conseguirlos de una manera más eficiente (*ephemeral associations* [Bouron and Collinot, 1992]). Consideremos, por ejemplo, un entorno que simula la construcción de un edificio 3D poblado por un cierto número de albañiles, cada uno ayudado por un peón. Supongamos, además, que diversos albañiles encargan a su peón que prepare una carretilla de cemento y que los recursos para hacerlo sean compartidos (p. ej. los sacos de cemento, la arena, el agua, etc.). Al principio, los peones podrían no ser conscientes de esta asignación común, entonces no habría ningún equipo formado y los agentes actuarían de forma independiente. Sin embargo, a medida que se dieran cuenta de este objetivo común, se comprometerían a conseguirlo de forma conjunta. Posteriormente, si alguno de estos peones fuera mandado a mover un palet de azulejos, debería dejar el grupo de trabajo en el que estaba inmerso.

El proceso de agrupación entre los actores sintéticos necesita la resolución de diversos subproblemas, como por ejemplo: la detección de dependencias, el intercambio de información mediante conversaciones y la identificación de sinergias entre los objetivos de los personajes. Referente a la **detección de dependencias**, hace falta que los personajes se den cuenta de cuando su actividad está siendo afectada o interrumpida por las acciones de otros individuos. La figura 5.2 muestra la máquina de estados genérica que hemos implementado para gobernar la actuación de los personajes 3D basados en el planificador *miniMin-HSP* y detectar sus dependencias [1]. El estado principal es *SEARCH*, el cual invoca al planificador para que tome la decisión de la siguiente tarea a realizar. Su ejecución (estado *WORKING*) puede necesitar el movimiento del agente (estado *NAVIGATE*) o la realización de acciones sobre los objetos del entorno (p. ej. coger un objeto), de las cuales hay que esperar el resultado (estado *WAITING*). Dentro de este controlador, la necesidad de comunicación se genera cuando una acción falla a causa de la interferencia de otro agente (p. ej. un actor coge un objeto que otro carácter deseaba). Los actores que definen su operativa basada en el modelo STRIPS pueden detectar las interferencias mediante la comprobación de las precondiciones de las tareas, no sólo durante el proceso de búsqueda sino también a lo largo de su ejecución. Para resolver esta dependencia, el carácter deberá de comunicarse con aquel que interfirió en el éxito de la ejecución (estado *COMMUNICATE*).

[1] Para una definición completa de la arquitectura de los agentes ver la sección 4.4


*Figura 5.2: Detección de dependencias en el módulo controlador de los agentes basados en el planificador* miniMin-HSP.

Una vez identificada la necesidad de comunicación (p. ej. una precondición ha sido violada) es necesario determinar qué tipo de conversación o protocolo de comunicación es el adecuado a la situación. Por ejemplo, el **protocolo de pregunta de la tarea actual**, descrito en la figura 4.8a, es interesante cuando las interferencias entre los personajes son esporádicas (p. ej. menores a una frecuencia fijada). Por medio de este diálogo, el agente que ha interferido comunica la tarea que se dispone a realizar (p. ej. *mover Silla_1 desde el Dormitorio a la Cocina*). Esta información entra en la memoria del agente interrumpido en forma de creencias sobre las tareas actuales de otros agentes (ver la figura 5.5). Así, los actores rellenan su memoria interna con la representación de otros actores externos.

Cuando se trabaja con creencias sobre tareas ejecutadas por otros personajes 3D, es necesario resolver dos cuestiones:

**¿En qué circunstancias una información se ha de convertir en una creencia?:** Cualquier información, obtenida como resultado de una comunicación con otro agente, no debería de convertirse directamente en una creencia en la memoria de los agentes; ya que este funcionamiento podría llevar al actor a un estado mental incoherente. Por ejemplo, imaginemos que en una conversación previa, un agente $A_0$ ha sido informado de que un objeto $O_1$ será dejado encima de una mesa $T_1$ por un agente $A_1$. Posteriormente, otro agente $A_2$ le transmite su intención de colocar un objeto diferente $O_2$ encima de la misma mesa $T_1$. Entonces, si suponemos que la mesa $T_1$ sólo puede alojar un único objeto, no tiene sentido que el agente $A_0$ crea a ambos interlocutores. En un contexto carente de marcas temporales que aseguren cuando

se realizarán las tareas (como es el caso del planificador *miniMin-HSP*), no es coherente dar validez a ambas afirmaciones, ya que llevan a una situación imposible. Así pues, la decisión sobre la **credibilidad** implica un proceso de razonamiento sobre la información recibida. Una simplificación razonable es el hecho de creer que el primer agente que comunicó sus intenciones lleva una cierta ventaja sobre el segundo; de tipo físico (p. ej. empezó antes la tarea) o de tipo personal (p. ej. fue el primero en comunicar que lo haría). A pesar de que este criterio no garantiza que la decisión sea la correcta, es muy sencillo y funcional. En todo caso, estamos hablando de creencias, y los personajes 3D podrán también estar equivocados, como a menudo lo estamos los seres humanos. Por tanto, después de recibir un mensaje de comunicación, el agente comprobará la compatibilidad de la nueva información con las creencias previamente adoptadas. Esta comprobación mirará si los objetos sobre los cuales actúa la tarea no estaban ya siendo afectados por la tarea de alguna creencia previa. En resumen, el actor confía en el primer agente que notifica sus intenciones y descarta las creencias posteriores que afecten a los mismos recursos.

**¿Cuándo una creencia deja de serlo?:** La duración de las creencias es limitada, sin embargo, los personajes que nos las comunicaron no volverán, habitualmente, para informarnos sobre si han conseguido o no sus propósitos. Por tanto, es el agente quien, de forma unilateral y mediante la observación del entorno, ha de decidir cuando una creencia ha de ser eliminada de la memoria de agentes. La figura 5.3 muestra el mecanismo implementado al efecto. Cada vez que una creencia se introduce en la memoria, el agente marca todos los objetos afectados por ella. De esta manera, cuando percibe alguna cosa relacionada con un objeto marcado, el agente hace una **validación** de la creencia asociada. En un contexto de definición de operadores de tipo STRIPS, la validación consiste en comprobar las precondiciones de la tarea creída. No importa si la tarea ha sido completada con éxito o si se ha interrumpido; las precondiciones no continuarán cumpliéndose. Este tipo de validación, por tanto, no discrimina el éxito o el fracaso de una tarea (cosa innecesaria ya que el agente percibe el estado del mundo que es lo que importa), sino simplemente reconoce cuando una creencia ya no es viable y ha de ser eliminada de la memoria (invalidación en la figura 5.3).

En la sección 5.3.2 exponemos cómo los agentes virtuales inteligente pueden utilizar esta representación en memoria de los agentes externos para coordinar el acceso a los recursos compartidos y mejorar, pues, la calidad de la animación comportamental.

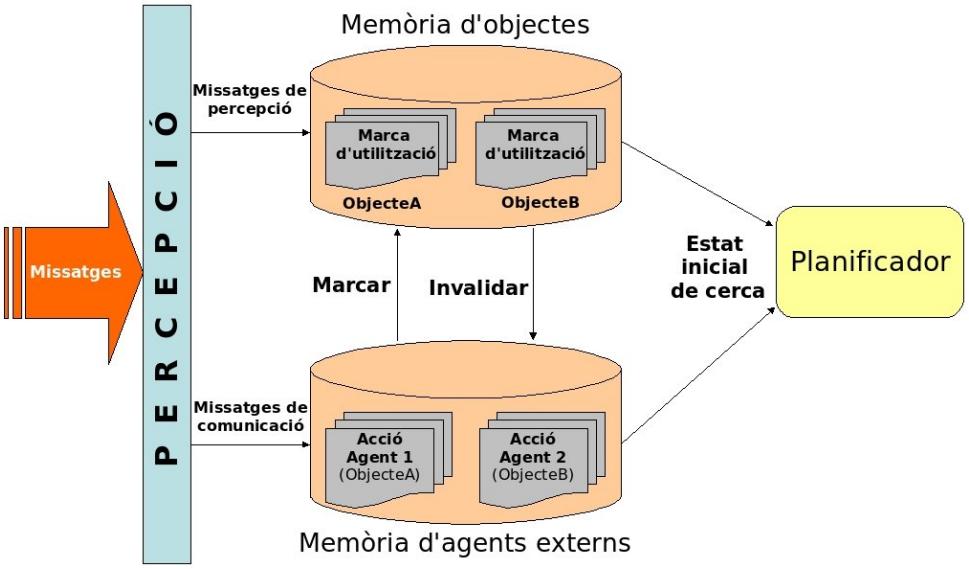


*Figura 5.3: Estructura de la memoria de los actores sintéticos.*

## *Formación de equipos*

Los personajes que no se interfieren muy a menudo, normalmente, no necesitan cooperar para alcanzar sus objetivos. Sin embargo, conviene que aquellos que interrumpen constantemente sus actividades consideren la posibilidad de formar un equipo temporal de trabajo en el que los miembros cooperen para conseguir un objetivo conjunto con más eficacia. Esta evaluación se hará a través del **protocolo de formación de equipos** de la figura 4.8b. En este tipo de conversación, los agentes intercambian sus objetivos, comprueban su nivel de acoplamiento y deciden si juntan sus objetivos y cooperan para conseguirlos o no. Sean $A$ y $B$ dos agentes, cuyos objetivos están formados por una lista de hechos ($G_A = \{f_i\}, G_B = \{f_j\}$), distinguimos tres niveles diferentes de acoplamiento:

- Los objetivos están *totalmente relacionados* cuando comparten el mismo conjunto de hechos ($G_A = G_B$) o cuando son un subconjunto el uno del otro ($G_A \subset G_B$ o $G_A \supset G_B$).

- Los objetivos están *parcialmente relacionados* cuando su intersección no es completa, es decir, hay hechos que sólo pertenecen a un agente ($G_A \cap G_B \neq \phi$).

- Los objetivos están *no relacionados* cuando no hay intersección entre ellos ($G_A \cap G_B = \phi$).

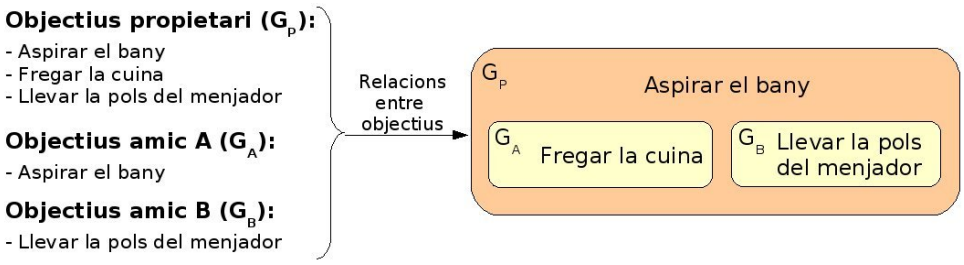


*Figura 5.4: Ejemplo de las relaciones entre los objetivos de tres agentes que están limpiando un apartamento.*

Los objetivos totalmente relacionados pueden ser problemáticos, ya que, si diversos agentes tratan de conseguirlos de forma simultánea, producirán muchas dependencias y conflictos (p. ej. los agentes querrán los mismos objetos en el mismo momento). Así pues, el protocolo de formación de equipos propondrá la creación de un grupo cuando los dos candidatos afectados por la conversación tengan **objetivos totalmente relacionados**. Por ejemplo, consideremos una posible extensión del conocido problema del *funny dinner-date* [Cavazza et al., 2002], en el que tres actores 3D deben limpiar el apartamento de uno de ellos. En concreto, el propietario quiere limpiar el piso entero, mientras que los dos amigos que lo acompañan sólo desean ayudarlo en ciertas tareas (ver la figura 5.4). En este escenario, el propietario ($P$) se puede juntar con el amigo $A$ para fregar la cocina y con el amigo $B$ para quitar el polvo del comedor. Sin embargo, la transitividad no puede ser aplicada cuando se forman los grupos. A pesar de que los objetivos $G_P$ i $G_A$ están totalmente relacionados, $G_A$ i $G_B$ no están relacionados. Por tanto, $A$ se junta con $P$ pero no con $B$. Es decir, el protocolo de formación de equipos se limita a decidir si dos actores sintéticos cooperan, sin afectar sus compromisos previos. No obstante, los equipos formados por más de dos agentes son posibles de reproducir, ya que los actores pueden crear de manera separada tantas parejas como necesiten.

Mientras que los personajes forman parte de un equipo, comunican continuamente sus intenciones a sus compañeros para facilitar la **coordinación de tareas** (sección 5.3.2). Además, sus objetivos pasan a formar parte de la representación en memoria que tienen los unos sobre los otros. Entonces, el **comportamiento cooperativo** de los miembros de un equipo se consigue mediante la inclusión de estos objetivos externos en el proceso de planificación, aspecto que se trata en la sección 5.3.3.

Por último, los actores sintéticos han de ser capaces de dejar los equipos, esto es, de comunicar el cese de la cooperación para alcanzar los objetivos externos. Análogamente a la formación de los equipos, esto sucederá cuando los objetivos de un agente dejen de estar totalmente relacionados con los de aquellos con quien estaba agrupado.

### 5.3.2. Coordinación de tareas con recursos compartidos

A la hora de **reducir los problemas de obstrucción** característicos de las simulaciones multiagente en entornos con recursos compartido, los actores pueden coordinar su operativa si conocen las intenciones de los agentes que los rodean. La información sobre las tareas que otros individuos pretenden completar puede ser utilizada para gestionar las restricciones impuestas por ellas (p. ej. un barman virtual puede aplazar la resolución de un pedido y atender a otros clientes mientras la botella que necesita esté en posesión de otro barman). Para representar las intenciones de los agentes externos, los actores 3D utilizan un modelo extendido de memoria que, a parte del estado del mundo percibido, contiene las creencias sobre las tareas actuales; recibidas a través de conversaciones con otros cohabitantes del mundo virtual (ver la figura 5.5). Consideramos una creencia como una conjetura sobre un cambio del mundo que un agente externo se dispone a realizar.

A continuación, presentamos como esta información ha sido usada por el formalismo de planificación del agente para realizar un acceso coordinado a los recursos compartidos. Uno espera que los actores sintéticos benevolentes coordinen sus tareas y eviten estorbarse los unos a los otros. La coordinación hace tender a los actores a trabajar en tareas separadas o sobre diferentes objetos 3D; que son los recursos del problema. Parece razonable, pues, que los personajes planifiquen su comportamiento (las tareas a realizar) mientras dejan al resto de individuos finalizar sus propósitos. Para conseguir esto, los actores han de completar el estado del mundo con las actividades ya planificadas por otros personajes, cuya ejecución se considera en marcha. De esta manera, el formalismo de planificación tendrá más información para seleccionar una tarea que reduzca las interferencias.

La figura 5.5 muestra esquemáticamente como el planificador utilizado por nuestros agentes socialmente inteligentes (*miniMin-HSP* [Lozano, 2005]) ha sido modificado para comenzar la búsqueda desde un **estado futuro virtual**, que es el resultado de aplicar las creencias sobre el estado actual percibido. Esta ejecución mental de las tareas de otros agentes externos se realiza aplicando las listas *add* y *delete* de la definición STRIPS de

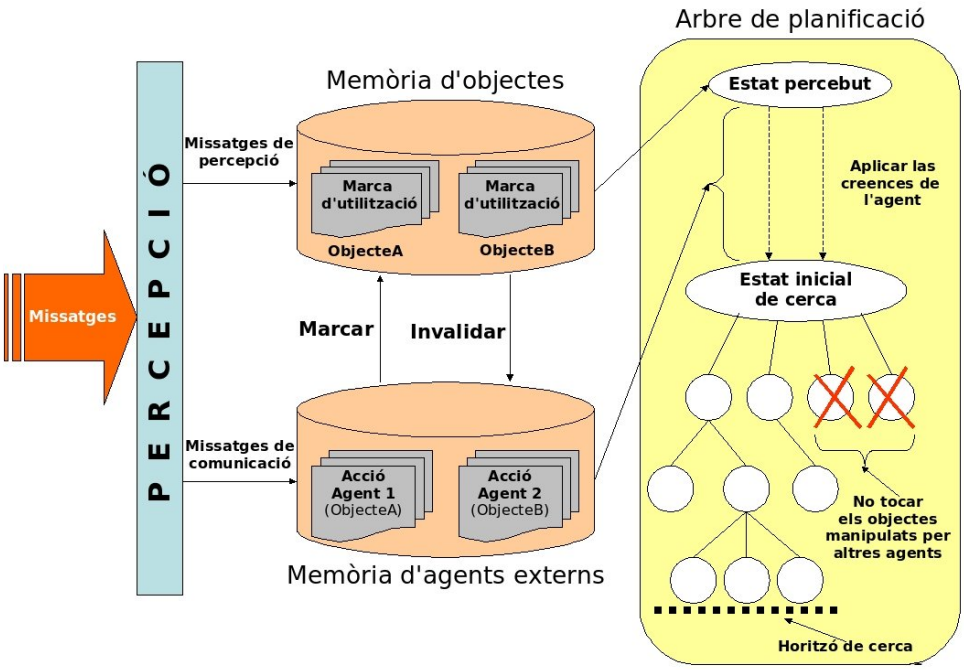


*Figura 5.5: Modelo de coordinación de tareas con recursos compartidos.*

los operadores. De esta manera, el agente construye una situación prospectiva que trata de pasar por encima de las dependencias actuales. El objetivo es que el agente planifique a partir de un punto en el tiempo en el que el resto de actores ha completado su tarea actual y evite, por tanto, interrumpirlos; cosa que se reflejará en una animación más coordinada.

Las creencias corresponden con tareas que todavía no se han completado: de hecho, nada garantiza que finalmente se completen satisfactoriamente. Por tanto, comenzar sólo la planificación desde un estado en el que todas las creencias se han acabado no es suficiente para conseguir **evitar la competición de recursos**. Por ejemplo, un agente podría decidir utilizar un objeto que se está utilizando en este momento por otro agente para completar una creencia. De cara a evitar este tipo de conflictos, cada creencia bloqueará los objetos que utiliza en el primer nivel del árbol de búsqueda; que es el nivel del que sale la siguiente tarea a realizar. Este funcionamiento prohíbe que un agente comprometa el éxito de una tarea previamente iniciada, ya que no podrá manipular los objetos que están implicados en ella (ver la figura 5.5). La planificación con estos recursos estará de nuevo disponible a partir del segundo nivel del árbol de búsqueda. En resumen, el agente puede planificar con los recursos asociados a creencias, pero se prohíbe su uso inmediato.

El mecanismo de bloqueo de recursos sólo en el primer nivel del árbol es suficiente en un contexto de planificación continua (*continous planning* [Avradinis et al., 2005]), como el propuesto por nuestros 3DIVA. El resultado de este tipo de planificaciónl, que intercala percepción, planificación y acción, no es un plan completo sino sólo la siguiente tarea a ejecutar; de manera que una vez finalizada ésta se vuelve a invocar al planificador. Por consiguiente, si un agente finaliza la primera tarea de su plan antes de invalidad una creencia previa (sin que otro agente acabe o interrumpa su tarea objetivo), la restricción de no utilizar los recursos en el primer nivel del árbol de búsqueda volverá a impedir que el planificador seleccione una tarea que utilice los objetos afectados por la creencia.

La coordinación de tareas dentro de un equipo de trabajo requiere la publicación, en forma de creencias, de las tareas en proceso de realización. Según el procedimiento descrito anteriormente, la inclusión en el proceso de planificación de estas tareas servirá para que los miembros del grupo eviten estorbar a sus compañeros y para relajar la competición de recursos. Por otro lado, la participación en un equipo implica el compromiso de alcanzar unos objetivos comunes. De acuerdo con esto, en la siguiente sección presentamos un mecanismo de sopesado heurístico para incluir los objetivos de otros agentes en el proceso de planificación y producir comportamientos cooperativos entre los miembros de un equipo.

### 5.3.3. Cooperación mediante sopesado heurístico

En esta sección presentamos una segunda manera de reducir las dependencias entre las actividades de los personajes 3D: la **cooperación** para conseguir un **objetivo global**. Según hemos explicado en la sección 5.3.1, la formación de un equipo implica que los actores se comprometen a compartir sus objetivos. La planificación sobre un objetivo común, formado por los subobjetivos propios y por otros objetivos externos adquiridos, puede ser muy beneficiosa a la hora de reducir las obstrucciones entre los miembros de un grupo (p. ej. los agentes harán tareas que beneficien al resto). Sin embargo, es necesario que los agentes utilicen alguna técnica para reflejar la importancia social de los objetivos propios respecto de los objetivos del resto de actores. A continuación, proponemos la división del objetivo común en **subobjetivos independientes**; sobre los cuales los agentes pueden aplicar diferentes **pesos heurísticos**.

El mecanismo de sopesado heurístico, que estudiamos en esta sección, puede ser utili-

zado sobre heurísticas independientes del dominio que trabajen sobre **problemas de tipo STRIPS**. Un problema STRIPS se define como un vector $P =< A, O, I, G >$, en el que: $A$ es un conjunto de predicados atómico o hechos; $O$ es un conjunto de acciones (operadores instanciados); por último, $I \subseteq A$ y $G \subseteq A$ corresponden con las situaciones inicial y final.

Una acción ser representa como el vector $o =< pre(o), add(o), del(o) >$, en el que: $pre(o)$ es la lista con las precondiciones de $o$; $add(o)$ y $del(o)$ son las listas con los átomos que deben ser añadidos y borrados, respectivamente, cuando se ejecuta $o$. De acuerdo con estas listas, una acción ($o$) es aplicable a un estado ($S$) si y sólo sí $pre(o) \subseteq S$. Entonces, el resultado de ejecutar una acción simple sobre un estado se define según la ecuación 5.1. Asimismo, el resultado de aplicar una secuencia de acciones a un estado se define recursivamente como muestra la ecuación 5.2.

$$Result(S, o) = S \cup add(o) \backslash del(o) \tag{5.1}$$

$$Result(S, < o_1, \ldots, o_n >) = Result(Result(S, < o_1, \ldots, o_{n-1} >), o_n) \tag{5.2}$$

Dado un problema STRIPS, un plan (o posible solución del problema) es una secuencia $P =< o_1, \ldots, o_n >$ de acciones de $O$ con cuya ejecución se consigue que $G \subseteq Result(I, P)$. Así pues, el objetivo de la planificación es encontrar la mínima secuencia de acciones que transforme el estado inicial del sistema en un estado objetivo deseado. Diversas familias de algoritmos han sido desarrolladas, entre las más notables: Graphplan [Blum and Furst, 1997], FastForward [Hoffmann and Nebel, 2001], *Heuristic Search Planning* [Bonet and Geffner, 2001], etc. La familia de los planificadores heurísticos (*Heuristic Search Planner* o HSP [Bonet and Geffner, 2001]) utiliza las **funciones heurísticas** para guiar la búsqueda de esta solución.

La función heurística estima el número mínimo de acciones necesarias para llegar a un estado deseado y toma la forma de la ecuación 5.3. Esto es, una función definida sobre el conjunto de los reales ($\mathbb{R}$), que estima la distancia que hay desde el estado que se está evaluando ($s$) hasta el estado objetivo ($G$). Hay muchas maneras de definir $g(G, s)$. Una forma muy común es definir esta distancia como la suma de las distancias a cada

una de las proposiciones que forman el estado objetivo (heurísticas aditivas). La ecuación 5.4 formaliza este cálculo de la función heurística, donde $p$ son las proposiciones de $G$ y $g(p, s)$ es el coste de conseguir $p$ desde el estado $s$.

$$h_G(s) = g(G, s) \tag{5.3}$$

$$g(G, s) = \sum_{p \in G} g(p, s) \tag{5.4}$$

El tipo de heurística expresado en la ecuación 5.4, sin embargo, no es admisible[2]. La causa radica en que el proceso para obtener una proposición del estado objetivo, desde el estado evaluado, puede también conseguir otras proposiciones deseadas como efecto colateral. La ecuación 5.5 representa una versión admisible de la función anterior [Haslum and Geffner, 2000], que utiliza el cuantificador máximo en vez de la suma de distancias; cosa que hará que esta heurística sea menos informativa. Ambas heurísticas definen el coste de conseguir una proposición $p$ desde un estado $s$ ($g(p, s)$) como 0 si $p$ pertenece al estado $s$ ($p \in s$). En caso contrario, le asignan $\infty$ como el valor inicial y realizan una revisión incremental, definida por la ecuación 5.6, hasta que el coste $g(p, s)$ no cambie.

$$g(G, s) = \max_{p \in G} g(p, s) \tag{5.5}$$

$$g(p, s) = \min_{a \in O(p)} [g(p, s), 1 + g(Prec(a), s)] \tag{5.6}$$

Las heurísticas anteriores asumen que las proposiciones que forman el objetivo global son **independientes**. Esto es, que los planes para conseguir un subobjetivo no tendrán ningún efecto en los planes para conseguir otros objetivos. De acuerdo con esta suposición, es posible introducir un peso asociado a cada proposición objetivo, de manera que refleje la importancia de cada proposición, tal y como muestra la siguiente fórmula:

[2]Es decir, la búsqueda mediante un algoritmo de tipo $A^*$ no tiene garantizado encontrar la solución óptima

$$H_w(G, s) = \sum_{p \in G} w(p) * g(p, s) \tag{5.7}$$

Los pesos $w(p)$ varían entre 0 y 1. De un lado, cuando todos los pesos tienen valor 1, la ecuación 5.7 es equivalente a la ecuación 5.4. De otro lado, cuando una proposición recibe un peso de valor 0 no tendrá ningún efecto en el valor final de la heurística. Es decir, la heurística se comportará como si esta proposición no existiera en el estado objetivo. Los pesos $w(p)$ pueden ser muy útiles a la hora de incluir objetivos externos de forma dinámica y combinarlos con los propios de manera cooperativa. Dentro de este contexto, se puede concebir el sopesado como un mecanismo para transformar un objetivo global en una agenda, ya que los subobjetivos son ordenados de acuerdo con su significancia.

Es importante considerar cómo el sopesado de las proposiciones afecta al comportamiento de la heurística y, en consecuencia, a la respuesta del planificador; esto es, la eficiencia y el orden en el que las proposiciones son obtenidas. Por ejemplo, el hecho de asignar un peso bajo a aquellas proposiciones que han de ser cumplidas necesariamente antes que otras (con un peso mayor), puede producir una solución de baja calidad y/o un tiempo de resolución del plan malo. Incluso, puede provocar que el planificador no encuentre ninguna solución. La causa de este problema es que, generalmente, las proposiciones que forman parte del objetivo no son independientes; a pesar de que se ha supuesto en 5.4 i 5.5 como simplificación del problema.

Para eludir este comportamiento no deseado, proponemos la separación del objetivo global en grupos de proposiciones o **subobjetivos independientes**. Entonces, los pesos se aplicarán a cada subobjetivo en vez de afectar a cada proposición. En problemas complejos, la identificación de subobjetivos independientes puede llegar a ser tan difícil como el propio proceso de planificación. Sin embargo, en los entornos virtuales, los problemas de planificación con objetos 3D suelen estar acotados en departamentos estancos, ya que cada objeto sólo se relaciona con unos pocos objetos de su alrededor. Por tanto, hemos definido un estimador que divide el estado objetivo atendiendo únicamente a los objetos que aparecen en las proposiciones que lo forman, para el cual usamos la siguiente relación:

$$f_1 \sim f_2 \Leftrightarrow \exists o : (f_1(o, o') \vee f_1(o', o)) \wedge ((f_2(o, o') \vee f_2(o'', o))) \tag{5.8}$$

Esto es, dos proposiciones $f_1$ y $f_2$ están relacionadas ($(f_1 \sim f_2)$) si hacen referencia a un objeto común. A pesar de que la ecuación 5.8 define la relación mediante proposiciones de aridad 2, la definición puede ser extendida a cualquier proposición de manera trivial. Así pues, la partición del estado objetivo ($G$) vendrá dada como el conjunto cociente de esta relación:

$$\Gamma_G = G/\sim \tag{5.9}$$

Este estimador no garantiza que los subobjetivos sean independientes, pero permite calcular de forma eficiente un conjunto finito de **subobjetivos casi independientes**. El conjunto de estados subobjetivo ($SubG_i$), resultado de la partición de la ecuación 5.9, será sopesado para considerar su importancia según la ecuación siguiente:

$$H_w(G,s) = \sum_{SubG_i \in \Gamma_G} w(SubG_i) * g(SubG_i, s) \tag{5.10}$$

Casi cualquier función heurística puede ser escrita de esta manera. En nuestro planificador, utilizamos una heurística similar a la ecuación 5.5, pero consideramos la partición en subobjetivos de la siguiente manera:

$$H_w(G,s) = \sum_{SubG_i \in \Gamma_G} (w(SubG_i) * \max_{p \in SubG_i} [g(p,s)]) \tag{5.11}$$

donde $g(p,s)$ calcula una estimación de lo que cuesta conseguir la proposición $p$ desde el estado $s$, de manera similar a como fue expresado en la ecuación 5.6.

A pesar de que esta heurística es muy similar a la heurística admisible presentada por Blai Bonet en [Bonet and Geffner, 2001], conviene destacar que esta nueva estimación trata de obtener el valor de las relaciones y de las restricciones establecidas entre los recursos. Cabe puntualizar, asimismo, que esta técnica de sopesado de la función heurística podría no funcionar como se espera en algunos algoritmos de planificación. En particular, puede tener problemas si el algoritmo poda algunas acciones antes de evaluar la heurística o si no evalúa todos los estados sucesores (como sucede en el algoritmo de ascenso de

colinas o *enforced hill climbing* [Hoffmann and Nebel, 2001]). Para los experimentos llevados a cabo en esta tesis, hemos utilizado el algoritmo **Minimin** [Lozano, 2005], a pesar de que también hubiera sido posible usar un algoritmo de búsqueda del tipo 'el primero mejor' o *best-first search algorithms* (p. ej. $A^*$, $RTA^*$).

Esta sección deja abiertas ciertas cuestiones, que han requerido de una fase de experimentación, como por ejemplo: qué valores son los adecuados para el sopesado de los subobjetivos adquiridos de los agentes externos y cómo afectan el orden en el que las proposiciones son conseguidas, así como la eficiencia de la planificación. En la sección 6.2 daremos respuesta a estas cuestiones y demostraremos el grado de coordinación y cooperación alcanzado por los 3DIVA basados en planificadores heurísticos.

Hasta ahora, hemos estudiado la introducción de criterios sociales en la toma de decisiones basada en planificadores heurísticos. Lo que resta de capítulo trata sobre un nuevo tipo de toma de decisiones social que hemos llamado MADeM (*Multi-modal Agent Decision Making*) y que puede ser utilizada por agentes de tipo BDI para la toma de decisiones socialmente aceptables.

## 5.4. MADEM: MULTI-MODAL AGENT DECISION MAKING

La toma de decisiones es el proceso cognitivo que permite a un agente seleccionar el curso de sus acciones entre un conjunto de posibilidades. Aunque hay diversos factores que influyen en este proceso, tal vez, uno de los más importantes sea el criterio utilizado por el agente a la hora de realizar la decisión. Desde un punto de vista racional, la toma de decisiones ha seguido habitualmente el criterio de la eficiencia. Esto es, la decisión adoptada ha sido aquella que conduce al agente al estado objetivo de una manera más rápida o consumiendo la cantidad mínima de recursos. Sin embargo, este comportamiento, tan arraigado en el campo de la robótica, no es precisamente el más verosímil cuando simulamos humanoides virtuales. Por contra, el razonamiento humano a menudo evalúa diferentes criterios o **puntos de vista**, que deberán de ser tenidos en cuenta en el proceso de toma de decisiones de los actores sintéticos. Por ejemplo, un cliente que entra en un bar virtual y tiene que decidir a qué camarero hacer tu pedido podría considerar diversos puntos de vista, como: a) el cansancio (p. ej. pedir al camarero que esté más cerca); b) la utilidad (p. ej. pedir al camarero menos ocupado); o c) la amistad (p. ej. pedir a un

camarero que sea amigo).

Además, la toma de **decisiones socialmente aceptables** en contextos multiagente requiere que los agentes evalúen el impacto social de sus acciones. Por ejemplo, los estudios sociológicos realizados por Jean Piaget [Piaget, 1995] determinan que la sociabilidad de un grupo se manifiesta como un intercambio de servicios entre sus miembros. Estos intercambios producen ganancias y pérdidas de elementos energéticos (p. ej. tiempo, dinero, energía, etc.) y motivadores (p. ej. inversión en servicios hechos y servicios esperados de retorno). En tal caso, para medir estas variaciones se han definido los valores de intercambio o *exchange values*. Los valores de intercambio pueden servir para modelar el intercambio social entre los agentes de una sociedad artificial e incluir criterios personales en la evaluación de estos intercambios [Ribeiro et al., 2003]. Uno de los retos actuales en esta línea consiste en integrar este tipo de técnicas sociales en los sistemas multiagente, de manera que se puedan obtener herramientas para el desarrollo de simulaciones sociales multiagente (p. ej. MAS-SOC [Bordini et al., 2005]).

Basándonos en los requisitos anteriores, en esta sección presentamos **MADeM (*Multi-modal Agent Decision Making*)**, un proceso de toma de decisiones para que los actores sintéticos puedan tomar decisiones sociales multimodales. Definimos este tipo de decisiones como aquellas que consideran diferentes puntos de vista que provienen de diversas fuentes de información (agentes externos). De un lado, cada punto de vista o foco de atención proporcionará un criterio diferente a la toma de decisiones autointeresada; que estará, de esta manera, más informada. Por ejemplo, un agente podría sopesa diferentes aspectos como: la eficiencia, el cansancio, la maña, etc. De otro lado, MADeM integra la influencia social a través de la consulta a otros agentes externos sobre aquellos puntos de vista en los que el agente está interesado. Esta realimentación social suministrará las preferencias del resto de agentes para cada una de las posibles soluciones del problema. Con toda esta información, MADeM realiza una toma de decisiones que podemos adjetivar como **sociales** y **multimodales**.

En la subsección siguiente describimos la notación y los elementos que definen una toma de decisiones de tipo MADeM. A continuación, las subsecciones 5.4.2 y 5.4.3 tratan, respectivamente, sobre la representación de las soluciones a un problema y sobre la expresión de preferencias mediante funciones de utilidad. Finalmente, en la subsección 5.4.4 explicamos el procedimiento de toma de decisiones realizado por MADeM.

### 5.4.1. Elementos y notación

La toma de decisiones social llevada a cabo por MADeM ha sido inspirada por la teoría de la **repartición de recursos en entornos multiagente** (**Multi-Agent Resource Allocation** o MARA [Chevaleyre et al., 2006]); que introdujimos en la sección 3.4.3. MARA es el proceso de distribución de un número de elementos (recursos) entre un conjunto de agentes. La definición de este proceso necesita resolver cuestiones como qué tipos de recursos se distribuirán o cuál es el procedimiento de asignación utilizado. MADeM comparte el mismo dominio de definición de MARA, pero le añade algunas características nuevas. Los elementos que definen el proceso de toma de decisiones de tipo MADeM se pueden resumir en los puntos siguientes:

- Un conjunto de agentes $A = \{a_1, \ldots, a_n\}$, donde cada $a_i$ representa un agente particular involucrado en la decisión. Asociado a cada uno de estos agentes hay un vector de pesos $\overrightarrow{w} =< w_1, ..., w_n >$, que representa su actitud personal hacia el resto de agentes (p. ej. el egoísmo, el altruismo, la indiferencia, etc.). De acuerdo con estos pesos, MADeM es capaz de sopesar la información recibida desde los otros agentes (ver la sección 5.4.4).

- Un conjunto de recursos que los agentes han de asignar $R = \{r_1, \ldots, r_m\}$. En MADeM, las soluciones a un problema dado son representadas como asignaciones de recursos. La definición de los recursos que utiliza MADeM es ligeramente diferente a la que encontramos en la literatura clásica de MARA y será tratada en detalle en la sección 5.4.2.

- Un conjunto de funciones de utilidad $\{U^1, U^2, ..., U^q\}$, para que cada agente evalúe las asignaciones de recursos arriba comentadas desde diferentes puntos de vista[3]. Las propiedades que han de cumplir estas funciones de utilidad son estudiadas en la sección 5.4.3. Relacionado con estas funciones, cada agente utiliza un vector de pesos de utilidad $\overrightarrow{w_u} =< w_{u1}, \ldots, w_{uq} >$, que representa la importancia que da el agente a cada punto de vista en la toma de decisiones multimodal. En el ejemplo de la sección anterior, donde un cliente de un bar virtual tenía que decidir el camarero al que hacerle su pedido, los pesos de utilidad expresarían tendencias personales como: la pereza, la impaciencia o el fomento de la amistad.

[3] Al contrario de los problemas clásicos de MARA, en los que los agentes sólo evaluan una única función de utilidad

### 5.4.2. Tipos de recursos

MADeM proporciona a los agentes la habilidad de preguntar la opinión de los otros sobre las diferentes soluciones a un problema (p. ej. cuando un agente recibe un encargo, puede evaluar si le interesa más realizarlo o pasar la tarea a otro agente). De acuerdo con la filosofía MARA, MADeM representa cada una de estas soluciones como un problema de asignación de recursos, en el que los recursos considerados son las tareas. La asignación de tareas ha sido estudiada ampliamente en la comunidad de sistemas multiagente pero su aplicación a los agentes virtuales sociales no es muy común y necesita tener en cuenta nuevos aspectos.

La principal novedad de nuestra aproximación radica en que MADeM no considera únicamente la ejecución de una tarea, como se puede encontrar normalmente en la literatura. Por el contrario, el tipo de recurso que será asignado es lo que nosotros llamamos **ranuras de tareas** o *task slots*. Consideramos las ranuras de tareas como parámetros que necesitan ser asignados para que una tarea pueda ser ejecutada (de conformidad con los esquemas de acción presentados en la sección ). Asociados a cualquier tarea, distinguimos dos tipos principales de ranuras: a) las ranuras de agentes o *agent slots*, que corresponden con los agentes que interpretan algún rol en la tarea (p. ej. el ejecutor o el beneficiario de la tarea); y b) las ranuras de objetos o *object slots*, que hacen referencia a los objetos involucrados en la tarea (p. ej. para clavar un clavo hace falta un martillo o algún objeto con el que golpear).

Dependiendo del rol que jueguen en la ejecución de la tarea, podemos identificar tres ranuras de tareas generales. Hay una ranura que estará presente en cualquier tipo de tarea, que es el agente que la lleva a cabo ($Ag_e$). Además podemos considerar también como casos generales dos ranuras que están presentes en muchas tareas: el objeto principal de la acción ($Obj_m$) y el receptor, en cualquier sentido, de los efectos de la acción ($Ag_d$). Por ejemplo, en la tarea *Dar*, $Ag_e$ es el agente que tiene el objeto al inicio, $Obj_m$ es el objeto que se está transfiriendo y $Ag_d$ es el agente que recibe el objeto. Con todo, siempre es posible considerar tantas ranuras de tareas como sean necesarias para un problema particular.

Como se puede apreciar, la asignación clásica de tareas se encuentra subsumida por nuestra aproximación, ya que corresponde a la asignación de la ranura $Ag_e$. Asimismo, la repartición de objetos entre un conjunto de agentes también es posible si la modelamos

como una ranura de una tarea adecuada (p. ej. la ranura $Obj_m$ de la acción *Poseer*).

En resumen, MADeM representa los recursos en forma de ranuras de tareas ($r = task(slot)$). La asignación de un elemento (agente u objeto) a una ranura se representa, entonces, como $task(slot) \leftarrow element$. De acuerdo con esto, una distribución ($P$) de elementos en ranuras de tareas tiene el siguiente aspecto:

$$P = \{t_1(s_1) \leftarrow e_1, t_1(s_2) \leftarrow e_2, \ldots, t_1(s_n) \leftarrow e_n, t_2(s_1) \leftarrow e_{n+1}, \ldots, t_m(s_n) \leftarrow e_{n*m}\} \tag{5.12}$$

MADeM representa cada una de las soluciones consideradas para un problema de decisión como una distribución del estilo de la ecuación 5.12. Por ejemplo, supongamos que en un bar virtual modelamos la tarea $Servir(Camarero, Producto, Cliente)$, que será utilizada por los camareros cuando sirvan un producto a un cliente. Esta tarea contiene dos ranuras que pueden ser subastadas: la ranura $Ag_e$, que corresponde al camarero que sirve; y la ranura $Ag_d$, que es el cliente que recibe el producto. Estas ranuras serían representadas como $Servir(Ag_e)$ y $Servir(Ag_d)$. Por tanto, una posible asignación para estas ranuras podría ser $\{Servir(Ag_e) \leftarrow a_1, Servir(Ag_d) \leftarrow a_5\}$, considerando que $a_1$ fuera un camarero y $a_5$ un cliente.

### 5.4.3. Expressión de preferencias

Los agentes involucrados en un proceso de toma de decisiones MADeM han de expresar sus preferencias para cada una de las soluciones a un problema dado. A la hora de hacer esto, los agentes utilizan las **funciones de utilidad** [Fishburn, 1970]. Estas funciones evalúan, de forma numérica, el interés que tiene un agente por una distribución de elementos en ranuras de tareas (ver la ecuación 5.12).

El tipo de representación de las funciones de utilidad es un aspecto a tener en cuenta. Por ejemplo, la enumeración de los valores de utilidad para cada una de las asignaciones posibles es normalmente inviable, ya que el número de combinaciones crece de manera exponencial. De acuerdo con esto, MADeM utiliza un tipo de representación más concisa y eficiente, las funciones de utilidad *k-aditivas*. Una función de utilidad, para una asig-

nación de recursos $P$, es *k-aditiva* si se puede expresar como un sumatorio de valores de utilidad asignados a subconjuntos $T$ de tamaño máximo $k$ (ver la ecuación 5.13).

$$U(P) = \sum_{T \in P} u(T) \quad |T| \leq k \tag{5.13}$$

La representación k-aditiva de las funciones de utilidad es totalmente expresiva, ya que puede describir cualquier función de utilidad siempre que $k$ sea suficientemente grande. No obstante, según afirma [Chevaleyre et al., 2006], en muchos dominios de aplicación es razonable asumir un valor reducido de $k$. En consecuencia, los agentes MADeM utilizan **funciones de utilidad 2-aditivas no negativas**, tal y como especifica la ecuación 5.14.

$$U(P) = \sum_{p \in P} u(p) + \sum_{p_1, p_2 \in P} u(p_1, p_2) \tag{5.14}$$

Esto es, la utilidad de una distribución de elementos $P$ se calcula como la suma de las utilidades de cada asignación ($u(p)$) más los valores de utilidad extra que aportan las asignaciones en pareja $u(p_1, p_2)$. De cara a obtener las funciones de utilidad normalizadas ($U(P) \in [0, 1]$), las funciones de utilidad de todos los agentes deben ser divididas por la misma constante, que dependerá del problema particular.

Por último, las funciones de utilidad en MADeM expresan beneficio, a pesar de que los tipos de recursos son ranuras de tareas. Por tanto, a la hora de tomar la decisión, el objetivo de los agentes será maximizar su utilidad (ver la sección siguiente). El uso de funciones de utilidad que expresen costes también hubiera sido posible. En tal caso, los agentes deberían de minimizar estos costes en su toma de decisiones.

### 5.4.4. Procedimiento de toma de decisiones

Para reunir las preferencias externas sobre las diferentes soluciones a un problema, MADeM utiliza las **subastas combinatorias de una ronda con apuestas no visibles** (*one-round sealed-bid combinatorial auctions*). En esta aproximación, los roles de subastador y apostador no los juegan agentes fijos del sistema. Al contrario, los agentes adopta-

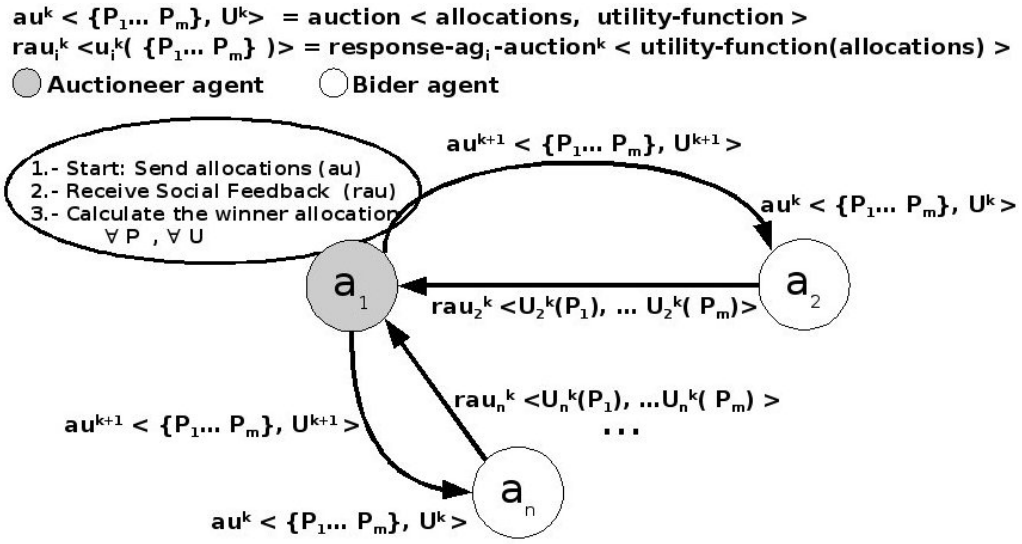


*Figura 5.6: Procedimiento de toma de decisiones MADeM.*

rán dinámicamente estos roles, dependiendo de sus necesidades o intereses. Por ejemplo, un agente que quiere decidir si realiza una tarea o pasa su ejecución a otro, actuaría como el subastador. Mientras que los agentes que recibieran la subasta apostarían con sus valores de utilidad, siempre que estuvieran interesados en las ranuras de tareas que se están subastando. Sin embargo, debido a que las decisiones complejas pueden necesitar la consideración de más de un punto de vista, el subastador puede requerir que el resto de agentes expresen sus preferencias mediante diversas funciones de utilidad. Por tanto, MADeM permite ejecutar más de una subasta a la vez. Asimismo, los agentes pueden participar en diversas subastas de manera simultánea. En consecuencia, la toma de decisiones de tipo MADeM se encuentra entre la asignación de recursos basada en el mercado centralizada (porque hay un subastador central que decide) y distribuida (porque todos los agentes pueden ser subastadores).

La figura 5.6 muestra el esquema del **procedimiento de toma de decisiones** llevado a cabo por MADeM para generar decisiones sociales multimodales. Este procedimiento se subdivide, fundamentalmente, en los siguientes pasos:

**1** *Fase de subasta*: Esta fase es realizada por un agente($a_1$) que desea resolver un problema de decisión con información social (p. ej. en qué mesa sentarse en un bar virtual). Entonces, el agente construye el conjunto de distribuciones que representan las posibles soluciones al problema ($\{P_1, P_2, \ldots, P_m\}$). Cada una de estas distribuciones estará formada por un conjunto de asignaciones de elementos a ranuras de tareas, como por ejemplo $Sentar(Obj_m) \leftarrow mesa_1$. A continuación, el

agente subasta estas distribuciones a un grupo particular de agentes, que llamaremos agentes objetivo. Cada subasta, además, incluye el tipo de función de utilidad o punto de vista considerado ($au^k(\{P_1, P_2, \ldots, P_m\}, U^k)$). Como los procesos de decisión complejos a menudo requieren la consideración de más de un punto de vista, el agente subastador puede comenzar diferentes subastas para las mismas distribuciones pero con diferentes funciones de utilidad a evaluar (desde $au^1$ hasta $au^q$).

El mecanismo de selección de los agentes objetivo de una subasta varía dependiendo de las herramientas que el subastador tenga a su alcance. Si el agente dispone de un sistema de clasificación de los objetos del entorno (p. ej. la base ontológica definida en la sección 4.3), cada ranura puede ser considerada como un parámetro con un cierto tipo. De manera que cuando se trabaja con ranuras de agente, los agentes objetivo pueden ser extraídos utilizando el tipo de ranura que se esté subastando. Por ejemplo, la ranura $Hacer\_Caf(Ag_e)$ debería ser subastada sólo a los agentes de la clase *Camarero*, ya que son los únicos que pueden realizar esta tarea en un bar virtual. Por otro lado, cuando se asignan ranuras de objetos, los agentes objetivo podrían ser aquellos agentes que están relacionados de alguna manera con algún objeto del tipo de la ranura que se está subastando. Por ejemplo, la ranura de tarea $Sentarse(Obj_m)$, si $Obj_m$ corresponde con la mesa donde el agente se ha de sentar, podría ser subastada a todos los agentes que están relacionados con alguna mesa del entorno. Si no hay un sistema de tipos disponible, los agentes objetivo pueden extraerse siguiendo criterios de visibilidad o, en una situación extrema, ser el conjunto de todos los agentes.

**2** *Fase de apuestas*: Cuando se inicia la subasta, el subastador informa tanto sobre las distribuciones (soluciones posibles) que quiere evaluar como sobre la función de utilidad que ha de ser considerada. Por tanto, los agentes objetivo (apostadores) simplemente tienen que computar la función de utilidad solicitada y enviar los valores obtenidos para cada distribución en forma de apuesta ($rau_i^k =< U_i^k(P_1), ..., U_i^k(P_m) >$).

**3** *Fase de resolución del ganador*: Esta fase consta de dos partes, referidas como la determinación del ganador de cada subasta y la toma de decisiones multimodal. La primera parte sirve para que el subastador seleccione cuál es la distribución ganadora de cada una de las subastas lanzadas. Esta decisión seguirá el criterio de maximizar el bienestar de la sociedad artificial (ver [Chevaleyre et al., 2004] para un buen conjunto de ejemplos de bienestar social). Posteriormente, la toma de deci-

siones multimodal escogerá la distribución ganadora final (solución adoptada) del conjunto de distribuciones ganadoras de las subastas.

A continuación analizamos en detalle cómo MADeM realiza la determinación del ganador de cada subasta y la toma de decisiones multimodal.

### *Determinación del ganador de cada subasta*

El proceso de determinación del ganador de una subasta comienza una vez que todos los apostadores han enviado su apuesta[4]. Como información de entrada, el agente dispone de los valores de utilidad ($U_i(P_j)$) enviados por los agentes de la sociedad ($a_i \in A$) para cada distribución que se esté evaluando ($P_j$). La ecuación 5.15 agrupa estos valores de utilidad en un conjunto de **vectores de utilidad**, uno para cada posible solución.

$$\overrightarrow{U(P_j)} =< U_1(P_j), \ldots, U_n(P_j) > \quad \forall j \in [1..m] \tag{5.15}$$

Asimismo, recordemos que cada agente dispone de un vector de pesos que representan su actitud personal hacia los otros (ver la ecuación 5.16). Por tanto, el agente subastador puede aplicar un diferencial a los vectores de utilidad mediante la multiplicación componente a componente con el vector de actitudes, como demuestra la ecuación 5.17.

$$\overrightarrow{w} =< w_1, \ldots, w_n > \tag{5.16}$$

$$\overrightarrow{U_w(P_j)} = \overrightarrow{U(P_j)} * \overrightarrow{w} \quad \forall j \in [1..m] \tag{5.17}$$

Los **pesos de actitud** se utilizan para modelar el comportamiento social a nivel personal, esto es, entre el agente subastador y el resto de agentes de la sociedad. Por ejemplo,

[4]Se puede considerar que aquel que no responda no tiene ninguna preferencia al respecto, es decir, su utilidad es cero

el egoísmo extremo se puede representar a través de un peso de valor 1 para las utilidades del propio agente y 0 para el resto de agentes. De hecho, se puede modelar todo un rango de comportamientos entre el egoísmo y el altruismo mediante un vector de pesos como el que aparece en la ecuación 5.18. En este modelo, $p = 0$ corresponde con el comportamiento que acabamos de definir, $p = 1$ representa el altruismo máximo y $p = 0{,}5$ representa un comportamiento igualitario o indiferente hacia el resto de agentes. Por último, con el vector de pesos $\overrightarrow{w}$ también es posible modelar actitudes recíprocas. La ecuación 5.19 muestra un ejemplo sencillo, en el que los pesos se calculan en función del número de favores intercambiados entre los agentes.

$$\begin{aligned} &Egoisme - Altruisme: \quad \overrightarrow{w} =< p, \ldots, p, 1-p, p, \ldots, p > \\ &w_i = 1 - p, w_{j \neq i} = p, i = Myself \end{aligned} \tag{5.18}$$

$$Reciprocitat: \quad w_i = \frac{Favors\_from(i)}{Favors\_to(i)} \tag{5.19}$$

En este contexto de preferencias individuales de los agentes, la determinación del ganador de una subasta consiste en encontrar aquella distribución $P_j$ que se pueda considerar óptima respecto de alguna métrica global a la sociedad. La definición de métricas globales como agregación de preferencias individuales ha sido tratada, a menudo, por el **bienestar social** (*social welfare*); concepto estudiado por la economía del bienestar (*Welfare Economics*) y por la teoría de la decisión social (*Social Choice Theory*). En el caso habitual en el que las preferencias se expresan mediante funciones de utilidad, el bienestar social se define a través de **funciones de utilidad colectivas** (*Collective Utility Functions* o CUFs). Así pues, la ecuación 5.20 define el bienestar social en MADeM en función de las utilidades sopesadas de la ecuación 5.17. MADeM permite escoger entre diferentes CUFs a la hora de evaluar el bienestar social de una asignación (ver la ecuación 5.21). Cada una de estas funciones está relacionada con un tipo de sociedad [Sandholm and Suri, 2001], respectivamente: utilitaria, igualitaria, elitista y tipo Nash.

$$sw(P) = cuf(\overrightarrow{U_w(P)}) \tag{5.20}$$

$$\begin{aligned} cuf_{utilitarian} &= \Sigma u_w(i) \\ cuf_{egalitarian} &= \text{mín}\{u_w(i)\} \\ cuf_{elitist} &= \text{máx}\{u_w(i)\} \\ cuf_{nash} &= \Pi u_w(i) \end{aligned} \tag{5.21}$$

La función de bienestar social ($sw$) define un orden social tal que: una distribución $Q$ es socialmente preferible a una distribución $P$ si y sólo sí $sw(P) \leq sw(Q)$. De acuerdo con esto, la distribución ganadora de una subasta será aquella que maximice el bienestar de la sociedad (ver la ecuación 5.22).

$$Winner = P_w \longleftrightarrow sw(P_w) = \max_{j \in [1..m]} sw(P_j) \tag{5.22}$$

### *Toma de decisiones multimodal*

La toma de decisiones multimodal es el último paso que se realiza y su resultado es la solución MADeM a un problema dado. Como hemos analizado antes, un agente que ejecuta una toma de decisiones de tipo MADeM considera las preferencias de otros agentes según diferentes puntos de vista (p. ej. la eficiencia, el cansancio, etc.). Para hacerlo, el agente lanza diversas subastas, cada una con una función de utilidad diferente (ver el parámetro $U^k$ de las subastas en la figura 5.6). Una vez que todas estas subastas han sido resueltas, el agente subastador tiene la distribución ganadora según cada punto de vista, así como el bienestar social que se obtiene con ella (ver la ecuación 5.23).

$$\begin{aligned} auction_1(\{P_1, P_2, ..., P_m\}, U^1) &\longrightarrow (P_{w1}, sw(P_{w1})) \\ &\dots \\ auction_k(\{P_1, P_2, ..., P_m\}, U^k) &\longrightarrow (P_{wk}, sw(P_{wk})) \end{aligned} \tag{5.23}$$

De esta manera, la selección de la distribución ganadora final se realiza haciendo uso del vector de pesos de utilidad $\overrightarrow{w_u}$ definido en la sección 5.4.1. La toma de decisiones

multimodal utiliza los valores de $\overrightarrow{w_u}$ como los pesos asignados a cada punto de vista o función de utilidad. En consecuencia, la distribución ganadora final ($P_f$) será aquella que maximice el bienestar de la sociedad, después de haber multiplicado el bienestar obtenido con cada punto de vista por su correspondiente peso de utilidad (ver la ecuación 5.24).

$$P_f = P_{wi} \longleftrightarrow sw(P_{wi}) = \max_{i \in [1..k]} w_u(i) * sw(P_{wi}) \tag{5.24}$$

En resumen, el uso de subastas es una aproximación muy flexible para la evaluación de las diferentes soluciones a un problema en un entorno multiagente. Sin embargo, está claro que la animación de todas estas subastas produciría un comportamiento del todo inverosímil (p. ej. humanoides que subastan tareas continuamente y reciben respuestas numéricas ininteligibles). Por tanto, hay que separar el proceso de toma de decisiones de la animación de la decisión adoptada. De acuerdo con esto, MADeM puede ser utilizado como el módulo social para la toma de decisiones en la arquitectura de agentes presentada en la sección 4.4. Sin embargo, MADeM se ejecutará de manera transparente al observador externo. Esto es, las comunicaciones establecidas entre los agentes para llevar a cabo las subastas no se animan sino que se realizan de forma interna. Entonces, una vez se ha seleccionado la decisión ganadora, el agente se ha de hacer cargo de animar la situación correspondiente (p. ej. mediante protocolos de animación de conversaciones entre personajes 3D). En este contexto, MADeM es una fuente de información muy útil para la simulación de acuerdos, desacuerdos y comportamientos sociales puros (p. ej. diálogos banales). En la sección 6.4.4, mostramos como ha sido resuelta la animación de acuerdos sociales en un entorno 3D donde los personajes utilizan MADeM para la toma de decisiones sociales.

## 5.5. CONCLUSIONES

En este capítulo hemos analizado las dos aportaciones principales de esta tesis en lo que hace referencia a la integración de habilidades sociales en la toma de decisiones de los actores sintéticos.

En primer lugar, hemos presentado una técnica para conseguir comportamientos colaborativos en aquellos personajes 3D que utilizan los planificadores heurísticos (modelo

STRIPS) como el mecanismo de selección dinámica de acciones. Esta técnica dota a los actores de la habilidad de adaptarse y actuar de forma eficiente en los entornos poblados por diversos caracteres autónomos (p. ej. juegos, *storytelling*, simulación civil, etc.). En estos contextos, la detección de dependencias y la comunicación directa entre los personajes sirve para construir una representación de los agentes externos. De manera que hemos propuesto dos mecanismos que utilizan este información para incluir criterios sociales en el proceso de planificación. De un lado, hemos incorporado una fase de pre-planificación que evita obstruir las acciones de otros agentes. De otro lado, hemos desarrollado un mecanismo de sopesado de la función heurística que permite a los agentes cooperar en la consecución de objetivos externos.

En segundo lugar, hemos presentado MADeM (*Multi-modal Agent Decision Making*). MADeM es un proceso de toma de decisiones multimodal y social. Esto es, que permite evaluar diversos puntos de vista (p. ej. eficiencia, cansancio, etc.) así como consultar las preferencias de otros agentes acerca de las diferentes soluciones a un problema dado. Las preferencias individuales son convertidas en preferencias colectivas a través del bienestar social, que define el tipo de sociedad simulada. MADeM es capaz de simular diferentes tipos de sociedades (p. ej. elitistas, igualitarias, utilitarias, etc.) así como actitudes sociales entre sus miembros (p. ej. egoísmo, altruismo, indiferencia o reciprocidad).

En el capítulo siguiente analizaremos los efectos sociales que provocan ambas aportaciones en el comportamiento de los personajes 3D. Para ello, repasaremos tres ejemplos de complejidad creciente: a) el sopesado heurístico en el mundo de bloques clásico; b) la colaboración en el mundo de bloques 3D multipersonaje; c) la toma de decisiones socialmente aceptables dentro de un bar virtual.

CAPÍTULO 6

# RESULTADOS

En este capítulo presentamos tres ejemplos que muestran los efectos producidos por los mecanismos de toma de decisiones sociales propuestos en esta tesis. En primer lugar, definimos un conjunto de experimentos basados en el mundo de bloques clásico, de cara a estudiar el impacto del sopesado de objetivos en los planificadores de búsqueda heurística. En segundo lugar, hemos creado diversos escenarios 3D inspirados en el mundo de bloques multipersonaje. En estos entornos dinámicos, los actores sintéticos (basados en planificadores heurísticos) muestran su sociabilidad a través de la **coordinación** de tareas y de la **cooperación** con los objetivos de otros personajes. Finalmente, hemos desarrollado un entorno dinámico con un grado de complejidad mayor, consistente en un bar virtual donde actores que ejecutan el rol de camareros sirven los pedidos de otros actores con el rol de clientes. En este contexto, ambos tipos de actores utilizan MADeM para expresar su **sociabilidad** según diversos criterios (p. ej. eficiencia, cansancio, etc.), así como para modelar diferentes actitudes personales (p. ej. reciprocidad, altruismo, etc.).

## 6.1. DESCRIPCIÓN DE LOS EXPERIMENTOS PLANTEADOS

Las dificultades a las que se ha tenido que enfrentar la animación comportamental de un actor sintético individual (p. ej. fidelidad visual, percepción, navegación, etc.) han provocado que justo empiecen a integrarse comportamientos sociales complejos en la toma de decisiones de los personajes 3D [Prada and Paiva, 2005, Pelechano et al., 2007]. En este contexto, un problema importante es cómo validar los modelos sociales incorporados en los agentes. Sin embargo, la investigación en actores sintéticos carece hoy en día de métodos de validación comúnmente aceptados (*benchmarks*), para poder comparar el

comportamiento social de actores sintéticos gobernados por mecanismos diferentes. Por el contrario, cada sistema ha utilizado el conjunto de experimentos adecuados para demostrar los efectos sociales que se pueden conseguir con él. De acuerdo con esto, en este capítulo presentamos tres ejemplos de complejidad creciente, que nos permiten demostrar el efecto de las técnicas sociales desarrolladas en esta tesis de una forma cuantitativa, así como comprobar que los mecanismos propuestos permiten controlar la integración de diferentes habilidades sociales en el comportamiento de los 3DIVA.

Los dos primeros ejemplos están relacionados con la técnica de colaboración mediante el uso de un planificador heurístico. Primeramente, hemos realizado un conjunto de **experimentos basados en el mundo de bloques clásico** (ver el punto 6.2). En estos problemas monoagente, hemos definido el estado objetivo del agente como un conjunto de subobjetivos propios y externos. Así pues, podemos analizar la influencia que tiene el sopesado heurístico de los objetivos externos en los entornos multiagente, pero simplificando el problema con un entorno estático donde no se producen cambios exógenos. El segundo ejemplo que consideramos ha sido la **animación 3D del mundo de bloques multipersonaje** (ver el apartado 6.3). El dinamismo de estos escenarios conducirá a los actores ante situaciones conflictivas, que deberán ser resueltas de manera reactiva. En estos entornos orientados a la consecución de tareas, medimos la sociabilidad de los personajes en función de la coordinación de tareas (sección 6.3.1) y de la cooperación con los objetivos del resto de agentes (sección 6.3.2).

En tercer lugar, hemos construido un entorno 3D más complejo que simula un **bar universitario virtual** poblado por camareros y clientes (ver el apartado 6.4). Los numerosos objetos interactivos considerados (p. ej. vasos, platos, etc.), han sido representados siguiendo las definiciones de una ontología específica del dominio que también permite la definición de relaciones sociales entre los actores (secciones 6.4.1 y 6.4.2). Entonces, los personajes utilizan MADeM para tomar decisiones sociales multimodales. En la sección 6.4.3, mostramos las funciones de utilidad que hemos diseñado para que los actores expresen sus preferencias según puntos de vista diferentes (p. ej. la eficiencia, el cansancio, las ganas de hablar, la pereza, etc.). Finalmente, en las secciones 6.4.5 y 6.4.6 repasamos los resultados sociales obtenidos para diversos tipos de personajes (p. ej. coordinados, habladores, perezosos, etc.) y para diferentes modelos de actitud (p. ej. egoísmo, altruismo, reciprocidad, etc.).

Como detalle de implementación, los resultados mostrados a lo largo de este capítulo

se han obtenido utilizando un Pentium IV a 2.66 GHz y con 1 Gb de RAM. Por tanto, podemos afirmar que los modelos sociales propuestos no necesitan una gran potencia de cómputo y que pueden ser aplicados a los entornos virtuales 3D que animan los ordenadores personales de hoy en día.

## 6.2. SOSPESADO HEURÍSTICO EN EL MUNDO DE BLOQUES

En este apartado mostramos los resultados de un conjunto de experimentos que hemos realizado para estudiar el impacto del sopesado de objetivos en los planificadores de búsqueda heurística. Los experimentos están basados en el mundo de bloques clásico, que consiste en que un agente mueva (apile/desapile) un conjunto de bloques para conseguir una distribución objetivo de la manera más eficiente posible. Sin embargo, para poder evaluar el grado de cooperación en entornos multiagente, hemos completado el estado objetivo del agente con un número finito de subobjetivos externos que representan los objetivos de otros agentes. Entonces, hemos introducido el mecanismo de sopesado heurístico explicado en la sección 5.3.3 para modular la influencia de estos subobjetivos externos en la toma de decisiones del agente.

El conjunto de experimentos utilizado consta de una batería de 10 problemas para cada una de las siguientes tallas: 8, 12, 16, 20 y 24 bloques. La disposición inicial y final (objetivo) de los bloques se ha generado de forma aleatoria. Hemos dividido el objetivo en un conjunto de subobjetivos independientes. De estos subobjetivos consideramos una parte como objetivos propios del agente y otra parte como objetivos externos. La proporción utilizada ha sido 1:1, esto es, una mitad de objetivos externos y la otra de objetivos propios. Entonces, hemos ejecutado todos estos problemas utilizando 11 pesos diferentes para los objetivos externos ($w \in [0,0{,}1,0{,}2,..,1{,}0]$). De cara a calcular una media estable de los resultados, hemos resuelto cada problema 10 veces por peso. Los problemas se resuelven por un único agente que utiliza *miniMin-HSP* [Lozano, 2005] para calcular la secuencia de acciones que le permita conseguir los objetivos propios. Aunque en estos experimentos no se producen interferencias entre los agentes, la cantidad de objetivos externos conseguidos puede ser un buen estimador del **grado de cooperación**.

La figura 6.1a muestra la media de objetivos externos (medidos en número de hechos o proposiciones) conseguidos por un agente mientras cumple con sus objetivos propios. La

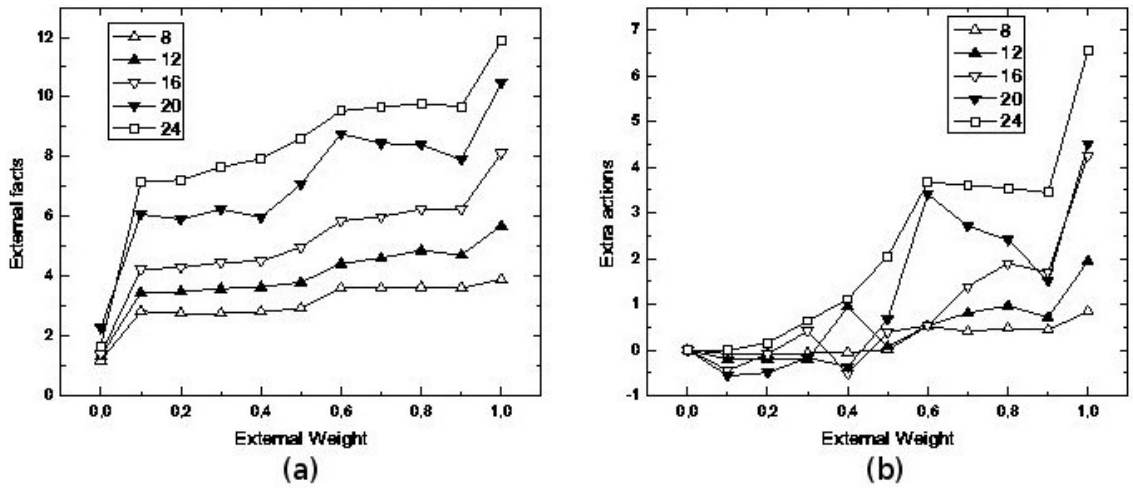


*Figura 6.1: Cooperación en función del peso heurístico: a) media de objetivos externos conseguidos; y b) media de acciones extras realizadas.*

influencia de los pesos aplicados sobre la heurística de los objetivos externos modificará su importancia en el proceso de toma de decisiones y, por tanto, la cantidad de objetivos externos alcanzados. Por ejemplo, cuando $w_{ext} = 0$, los objetivos externos no son considerados por la heurística y el número de hechos externos conseguidos es muy bajo. Uno podría pensar que este valor debería de ser 0. Sin embargo, es fácilmente comprensible que algunos hechos externos se puedan conseguir de manera casual (p. ej. como efectos colaterales de las acciones realizadas para cumplir con los objetivos propios). A medida que aumentan los pesos, también lo hace la proporción de objetivos externos conseguidos. Por ejemplo, cuando $w_{ext} \in [0{,}1.{,}0{,}5]$, el número de hechos externos alcanzados se mueve entre un 30 % y un 35 % de los objetivos externos considerados. Este crecimiento se produce en todos los problemas, que llegan al 50 % de los objetivos totales cuando la importancia entre los objetivos externos y los propios se iguala ($w_{ext} = 1$); resultado que corresponde con el ratio entre los objetivos propios y externos.

La figura 6.1b muestra la media de acciones extras ejecutadas a causa de la cooperación con los objetivos de los agentes externos. El propósito de esta gráfica es evaluar el esfuerzo adicional que realiza un agente dependiendo de los pesos utilizados para el sopesado de objetivos externos. Como se puede apreciar, cuando los pesos son bajos ($w_{ext} \in [0{,}1.{,}0{,}3]$), el agente no realiza más acciones que aquellas que necesita cuando no hay cooperación ($w_{ext} = 0$). Recordemos, además, que con este rango de pesos ya se conseguían algunas proposiciones asociadas a objetivos externos (como vimos en

la figura 6.1a). Este hecho se produce gracias al oportunismo de ciertas acciones. Por ejemplo, cuando un agente tiene que quitar un bloque de encima de otro porque quiere coger el bloque de abajo, puede decidir moverlo a cualquier posición disponible o bien, dejarlo en una posición que sirva para conseguir algún objetivo externo. Si esto no es posible, al menos, el agente puede dejar el bloque en algún sitio que no moleste a otros agentes, es decir, que no aleje la consecución de otros objetivos externos. La influencia del sopesado heurístico en la actividad del agente se manifiesta de una manera más clara cuando $w_{ext} > 0{,}5$. Por ejemplo, los problemas de 8 bloques normalmente se resuelven con planes formados por una media de 6 acciones. Sin embargo, la consideración de objetivos externos con la misma prioridad que los propios ($w_{ext} = 1$) hace que los agentes ejecuten una acción extra por término medio. Este comportamiento se reproduce también en el resto de experimentos. Por ejemplo, en los problemas de 24 bloques, la media de acciones extras llega casi a 7 acciones. Estos resultados muestran como los agentes, en su intento de actuar de forma cooperativa, pueden incurrir en un esfuerzo innecesario, ya que estas acciones extras se realizan únicamente para conseguir los objetivos asociados a otros agentes (ver también la figura 6.1a).

El sopesado heurístico no sólo facilita la cooperación entre los agentes sino que, en ciertas circunstancias, puede tener un impacto positivo en la eficiencia del proceso de toma de decisiones. La figura 6.2 muestra la media de tiempo de CPU que necesita el planificador para obtener un plan completo que lleve al agente a conseguir sus objetivos propios. De manera general, se puede constatar que el tiempo consumido por el planificador es superior cuando $w_{ext} = 0$ que cuando $w_{ext} > 0$. La razón de este comportamiento radica en que, cuando se consideran sólo los objetivos propios para guiar la búsqueda, pueden haber más caminos de la misma longitud que lleguen al estado objetivo. Entonces, el planificador expandirá más nodos cuando busque el mejor camino hacia el objetivo. Este hecho todavía es más notorio cuando aumenta el número de objetos del entorno, ya que hay muchos objetos que no tienen ninguna influencia en los objetivos propios (ver el alto tiempo de ejecución de los problemas con 24 bloques y $w_{ext} = 0$). Sin embargo, si se consideran los objetivos externos en el cálculo de la heurística, aquellos caminos que consigan más objetivos externos se verán favorecidos y, en consecuencia, el número de nodos explorados será menor. No obstante, el hecho de aumentar mucho el peso de los objetivos externos implica reducir la prioridad de resolución de los objetivos propios respecto de los externos. En un caso extremo, el agente podría comenzar a perder el tiempo tratando de conseguir otros objetivos externos en vez de los propios, que son los que hacen que finalice el proceso de planificación. La figura 6.2 refleja este hecho en forma de una ligera

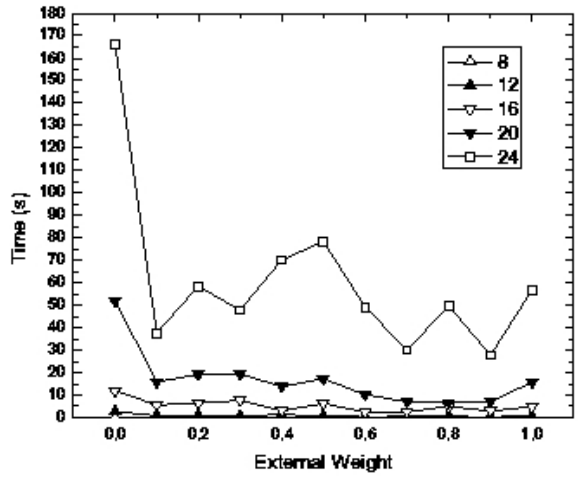


*Figura 6.2: Comparativa de los tiempos de resolución en función del peso heurístico.*

tendencia al alza en los tiempos de CPU cuando $w_{ext} = 1$.

En resumen, el estudio anterior ha demostrado que la inclusión de objetivos externos en el proceso de planificación permite conseguir **comportamientos cooperativos** sin aumentar la complejidad de las decisiones. En este contexto, el peso heurístico utilizado por los objetivos externos determinará el grado de cooperación alcanzado. Por una parte, a la hora de desarrollar agentes que cooperen con otros sin incrementar su esfuerzo (el número de acciones realizadas), un buen valor para los pesos $w_{ext}$ se encontrará en el rango $[0{,}1.{,}0{,}3]$. Por otra, si se desea implementar un agente que coopere en un grado mayor, aunque eso suponga un esfuerzo extra, se pueden usar valores de $w_{ext}$ en el rango $[0{,}6.{,}1{,}0]$.

## 6.3. MUNDO DE BLOQUES 3D MULTIPERSONAJE

Para evaluar la técnica de colaboración propuesta por los actores sintéticos basados en planificadores heurísticos en un entorno dinámico, hemos creado diversos escenarios 3D inspirados en el mundo de bloques. Estos entornos están poblados por un conjunto de actores que manipulan los objetos del entorno, fundamentalmente, para situarlos en una posición final deseada. La operativa de cada uno de los actores sigue el bucle típico que

intercala percepción, planificación y actuación. En este contexto, los personajes 3D utilizan el planificador miniMin-HSP con las extensiones sociales presentadas en la sección 5.3 para decidir la siguiente tarea a realizar (p. ej. coger o dejar un objeto encima de otro). Por lo que respecta a la visualización gráfica, en este ejemplo hemos utilizado el motor del juego *Unreal Tournament* [Epic games, 2008].

La simulación 3D del mundo de bloques multipersonaje se enfrenta a menudo con situaciones conflictivas, a causa de la fuerte competencia por los recursos compartidos. Por ejemplo, aunque dos actores decidan coger el mismo objeto, sólo uno de ellos lo puede conseguir finalmente. Para detectar las interferencias que se producen con otros agentes, los actores comprueban continuamente las precondiciones de las tareas (operadores STRIPS) que ejecutan. Entonces, la detección de dependencias provoca el lanzamiento de los protocolos de pregunta de la tarea actual y de formación de equipos, descritos en la sección 5.3.1. El propósito de estas conversaciones es el intercambio de información sobre la tarea y los objetivos actuales. Esta información representa el **contexto social**, que se utiliza posteriormente por el planificador.

En este tipo de entornos orientados a la consecución de tareas, la sociabilidad de los personajes se asocia con la eficiencia de su actuación. Por tanto, el grado de comportamiento social alcanzado se puede medir a través de la **coordinación de tareas** y la **cooperación con los objetivos del resto de agentes**; aspectos que estudiamos en las secciones 6.3.1 y 6.3.2, respectivamente.

### 6.3.1. Coordinación de tareas

La carencia de recursos y los conflictos que surgen por su manipulación compartida pueden reducir la verosimilitud de los personajes 3D. Para ilustrar este hecho, comencemos con un sencillo entorno donde el objetivo sea apilar cuatro cajas según un cierto orden establecido (ver la figura 6.3). Mientras que un único agente necesita 12 tareas (coger/-soltar) para completar la pila, sería deseable que dos agentes se repartieran las tareas para montar la pila en un tiempo inferior. Sin embargo, cuando los agentes no están coordinados, el número de tareas ejecutadas por cada agente supera el caso monoagente y llega a un total de 26 tareas (ver la tabla 6.1). La causa se encuentra en que, cuando los agentes planifican de manera independiente, la actuación de un agente interrumpirá los planes de los otros agentes (hasta 10 veces en el sencillo ejemplo planteado). Esta sensación de

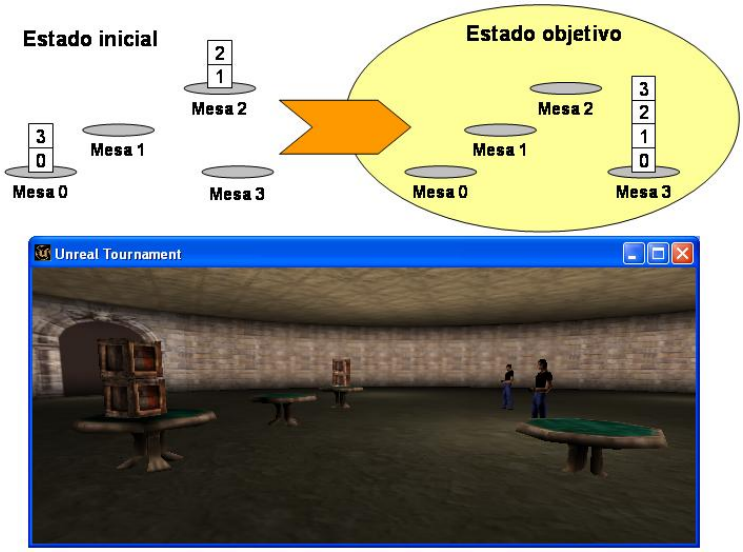


*Figura 6.3: Problema de apilar cuatro cajas entre dos personajes.*

| | Planes interrumpidos | | Tareas ejecutadas | |
|---|---|---|---|---|
| Agente | Simple | Coordinado | Simple | Coordinado |
| A | 5 | 0 | 14 | 8 |
| B | 5 | 4 | 12 | 4 |

*Tabla 6.1: Resultados de coordinación para dos agentes en el mundo de bloques 3D.*

**descoordinación** reducirá la verosimilitud del comportamiento de los actores.

En la técnica colaborativa presentada en esta tesis, las interferencias se resuelven mediante una conversación donde los agentes se preguntan sobre sus tareas actuales. Esta información, que entre en forma de creencia en la memoria del agente, se utilizará en el proceso de planificación para comenzar la búsqueda desde un posible estado futuro en el que no se estorbe al otro agente. La tabla 6.1 muestra también los resultados obtenidos cuando los agentes comunican sus acciones y coordinan sus tareas futuras de acuerdo con la información intercambiada. En este caso, el número total de acciones completadas entre los dos agentes es 12, como era de esperar. El agente $A$ no ha visto interrumpida ninguna de sus tareas, mientras que el agente $B$ ha utilizado sus interrupciones para coordinarse con el agente $A$. La figura 6.4 muestra la secuencia de pasos realizada por los agentes para apilar las cajas. Las instantáneas *b*, *e* y *g* muestran las conversaciones establecidas entre los agentes para comunicarse su tarea actual.

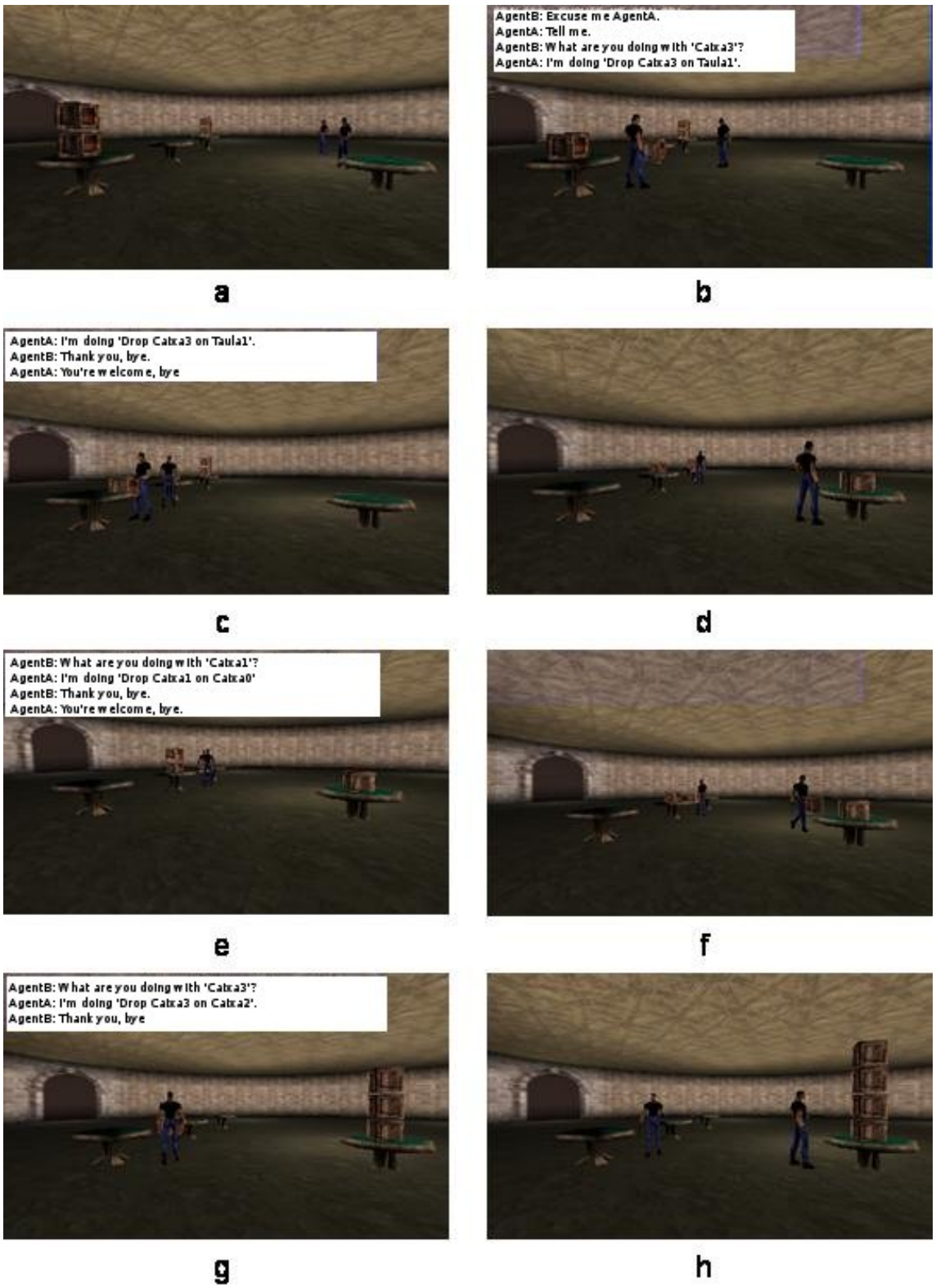


*Figura 6.4: Animación 3D de la construcción de la pila de cajas entre dos actores.*

| | Planes interrumpidos | | Tareas ejecutadas | | Llamadas al planificador | |
|---|---|---|---|---|---|---|
| Agente | Simple | Coordinado | Simple | Coordinado | Simple | Coordinado |
| A | 4 | 5 | 10 | 8 | 14 | 13 |
| B | 3 | 2 | 6 | 3 | 9 | 5 |
| C | 4 | 2 | 11 | 7 | 15 | 9 |
| D | 7 | 2 | 8 | 4 | 15 | 6 |

*Tabla 6.2: Resultados de coordinación para cuatro agentes en el mundo de bloques 3D.*

Para estudiar la **coordinación de tareas** en un ejemplo más complejo, hemos animado un mundo de bloques 3D con 21 cajas y poblado por 4 actores que comparten un objetivo común. Según establece Jacques Ferber [Ferber, 1999], la coordinación entre un conjunto de agentes puede ser medida a través de tres indicadores: la **supervivencia**, la **resolución de conflictos** y el **aumento de la eficiencia**. El indicador de supervivencia no tiene sentido en este problema, ya que todos los agentes conseguirán su objetivo. El nivel de resolución de conflictos puede ser estimado mediante el número de planes interrumpidos. El incremento de la eficiencia puede ser estimado de acuerdo con la cantidad de tareas ejecutadas. Finalmente, el número de llamadas al planificador es un estimador de coordinación que mezcla ambas métricas, ya que corresponde con la suma de los planes interrumpidos más las tareas completadas con éxito.

La tabla 6.2 muestra los valores que toman los tres estimadores anteriores para cada uno de los actores sintéticos. Como se puede apreciar en la columna *Tareas ejecutadas*, los actores simples o no coordinados necesitan ejecutar 35 tareas para llegar al objetivo. Sin embargo, los actores coordinados reducen esta cantidad a 22 tareas. Además, estas tareas son no conflictivas, ya que no utilizan ningún recurso que esté siendo utilizado por otro agentes o que vaya a ser utilizado en un futuro cercano. Por tanto, el objetivo común puede ser alcanzado de una manera más rápida por un conjunto de actores coordinados, ya que pueden ejecutar sus tareas de forma paralela. Por lo respecta a la resolución de conflictos, el número de planes interrumpidos también se reduce de manera general en los agentes coordinados, ya que sus tareas interfieren con menos frecuencia. En resumen, los personajes coordinados necesitarán invocar menos veces al planificador para la selección de la siguiente acción, cosa que aumentará la fluidez de su comportamiento (ver la columna *Llamadas al planificador*).

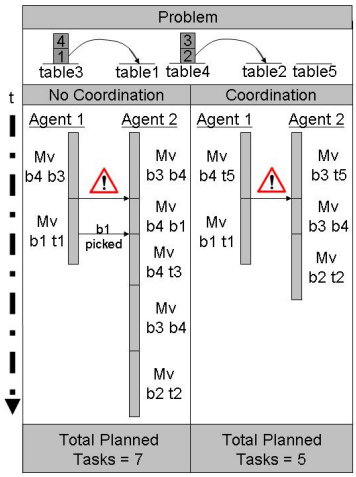


*Figura 6.5: Traza simple de cooperación entre dos agentes en el mundo de bloques 3D.*

En resumen, la inclusión de las tareas de los agentes externos en el proceso de planificación reduce los conflictos y produce un conjunto coordinado de tareas, cuya ejecución puede ser simultaneada por diversos agentes. A continuación, analizamos el impacto adicional conseguido con la inclusión de los objetivos externos, diferentes de los propios, en el proceso de búsqueda heurística.

### 6.3.2. Cooperación con objetivos externos

La cooperación con los objetivos de otros personajes puede servir para **reducir las obstrucciones y mejorar la eficacia** de los actores sintéticos. La traza de la figura 6.5 muestra un ejemplo sencillo de obstrucción entre dos actores que quieren mover las cajas que se encuentran bajo de las pilas, respectivamente, sobre las mesas llamadas `table1` y `table2`. Para ello disponen, además, de una mesa auxiliar (`table5`). Sin embargo, esta mesa está demasiado lejos para que el actor la considere como la primera opción a la hora de desapilar la caja que se encuentra en la cima de la pila, ya que hay movimientos posibles que requieren menos esfuerzo. Entonces, cuando los personajes no son conscientes de los objetivos externos, su actitud puede perjudicar los intereses de los actores que los rodean (ver parte izquierda de la figura 6.5). Por el contrario, la parte derecha de la figura 6.5 muestra la actuación cooperativa de los actores cuando conocen los objetivos del otro. En esta situación, los personajes deciden usar la mesa auxiliar para evitar obstruir al otro compañero, de manera que el número total de tareas planificadas se ve reducido.

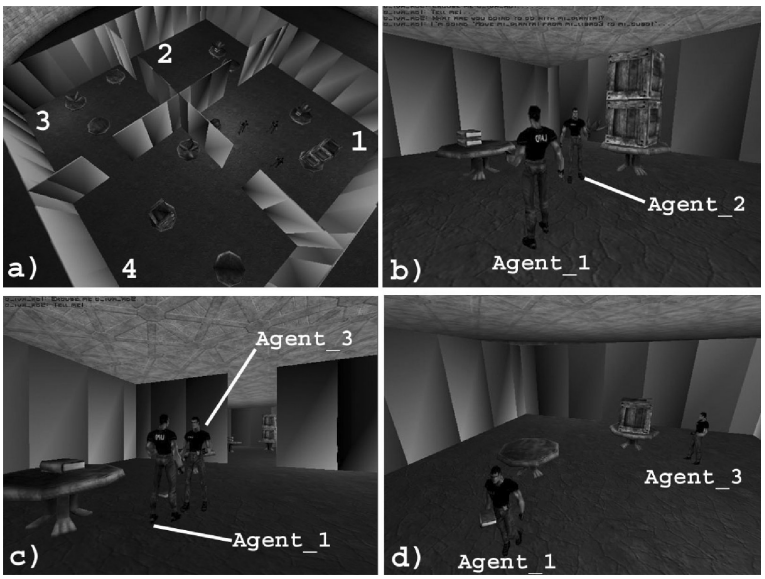


*Figura 6.6: Cooperación entre tres actores sintéticos en un entorno 3D.*

Los actores sociales basados en planificadores heurísticos, presentados en esta tesis, publican sus objetivos mediante el protocolo de formación de equipos explicado en la sección 5.3.1. Entonces, según la relación que haya entre los objetivos, los agentes deciden si quieren o no adquirir los objetivos externos y cooperar en su consecución. Esta aproximación, más flexible que la definición de una base de conocimiento global por parte del diseñador, se ajusta mucho al dinamismo de los entornos virtuales habitados (p. ej. permite la creación dinámica de equipos de trabajo).

Para estudiar la **adquisición de objetivos externos** hemos desarrollado un escenario 3D donde tres personajes deben organizar los objetos de un piso compuesto por 4 habitaciones (ver la figura 6.6a). Los objetivos de los personajes son los siguientes: $Agent\_1$ quiere llevar todos los libros a la habitación número 4; $Agent\_2$ desea juntar todas las plantas en la habitación 3; por último, $Agent\_3$ tiene un objetivo más grande formado por los dos objetivos previos y el objetivo de apilar todas las cajas en la habitación 4.

Al inicio, los actores no son conscientes de los objetivos del resto. Sin embargo, mientras avanza la simulación y su actuación se ve interrumpida, los personajes irán comunicándoselos. Por ejemplo, en la instantánea 6.6b, $Agent\_1$ tenía la intención de apartar una planta que estaba encima de los libros, sin embargo, $Agent\_2$ la ha cogido primero. En consecuencia, los actores inician los protocolos que intercambian tanto su tarea actual como sus objetivos. En esta situación, sin embargo, $Agent\_1$ y $Agent\_2$ recha-

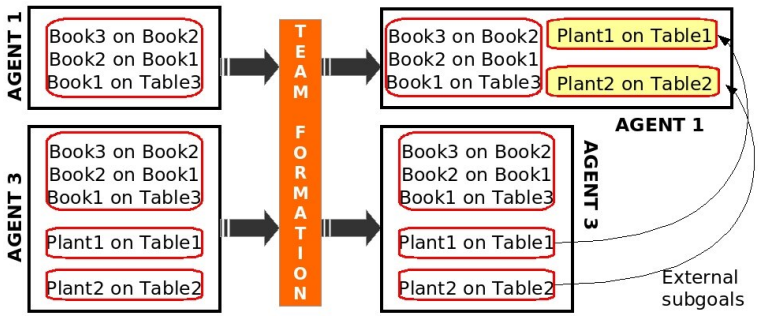


*Figura 6.7: Ejemplo de adquisición de objetivos externos.*

zan la cooperación ya que sus objetivos no están relacionados y consideran que sólo la coordinación con la tarea actual será suficiente. No obstante, cuando los objetivos están totalmente relacionados, los personajes forman un equipo de trabajo cooperativo, ya que los personajes no quieren estar continuamente molestándose. Por ejemplo, en la fotografía 6.6c, $Agent\_1$ y $Agent\_3$ compiten por un libro. La figura 6.7 muestra parcialmente los objetivos intercambiados y como $Agent\_1$ adquiere los dos subobjetivos asociados a las plantas como objetivos externos. Mientras que los actores forman parte de un equipo, la coordinación de tareas se aplica continuamente. Por ejemplo, cuando $Agent\_1$ mueve el libro a la habitación 4 en la fotografía 6.6d, $Agent\_3$ cambiará a otro subobjetivo para evitar interferencias (p. ej. apilar cajas en vez de recoger los libros).

Para medir el beneficio de la **cooperación** en función del peso heurístico asociado a los objetivos externos, hemos considerado tres estimadores en el ejemplo anterior: a) el número de tareas ejecutadas; b) el número de interrupciones o fallos; y c) el número de interacciones ($I$) que cada agente realiza sobre el mismo objeto. Los dos primeros están relacionados, como ja razonamos en la sección anterior, con la eficiencia y con la resolución de conflictos. El último estimador se utiliza como una aproximación de la cantidad de tareas repetidas por un agente, que serán innecesarias la mayoría de las veces.

La tabla 6.3 muestra los valores obtenidos para tres pesos diferentes, que uno puede asociar a diversos grados de cooperación. La primera fila de la tabla ($w_{ext} = 0$), corresponde con la situación en que los actores no prestan ninguna atención a los objetivos externos, es decir, no cooperan. Sin cooperación, los personajes necesitan ejecutar numerosas tareas para organizar los objetos del piso. La causa de ello la encontramos en la alta cantidad de interrupciones y de acciones repetidas. En término medio, los actores interaccionan con los objetos alrededor de 1.5 veces, cosa que nos lleva a pensar que la

| $w_{ext}$ | Agente | Tareas | Fallos | I(Avg) | I(Max) |
|---|---|---|---|---|---|
| 0 | A | 10 | 3 | 1.4 | 2 |
| | B | 15 | 7 | 1.6 | 2 |
| | C | 12 | 3 | 1.5 | 2 |
| 1 | A | 7 | 1 | 1 | 1 |
| | B | 5 | 1 | 1.3 | 2 |
| | C | 8 | 4 | 1 | 1 |
| 0.2 | A | 4 | 0 | 1 | 1 |
| | B | 5 | 1 | 1 | 1 |
| | C | 6 | 0 | 1.2 | 2 |

*Tabla 6.3: Diferentes grados de cooperación con objetivos externos.*

mitad de las tareas que las han de repetir dos veces (ver las columnas $I(Avg)$ i $I(Max)$). La segunda fila de la tabla ($w_{ext} = 1$), corresponde a la situación en la que los objetivos externos reciben la misma importancia que los propios. Esta cooperación máxima reduce los fallos, la cantidad de interacciones repetidas y, por tanto, el número de tareas ejecutadas por los personajes. Como discutimos en la sección 6.2, el uso de pesos elevados puede provocar que los personajes comiencen a trabajar para conseguir los objetivos de los otros y dejen de lado sus objetivos. En consecuencia, se puede ajustar el peso heurístico de los objetivos externos para priorizar la consecución de los objetivos propios, cosa que dará juego en la generación de otros roles con diferentes niveles de eficiencia. Por ejemplo, la tabla 6.3 muestra que, si rebajamos $w_{ext}$ a 0.2, los actores reducen ligeramente el número de tareas ejecutadas y consiguen su objetivo más rápidamente.

En resumen, la inclusión de los objetivos externos en el proceso de planificación permite que un actor sintético coopere con el resto de personajes que lo rodean. En este contexto, el sopesado de la heurística para los objetivos externos permite ajustar el **grado de cooperación**, cosa que afectará a la eficiencia, la capacidad de resolución de conflictos y, por tanto, a la calidad de los roles interpretados por los actores sintéticos.

### 6.3.3. Conclusiones a la colaboración con planificadores heurísticos

La técnica de colaboración con planificadores heurísticos, presentada en esta tesis, junta la toma de decisiones racional con un modelo social de **trabajo colaborativo**, de manera que permite desarrollar actores sintéticos que muestran ambas capacidades: racionalidad y sociabilidad.

El comportamiento social en entornos multipersonaje apunta, normalmente, a la capacidad de un actor de adaptarse a la actuación de los actores que lo rodean. De acuerdo con esto, el mecanismo de **detección de dependencias** incluido en los 3DIVA permite a los actores reaccionar a las interferencias externas. Esto es, no es el diseñador quien decide cuando interaccionan los actores ni tampoco quien establece los detalles concretos de la colaboración. Por el contrario, el comportamiento colaborativo surge de manera dinámica como el resultado de las interferencias entre los actores; cosa que proporcionará una flexibilidad superior a los mecanismos de animación comportamental comunes para personajes 3D (p. ej. guiones, máquinas de estados, HTN, etc.).

A la hora de colaborar, los actores utilizan dos mecanismos: la **coordinación de tareas** y la **cooperación con los objetivos externos**. Por un lado, la inclusión de las tareas de los agentes externos, en una fase de pre-planificación reduce los conflictos y produce un conjunto coordinado de tareas en sistemas que intercalan planificación y acción; en los que la construcción de un plan conjunto coordinado, en una fase de post-planificación, no es viable debido a la alta variabilidad del entorno. Por otro lado, la incorporación de los objetivos externos en el proceso de búsqueda genera decisiones cooperativas. En este contexto, el peso heurístico que los personajes dan a los objetivos externos sirve para ajustar su grado de cooperación. La capacidad de graduar la cantidad de cooperación es una característica fundamental para animar comportamientos verosímiles en actores sintéticos situados en entornos virtuales habitados. Así pues, el mecanismo de sopesado heurístico permitirá la simulación de diferentes modelos de personalidad de una manera sencilla. Por ejemplo, un personaje generoso utilizará un peso heurístico alto para dar importancia a la consecución de los objetivos externos.

La colaboración mediante los planificadores heurísticos es una buena técnica para seleccionar acciones en contextos sociales orientados a la ejecución de tareas, sin embargo, la búsqueda heurística tiene unos problemas que conviene discutir. En primer lugar tenemos su alto coste computacional, que crece de manera exponencial en función del número

de objetos de interacción de la escena. De manera que, si se quiere mantener la toma de decisiones dentro de un tiempo razonable con la verosimilitud de los personajes 3D, es necesario realizar una percepción inteligente que excluya los objetos innecesarios del estado de búsqueda. En segundo lugar se encuentra el criterio de decisión, que siempre está relacionado con la eficiencia; es decir, los mejores planes son aquellos que son más cortos. A nivel social, este criterio se manifiesta con la coordinación y la cooperación con el resto de agentes. Sin embargo, como los actores sintéticos no son robots ni han de comportarse como tales, se pueden encontrar criterios que sean sociales pero no eficientes. Por ejemplo, en un grupo de amigos, los actores podrían preferir competir por un recurso para poder estar juntos y hablar, en vez de coordinarse y realizar tareas separadas. De acuerdo con esto, en el ejemplo de la sección siguiente tratamos ambos aspectos: la representación semántica de mundos con muchos objetos de interacción y la toma de decisiones social multimodal mediante MADeM.

## 6.4. BAR UNIVERSITARIO VIRTUAL

En esta sección presentamos un entorno 3D complejo que simula un bar universitario virtual, donde un grupo de camareros sirven los pedido de un conjunto de clientes. El marco de simulación para animar este escenario ha sido implementado utilizando *Jason* [Bordini and Hübner, 2007], un sistema multiagente de propósito general que permite la definición de agentes de tipo BDI. Para la visualización 3D, hemos utilizado la librería gráfica *OpenSceneGraph* [OSG, 2008].

El bar virtual contiene un gran número de objetos interactivos (p. ej. vasos, platos, etc.), que han sido representados como instancias de clases definidas en una **ontología** específica del problema (ver la sección 6.4.1). Por ejemplo, algunos objetos típicos en un bar han sido modelados como dispensadores que tienen un tiempo asociado para proporcionar sus productos (p. ej. una cafetera tarda 2 minutos en hacer un café). En este contexto, los personajes deben coordinarse para evitar los conflictos por el uso de los recursos compartidos; que no puedan ser utilizados por dos agentes a la vez. Asimismo, los actores pueden estar relacionados socialmente mediante los conceptos definidos en la ontología (ver la sección 6.4.2). Por ejemplo, los camareros establecen relaciones de amistad con sus compañeros de trabajo. Esta red social deberá tenerse en cuenta a la hora de tomar decisiones sociales (p. ej. realizar favores, promover encuentros, etc.).

En conjunto, consideramos el ejemplo del bar virtual como un entorno adecuado para comprobar los efectos de **MADeM**, como el proceso para la toma de decisiones social de los actores sintéticos (ver el punto 5.4). En la sección 6.4.3, mostramos las funciones de utilidad que hemos diseñado para que los actores expresen sus preferencias según puntos de vista diferentes (p. ej. la eficiencia, el cansancio, las ganas de hablar, la pereza, etc.). Posteriormente, en función de la decisión ganadora, los agentes animarán el acuerdo social correspondiente (ver la sección 6.4.4). En la sección 6.4.5, repasamos los resultados sociales obtenidos por diversos tipos de personajes (p. ej. coordinados, habladores, perezosos, etc.) y por diferentes modelos de actitud (p. ej. egoismo, altruismo, reciprocidad, etc.). En la sección 6.4.6, analizamos con más detalle el efecto de la coordinación y la sociabilidad en las tareas realizadas por los actores sintéticos. Para finalizar, enunciamos las conclusiones de la toma de decisiones social multimodal llevada a cabo por MADeM.

### 6.4.1. Definición del entorno

Para construir el bar virtual hemos utilizado la **ontología base** que fue presentada en la sección 4.3.1 (para una definición completa en OWL ver el anexo A.1). Esta base semántica nos permite clasificar los objetos interactivos del mundo, así como sus interrelaciones. Por ejemplo, la figura 6.8 muestra una posible configuración de un bar virtual, cuya instanciación se encuentra en el anexo A.3. En este escenario, la clase *ServiceContainer* ha sido usada para modelar tanto una cafetera, como una bandeja con pasteles o una cubitera. Asimismo, diversos objetos han sido representados como instancias de la clase *HeterogeneousObjectContainer*, como por ejemplo *shelf-B*, un estante que contiene botellas de bebidas alcohólicas. Además, hemos implementado una **ontología específica del dominio** para definir los objetos típicos que se pueden encontrar en un bar, como las botellas (clase *Bottle*), platos (clase *Dish*), vasos (clase *Glass*), etc. (ver el anexo A.2).

La definición semántica del entorno puede servir para mejorar la **percepción** de escenas complejas, esto es, con muchos objetos de interacción. Por ejemplo, algunos objetos contenedores (p. ej. *refrigerator*, *cabinet* o *cupboard*) han sido configurados con el filtro de percepción *CLASSES-ONLY*, de manera que, mientras estén cerrados, sólo publican las clases de los objetos que contienen. Otro ejemplo de reducción sensorial lo encontramos en el objeto *shelf-A*, un estante con 20 vasos limpios, que hemos modelado como una instancia de la clase *HomegeneousObjectContainer*. La representación completa de este objeto inundaría fácilmente cualquier estado mental (debido a la gran cantidad de

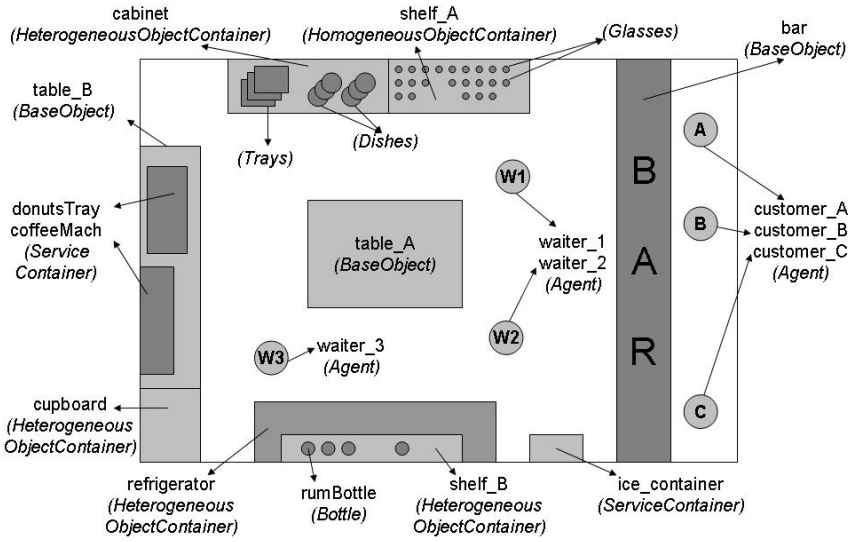


*Figura 6.8: Definición semántica del entorno del bar virtual.*

información asociada a los objetos). Además sería innecesaria, ya que todos los vasos son similares. Por tanto, la capa semántica resume la información percibida en tiempo de simulación. De manera que, a pesar de que informa sobre la cantidad total de vasos contenidos, sólo proporciona una descripción completa de un conjunto reducido de vasos. El escenario mostrado en la figura 6.8 contiene 68 objetos interactivos. Sin ningún tipo de filtrado, la cantidad de información necesaria para describir en lógica proposicional las propiedades y las relaciones de todos estos objetos sería de 612 literales. Sin embargo, el filtro basado en la ontología realizado por la capa semántica reduce esta cantidad a 252 literales, referidos a 28 objetos interactivos. Por tanto, produce una reducción del 60 % en el flujo de información que describe la escena. Así pues, podemos garantizar estados mentales de un tamaño razonable para los 3DIVA que lo perciben.

### 6.4.2. Definición de los actores sintéticos

Al igual que los objetos que forman el escenario, los actores sintéticos se interrelacionan mediante los conceptos (clases, propiedades y relaciones) definidos en la ontología. Por ejemplo, la figura 6.9 muestra algunos ejemplos de **relaciones sociales** definidas entre los personajes que habitan el bar universitario virtual. Todos los camareros se relacionan a través de una *groupSocialRelation* con el grupo *Waiters*, una clase que representa su rol. Asimismo, los camareros pueden relacionarse individualmente con otros camareros que sean amigos de trabajo (*workFriend*). Esta es una relación bidireccional pero no transiti-

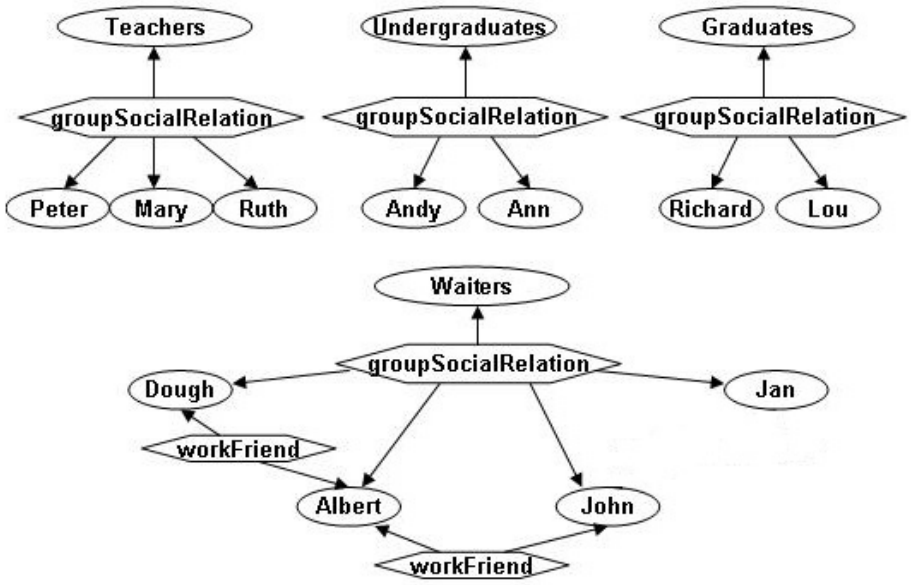


*Figura 6.9: Relaciones sociales entre los personajes que habitan el bar virtual.*

va. Por ejemplo, en la figura 6.9, Albert es amigo de Dough y John, pero los últimos no lo son entre ellos. Referente a los clientes, hemos especificado tres grupos o clases sociales: profesores, estudiantes no graduados y graduados. La red social definida por estas relaciones se utiliza para promover encuentros sociales entre los personajes del entorno.

Los controladores de los agentes han sido definidos mediante planes implementados en la versión extendida del lenguaje *AgentSpeak(L)* [Rao, 1996], que proporciona *Jason* para la definición de los agentes. De acuerdo con estos planes, la **actuación de los camareros** está gobernada por la máquina de estados finitos de la figura 6.10a. Los camareros sirven los pedidos, básicamente, en dos pasos: a) usan el dispensador correspondiente para obtener el producto demandado (p. ej. la plancha para hacer un bocadillo); y b) entregan el producto al cliente. Sin embargo, la ejecución de estas tareas se subasta usando MA-DeM, de cara a determinar buenas asignaciones sociales. La tabla 6.4 resume las tareas ejecutadas por los actores y las ranuras consideradas para cada una de ellas. Tal i como hemos dicho antes, los encargados de las tareas *Use* y *Give* son los camareros, que tratan de asignarlas al mejor candidato. Por tanto, en ambos casos, la ranura que se subasta será la del agente ejecutor de la tarea ($Ag_e$). El conjunto de asignaciones entre las que decidir está representado por la ecuación 6.1. Es decir, los camareros evalúan la posibilidad de realizar cada tarea contra la posibilidad de pasarla a otro actor, de manera que puedan encargarse de la siguiente tarea pendiente. En la sección 6.4.3, mostramos las funciones de utilidad utilizadas por los camareros para expresar sus preferencias sobre cada una de

| Tasks/Slots | $Ag_e$ | $Obj_m$ | $Ag_d$ |
|---|---|---|---|
| Use | Waiter | ProductDispenser | - |
| Give | Waiter | Product | Customer |
| SitAt | Customer | SeatingPlace | - |

*Tabla 6.4: Tareas subastadas por los agentes en el bar virtual.*

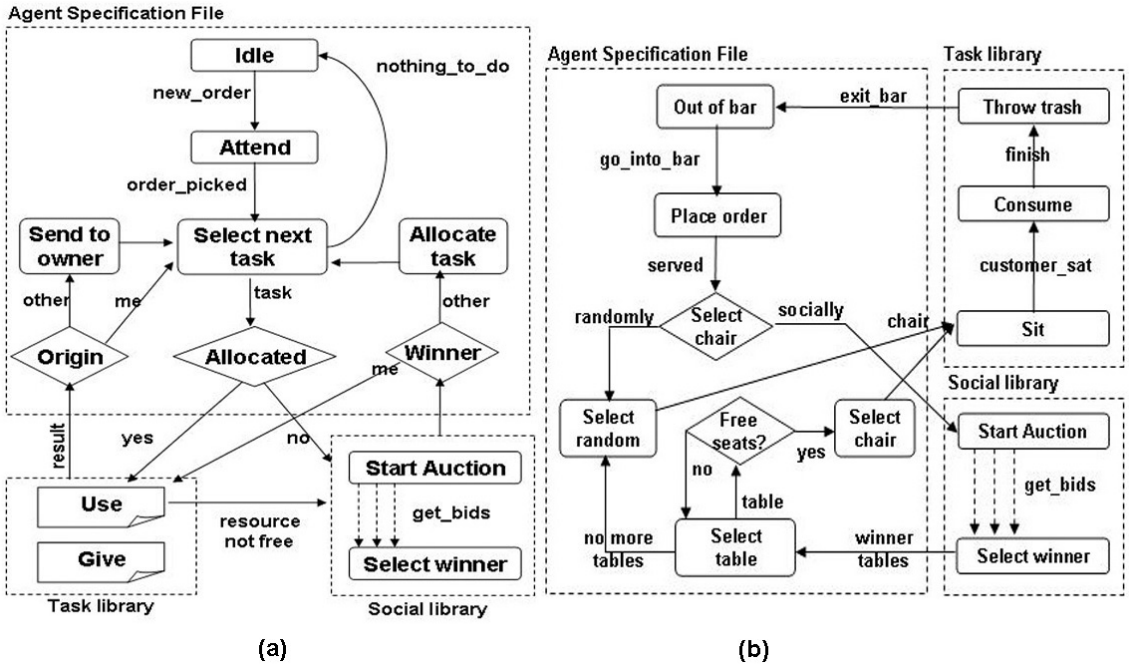


*Figura 6.10: Controladores de los agents: (a) camareros y (b) clientes.*

estas posibilidades.

$$\begin{cases} P_0 = \{t(Ag_e) \leftarrow Myself\} \\ P_i = \{t(Ag_e) \leftarrow a_i, t_{next}(Ag_e) \leftarrow Myself\} \quad \forall a_i \in A \end{cases} \tag{6.1}$$

En segundo lugar, la figura 6.10b muestra el **controlador de los clientes** del bar virtual. Los clientes hacen sus pedidos en la barra y consumen los productos cuando los camareros les han servido. En esta ocasión, estamos interesados en animar comportamientos sociales de clase, de manera que los clientes deseen consumir junto con aquellos actores que sean de la misma clase social. Por tanto, los clientes utilizan MADeM para

decidir donde sentarse a consumir. Como el problema de decisión de los clientes será escoger entre diferentes asientos (*SeatingPlaces*) para consumir, los clientes subastan la ranura $Obj_m$ de la acción *SitAt*, que corresponde con el sitio donde sentarse (ver la tabla 6.4). Las clases de los objetos, usadas para describir las ranuras de las tareas, se extraen de la taxonomía de objetos definida en la ontología del bar virtual. Esto es, objetos como las mesas o la barra del bar serán implementados como instancias de la clase *SeatingPlace* y, por tanto, serán considerados como lugares posibles donde sentarse a consumir. La ecuación 6.2 define el conjunto de posibilidades entre las que decidir, es decir, los sitios candidatos para sentarse son todas las mesas del entorno así como la propia barra del bar.

$$\begin{cases} P_0 = \{SitAt(Obj_m) \leftarrow bar\} \\ P_j = \{SitAt(Obj_m) \leftarrow table\} \quad \forall table \in Tables \end{cases} \tag{6.2}$$

A continuación, mostramos el conjunto de funciones de utilidad utilizadas por los camareros y por los clientes del bar virtual. Estas funciones expresan, según diversos puntos de vista, las **preferencias** sobre cada una de las posibles soluciones a un problema dado, representadas en las ecuaciones 6.1 y 6.2.

### 6.4.3. Expresión de preferencias

En el ejemplo del bar virtual, los camareros tienen en cuenta tres puntos de vista cuando expresan sus preferencias: la **eficiencia**, la **sociabilidad** y el **cansancio**. Las ecuaciones 6.3 y 6.4 definen los valores de utilidad retornados por la función de utilidad de eficiencia para las tareas *Use* y *Give*. Esta función tiene el propósito de maximizar el número de tareas realizadas simultáneamente y representa el deseo de los camareros de servir los pedidos tan rápido como sea posible. El comportamiento social, definido por los camareros, está orientado a animar conversaciones con sus amigos del trabajo (aquellos con quien tienen una relación de tipo *workFriend*). Para hacerlo, los camareros implementan la función de utilidad social que muestran las ecuaciones 6.5 y 6.6; en las que *Near* computa la distancia que hay entre los agentes mientras ejecutan un par de tareas. La función de utilidad social evalúa el interés social como la posibilidad de encontrar un amigo en un futuro próximo y, por tanto, poder animar un diálogo con e?. Por último, la ecuación 6.7 define la función de utilidad del cansancio de un camarero. Esta función implementa el principio básico de mínima energía, que los humanos aplicamos a menudo en el trabajo.

El tipo de sociedad simulada para los camareros es **elitista**, en consecuencia, MADeM escogerá aquellas asignaciones que maximicen las funciones de utilidad arriba descritas.

$$U^{perf}(\text{Use}(Ag_e) \leftarrow a) = \begin{cases} 0 & \text{if } a \neq Myself \\ 1 & \text{if } (a = Myself) \wedge \\ & IsUsing(Myself, Obj_m) \wedge \\ & not(IsComplete(Obj_m))] \\ \frac{1}{t_{tobefree}+t_{queue}} & \text{Otherwise} \end{cases} \tag{6.3}$$

$$U^{perf}(\text{Give}(Ag_e) \leftarrow a) = \begin{cases} 0 & \text{if } a \neq Myself \\ 1 & \text{if } (a = Myself) \wedge \\ & CurrentTask = \text{'Give'} \wedge \\ & not(HandsBusy(Myself) < 2)] \\ \frac{1}{t_{tobefree}} & \text{Otherwise} \end{cases} \tag{6.4}$$

$$U^{soc}(t_1(Ag_e) \leftarrow a_1, t_2(Ag_e) \leftarrow a_2) = \begin{cases} 0 & \text{if } a_1 \neq Myself \vee a_2 \neq Auctioneer \\ 1 & \text{if } (a_1 = Myself) \wedge a_2 = Auctioneer \wedge \\ & IsFriend(a_1, a_2) \wedge Near(Pos(t_1), Pos(t_2)) \wedge \\ & ExecTime(t_2) > RemainTime(CurrentTask) \\ 0 & \text{Otherwise} \end{cases} \tag{6.5}$$

$$U^{soc}(t(Ag_e) \leftarrow a) = \begin{cases} 0 & \text{if } a \neq Auctioneer \\ 1 & \text{if } a = Auctioneer \wedge IsFriend(a, Myself) \\ & \wedge Near(Pos(CurrentTask), Pos(t)) \wedge \\ & \wedge TimeToStart(t, a) = Now \\ 0 & \text{Otherwise} \end{cases} \tag{6.6}$$

$$U^{tir}(t(Ag_e) \leftarrow a) = \begin{cases} 1 - \frac{tasks\_done}{total\_tasks} & \text{if } a = Myself \\ 0 & \text{Otherwise} \end{cases} \tag{6.7}$$

Por lo que respecta a los clientes, consideramos dos puntos de vista: la **sociabilidad** y la **pereza**. Las ecuaciones 6.8 y 6.9 corresponden con la función de utilidad social definida para los clientes. Esta función asigna el valor máximo a una mesa en el caso

en que, sentado en ella, haya un cliente en de la misma clase social (profesor, estudiante graduado o no graduado). El hecho de consumir solo en la barra no tiene ningún interés social, por tanto, su valor de utilidad es cero. La función de utilidad de la pereza está representada por las ecuaciones 6.10 y 6.11. Esta función evalúa las mesas en función de su distancia al cliente y, al contrario de la sociabilidad, que el cliente consuma en la barra se considera la mejor opción. El tipo de sociedad simulada para los clientes es **utilitaria**, así pues, MADeM escogerá aquella asignación que maximice la suma de los valores de utilidad previamente definidos.

$$U^{soc}(SitAt(Obj_m) \leftarrow table) = \begin{cases} 1 & \text{if } AtTable(Myself, table) \wedge \\ & IsSameClass(Myself, Auctioneer) \wedge \\ & AvailablePlace(table) \\ 0 & \text{Otherwise} \end{cases} \tag{6.8}$$

$$U^{soc}(SitAt(Obj_m) \leftarrow bar) = 0 \tag{6.9}$$

$$U^{laz}(SitAt(Obj_m) \leftarrow table) = \begin{cases} \frac{\min_{t \in Tables}\{DistanceTo(t)\}}{DistanceTo(table)} & \text{if } Auctioneer = Myself \wedge \\ & AvailablePlace(table) \\ 0 & \text{Otherwise} \end{cases} \tag{6.10}$$

$$U^{laz}(SitAt(Obj_m) \leftarrow bar) = \begin{cases} 1 & \text{if } Auctioneer = Myself \wedge \\ & AvailablePlace(bar) \\ 0 & \text{Otherwise} \end{cases} \tag{6.11}$$

Las funciones de utilidad, definidas previamente para los camareros y los clientes, expresan una preferencia diferente de cero sólo bajo ciertas condiciones. Como un valor de utilidad cero es similar a no expresar ninguna preferencia, las utilidades cero no se envían al subastador. Mediante este detalle de implementación, el problema de determinación del ganador se ve simplificado, ya que no considera apuestas que no aportan nada a la decisión.

### 6.4.4. Animación de acuerdos sociales

Los controladores de los camareros y de los clientes invocan MADeM cuando el actor ha de tomar una decisión social (ver la figura 6.10). Posteriormente, es necesario recoger la decisión ganadora y animar el **acuerdo social** que corresponda. Para animar los acuerdos sociales, el módulo de comunicación proporciona diversos protocolos de animación de conversaciones entre personajes 3D. Por ejemplo, los diálogos *tell* y *shout* sirven para que los agentes se soliciten la ejecución de tareas, respectivamente, con un tono de voz normal o a gritos.

Como caso de uso, consideremos que un camarero ($A$) evalúa dos puntos de vista a la hora de subastar la ejecución de una tarea ($t$): la eficiencia y la sociabilidad. La tabla 6.5 resume los planes animados por los camareros dependiendo del criterio ganador. La mitad superior de la tabla corresponde con la eficiencia, esto es, cuando el ganador es aquel que realiza la tarea en el menor tiempo. Si es un agente externo $B$ el que gana la subasta, el agente subastador $A$ puede animar el acuerdo de diversas maneras. Por ejemplo, puede acercarse y empezar un diálogo donde le solicite que ejecute la tarea $t$, o bien, notificárselo a gritos (p. ej. el agente $A$ lanza el grito: *"Por favor $B$, hazme un café!"*). En ambos diálogos, hace falta una respuesta positiva por parte de $B$ para que la animación sea consistente con el resultado obtenido de MADeM. A menudo, cuando el agente subastador es también el ganador, los agentes no animan ningún acuerdo social sino que simplemente el agente $A$ comienza a ejecutar la tarea. No obstante, la animación de estas situaciones puede ser también muy útil para reflejar desacuerdos entre los agentes, es decir, conversaciones con una respuesta negativa (p. ej. cuando el agente $B$ responde: *"Lo siento, no puedo hacerlo porque estoy muy ocupado"*).

La mitad inferior de la tabla 6.5 muestra un ejemplo de los diálogos que se pueden animar cuando el criterio ganador ha sido la sociabilidad. En este caso, los personajes planifican una cita mediante la acción *plan_meeting*. La planificación de citas es un diálogo en el que los actores acuerdan un encuentro para poder hablar (p. ej. *"Por favor, ven a la cafetera y hablamos un rato"*). Posteriormente, la animación de la conversación (*chat*) entre los personajes se iniciará cuando los actores tengan una cita previa y estén a la distancia adecuada. Como hemos visto en ambos casos, la decisión sobre la animación a utilizar depende de unas precondiciones, como por ejemplo el nivel de ruido del ambiente (*noise*) o la distancia entre los agentes y los recursos (*near*). El cumplimiento de estas precondiciones se realizará en función del estado de la escena. Por ejemplo, el nivel de

| Performance winner | | | |
|---|---|---|---|
| Winner | Preconditions | Action | Response |
| $A$ | None | None | None |
| | near($B$,Resource)<br>not(near($B$,$A$)) | shout($A$,$B$,make($t$)) | shout($B$,$A$,no) |
| | near($B$,Resource)<br>near($B$,$A$) | tell($A$,$B$,make($t$)) | tell($B$,$A$,no) |
| $B$ | noise(high) | approach($B$), tell($A$,$B$,make($t$)) | tell($B$,$A$,yes) |
| | noise(low) | shout($A$,$B$,make($t$)) | shout($B$,$A$,yes) |
| Sociability winner | | | |
| $A$ | None | plan_meeting($A$,$B$),chat($A$,$B$) | chat($B$,$A$) |
| $B$ | noise(high) | approach($B$), tell($A$,$B$,make($t$)),<br>plan_meeting($A$,$B$),chat($A$,$B$) | tell($B$,$A$,yes)<br>chat($B$,$A$) |
| | noise(low) | shout($A$,$B$,make($t$)),<br>plan_meeting($A$,$B$),chat($A$,$B$) | shout($B$,$A$,yes)<br>chat($B$,$A$) |

*Tabla 6.5: Ejemplos de conversaciones para animar acuerdos sociales.*

ruido puede ser fácilmente calculado de acuerdo con el número de agentes de la escena o sólo teniendo en cuenta aquellos que se encuentran cerca del ganador.

### 6.4.5. Resultados sociales

En esta sección mostramos un conjunto de resultados sociales, obtenidos con diferentes camareros y clientes que utilizan MADeM como su mecanismo de toma de decisiones social. Los valores que analizamos a continuación corresponden a simulaciones en las que 10 camareros sirven las peticiones de 100 clientes en el bar universitario virtual.

Como presentamos en la sección 6.4.3, hemos modelado una sociedad elitista de camareros en la que los actores tienen en cuenta tres puntos de vista (eficiencia, sociabilidad y cansancio); cada uno representado por su función de utilidad. En este contexto, los pesos de utilidad pueden ser ajustados para crear camareros con diversos tipos de **personalidad**. Por ejemplo, un camarero *coordinado* podría dar el triple de importancia al criterio de la eficiencia que a los criterios de sociabilidad y cansancio. De acuerdo con esto, un vector de pesos de utilidad para un camarero *coordinado* sería $\overrightarrow{w_u} =< 0{,}75, 0{,}125, 0{,}125 >$, donde cada componente representa la importancia dada a cada una de las funciones de utilidad evaluadas. De manera similar, hemos creado los camareros *sociales* (o habladores) como actores que utilizan el siguiente vector de pesos de utilidad $\overrightarrow{w_u} =< 0{,}125, 0{,}75, 0{,}125 >$ y los camareros *igualitarios* (o calculadores) como los que utilizan vector $\overrightarrow{w_u} =< 0{,}125, 0{,}125, 0{,}75 >$.

La tabla 6.6 resume las tareas realizadas por los camareros anteriores (*coordinados*, *sociales* i *igualitarios*) en comparación con un conjunto de camareros *autointeresados*; esto es, que no incluyen ningún mecanismo de toma de decisiones social. Los camareros *coordinados* muestran una mejor eficiencia (ver la columna $Tasques/s$), ya que la mayoría de conflictos causados por la voluntad de usar el mismo recurso (p. ej. la cafetera) se resuelven mediante la **especialización**. Esto es, delegando la ejecución de la tarea en otro camarero que esté ya utilizando el susodicho recurso. Por otro lado, los camareros *sociales* tardan más en servir los pedidos de los clientes, ya que animan más conversaciones entre los camareros (ver la media de diálogos animados en la columna $\overline{NXats}$). Los camareros *igualitarios* dan más importancia a la función de utilidad de cansancio y, por tanto, tratan de asignar la ejecución de la tarea al camarero que esté más descansado. En consecuencia, la desviación estándar del número de tareas ejecutadas por cada agente

| Tipo de actor | $\sigma_{NTasques}$ | $\overline{NXats}$ | $Tasques/s$ |
|---|---|---|---|
| Coordinado | 6.73 | 5 | 0,91 |
| Social | 4.37 | 29.4 | 0.65 |
| Igualitario | 2.74 | 6.6 | 0.62 |
| Autointeresado | - | - | 0.17 |

*Tabla 6.6: Tareas realizadas por diversos tipos de camareros.*

tiende a cero para estos actores (ver la columna $\sigma_{NTasques}$). Por último, los camareros *autointeresados* demuestran trabajar de forma menos eficiente que cualquier tipo de camarero con habilidades sociales. Como estos actores no realizan reasignación de tareas ni diálogos sociales, las columnas $\sigma_{NTasques}$ y $\overline{NXats}$ no han sido consideradas.

A parte de definir la importancia de cada punto de vista mediante el vector de pesos de utilidad ($\overrightarrow{w_u}$), MADeM permite la definición de un vector de pesos personales ($\overrightarrow{w}$), que modelan la actitud de un actor hacia el resto de individuos. La tabla 6.7 muestra los resultados obtenidos por los camareros definidos previamente usando los cuatro **modelos de actitud** presentados en la sección 5.4.4: **indiferencia, reciprocidad, altruismo y egoísmo**. Los actores que utilizan el modelo de indiferencia no realizan ninguna modificación sobre las utilidades recibidas del resto de personajes, por tanto, consideramos los resultados obtenidos para la indiferencia como los valores base con los que comparar el resto de actitudes. La reciprocidad sopesa las utilidades de acuerdo con el número de favores intercambiado entre los agentes. Esta actitud produce el equilibrio entre el número de favores intercambiados, cosa que se puede apreciar en el bajo valor que toma $\sigma_{Favors}$ para los tres tipos de camareros recíprocos ($< 1{,}15, 1{,}76, 2{,}4 >$); en comparación con los valores obtenidos con el modelo de indiferencia ($< 7{,}57, 3{,}52, 7{,}58 >$). El altruismo ha sido implementado de manera que el peso que los agentes dan a su utilidad es 0.25, mientras que el peso dado a las utilidades del resto de personajes es 0.75. Como consecuencia, los agentes altruistas realizan un mayor número de favores, ya que la importancia que dan a sus opiniones es tres veces inferior a la dada a las preferencias externas (ver los altos valores de la media de favores intercambiados $\overline{Favors} < 17, 12{,}7, 17{,}9 >$). Por el contrario, el egoísmo sopesa con 0.75 las preferencias propias y con 0.25 las externas, así pues, los actores raramente hacen favores (ver los favores tan bajos de la columna $\overline{Favors}$ para el egoísmo $< 0{,}7, 0{,}4, 0{,}1 >$).

| | Coordinado | | Social | | Igualitario | |
|---|---|---|---|---|---|---|
| Actitud | $\sigma_{Favors}$ | $\overline{Favors}$ | $\sigma_{Favors}$ | $\overline{Favors}$ | $\sigma_{Favors}$ | $\overline{Favors}$ |
| Indiferencia | 7.57 | 6.9 | 3.52 | 8.7 | 7.58 | 13.6 |
| Reciprocidad | 1.15 | 8.8 | 1.76 | 7.8 | 2.4 | 15.5 |
| Altruismo | 5.94 | 17 | 6.66 | 12.7 | 4.44 | 17.9 |
| Egoismo | 1.41 | 0.7 | 0.81 | 0.4 | 0.47 | 0.1 |

*Tabla 6.7: Favores intercambiados con diferentes modelos de actitud personal.*

Las actitudes personales pueden, a veces, ir en contra de las preferencias de los actores, y viceversa. Por ejemplo, mientras que la reciprocidad trata de balancear el número de favores, el cansancio tiende a asignar las tareas al camarero menos cansado (ver la alta $\sigma_{Favors}$ de los personajes igualitarios recíprocos en la tabla 6.7). Otro ejemplo lo encontramos en el egoísmo aplicado a los camareros igualitarios, que no intercambian prácticamente la ejecución de ninguna tarea ($\overline{Favors} = 0{,}1$). Sin embargo, las actitudes personales también pueden fortalecer las preferencias de los actores. Por ejemplo, el altruismo aplicado a los camareros coordinados produce un alto nivel de especialización. Este tipo de actores genera grandes valores de $\sigma_{Favors}$, ya que los agentes que comienzan a usar un recurso (p. ej. una máquina de hacer zumos) permanecerán extrayendo productos de acuerdo con su política de comportamiento altruista y coordinado. A pesar de estas variaciones en los efectos de los modelos de actitud, los pesos de actitud han demostrado producir efectos similares en los personajes, independientemente del tipo de camarero que se considere: coordinado, social o igualitario.

Al contrario de los camareros, los clientes toman sus decisiones dentro de una sociedad utilitaria, en la que consideran dos puntos de vista (sociabilidad y pereza). La figura 6.11 muestra una comparativa del comportamiento obtenido en función del peso de utilidad asignado a cada uno de estos criterios. En esta gráfica comparamos dos métricas: a) la media de encuentros sociales producidos entre los clientes; y b) la distancia media recorrida por un cliente a la hora de consumir. Los clientes perezosos, que aplican un peso de utilidad bajo a la sociabilidad, deciden consumir preferiblemente en la barra o sentarse en una mesa que esté cerca de ellos (ver el punto $\overrightarrow{w_u} =< 0{,}3, 0{,}7 >$). En este caso, la distancia recorrida hasta el punto de consumo es corta pero sólo unos pocos encuentros sociales son animados. Por otro lado, los clientes sociales, que aplican pesos más altos a la sociabilidad, realizan más encuentros sociales pero también necesitan recorrer distancias

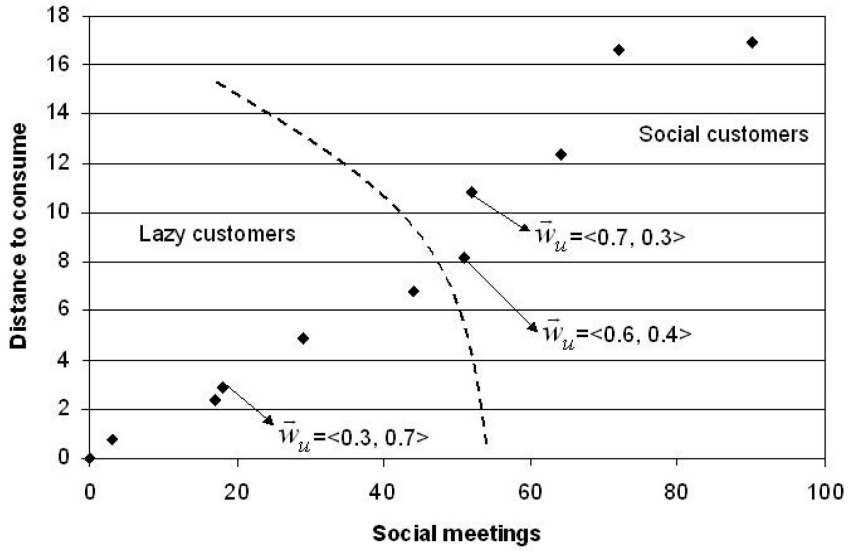


*Figura 6.11: Comparativa entre los clientes perezosos y los clientes sociales.*

más largas para encontrar a sus compañeros. Por ejemplo, los puntos $\overrightarrow{w_u} =< 0{,}6, 0{,}4 >$ y $\overrightarrow{w_u} =< 0{,}7, 0{,}3 >$ de la figura 6.11 son un ejemplo de los resultados obtenidos con clientes sociales.

Los efectos de la **sociabilidad** sobre los clientes se pueden ver en la figura 6.12. Esta instantánea muestra un bar virtual poblado por 7 camareros y 17 clientes. Los clientes pertenecen a los tres grupos sociales definidos previamente: profesores, estudiantes no graduados y graduados. En esta simulación los clientes usan el vector de pesos de utilidad $\overrightarrow{w_u} =< 0{,}7, 0{,}3 >$. Por consiguiente, una vez que han sido servido, los actores normalmente deciden consumir junto con otros clientes de su grupo social. Para distinguir a los miembros de cada grupo social, utilizamos avatares diferentes. En la figura 6.12 se aprecia como los avatares del mismo tipo se juntan para consumir en la misma mesa siempre que pueden.

En resumen, en esta sección hemos presentado los efectos de la **multimodalidad** de criterios y de las **actitudes personales** en la toma de decisiones de los actores sintéticos. A continuación, analizamos más detalladamente los resultados obtenidos para dos criterios muy importantes dentro del comportamiento en grupo: la coordinación y la sociabilidad.

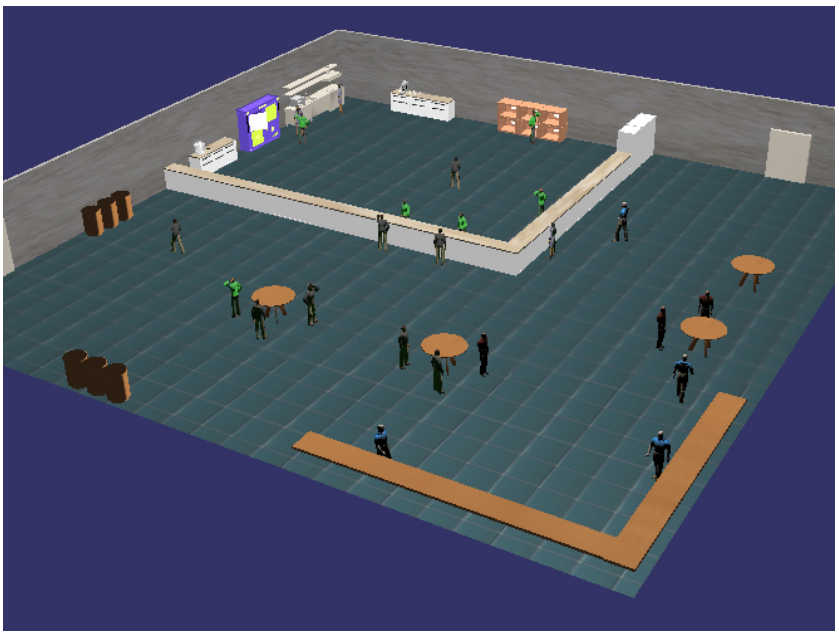

*Figura 6.12: Entorno 3D habitado que simula un bar virtual.*

### 6.4.6. Coordinación vs. Sociabilidad

En el campo de los sistemas multiagente, el comportamiento social ha estado asociado a menudo con la capacidad de los agentes de actuar de forma coordinada. Sin embargo, los personajes 3D pueden mostrar actitudes sociales con **comportamientos no necesariamente orientados a la eficiencia**. Por ejemplo, los camareros y los clientes descritos en las secciones anteriores implementan la sociabilidad como la habilidad de conversar con otros personajes. De acuerdo con esto, en esta sección comparamos los efectos de la coordinación y de la sociabilidad en el comportamiento de los actores sintéticos. Para hacerlo, utilizamos los camareros definidos con anterioridad, pero no tenemos en cuenta el criterio de cansancio ($w_{tir} = 0$). Así pues, el peso de utilidad asignado a la sociabilidad $w_{soc}$, que aquí lo llamamos *factor de sociabilidad*, es el parámetro con el que balancearemos eficiencia y sociabilidad, ya que $w_{perf} = 1 - w_{soc}$.

En primer lugar, consideremos unas trazas sencillas de comportamiento donde intervienen un conjunto reducido de camareros y clientes. La figura 6.13 esquematiza dos simulaciones en las que 2 camareros reciben los pedidos de 4 clientes en el bar virtual. En estas simulaciones, todos los clientes piden un bocadillo pero la plancha, recurso necesario para prepararlos, se comparte entre los camareros. A pesar de que la plancha no puede ser utilizada por más de un camarero a la vez, hemos proporcionado a este objeto

la capacidad de hacer 4 bocadillos de manera simultánea. Entonces, los camareros pueden coordinarse para servir a los clientes rápidamente. La figura 6.13a muestra la traza resultante cuando los camareros tienen un grado de coordinación máxima ($w_{perf} = 1$). En este caso, *Waiter1* se especializa en hacer bocadillos, mientras que *Waiter2* se encarga de coger los pedidos y de servir a los clientes. Esto es, los camareros trabajan de la manera más eficiente posible.

Sin embargo, un exceso de eficiencia es más común en el comportamiento de los robots que en las sociedades humanas. Por tanto, el factor de sociabilidad puede ser ajustado para incorporar interacciones sociales que respondan a las relaciones definidas entre los actores, com por ejemplo la amistad en el trabajo (*workFriends*). Por ejemplo, la figura 6.13b corresponde a la situación en la que los camareros utilizan $w_{soc} = 0{,}6$. En esta configuración, *Waiter2* primero decide ser eficiente y pasa la acción de preparar el bocadillo a *Waiter1*, que ya estaba utilizando la plancha para preparar el bocadillo de *Customer1*. Sin embargo, cuando *Waiter2* atiende a *Customer3*, la opción escogida no es pasar la ejecución de la tarea sino hablar con su compañero *Waiter1*, mientras éste acaba de preparar los bocadillos que tiene en marcha. Como consecuencia, los clientes han de esperar más tiempo para ser servidos pero el comportamiento animado por los camareros es más rico gracias a la **variabilidad** que desprenden las nuevas situaciones sociales animadas.

La figura 6.14 muestra la animación 3D de la traza de la figura 6.13b. Los camareros atienden a los clientes por orden de aparición mediante un diálogo estándar como el que aparece en la instantánea 6.14a. Cuando el recurso necesario para ejecutar una tarea (p. ej. la plancha para hacer un bocadillo) está siendo usado por otro personaje, el camarero considera dos posibilidades: a) pasar la tarea y continuar atendiendo otros pedidos; o b) animar una conversación on diálogo social mientras espera que el recurso se libere. Por ejemplo, en la fotografía 6.14b, *Waiter2* se da cuenta de que *Waiter1* está ya preparando un bocadillo y le pide que le haga otro. Cuando la tarea reasignada ha sido completada, el camarero que la realizó informa al solicitante del resultado. Por ejemplo, en la viñeta 6.14c, *Waiter1* comunica a *Waiter2* que el bocadillo ya está preparado. Análogamente a usar un recurso para resolver diversas tareas de manera simultánea, los camareros pueden usar sus manos para llevar más de un producto a la vez. Por ejemplo, en la viñeta 6.14d, *Waiter2* lleva dos bocadillos para *Customer2* y *Customer3*.

Para estimar numéricamente los efectos del **balanceo entre la coordinación y la sociabilidad** hemos creado un conjunto de simulaciones que varían el factor de sociabilidad

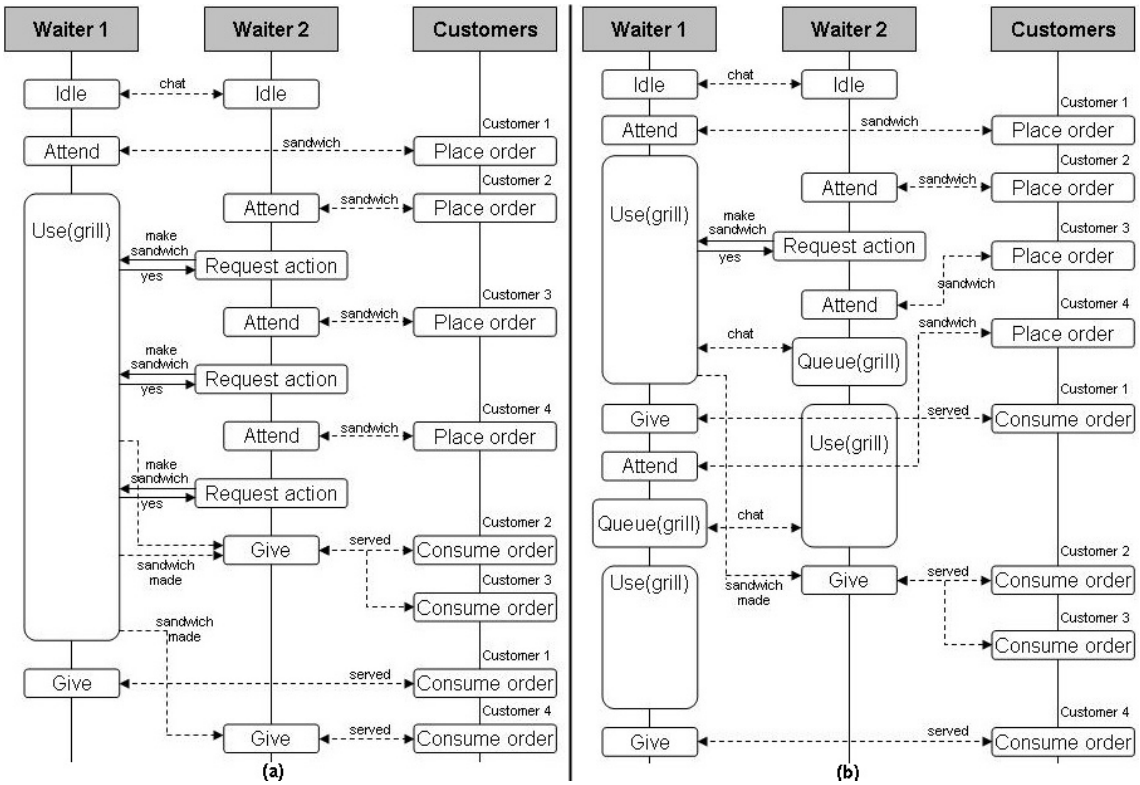


*Figura 6.13: Trazas sencillas de comportamiento: (a) coordinación máxima entre camareros; (b) balanceo entre coordinación y sociabilidad.*

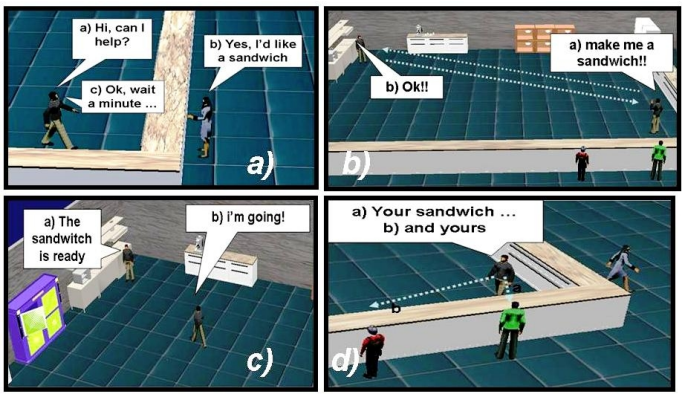


*Figura 6.14: Animación de diversas situaciones interactivas: (a) Atender a un cliente, (b) Pedir un favor a un compañero, (c) Informar del resultado de una acción, y (d) Servir a un cliente.*

de los camareros, desde el valor mínimo $w_{soc} = 0$ al valor máximo $w_{soc} = 1$. Medimos la eficiencia de un grupo de actores mediante el estimado $Throughput$. Este estimador es un indicador que toma valores en el rango $[0, 1]$ y que estima cómo de cerca se encuentra una simulación de la situación ideal, en la que la carga de trabajo puede ser distribuida entre todos los miembros del grupo y no se produce ninguna colisión. De acuerdo con esto, la ecuación 6.12 define $Throughput$ como la relación entre el tiempo de simulación ideal ($T^*_{sim}$) y el tiempo de simulación real ($T_{sim}$), donde $N_{tasks}$ y $N_{agents}$ es el número de tareas y de actores respectivamente mientras que $\overline{T_{task}}$ es el tiempo medio que cuesta ejecutar una tarea.

$$Throughput = \frac{T^*_{sim}}{T_{sim}} = \frac{N_{tasks} * \overline{T_{task}}/N_{agents}}{T_{sim}} \tag{6.12}$$

La figura 6.15a muestra los valores de $Throughput$ obtenidos con los diferentes camareros sociales, en comparación con los camareros autointeresados (sin ningún mecanismo social incluido). Como ya sabemos, los actores autointeresados colisionan constantemente a causa de la competición por el uso de los recursos compartidos. Estas colisiones producen tiempos de espera elevados que reducen la eficiencia en tanto que el número de actores crece, cosa que producirá una animación comportamental de baja calidad. La eficiencia en el comportamiento puede ser mejorada con personajes totalmente coordinados ($w_{soc} = 0$), que intercambien las tareas con otros que las puedan ejecutar en paralelo y, por tanto, reduzcan el tiempo de espera de los recursos. Sin embargo, este comportamiento extremadamente coordinado es más propio de los robots (que están continuamente trabajando si tienen la oportunidad) que de los humanos (que manifiestan sus relaciones sociales, por ejemplo, mediante las conversaciones con otros amigos). Por consiguiente, el factor de sociabilidad, $w_{soc}$, puede ser usado para balancear coordinación y sociabilidad y garantizar, así pues, comportamientos más humanos que robóticos. En este contexto, el valor de $Throughput$ para el tipo de animaciones que consideramos interesantes se encuentra entre el comportamiento totalmente coordinado ($w_{soc} = 0$) y el comportamiento totalmente social ($w_{soc} = 1$), como destaca la zona sombreada de la figura 6.15a.

$Throughput$ es un estimador del grado de coordinación alcanzado por los personajes 3D. Sin embargo, a pesar de que la eficiencia es un requerimiento básico a la hora de simular grupos de actores sintéticos, éste no es el único criterio que conviene evaluar cuando se pretende crear simulaciones verosímiles. De acuerdo con esto, hemos definido un estimador que tiene en cuenta el tiempo que el diseñador de la simulación desea que

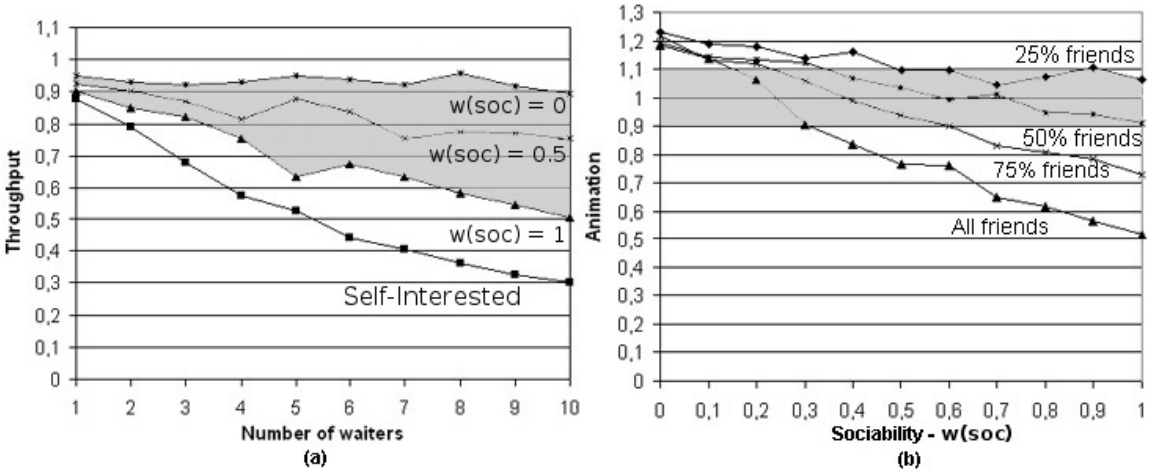


*Figura 6.15: Estimadores de coordinación y sociabilidad para un grupo de camareros: (a) $Throughput$ y (b) $Animation$.*

los personajes dediquen a la animación de interacciones sociales. La ecuación 6.13 contiene la definición de este estimador, que llamamos $Animation$. En esta ecuación, $T_{social}$ representa el tiempo dedicado a la animación de conversaciones y acuerdos sociales entre los personajes. En nuestro bar virtual, hemos definido este tiempo como el 35 % del tiempo ideal de simulación ($T^*_{sim}$).

$$Animation = \frac{T^*_{sim} + T_{social}}{T_{sim}} \tag{6.13}$$

La figura 6.15b muestra los valores de animación obtenidos para el grupo de camareros cuando se consideran 4 configuraciones de amistad: todos los camareros son amigos, el 75 % de los camareros son amigos, la mitad de los camareros son amigos y sólo el 25 % de los camareros son amigos. Como ya habíamos comentado antes, los valores bajos de sociabilidad producen simulaciones con un bajo nivel de verosimilitud, cosa que se refleja en la función de animación, que se encuentra por encima del valor de referencia ($Animation = 1$). Asimismo, valores muy altos de sociabilidad también conducirán a simulaciones de baja credibilidad, especialmente cuando el grado de amistad es algo. En estas configuraciones, el número de conversaciones sociales animadas es demasiado grande para ser realista y el valor de animación se encuentra lejos del valor de referencia. El estimador de animación puede ser utilizado para extraer el rango de valores adecuado para el factor de sociabilidad ($w_{soc}$), dependiendo de la situación que se esté simulando.

Por ejemplo, en nuestro bar virtual hemos considerado como simulaciones de buena calidad aquellas que caen dentro de la franja del $\pm 10\,\%$ del valor de referencia (ver la zona sombreada de la figura 6.15b). De acuerdo con esto, cuando todos los camareros son amigos, las buenas animaciones emergen para $w_{soc} \in [0{,}1, 0{,}3]$ (ver la línia *All friends* de la figura 6.15b).

Por último, la tabla 6.8 compara la cantidad de tiempo dedicado a ejecutar cada tipo de tarea por cada uno de los 10 camareros con los valores extremos de factor de sociabilidad: coordinación máxima ($w_{soc} = 0$) y sociabilidad máxima ($w_{soc} = 1$). En el caso de la coordinación máxima, los valores irregulares de las columnas $T_{use}$ y $T_{give}$ de la parte izquierda de la tabla demuestran cómo algunos camareros se han especializado en la ejecución de ciertas tareas. Por ejemplo, los camareros 2, 5, 9 y 10 destinan la mayor parte de su tiempo a entregar productos a los clientes, mientras que los camareros 3 y 7 se dedican principalmente a usar los recursos del bar (p. ej. la máquina de hacer zumos). A pesar de que la especialización es un efecto deseable en muchos sistemas multiagente, las sociedades humanas, a menudo de tipo igualitario, suelen balancear la carga de trabajo asignada a cada agente. Según muestra la parte derecha de la tabla, los camareros completamente sociales equilibran los tiempos de uso de los recursos del entorno y de entrega de productos a los clientes (ver las columnas $T_{use}$ y $T_{give}$). Además, si el modelo de actitud personal es la reciprocidad, la diferencia de favores intercambiados tiene a cero (comparar las columnas llamadas $Balance$). Un efecto colateral de este equilibrio es el incremento en los tiempos de espera, ya que los camareros sociales a veces prefieren esperar a que otro camarero libere el recurso que pasar la tarea (comparar las columnas $T_{wait}$). Como consecuencia, surge un nuevo porcentaje de tiempo ($T_{chat}$) en el que los camareros pueden animar interacciones sociales como por ejemplo diálogos entre amigos..

En conclusión, a pesar de que la coordinación es un comportamiento muy deseado en contextos multiagente, generalmente, no es el único camino para expresar la sociabilidad. Al contrario, el **balanceo** de la coordinación con comportamientos sociales no orientados a la eficiencia (p. ej. que respondan a relaciones sociales de índole diversa como por ejemplo la amistad, el rol o la clase social), aporta esa **variabilidad** que distingue el razonamiento humano del robótico y que, por tanto, mejorará la verosimilitud de los actores sintéticos.

| | $w_{soc} = 0$ | | | | | $w_{soc} = 1$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Agente | $T_{wait}$ | $T_{chat}$ | $T_{use}$ | $T_{give}$ | $Balance$ | $T_{wait}$ | $T_{chat}$ | $T_{use}$ | $T_{give}$ | $Balance$ |
| 1 | 0 | 0 | 32 | 19 | -6 | 16 | 36 | 69 | 34 | -2 |
| 2 | 3 | 0 | 4 | 26 | -3 | 18 | 62 | 58 | 24 | -2 |
| 3 | 14 | 0 | 52 | 1 | 28 | 41 | 66 | 45 | 16 | 0 |
| 4 | 3 | 0 | 16 | 28 | -3 | 48 | 61 | 60 | 27 | 3 |
| 5 | 0 | 0 | 7 | 30 | -16 | 34 | 68 | 58 | 12 | -1 |
| 6 | 3 | 0 | 37 | 17 | -1 | 48 | 74 | 64 | 14 | -2 |
| 7 | 0 | 0 | 67 | 4 | 21 | 18 | 66 | 48 | 24 | 1 |
| 8 | 0 | 0 | 45 | 17 | 1 | 33 | 76 | 45 | 24 | 4 |
| 9 | 7 | 0 | 5 | 23 | -11 | 46 | 58 | 36 | 21 | 0 |
| 10 | 1 | 0 | 6 | 41 | -10 | 27 | 69 | 56 | 20 | -1 |

*Tabla 6.8: Distribución temporal (en segundos) de las tareas ejecutadas per 10 camareros.*

### 6.4.7. Conclusiones MADeM

La toma de decisiones de tipo MADeM proporciona diversas características interesantes a la hora de incorporar razonamiento social en la animación de personajes 3D. En primer lugar, MADeM permite que los agentes consideren diferentes puntos de vista para evaluar sus acciones (**multimodalidad**). El uso de las funciones de utilidad para expresar estos puntos de vista es una aproximación modular que permite definir diferentes criterios de manera sencilla. Por ejemplo, hemos desarrollado grupos de actores sintéticos que toman sus decisiones de acuerdo a criterios como la eficiencia, la sociabilidad, el cansancio o la pereza. En este contexto, el sopesado de las diferentes funciones de utilidad permite definir **personalidades** diversas que atienden más de un punto de vista que a otro (p. ej. actores coordinados, habladores, calculadores, perezosos, etc.). En segundo lugar, MADeM recoge las preferencias del resto de miembros de la sociedad mediante la subasta de las posibles soluciones a un problema de decisión. En este contexto, el uso de distintas funciones de bienestar social servirá para simular diferentes **modelos de sociedad**. En el ejemplo del bar virtual, la sociedad simulada por los camareros correspondía con el bienestar elitista mientras que los clientes seguían un bienestar igualitario. En tercer lugar, por medio de la aplicación de un diferencial a las preferencias recibidas, MADeM pro-

porciona a los personajes la posibilidad de modelar **actitudes personales** hacia el resto de individuos de la sociedad artificial. Estas actitudes personales pueden variar desde la indiferencia, pasando por la reciprocidad, hasta el altruismo y el egoísmo.

En conclusión, MADeM ha demostrado su capacidad para reproducir comportamientos sociales como por ejemplo: la coordinación, la especialización, el intercambio de favores, el fomento de encuentros sociales, el balanceo de tareas ejecutadas, etc. En consecuencia, MADeM proporciona un conjunto de herramientas adecuado para dotar a los actores sintéticos de un mecanismo de **toma de decisiones socialmente aceptables**.

CAPÍTULO 7

# CONCLUSIONES, APORTACIONES Y TRABAJOS FUTUROS

## 7.1. CONCLUSIONES GENERALES Y APORTACIONES

La investigación llevada a cabo en esta tesis, para conseguir la integración de habilidades sociales en la animación comportamental de los actores sintéticos, ha generado los resultados y aportaciones que resumimos en los puntos siguientes:

- La simulación de entornos virtuales habitados se enfrenta a la resolución de problemas complejos, provenientes del campo de la animación gráfica y de la inteligencia artificial. Para poder abarcar este conjunto de problemas, hemos propuesto un **marco de simulación multiagente** que separa el modelo semántico, el modelo comportamental y el modelo geométrico. De acuerdo con esto, el sistema de simulación está formado por: a) un **entorno virtual semántico**, que crea y mantiene el estado del mundo; b) un conjunto de **agentes socialmente inteligentes**, que genera el comportamiento de los personajes; y c) un **motor gráfico 3D**, a cargo de la visualización.

- La evolución de la animación comportamental de los personajes 3D ha ido ligada al nivel de representación del estado del mundo. En este contexto, proponemos un **modelo general de entorno virtual semántico basado en ontologías** con el que se pueden conseguir tres objetivos básicos para la animación de actores sintéticos: a) la mejora de la sensorización de escenas complejas; b) la definición de operativas generales que los personajes puedan reutilizar en situaciones diversas; y c) la definición de las relaciones sociales y de organización establecidas entre los miembros de una sociedad artificial.

- El desarrollo de personajes 3D conlleva la resolución de una multitud de proble-

mas complejos, como por ejemplo: la percepción, el control motor, la navegación, la comunicación, la selección de acciones, etc. En consecuencia, proponemos una **arquitectura de agente socialmente inteligente** formada por un conjunto de módulos a cargo de estos aspectos. El grado de verosimilitud comportamental de estos agentes dependerá, fundamentalmente, de sus capacidades racionales y sociales. Por tanto, las aportaciones principales de esta tesis se han centrado en la inclusión de criterios sociales en dos formalismos bien conocidos para la selección dinámica de acciones en 3DIVA: la planificación de tareas de tipo STRIPS y el modelo BDI.

- En primer lugar, hemos presentado una **técnica para conseguir comportamientos de tipo colaborativo en aquellos personajes 3D que utilizan planificadores heurísticos como el mecanismo de selección dinámica de acciones**. Esta técnica proporciona a los actores la habilidad de adaptarse y actuar de forma eficiente en los entornos habitados por diversos caracteres autónomos (p. ej. juegos, *storytelling*, simulación civil, etc.).En estos contextos, hemos usado la comunicación directa entre los personajes para construir la representación en memoria de los agentes externos. Asimismo, hemos incorporado una fase de pre-planificación que favorece la **coordinación** con las acciones de otros agentes y hemos desarrollado un mecanismo de sopesado de la función heurística que permite a los agentes **cooperar** en la consecución de objetivos externos.

- En segundo lugar, hemos presentado **MADeM (*Multi-modal Agent Decision Making*), un proceso de toma de decisiones socialmente aceptables para actores sintéticos**. Por un lado, MADeM permite evaluar diversos puntos de vista o preferencias personales (p. ej. la eficiencia, la sociabilidad, el cansancio, etc.). Por otro lado, MADeM es capaz de incorporar las preferencias de otros agentes externos mediante una técnica basada en el uso de subastas de tareas. En este contexto, MADeM permite definir diversos órdenes sociales en una sociedad (p. ej. el elitista, el utilitario, etc.) así como actitudes personales de sus miembros (p. ej. el egoísmo, el altruismo, etc.). Finalmente, MADeM ha demostrado su capacidad para reproducir habilidades sociales en actores de tipo BDI como por ejemplo: la coordinación, la especialización, el intercambio de favores, el fomento de encuentros sociales, el balanceo de tareas ejecutadas, etc.

En resumen, hemos diseñado, implementado y probado un marco de simulación de entornos virtuales habitados por actores socialmente inteligentes. Este sistema utiliza las ontologías como una solución genérica para la representación semántica del estado del

mundo. Por lo que respecta a la integración de habilidades sociales, hemos probado dos aproximaciones, una basada en la colaboración con planificadores heurísticos y otra en un proceso de toma de decisiones socialmente aceptables para agentes de tipo BDI. Por tanto, podemos considerar alcanzados los objetivos propuestos al inicio de este documento; a pesar de que han quedado abiertas una serie de líneas de investigación, que resumimos en el punto siguiente.

## 7.2. TRABAJOS FUTUROS

Muchas son las líneas de trabajo que quedan abiertas como resultado de esta tesis. A continuación resumimos aquellas que consideramos más interesantes:

- **Incorporación de sistemas de normas a la dinámica de las relaciones que ligan a los miembros de una sociedad artificial**. Los sistemas de normas pueden ser aplicados, por ejemplo, a la definición del contrato social, que establece el conjunto de reglas que gobiernan las agrupaciones de personajes 3D. De acuerdo con este contrato, la actuación de un actor sintético puede implicar la activación de una serie de reglas que modifican las actitudes del resto de miembros hacia él (p. ej. si un personaje perjudica al grupo con su actuación, el resto de actores podrían recriminarle verbalmente o incluso modificar las relaciones sociales que tiene con él). Esta evolución social favorecerá la variabilidad en el comportamiento de los actores sintéticos y aumentará, pues, su grado de verosimilitud.

- Los mecanismos de toma de decisiones sociales presentados en esta tesis responden a situaciones en que los objetivos son compatibles y los actores sintéticos son benevolentes. La compatibilidad y la benevolencia se pueden ver como el resultado de una negociación previa entre personajes con objetivos incompatibles. Por tanto, conviene estudiar la **aplicación de mecanismos de negociación como el regateo o las votaciones**. La negociación es un aspecto interesante en los entornos 3D multiusuario de última generación (p. ej. SecondLife), en los que la simulación no está bajo el control de un único diseñador y los objetivos incompatibles aparecen de manera natural.

- A lo largo de esta tesis, hemos presentado la planificación heurística y el modelo BDI como formalismos adecuados para la toma de decisiones en personajes 3D. Consideramos, además, que ambas aproximaciones pueden utilizarse de forma

complementaria. Por ejemplo, se puede pensar en combinar la planificación para la secuenciación de tareas en el bajo nivel y el modelo BDI para el mantenimiento de los objetivos de más alto nivel. Por consiguiente, la **integración de la colaboración con planificadores heurísticos y MADeM** permitirá juntar los comportamientos sociales orientados a la eficiencia (p. ej. coordinación, cooperación, etc.) con los comportamientos orientados a garantizar el bienestar social de un grupo de actores sintéticos (p. ej. elitismo, reciprocidad, altruismo, egoísmo, etc.).

- En el capítulo 6, hemos demostrado el efecto de las técnicas sociales desarrolladas de forma cuantitativa. Sin embargo, una **validación cualitativa de la sociabilidad apreciada por los usuarios humanos** puede servir para extraer datos importantes. En este sentido, se pueden realizar pruebas clásicas en las que un conjunto de individuos visualiza una selección de vídeos y responde a las preguntas de una encuesta; que evalúa las cualidades sociales de los actores sintéticos desde el punto de vista de un observador externo. Otra posibilidad más sofisticada consiste en realizar pruebas de presencia de última generación, en las que el usuario se sumerge en la escena mediante algún dispositivo inmersivo (p. ej. gafas estereoscópicas). De esta manera, el usuario podrá evaluar la experiencia social resultante de la interacción con los humanoides virtuales.

## 7.3. PUBLICACIONES DERIVADAS DE ESTA TESIS

La integración de habilidades sociales en la toma de decisiones de los actores sintéticos ha dado lugar a las siguientes publicaciones, ordenadas en relación con los capítulos anteriores:

- **Semantic Virtual Environments for interactive planning agents**. Fran Grimaldo, Fernando Barber, Miguel Lozano, Juan M. Orduña. *International Digital Games Conference (iDiG'06)*. Portalegre (Portugal), septiembre de 2006.

- **An ontology-based approach for IVE+VA**. Fran Grimaldo, Fernando Barber, Miguel Lozano. *International Conference on Intelligent Virtual Environments and Virtual Agents (IVEVA'06)*. Aguascalientes (Mexico), octubre de 2006.

- **A multiagent framework to animate socially intelligent agents**. Francisco Grimaldo, Miguel Lozano, Fernando Barber. *The 2nd Hybrid Artificial Intelligence*

*Systems Workshop (CAEPIA-TTIA'2007)*. Springer Advances in Soft Computing, Salamanca (Spain), noviembre de 2007.

- **Integrating social skills in task-oriented 3D IVA**. Fran Grimaldo, Miguel Lozano, Fernando Barber, Juan M. Orduña. *The 5th International Working Conference on Intelligent Virtual Agents (IVA'05)*. Springer-Verlag LNAI, Kos (Greece), septiembre de 2005.

- **Coordination and sociability for intelligent virtual agents**. Francisco Grimaldo, Miguel Lozano, Fernando Barber. *The MALLOW Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN'07)*. Springer Verlag LNAI, Durham (UK), septiembre de 2007.

- **Balancing social and task oriented behaviors for animation agents**. Francisco Grimaldo, Miguel Lozano, Fernando Barber. *III Taller en Desarrollo de Sistemas Multiagente (CEDI'2007)*. Zaragoza (Spain), septiembre de 2007.

- **Social animation in complex environments**. Francisco Grimaldo, Miguel Lozano, Fernando Barber. *The 7th International Conference on Intelligent Virtual Agents (IVA'07)*. Springer-Verlag LNAI, Paris (France), septiembre de 2007.

- **Animating groups of socially intelligent agents**. Francisco Grimaldo, Miguel Lozano, Fernando Barber, Guillermo Vigueras. *Proceedings of the Cyberworlds 2007*. IEEE Computer Society Press. Hannover (Germany), octubre de 2007.

- **MADeM: a multi-modal decision making for social MAS**. Francisco Grimaldo, Miguel Lozano, Fernando Barber. *The 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*. ACM Press. Estoril (Portugal), mayo de 2008.

- **Simulating socially intelligent agents in Semantic Virtual Environments**. Francisco Grimaldo, Miguel Lozano, Fernando Barber, Guillermo Vigueras. *The Knowledge Engineering Review*. ISSN: 0269-8889. Cambridge University Press, diciembre de 2008.

# BIBLIOGRAFÍA


[pdd, 2008] (2008). Writing planning domains and problems in PDDL. Available at http://www.ida.liu.se/ TDDA13/labbar/planning/2003/writing.html. 4.4.1

[Activeworlds Corporation, 2008] Activeworlds Corporation (2008). Active worlds. Available at http://www.activeworlds.com/. 2.3.1

[Agotnes et al., 2007] Agotnes, T., der Hoek, W. V., Rodrguez-Aguilar, J. A., Sierra, C., and Wooldridge, M. (2007). On the logic of normative systems. In *Proc. of IJCAI07: International Joint Conference on AI*, pages 1175–1180. AAAI Press. 3.4

[ANA MAS, 2005] ANA MAS (2005). *Agentes software y sistemas multiagente: Conceptos, arquitecturas y aplicaciones*. Pearson Prentice Hall. 5.2.2

[Anastassakis et al., 2001] Anastassakis, G., Panayiotopoulos, T., and Ritchings, T. (2001). Virtual agent societies with the mVITAL intelligent agent system. In *Proc. of the 3rd International Workshop on Intelligent Virtual Agents*. 3.2

[Avradinis et al., 2005] Avradinis, N., Panayiotopoulos, T., and Aylett, R. (2005). *Intelligent techniques for planning*, chapter Continous Planning for Virtual Environments, pages 162–193. Idea Grup Publishing. 2.4.1, 5.3.2

[Aylett and Cavazza, 2001] Aylett, R. and Cavazza, M. (2001). Intelligent virtual environments: A state of the art report. In *Proc. of Eurographics*. 2.1

[Badler, 1997] Badler, N. (1997). Virtual humans for animating ergonomics and simulation. In *Proc. of the Workshop on non-rigid and articulation motion*, Puerto Rico. IEEE. 2.3.1

[Badler and Allbeck, 2000] Badler, N. and Allbeck, J. (2000). Towards behavioral consistency in animated agents. In *Proc. of Deformable avatars*. 2.1

[Badler et al., 2000] Badler, N., Bindiganavale, R., Bourne, J., Palmer, M., Shi, J., and Schuler, W. (2000). *A parametrized action representation for virtual human agents*, pages 256–284. MIT press. 2.2.2, 4.4.1

[Badler et al., 1993] Badler, N., Philips, C., and Webber, B. (1993). *Simulating Humans: Computer Graphics, Animation and Control.* Oxford University Press. 2.1

[Bates, 1992] Bates, J. (1992). The nature of character in interactive worlds and the oz project. Technical Report CMU-CS-92-200, School of Computer Science, Carnegie Mellon Universiti., Pittsburg, PA. 2.4.1

[Benford et al., 1997] Benford, S., Greenhalgh, C., and Lloyd, D. (1997). Crowded collaborative virtual environments. In *Proc. of SIGCHI: Conference on human factors in computing systems*. 3.1, 4.3.2

[Bergenty and Ricci, 2002] Bergenty, F. and Ricci, A. (2002). Three approaches to the coordination of multi-agent systems. In *Proc. of Symposium on Applied Computing*, pages 367–372. ACM Press. 3.3.2

[Bickmore and Cassell, 2001] Bickmore, T. and Cassell, J. (2001). Relational agents: A model and implementation of building user trust. In *Proc. of CHI'2001: Conference on Human Factors in Computing Systems*, London. ACM. 3.1

[Bioware, 2003] Bioware (2003). Starwars: Knights of the old republic. Available at http://lucasarts.com/products/swkotor. 3.1

[Blum and Furst, 1997] Blum, A. and Furst, M. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, 90:281–300. 2.4.1, 5.3.3

[Bonet and Geffner, 2001] Bonet, B. and Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, 129(1–2):5–33. 2.4.1, 2.4.1, 5.3.3, 5.3.3

[Bonet et al., 1997] Bonet, B., Loerincs, G., and Geffner, H. (1997). A fast and robust action selection mechanism for planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*. AAAI Press. 5.3

[Bordini et al., 2005] Bordini, R. H., da Rocha, A. C., Hübner, J. F., Moreira, A. F., Okuyama, F. Y., and Vieira, R. (2005). MAS-SOC: a social simulation platform based

on agent-oriented programming. *Journal of Artificial Societies and Social Simulation*, 8(3). 5.4

[Bordini and Hübner, 2007] Bordini, R. H. and Hübner, J. F. (2007). Jason. Available at http://jason.sourceforge.net/. 2.4.2, 3.2, 4.4.1, 6.4

[Bouron and Collinot, 1992] Bouron, T. and Collinot, A. (1992). SAM: a model to design computational social agents. In *Proc. of ECAI'92: European Conference on AI*, pages 239–243. 5.3.1

[Boutilier and Brafman, 2001] Boutilier, C. and Brafman, R. I. (2001). Partial order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14:105–136. 3.3.2

[Brainov and Sandholm, 1999] Brainov, S. and Sandholm, T. (1999). Power, dependence and stability in multiagent plans. In *Proc. of AAAI '99/IAAI '99: 6th national conference on Artificial intelligence and 11th Innovative Applications of Artificial Intelligence conference*, pages 11–16, Menlo Park, CA, USA. American Association for Artificial Intelligence. 3.4.3

[Bratman, 1987] Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Hardvard University Press. 2.4.2

[Brenner, 2003] Brenner, M. (2003). A multi-agent planning language. In *Proc. of ICAPS'03 Workshop on PDDL*. 3.3.2

[Caicedo and Thalmann, 2000] Caicedo, A. and Thalmann, D. (2000). Virtual humanoids: Let them be autonomous without losing control. In *Proc. of 3IA2000: 4th Conf. Computer Graphics and Artificial Intelligence*. (document), 2.4.2, 2.14

[Calderón et al., 2003] Calderón, C., Cavazza, M., and Díaz, D. (2003). Interactive problem solving in an intelligent virtual environment. In *Proc. of IUI'03: International conference on Intelligent User Interfaces*, pages 319–319, New York, USA. ACM Press. 2.1

[Cassell et al., 1994] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., and Stone, M. (1994). Animated conversation: Rule-based generation of facial expression, gesture and spoken intonational for multiple conversational agents. In *Proc. of SIGGRAPH'94*. 3.1

[Castelfranchi, 1998] Castelfranchi, C. (1998). Modeling social action for ai agents. *Artificial Intelligence*, 103:157–182. 3.4

[Castelfranchi and Falcone, 1998] Castelfranchi, C. and Falcone, R. (1998). Principles of trust in MAS: cognitive anatomy, social importance and quantification. In *Proc. of ICMAS'98*, pages 72–79, Paris. 3.4.2

[Castelfranchi et al., 1992] Castelfranchi, C., Miceli, M., and Cesta, A. (1992). Dependence relations among autonomous agents. In *Proc. of MAAMAW'92*, pages 215–227, Amsterdam. Elsevier Science Publishers. 3.4.1

[Cavazza et al., 2002] Cavazza, M., Charles, F., and Mead, S. J. (2002). Character based interactive storytelling. *IEEE Intelligent systems*, 17(4):17–24. (document), 2.4.1, 2.10, 3.1, 4.4, 5.3.1

[Chang et al., 2005] Chang, P. H.-M., Chien, Y.-H., Kao, E. C.-C., and Soo, V.-W. (2005). A knowledge-based scenario framework to support intelligent planning characters. In LNCS, S., editor, *Proc. of the 5th International Workshop of Intelligent Virtual Agents (IVA05)*, pages 134–145. (document), 2.2.2, 2.2

[Chevaleyre et al., 2006] Chevaleyre, Y., Dunne, P. E., Endriss, U., Lang, J., Lemaitre, M., Maudet, N., Padget, J., Phelps, S., Rodriguez-Aguilar, J. A., and Sousa, P. (2006). Issues in multiagent resource allocation. *Informatica*, 30:3–31. 3.4.3, 3.4.3, 5.4.1, 5.4.3

[Chevaleyre et al., 2004] Chevaleyre, Y., Endriss, U., Estivie, S., and Maudet, N. (2004). Welfare engineering in practice: On the variety of multiagent resource allocation problems. In *ESAW*, pages 335–347. 3

[Ciger, 2005] Ciger, J. (2005). *Collaboration with agents in VR environments*. PhD thesis, École Polytechnique Fédérale de Lausanne. (document), 2.4.1, 3.1, 3.1

[Cohen and Levesque, 1991] Cohen, P. and Levesque, H. (1991). Teamwork. *Nous, Special Issue on Cognitive Science and AI*. 3.3.3, 5.3.1

[Conte, 1999] Conte, R. (1999). Artifcial social intelligence: a necessity for agent systems' developments. *Knowledge Engineering Review*, 14:109–118. 3.3.3, 3.4.1

[Conte and Castelfranchi, 1995] Conte, R. and Castelfranchi, C. (1995). *Cognitive and Social Action*. UCL Press, London. 3.3.3

[Costa de Paiva et al., 2005] Costa de Paiva, D., Vieira, R., and Musse, S. R. (2005). Ontology-based crowd simulation for normal life situations. In *Proceedings of Computer Graphics*. IEEE. (document), 2.2.1, 2.1

[Cox and Durfee, 2005] Cox, J. S. and Durfee, E. H. (2005). An efficient algorithm for multiagent plan coordination. In *Procs. of AAMAS'05: Autonomous Agents and Multiagent Systems*, pages 828–827. 3.3.2

[de Weerdt, 2003] de Weerdt, M. M. (2003). Plan coordination. In *Proceedings of the Doctorial Consortium of the International Conferenence on AI Planning and Scheduling*, pages 142–145. 3.3.2

[Decker, 1998] Decker, K. S. (1998). *Simulating Organizations: Computational Models of Institutions and Groups*, chapter Task environment centered simulation, pages 105–128. AAAI Press / MIT Press, Menlo Park. 2.2.3

[Decker and Lesser, 1997] Decker, K. S. and Lesser, V. R. (1997). *Readings in Agents*, chapter Designing a family of coordination algorithms. Elsevier. 3.3.2

[Delgado-Mata and Aylett, 2004] Delgado-Mata, C. and Aylett, R. (2004). Emotion and action selection: Regulating the collective behaviour of agents in virtual environments. In *AAMAS*, pages 1304–1305. 3.1

[D'Inverno and Luck, 2004] D'Inverno, M. and Luck, M. (2004). *Understanding Agent Systems*. Springer-Verlag, 2nd edition. 5.2

[Distributed Systems and Information Systems Group - University of Hamburg, 2008] Distributed Systems and Information Systems Group - University of Hamburg (2008). JADEX: BDI agent system. Available at http://vsis-www.informatik.uni-hamburg.de/projects/jadex/. 2.4.2

[Doyle, 2002] Doyle, P. (2002). Believability through context using "knowledge in the world"to create intelligent characters. In Press, A., editor, *First International Joint Conference on Autonomous Agents and Multi-Agent Systems(AAMAS02)*, pages 342–349. 2.2.2

[Durfee, 1999] Durfee, E. H. (1999). *A modern approach to distributed artificial intelligence: Distributed Problem Solving and Planning*, chapter 3. The MIT Press. 3.3.2

[Elliott et al., 1999] Elliott, C., Rickel, J., and Lester, J. (1999). Lifelike pedagogical agents and affective computing: An exploratory synthesis. *Lecture Notes in Computer Science*, 1600. (document), 2.1, 2.4.2, 2.13

[Ephrati and Rosenschein, 1994] Ephrati, E. and Rosenschein, J. S. (1994). Divide and conquer in multi–agent planning. In *Procs. of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 375–380, Menlo Park, CA. AAAI Press. 3.3.2

[Epic games, 2008] Epic games (2008). Unreal tournament. Available at http://www.unreal.com. 2.2, 4.2, 6.3

[Falcone et al., 2004] Falcone, R., Pezzulo, G., Castelfranchi, C., and Calvi, G. (2004). Why a cognitive trustier performs better: Simulating trust-based contract nets. In *Proc. of AAMAS'04: Autonomous Agents and Multi-Agent Systems*, pages 1392–1393. ACM. (document), 3.4.2, 3.7

[Farenc et al., 1999] Farenc, N., Boulic, R., and Thalmann, D. (1999). An informed environment dedicated to the simulation of virtual humans in urban context. In Brunet, P. and Scopigno, R., editors, *Proc. of EUROGRAPHICS"99*, volume 18(3), pages 309–318. The Eurographics Association and Blackwell Publishers. 2.2.1

[Ferber, 1999] Ferber, J. (1999). *Multi-Agent Systems: An introduction to Distributed Artificial Intelligence*. Addison-Wesley Publishing Company, Longman. (document), 3.2, 3.2, 3.4.3, 5.2.1, 5.1, 6.3.1

[Ferber and Gutknecht, 1998] Ferber, J. and Gutknecht, O. (1998). A meta-model for the analysis and design of organizations in multi-agents systems. In *Proc. of the 3rd International Conference on Multi-Agent Systems (ICMAS'98)*, pages 128–135. IEEE Press. 2.2.3

[Fikes and Nilsson, 1971] Fikes, R. and Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208. 2.4.1

[Finin et al., 1994] Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). KQML as an agent communication language. In *Proc. of CIKM'94*. ACM Press. 3.3.2, 4.2

[FIPA (Foundation for Intelligent Physical Agents), 2008] FIPA (Foundation for Intelligent Physical Agents) (2008). FIPA ACL specifications. Available at http://www.fipa.org. 3.3.2, 4.2, 4.4.2

[Fishburn, 1970] Fishburn, P. (1970). *Utility Theory for Decision Making*. Robert E. Krieger Publishing Co., Huntington, NY. 5.4.3

[Fox et al., 1998] Fox, M. S., Barbuceanu, M., Gruninger, M., and Lon, J. (1998). *Simulating Organizations: Computational Models of Institutions and Groups*, chapter An organizational ontology for enterprise modeling., pages 131–152. AAAI Press / MIT Press, Menlo Park. 2.2.3

[Funge et al., 1999] Funge, J., Tu, X., and Terzopoulos, D. (1999). Cognitive modeling: knowledge, reasoning and planning for intelligent characters. In *Proc. of SIGGRAPH'99*, pages 29–38. ACM. (document), 2.3, 2.4, 2.4.1, 2.11

[Gardner, 1970] Gardner, M. (1970). MATHEMATICAL GAMES: The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223:120–123. 3.2

[Geib et al., 1994] Geib, C., Levison, L., and Moore, M. (1994). Sodajack: An architecture for agents that search and manipulate objects. Technical Report MS-CIS-94-16 / LINC LAB265, Dpt. Computer and Information Science, University of Pennsylvania. (document), 2.4.1, 2.9

[Georgeff, 1983] Georgeff, M. P. (1983). Communication and interaction in multi-agent planning. In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pages 125–129, Menlo Park, CA. AAAI Press. 3.3.2

[Giampapa and Sycara, 2002] Giampapa, J. A. and Sycara, K. (2002). Team-oriented agent coordination in the RETSINA multi-agent system. Tech. report CMU-RI-TR-02-34, Robotics Institute-Carnegie Mellon University. 3.3.3, 5.3.1

[Gibson, 1984] Gibson, W. (1984). *Neuromancer*. Ace Books. 1.1

[Gil, 2005] Gil, Y. (2005). Description logics and planning. *AI Magazine*, 26(2):73–84. 2.2.2

[Giménez-Lugo et al., 2005] Giménez-Lugo, G., Sichman, J., and Hübner, J. (2005). Addressing the social components of knowledge to foster communitary exchanges. *International Journal on Web Based Communities*, 1(2):176–194. 2.2.3

[Grosz et al., 1999] Grosz, B., Hunsberger, L., and Kraus, S. (1999). Planning and acting together. *AI Magazine*, 20(4):23–34. 3.3.3, 5.3.1

[Gutierrez, 2006] Gutierrez, M. (2006). *Semantic Virtual Environments*. PhD thesis, École Polytechnique Fédérale de Lausanne. 2.2.1

[Haslum and Geffner, 2000] Haslum, P. and Geffner, H. (2000). Addmissible heuristics for optimal planning. *Artificial Intelligence Planning Systems*, pages 140–149. 5.3.3

[Hayes-Roth, 2008] Hayes-Roth, B. (2008). Barbara hayes-roth website. Available at http://www.ksl.stanford.edu/people/bhr/. 3.1

[Helbing et al., 2005] Helbing, D., Buzna, L., Johansson, A., and Werner, T. (2005). Self-organized pedestrian crowd dynamics. *Transportation Science*, 39(1):1–24. 3.1

[Hexmoor, 2001] Hexmoor, H. (2001). From inter-agents to groups. In *Proc. of ISAI'01: International Symposium on Artificial Intelligence*. 3.3.3

[Hoffmann and Nebel, 2001] Hoffmann, J. and Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Artificial Intelligence Research*, 14:253–302. 2.4.1, 5.3.3, 5.3.3

[Hogg and Jennings, 2001] Hogg, L. M. and Jennings, N. (2001). Socially intelligent reasoning for autonomous agents. *IEEE Transactions on System Man and Cybernetics*, 31(5):381–393. 3.1

[Iglesias and Luengo, 2004] Iglesias, A. and Luengo, F. (2004). Intelligent agents in virtual worlds. In *Proc. of CW04: International Conference on Cyberworlds*, pages 62–69. IEEE Computer Society. 2.3

[Ioeger and Johnson, 2001] Ioeger, T. and Johnson, J. (2001). A formal model of responsibilities in agent-based teamwork. *Applied Artificial Intelligence*, 15(10):875–916. 3.3.3

[JAVA3D, 2008] JAVA3D (2008). JAVA 3D API. Available at http://java.sun.com/products/java-media/3D/. 2.2

[Jena, 2008] Jena (2008). A semantic web framework for java. Available at http://jena.sourceforge.net/. 4.3

[Jennings and Campos, 1997] Jennings, N. and Campos, J. (1997). Towards a social level characterisation of socially responsible agents. In *Proc. of Software Engineering*, volume 144, pages 11–25. IEEE. 3.4

[Joint US/EU ad hoc Agent Markup Language Committee, 2008] Joint US/EU ad hoc Agent Markup Language Committee (2008). DAM+OIL 2001 release. Available at http://www.daml.org/2001/03/daml+oil-index.html. 2.2.1, 4.3

[Kallmann and Thalmann, 1999] Kallmann, M. and Thalmann, D. (1999). Direct 3D interaction with smart objects. In Press, A., editor, *VRST '99: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 124–130. 2.2.1

[Kambhampati et al., 1991] Kambhampati, S., Tanembaum, M., and Lee, S. (1991). Combining specialized reasoners and general purpose planners. In *Proceedings of the 9th National Conference on Artificial Intelligence*. 3.3.2

[Kao et al., 2005] Kao, E. C.-C., H-M., C. P., Chien, Y.-H., and Soo, V.-W. (2005). Using ontology to establish social context and support social reasoning. In *Proc. of IVA'05: International Conference on Intelligent Virtual Agents*. Springer-Verlag LNAI. (document), 2.2.3, 2.3

[Kim, 2003] Kim, I.-C. (2003). KGBot: A BDI agent deploying within a complex 3D virtual environment. In T. Rist, R. Aylett, D. B. and Rickel, J., editors, *Intelligent Virtual Agents*, pages 192–196. Springer-Verlag LNAI. 2.4.2

[Kuffner, 1999] Kuffner, J. (1999). *Autonomous Agents for Real-time Animation*. PhD thesis, Stanford University, Stanford, CA. 4.4

[Kuffner and Latombe, 1999] Kuffner, J. and Latombe, J. (1999). Perception-based navigation for animated characters in real-time virtual environments. *The Visual Computer: Real-Time Virtual Worlds*, 3 (4). 4.4

[Laird et al., 1987] Laird, J., Newel, A., and Rosenbloom, P. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, pages 1–64. 2.3.2, 2.4.2

[Laird and Duchi, 2000] Laird, J. E. and Duchi, J. C. (2000). Creating human-like synthetic characters with multiple skill levels: A case study using the soar quakebot. *AAAI Fall Symposium Series: Simulating Human Agents*. 2.1, 2.4.2

[Lang, 2005] Lang, J. (2005). Some representation and computational issues in social choice. In *Proc. of . ECSQARU'05*, pages 15–26. 3.4.3

[Levison, 1996] Levison, L. (1996). *Connecting planning and acting via object-specific reasoning*. PhD thesis, University of Pennsylvania, PA, USA. 2.2.2, 4.4.1

[Lewis and Jacobson, 2002] Lewis, M. and Jacobson, J. (2002). Introduction. *SPECIAL ISSUE: Game engines in scientific research. Commun. ACM*, 45(1):27–31. 2.2

[Linden Lab, 2008] Linden Lab (2008). Second life. Available at http://secondlife.com/. 2.3.1, 3.1

[Lozano, 2005] Lozano, M. (2005). *Animación comportamental de personajes inteligentes 3D basada en miniMin-HSP (Heuristic Search Planning)*. PhD thesis, Escola Tècnica Superior d'Enginyeria. Universitat de València. (document), 1.1, 2.4.1, 2.12, 4.4, 4.4.1, 5.3, 5.3.2, 5.3.3, 6.2

[Lozano and Calderón, 2004] Lozano, M. and Calderón, C. (2004). Entornos virtuales 3D clásicos e inteligentes: hacia un nuevo marco de simulación para aplicaciones gráficas 3D interactivas. *Revista Ibero-Americana de Inteligencia Artificial*, 23. 4.2

[Luck and Aylett, 2000] Luck, M. and Aylett, R. (2000). Applying artificial intelligence to virtual reality: Intelligent virtual environments. *Applied Artificial Intelligence*, 14(1):3–32. 2.1

[Mamei et al., 2004] Mamei, M., Zambonelli, F., and Leonardi, L. (2004). Co-fields: A physically inspired approach to motion coordination. *IEEE Pervasive Computing*, 3(2):52–61. 4.4

[Marsh, 1994] Marsh, S. (1994). *Formalizing Trust As a Computational Concept*. PhD thesis, University of Stirling, U.K. 3.4.2

[Martin et al., 1999] Martin, D., Cheyer, A., and Moran, D. (1999). The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence*, 13(1/2):91–128. 3.3.3

[Meneguzzi et al., 2004] Meneguzzi, F., Zorzo, A., and Da Costa Mï¿½a, M. (2004). Propositional planning in BDI agents. In *SAC'04: Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 58–63. ACM Press. 2.4.2

[Molet et al., 1997] Molet, M., Aubel, A., Capin, T., Carion, S., Lee, E., Thalmann, N., Noser, H., and Thalmann, D. (1997). Anyone for tennis? *Presence*, 8(2):140–156. 2.3.2

[Molyneux, 2001] Molyneux, P. (2001). Black&white. Postmortem: Lionhead studio. 2.4.2

[Nebel et al., 1997] Nebel, B., Dimopoulos, Y., and Koehler, J. (1997). Irrelevant facts and operators in plan generation. Technical report, Institut für Informatik Freiburg. 2.4.1

[Niewiadomski and Pelachaud, 2007] Niewiadomski, R. and Pelachaud, C. (2007). Fuzzy similarity od facial expressions of embodied agents. In Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., and Pelé, D., editors, *Proc. of IVA'07: Intelligent Virtual Agents*, number 4722, pages 86–98, Paris. Springer-Verlag LNAI. 2.3.1

[Nilsson, 2001] Nilsson, N. (2001). *Inteligencia Artificial, una nueva síntesis*. McGraw-Hill. 2, 5.2.2

[Noma and Badler, 1997] Noma, T. and Badler, N. (1997). A virtual human presenter. In *Proc. of the IJCAI'97. Workshop on Animated Interface Agents*, Nagoya,Japan. 2.3.1

[Okuyama et al., 2005] Okuyama, F. Y., Bordini, R. H., and da Rocha, A. C. (2005). ELMS: An environment description language for multi-agent simulations. In Weyns, D., van Dyke Parunak, H., and Michel, F., editors, *Proc. of the 1st International Workshop on Environments for Multiagent Systems (E4MAS), held with AAMAS'04*, number 3374, pages 67–83. Springer-Verlag LNAI. 2.2.2

[Omicini and Zambonelli, 1999] Omicini, A. and Zambonelli, F. (1999). Tuple centres for the coordination of internet agents. In *Proc. of the ACM Symposium on Applied Computing*, pages 183–190. ACM Press. 4.3.2

[Osborne, 2004] Osborne, M. (2004). *An Introduction to Game Theory*. Oxford University Press. 3.4.3

[OSG, 2008] OSG (2008). Openscenegraph website. Available at http://www.openscenegraph.org/projects/osg. 2.2, 4.2, 6.4

[Otto, 2005] Otto, K. A. (2005). Towards semantic virtual environments. In *Workshop Towards Semantic Virtual Environments (SVE'05)*. 2.2.2

[Pandzic et al., 1998] Pandzic, I., Capin, T., Lee, E., Magnenat-Thalmann, N., and Thalmann, D. (1998). Autonomous actors in networked collaborative virtual environments. In *Proc. of MultiMedia Modeling'98*. IEEE Computer Society Press. 3.1

[Parunak et al., 2002] Parunak, H. V. D., Fleishcher, M., Brueckner, S., and Odell, J. (2002). Co-x: Defining what agents do together. In *Workshop on Teamwork and Coalition Formation, AAMAS*. 3.3

[Pelachaud, 2005] Pelachaud, C. (2005). Multimodal expressive embodied conversational agents. In *Proc. of MULTIMEDIA '05: The 13th annual ACM international conference on Multimedia*, pages 683–689, New York, NY, USA. ACM. (document), 2.1, 2.3.1, 2.6

[Pelechano et al., 2007] Pelechano, N., Allbeck, J., and Badler, N. (2007). Controlling individual agents in high-density crowd simulation. In Metaxas, D. and Popovic, J., editors, *Proc. of SCA'07: Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. ACM. 3.1, 4.4, 6.1

[Pellens et al., 2005] Pellens, B., Bille, W., De Troyer, O., and Kleinermann, F. (2005). VR-wise: A conceptual modelling approach for virtual environments. In *Methods and Tools for Virtual Reality (MeTo-VR 2005) workshop*. 2.2.1

[Perlin and Goldberg, 1996] Perlin, K. and Goldberg, A. (1996). IMPROV: a system for scripting interactive agents in virtual worlds. In *Proc. of SIGGRAPH'96*, pages 205–216. ACM. 2.3.2

[Piaget, 1995] Piaget, J. (1995). *Sociological studies*. Routledge. 5.4

[Prada and Paiva, 2005] Prada, R. and Paiva, A. (2005). Believable groups of synthetic characters. In *Proc. of AAMAS '05: Autonomous Agents and Multi-Agent Systems*, pages 37–43, New York, NY, USA. ACM. (document), 3.1, 3.1, 6.1

[Protégé Wiki, 2008] Protégé Wiki (2008). Protégé ontology library. Available at http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library. 4.3

[Racer Systems, 2008] Racer Systems (2008). RACER: Renamed abox and concept expression reasoner. Available at http://www.sts.tu-harburg.de/ r.f.moeller/racer/. 4.3

[Rao, 1996] Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In Verlag, S., editor, *Proc. of MAAMAW'96*, number 1038 in LNAI, pages 42–55. 4.4.1, 6.4.2

[Raupp and Thalmann, 2001] Raupp, S. and Thalmann, D. (2001). Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):152–164. 2.4.2

[Reilly, 1996] Reilly, W. S.Ñ. (1996). *Believable Social and Emotional Agents*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. 3.1

[Reynolds, 1987] Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proc. of SIGGRAPH'87*, pages 25–34. ACM. (document), 2.3.2, 2.8, 3.1

[Reynolds, 1999] Reynolds, C. (1999). Steering behaviors for autonomous characters. In *Proc. of Games Developer Conference*, pages 763–782. 2.3.2, 4.4

[Ribeiro et al., 2003] Ribeiro, M., da Rocha, A. C., and Bordini, R. H. (2003). A system of exchange values to support social interactions in artificial societies. In *Proc. of AAMAS'03: Autonomous Agents and Multi-agent Systems*. ACM. 5.4

[Rodríguez-García, 2004] Rodríguez-García, R. (2004). *Actores sintéticos en tiempo real: Nuevas estructuras de datos y métodos para su integración en aplicaciones de simulación*. PhD thesis, Escola Tècnica Superior d'Enginyeria. Universitat de València. 2.2

[Sabater and Sierra, 2002] Sabater, J. and Sierra, C. (2002). Reputation and social network analysis. In *Proc. of AAMAS'02: International Conference on Autonomous Agents and Multi-Agent Systems*, pages 475–482. 3.4.2

[Sandholm, 1998] Sandholm, T. (1998). Contract types for satisficing task allocation: I theoretical results. In *Proc. of AAAI Spring Symposium: Satisficing Models*. 3.4.3

[Sandholm and Suri, 2001] Sandholm, T. and Suri, S. (2001). Side constraints and non-price attributes in markets. In *Proc. of IJCAI Workshop on Distributed Constraint Reasoning*. 3.4.3, 5.4.4

[Schmitt and Rist, 2003] Schmitt, M. and Rist, T. (2003). Avatar arena: Virtual group-dynamics in multicharacter negotiation scenarios. In *Procs. of IVA'03: International Conference on Intelligent Virtual Agents*. Springer LNAI. 3.1

[Scott, 1991] Scott, J. (1991). *Social network analysis: A handbook*. Sage publications, London. 3.4

[Shao and Terzopoulos, 2005] Shao, W. and Terzopoulos, D. (2005). Autonomous pedestrians. In K. Anjyo, P. F., editor, *Proc. of Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. 4.4

[Shoham and Tennenholtz, 1995] Shoham, Y. and Tennenholtz, M. (1995). On social laws for artificial agent societies: Off-line design. *Artificial Intelligence*, 73(1–2):231–252. 3.3.2, 3.4, 5.2.2

[Sichman et al., 1994] Sichman, J., Conte, R., Castelfranchi, C., and Demazeau, Y. (1994). A social reasoning mechanism based on dependence networks. In Cohn, A., editor, *Proc. of ECAI'94: European Conference on Artificial Intelligence*, pages 188–192. Wiley. 3.4.1

[Sichman and Demazeau, 2001] Sichman, J. and Demazeau, Y. (2001). On social reasoning in multi-agent systems. *Revista Ibero-Americana de Inteligencia Artificial*, 13:68–84. 3.4.1

[Soto and Allongue, 2002] Soto, M. and Allongue, S. (2002). Modeling methods for reusable and interoperable virtual entities in multimedia virtual worlds. *Multimedia Tools and Appliclations*, 16:161–177. 2.2.2

[Stanford Medical Informatics, 2006] Stanford Medical Informatics (2006). Protégé. Available at http://protege.stanford.edu/. 4.3, 1

[Stanney, 2002] Stanney, K., editor (2002). *Handbook of Virtual Environments*. Lawrence Erlbaum Associates. 2.1

[Stephenson, 1992] Stephenson, N. (1992). *Snow crash*. Bantam Books. 1.1

[Sycara, 1998] Sycara, K. (1998). Multiagent systems. *AI Magazine*, 10(2):79–93. 3.2

[Tambe, 1997] Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124. (document), 3.3.3, 3.5

[Thalmann et al., 1998] Thalmann, N., Kalra, P., and Escher, M. (1998). Face to virtual face. In *Proceedings of the IEEE*, volume 86, pages 870–883. 2.3.1

[Thomas and Donikian, 2000] Thomas, G. and Donikian, S. (2000). Virtual humans animation in informed urban environments. In *Computer Animation 2000*. 2.2.1

[Tomlinson and Blumberg, 2002] Tomlinson, B. and Blumberg, B. (2002). Social synthetic characters. In *Proc. of Computer Graphics*, volume 26. 3.1

[Torres et al., 2003] Torres, J. A., Nedel, L. P., and Bordini, R. H. (2003). Using the BDI architecture to produce autonomous characters in virtual worlds. In *Procs. of IVA'03: International Conference on Intelligent Virtual Agents*. 3.2

[Tu and Terzopoulos, 1994] Tu, X. and Terzopoulos, D. (1994). Artificial fishes: physics, locomotion, perception, behavior. In *Proc. of SIGGRAPH'94*, pages 43–50. ACM. 2.3, 2.4.1

[van Kokswijk, 2007] van Kokswijk, J. (2007). *Digital Ego: Social and Legal Aspects of Virtual Identity*. 1.1, 2.3.1

[Vercouter et al., 2007] Vercouter, L., Casare, S. J., Sichman, J. S., and Brandao, A. A. F. (2007). An experience on reputation models interoperability based on a functional ontology. In *Proc. of IJCAI07: International Joint Conference on Artificial Intelligence*, Hyderabad, India. 3.4.2

[Viroli et al., 2006] Viroli, M., Ricci, A., and Omicini, A. (2006). Operating instructions for intelligent agent coordination. *Knowledge Engineering Review*, 21:49–69. 2.2.3

[Vosinakis and Panayiotopoulos, 2001] Vosinakis, S. and Panayiotopoulos, T. (2001). Simhuman: A platform for real time virtual agents planning capabilities. In *Proc. of IVA'01: International Conference on Intelligent Virtual Agents*. Springer-Verlag LNAI. 3.2

[Vosinakis and Panayotopoulos, 2003] Vosinakis, S. and Panayotopoulos, T. (2003). A task definition language for virtual agents. *Journal of WSCG*, 11:512–519. 2.2.2

[W3C, 2004] W3C (2004). OWL web ontology language guide. Available at http://www.w3.org/TR/owl-guide/. 2.2.3, 4.3

[Wooldridge and Jennings, 1994] Wooldridge, M. and Jennings, N. (1994). Towards a theory of cooperative problem solving. In *Proc. of MAAMAW094*, pages 15–26. 3.3.3

[Wooldridge and Jennings, 1995] Wooldridge, M. J. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152. 1, 5.2.2

[Yang, 1997] Yang, Q. (1997). *Intelligent Planning*. Springer. 3.3.2

[Young, 2001] Young, R. (2001). An overview of the mimesis architecture integrating intelligent narrative control into an existing gaming environment. Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, AAAI Press (2001). 2.4.1

# PARTE IV

# ANEXOS

ANEXO A

# ONTOLOGÍAS DEL ENTORNO VIRTUAL SEMÁNTICO

Este anexo recoge la ontología base del entorno virtual semántico, un ejemplo de ontología específica del dominio que define algunos objetos necesarios en un bar virtual y una posible instanciación de este bar.

## A.1. SVE CORE ONTOLOGY

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns="http://www.uv.es/~agentes/SVECore#"
  xml:base="http://www.uv.es/~agentes/SVECore">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="InterGroupSocialRelation">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="SocialRelation"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="MovableObject">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="GraphicalEntity"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Group"/>
  <owl:Class rdf:about="#GraphicalEntity">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
```

```
    >1</owl:cardinality>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="posx"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:cardinality
    rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="ry"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf
rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="height"/>
    </owl:onProperty>
    <owl:cardinality
    rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:cardinality
    rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="objectFree"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:cardinality
    rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</owl:cardinality>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="posy"/>
    </owl:onProperty>
  </owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty>
      <owl:FunctionalProperty rdf:ID="rx"/>
    </owl:onProperty>
    <owl:cardinality
```

```
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="posz"/>
        </owl:onProperty>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="yaw"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Agent">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="FilterClass"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#GraphicalEntity"/>
  </owl:Class>
  <owl:Class rdf:ID="CountableContainer">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="num"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Container"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="max"/>
        </owl:onProperty>
      </owl:Restriction>
```

```
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ObjectContainer">
  <rdfs:subClassOf rdf:resource="#CountableContainer"/>
</owl:Class>
<owl:Class rdf:ID="ServiceContainer">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:about="#num"/>
      </owl:onProperty>
      <owl:cardinality
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#CountableContainer"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:ID="containsSubstance"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="FrienshipRelation">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="favoursReceived"/>
      </owl:onProperty>
      <owl:maxCardinality
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:DatatypeProperty rdf:ID="favoursDone"/>
      </owl:onProperty>
      <owl:maxCardinality
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="AgentSocialRelation"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="HeterogeneousObjectContainer">
```

```
    <rdfs:subClassOf rdf:resource="#ObjectContainer"/>
  </owl:Class>
  <owl:Class rdf:about="#Container">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="full"/>
        </owl:onProperty>
        <owl:maxCardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#GraphicalEntity"/>
    <rdfs:subClassOf rdf:resource="#FilterClass"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:maxCardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="empty"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="HomogeneousObjectContainer">
    <rdfs:subClassOf rdf:resource="#ObjectContainer"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="currBaseObjects"/>
        </owl:onProperty>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:FunctionalProperty rdf:ID="objectBaseName"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
```

```
        <owl:FunctionalProperty rdf:ID="numBaseObjects"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="BaseObject">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:FunctionalProperty rdf:ID="baseFree"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#FilterClass"/>
  <rdfs:subClassOf rdf:resource="#GraphicalEntity"/>
</owl:Class>
<owl:Class rdf:ID="Substance">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="NonGraphicalEntity"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="GroupSocialRelation">
  <rdfs:subClassOf rdf:resource="#SocialRelation"/>
</owl:Class>
<owl:Class rdf:ID="UncountableContainer">
  <rdfs:subClassOf rdf:resource="#Container"/>
</owl:Class>
<owl:Class rdf:about="#AgentSocialRelation">
  <rdfs:subClassOf rdf:resource="#SocialRelation"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="domainAgentSocialRelation">
  <rdfs:range rdf:resource="#Agent"/>
  <rdfs:domain rdf:resource="#AgentSocialRelation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="targetAgentSocialRelation">
  <rdfs:domain rdf:resource="#AgentSocialRelation"/>
  <rdfs:range rdf:resource="#Agent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="pickedBy">
  <rdfs:domain rdf:resource="#MovableObject"/>
  <rdfs:range rdf:resource="#Agent"/>
  <rdfs:subPropertyOf>
    <owl:FunctionalProperty rdf:ID="senseDepends"/>
  </rdfs:subPropertyOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="domainGroupSocialRelation">
  <rdfs:domain rdf:resource="#Agent"/>
  <rdfs:range rdf:resource="#GroupSocialRelation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="domainInterGroupSocialRelation">
  <rdfs:range rdf:resource="#InterGroupSocialRelation"/>
```

```
    <rdfs:domain rdf:resource="#Group"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="in">
    <rdfs:subPropertyOf>
      <owl:FunctionalProperty rdf:about="#senseDepends"/>
    </rdfs:subPropertyOf>
    <rdfs:domain rdf:resource="#MovableObject"/>
    <rdfs:range rdf:resource="#Container"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#containsSubstance">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#ServiceContainer"/>
          <owl:Class rdf:about="#UncountableContainer"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range rdf:resource="#Substance"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="on">
    <rdfs:subPropertyOf>
      <owl:FunctionalProperty rdf:about="#senseDepends"/>
    </rdfs:subPropertyOf>
    <rdfs:range rdf:resource="#BaseObject"/>
    <rdfs:domain rdf:resource="#MovableObject"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="targetInterGroupSocialRelation">
    <rdfs:range rdf:resource="#InterGroupSocialRelation"/>
    <rdfs:domain rdf:resource="#Group"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="targetGroupSocialRelation">
    <rdfs:domain rdf:resource="#Group"/>
    <rdfs:range rdf:resource="#GroupSocialRelation"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:about="#favoursReceived">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#FrienshipRelation"/>
    <rdfs:subPropertyOf>
      <owl:DatatypeProperty rdf:ID="socialProperty"/>
    </rdfs:subPropertyOf>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="#socialProperty">
    <rdfs:domain rdf:resource="#SocialRelation"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="graphicalProperty"/>
  <owl:DatatypeProperty rdf:about="#favoursDone">
    <rdfs:subPropertyOf rdf:resource="#socialProperty"/>
    <rdfs:domain rdf:resource="#FrienshipRelation"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:about="#objectBaseName">
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
    <rdfs:domain rdf:resource="#HomogeneousObjectContainer"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="filterType">
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first
          rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >ALL</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:first
            rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
            >NOTHING</rdf:first>
            <rdf:rest rdf:parseType="Resource">
              <rdf:first
              rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >CLASSES-ONLY</rdf:first>
              <rdf:rest
              rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            </rdf:rest>
          </rdf:rest>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
    <rdfs:domain rdf:resource="#FilterClass"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#full">
    <rdfs:domain rdf:resource="#Container"/>
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#posz">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
    <rdfs:subPropertyOf rdf:resource="#graphicalProperty"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#posy">
    <rdfs:subPropertyOf rdf:resource="#graphicalProperty"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#numBaseObjects">
    <rdfs:domain rdf:resource="#HomogeneousObjectContainer"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
```

```
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#objectFree">
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#yaw">
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:subPropertyOf rdf:resource="#graphicalProperty"/>
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#senseDepends">
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:range rdf:resource="#FilterClass"/>
    <rdfs:comment
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Sensorial dependency</rdfs:comment>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#baseFree">
    <rdfs:domain rdf:resource="#BaseObject"/>
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#currBaseObjects">
    <rdfs:domain rdf:resource="#HomogeneousObjectContainer"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#ry">
    <rdfs:subPropertyOf rdf:resource="#graphicalProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#rx">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
    <rdfs:subPropertyOf rdf:resource="#graphicalProperty"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="closed">
    <rdfs:range
```

```
    rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#max">
    <rdfs:domain rdf:resource="#CountableContainer"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#num">
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#CountableContainer"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:comment
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Number of objects or services in the container</rdfs:comment>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#empty">
    <rdfs:comment xml:lang="en">Number of objects or services in
    the container</rdfs:comment>
    <rdfs:domain rdf:resource="#Container"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="typeObject">
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#HomogeneousObjectContainer"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#posx">
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:subPropertyOf rdf:resource="#graphicalProperty"/>
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:about="#height">
    <rdfs:range
    rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
    <rdfs:domain rdf:resource="#GraphicalEntity"/>
    <rdf:type
    rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:subPropertyOf rdf:resource="#graphicalProperty"/>
  </owl:FunctionalProperty>
</rdf:RDF>
```

## A.2. DOMAIN SPECIFIC ONTOLOGY: BAR VIRTUAL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.uv.es/agentes/sve/DSOBar.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:core="http://www.uv.es/~agentes/SVECore#"
  xml:base="http://www.uv.es/agentes/sve/DSOBar.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.uv.es/~agentes/SVECore"/>
  </owl:Ontology>
  <rdfs:Class rdf:about="http://www.uv.es/~agentes/
  SVECore#HeterogeneousObjectContainer"/>
  <rdfs:Class
  rdf:about="http://www.uv.es/~agentes/
  SVECore#MovableObject"/>
  <rdfs:Class rdf:about="http://www.uv.es/~agentes/
  SVECore#HomogeneousObjectContainer"/>
  <rdfs:Class rdf:about="http://www.uv.es/~agentes/
  SVECore#BaseObject"/>
  <rdfs:Class rdf:about="http://www.uv.es/~agentes/
  SVECore#UncountableContainer"/>
  <rdfs:Class rdf:about="http://www.uv.es/~agentes/
  SVECore#ServiceContainer"/>
  <owl:Class rdf:ID="Cup">
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#UncountableContainer"/>
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#MovableObject"/>
  </owl:Class>
  <owl:Class rdf:ID="Dish">
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#MovableObject"/>
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#UncountableContainer"/>
  </owl:Class>
  <owl:Class rdf:ID="TrayOfGlasses">
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#HomogeneousObjectContainer"/>
  </owl:Class>
  <owl:Class rdf:ID="Tray">
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#BaseObject"/>
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#MovableObject"/>
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#HeterogeneousObjectContainer"/>
  </owl:Class>
  <owl:Class rdf:ID="Glass">
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
```

```
    SVECore#UncountableContainer"/>
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#MovableObject"/>
  </owl:Class>
  <owl:Class rdf:ID="Cabinet">
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#HeterogeneousObjectContainer"/>
  </owl:Class>
  <owl:Class rdf:ID="Bottle">
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#ServiceContainer"/>
    <rdfs:subClassOf rdf:resource="http://www.uv.es/~agentes/
    SVECore#MovableObject"/>
  </owl:Class>
</rdf:RDF>
```

## A.3. INSTANCIACIÓN DEL MUNDO: BAR VIRTUAL

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.uv.es/~agentes/WorldBar#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dso="http://www.uv.es/~agentes/DSOBar#"
    xmlns:core="http://www.uv.es/~agentes/SVECore#"
  xml:base="http://www.uv.es/~agentes/WorldBar">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://www.uv.es/~agentes/DSOBar"/>
  </owl:Ontology>
  <dso:Cabinet rdf:ID="refrigerator">
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >50</core:max>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >15</core:num>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >-1.0</core:posx>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >CLASSES-ONLY</core:filterType>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >-4.7</core:posy>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.3</core:ry>
```

```
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >2.0</core:rx>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.0</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.5</core:posz>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
  </dso:Cabinet>
  <dso:Dish rdf:ID="dish_1">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:ry>
    <core:in>
      <dso:Cabinet rdf:ID="cabinet_1">
        <core:full
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</core:full>
        <core:max
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >20</core:max>
        <core:ry
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >0.5</core:ry>
        <core:filterType
        rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >CLASSES-ONLY</core:filterType>
        <core:posy
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >4.5</core:posy>
        <core:rx
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >1.5</core:rx>
        <core:posx
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >-2.0</core:posx>
        <core:height
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >1.0</core:height>
        <core:yaw
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >0.0</core:yaw>
        <core:posz
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >0.5</core:posz>
```

```
        <core:empty
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</core:empty>
        <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >9</core:num>
        <core:objectFree
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</core:objectFree>
      </dso:Cabinet>
    </core:in>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:rx>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:objectFree
     rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
  </dso:Dish>
  <dso:Dish rdf:ID="dish_4">
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:rx>
    <core:in rdf:resource="#cabinet_1"/>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
```

```
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:ry>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
  </dso:Dish>
  <dso:Tray rdf:ID="tray_1">
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:rx>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5</core:num>
    <core:baseFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:baseFree>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5</core:max>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:ry>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
    <core:in rdf:resource="#cabinet_1"/>
  </dso:Tray>
  <dso:Bottle rdf:ID="beerBottle_4">
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >ALL</core:filterType>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:in rdf:resource="#refrigerator"/>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:empty rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:containsSubstance>
      <core:Substance rdf:ID="Beer"/>
    </core:containsSubstance>
  </dso:Bottle>
  <core:ServiceContainer rdf:ID="coffeeMachine">
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.5</core:ry>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >100</core:max>
    <core:containsSubstance>
      <core:Substance rdf:ID="Coffee"/>
    </core:containsSubstance>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:rx>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >100</core:num>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:empty rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.5</core:height>
  </core:ServiceContainer>
  <core:HomogeneousObjectContainer rdf:ID="shelf_A">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.7</core:posz>
    <core:empty rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:typeObject
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Glass</core:typeObject>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:objectBaseName
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Glass</core:objectBaseName>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.0</core:posx>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.5</core:ry>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >50</core:max>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >50</core:num>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >4.5</core:posy>
    <core:numBaseObjects
    rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >3</core:numBaseObjects>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.4</core:height>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.5</core:rx>
  </core:HomogeneousObjectContainer>
  <dso:Dish rdf:ID="dish_6">
    <core:empty rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:rx>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:ry>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
```

```
    >false</core:objectFree>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:in rdf:resource="#cabinet_1"/>
  </dso:Dish>
  <dso:Dish rdf:ID="dish_3">
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:in rdf:resource="#cabinet_1"/>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:ry>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:rx>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
  </dso:Dish>
  <dso:Bottle rdf:ID="orangeBottle_3">
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:containsSubstance>
      <core:Substance rdf:ID="Orange"/>
    </core:containsSubstance>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:in rdf:resource="#refrigerator"/>
```

```
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
  </dso:Bottle>
  <dso:Bottle rdf:ID="rumBottle">
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:in>
      <core:HeterogeneousObjectContainer rdf:ID="shelf_B">
        <core:filterType
        rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >ALL</core:filterType>
        <core:empty
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</core:empty>
        <core:posz
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >1.7</core:posz>
        <core:full
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</core:full>
        <core:max
        rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >10</core:max>
        <core:rx
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >1.5</core:rx>
        <core:posy
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >-4.9</core:posy>
        <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >4</core:num>
        <core:objectFree
```

```
        rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
        >false</core:objectFree>
        <core:yaw
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >0.0</core:yaw>
        <core:ry
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >0.1</core:ry>
        <core:posx
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >-1.0</core:posx>
        <core:height
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >0.05</core:height>
      </core:HeterogeneousObjectContainer>
    </core:in>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:posz
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.3</core:height>
    <core:containsSubstance>
      <core:Substance rdf:ID="Rum"/>
    </core:containsSubstance>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:rx>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:max>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:num>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
  </dso:Bottle>
  <dso:Bottle rdf:ID="beerBottle_2">
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.2</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:empty rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:containsSubstance rdf:resource="#Beer"/>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:in rdf:resource="#refrigerator"/>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
  </dso:Bottle>
  <dso:Tray rdf:ID="tray_3">
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5</core:max>
    <core:in rdf:resource="#cabinet_1"/>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:ry>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5</core:num>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:baseFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:baseFree>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:on>
      <dso:Tray rdf:ID="tray_2">
        <core:on rdf:resource="#tray_1"/>
        <core:posz
        rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
        >0.0</core:posz>
        <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
```

```
      >5</core:max>
      <core:in rdf:resource="#cabinet_1"/>
      <core:baseFree
      rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >false</core:baseFree>
      <core:full
      rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >false</core:full>
      <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >0.25</core:ry>
      <core:posx
      rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >0.0</core:posx>
      <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >5</core:num>
      <core:filterType
      rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >ALL</core:filterType>
      <core:posy
      rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >0.0</core:posy>
      <core:height
      rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >0.01</core:height>
      <core:objectFree
      rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >false</core:objectFree>
      <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >0.25</core:rx>
      <core:empty
      rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
      >true</core:empty>
      <core:yaw
      rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
      >0.0</core:yaw>
    </dso:Tray>
  </core:on>
  <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.25</core:rx>
  <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.01</core:height>
  <core:objectFree
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</core:objectFree>
  <core:empty
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >true</core:empty>
  <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:yaw>
</dso:Tray>
<dso:Cup rdf:ID="cup_3">
  <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:yaw>
  <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
>0.03</core:rx>
<core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>0.0</core:posy>
<core:filterType
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>ALL</core:filterType>
<core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>0.07</core:height>
<core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>0.03</core:ry>
<core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>false</core:full>
<core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
>0.0</core:posz>
<core:empty
rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
>true</core:empty>
<core:in>
  <dso:Cabinet rdf:ID="cupboard">
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.0</core:height>
    <core:yaw
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:posz
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.8</core:posz>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >5</core:num>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >40</core:max>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.75</core:ry>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.5</core:rx>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >CLASSES-ONLY</core:filterType>
    <core:posx
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >-4.5</core:posx>
    <core:posy
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >-1.75</core:posy>
```

```
    </dso:Cabinet>
  </core:in>
  <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:posx>
  <core:objectFree
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</core:objectFree>
</dso:Cup>
<core:BaseObject rdf:ID="table_A">
  <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:posy>
  <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >-0.5</core:posx>
  <core:objectFree
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</core:objectFree>
  <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.5</core:rx>
  <core:filterType
  rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ALL</core:filterType>
  <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:yaw>
  <core:height
  rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.5</core:height>
  <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</core:ry>
  <core:baseFree
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >true</core:baseFree>
  <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.75</core:posz>
</core:BaseObject>
<core:BaseObject rdf:ID="table_B">
  <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.0</core:ry>
  <core:height
  rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.5</core:height>
  <core:objectFree
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</core:objectFree>
  <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.0</core:posy>
  <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >-4.5</core:posx>
  <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.75</core:posz>
  <core:filterType
  rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ALL</core:filterType>
  <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:yaw>
```

```
    <core:baseFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:baseFree>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.5</core:rx>
  </core:BaseObject>
  <core:Substance rdf:ID="Coke"/>
  <dso:Cup rdf:ID="cup_2">
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:rx>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.07</core:height>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:in rdf:resource="#cupboard"/>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
  </dso:Cup>
  <core:Agent rdf:ID="customerB">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >4.5</core:posx>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:ry>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >2.5</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.0</core:posz>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:rx>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >2.0</core:height>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:objectFree>
  </core:Agent>
  <dso:Bottle rdf:ID="beerBottle_3">
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:empty rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:containsSubstance rdf:resource="#Beer"/>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:in rdf:resource="#refrigerator"/>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
  </dso:Bottle>
  <dso:Bottle rdf:ID="beerBottle_5">
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:full rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:objectFree
```

```
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:in rdf:resource="#refrigerator"/>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:containsSubstance rdf:resource="#Beer"/>
  </dso:Bottle>
  <dso:Bottle rdf:ID="orangeBottle_5">
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:containsSubstance rdf:resource="#Orange"/>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:in rdf:resource="#refrigerator"/>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
  </dso:Bottle>
  <dso:Cup rdf:ID="cup_5">
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
```

```
    >false</core:objectFree>
    <core:in rdf:resource="#cupboard"/>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.07</core:height>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:rx>
  </dso:Cup>
  <core:ServiceContainer rdf:ID="donutsTray">
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:rx>
    <core:on rdf:resource="#table_B"/>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:containsSubstance>
      <core:Substance rdf:ID="Donut"/>
    </core:containsSubstance>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.4</core:ry>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
```

```
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >15</core:num>
    <rdf:type
    rdf:resource="http://www.uv.es/~agentes/SVECore#MovableObject"/>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >15</core:max>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
  </core:ServiceContainer>
  <core:ServiceContainer rdf:ID="iceContainer">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:containsSubstance>
      <core:Substance rdf:ID="Ice"/>
    </core:containsSubstance>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.5</core:posz>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >50</core:num>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >-4.75</core:posy>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.0</core:height>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >2.5</core:posx>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >50</core:max>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:ry>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:rx>
  </core:ServiceContainer>
  <dso:Bottle rdf:ID="cokeBottle_3">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:containsSubstance rdf:resource="#Coke"/>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:in rdf:resource="#refrigerator"/>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
  </dso:Bottle>
  <dso:Dish rdf:ID="dish_2">
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:in rdf:resource="#cabinet_1"/>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:ry>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.2</core:rx>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
  </dso:Dish>
  <dso:Bottle rdf:ID="ginBottle">
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:rx>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:in rdf:resource="#shelf_B"/>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.3</core:height>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:num>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:max>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:containsSubstance>
      <core:Substance rdf:ID="Gin"/>
    </core:containsSubstance>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
  </dso:Bottle>
  <dso:Cup rdf:ID="cup_1">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.03</core:rx>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:in rdf:resource="#cupboard"/>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.07</core:height>
  </dso:Cup>
  <dso:Bottle rdf:ID="cokeBottle_4">
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:in rdf:resource="#refrigerator"/>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:containsSubstance rdf:resource="#Coke"/>
  </dso:Bottle>
  <dso:Cup rdf:ID="cup_4">
```

```
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:in rdf:resource="#cupboard"/>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.07</core:height>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:rx>
  </dso:Cup>
  <core:Substance rdf:ID="Milk"/>
  <dso:Bottle rdf:ID="whiskyBottle">
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:containsSubstance>
      <core:Substance rdf:ID="Whisky"/>
    </core:containsSubstance>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.3</core:height>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:max>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:in rdf:resource="#shelf_B"/>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.03</core:rx>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:num>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
  </dso:Bottle>
  <dso:Bottle rdf:ID="cokeBottle_5">
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:in rdf:resource="#refrigerator"/>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:containsSubstance rdf:resource="#Coke"/>
  </dso:Bottle>
  <core:BaseObject rdf:ID="bar">
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >5.0</core:ry>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >3.5</core:posx>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
```

```
    <core:height rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >1.5</core:height>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.5</core:rx>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:baseFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:baseFree>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.75</core:posz>
  </core:BaseObject>
  <dso:Bottle rdf:ID="orangeBottle_1">
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:containsSubstance rdf:resource="#Orange"/>
    <core:posx
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:in rdf:resource="#refrigerator"/>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
  </dso:Bottle>
```

```
<core:Agent rdf:ID="customerC">
  <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >-2.5</core:posy>
  <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >1.0</core:posz>
  <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.25</core:rx>
  <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >4.5</core:posx>
  <core:filterType
  rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ALL</core:filterType>
  <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.25</core:ry>
  <core:objectFree
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >true</core:objectFree>
  <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:yaw>
  <core:height
  rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >2.0</core:height>
</core:Agent>
<dso:Bottle rdf:ID="orangeBottle_2">
  <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:posx>
  <core:containsSubstance rdf:resource="#Orange"/>
  <core:filterType
  rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >ALL</core:filterType>
  <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.015</core:rx>
  <core:in rdf:resource="#refrigerator"/>
  <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:yaw>
  <core:objectFree
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >false</core:objectFree>
  <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.0</core:posz>
  <core:full
  rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
  >true</core:full>
  <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >1</core:num>
  <core:height
  rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.2</core:height>
  <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
  >1</core:max>
  <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
  >0.015</core:ry>
  <core:posy
  rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.0</core:posy>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
  </dso:Bottle>
  <dso:Bottle rdf:ID="vodkaBottle">
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:rx>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:posz
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:num>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.03</core:ry>
    <core:containsSubstance>
      <core:Substance rdf:ID="Vodka"/>
    </core:containsSubstance>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >8</core:max>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.3</core:height>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:in rdf:resource="#shelf_B"/>
  </dso:Bottle>
  <core:Agent rdf:ID="customerA">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:ry>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >4.5</core:posx>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >1.0</core:posz>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:objectFree>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.25</core:rx>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >2.0</core:height>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >4.5</core:posy>
  </core:Agent>
  <dso:Bottle rdf:ID="cokeBottle_1">
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:containsSubstance rdf:resource="#Coke"/>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:in rdf:resource="#refrigerator"/>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
  </dso:Bottle>
  <dso:Dish rdf:ID="dish_5">
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >ALL</core:filterType>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:empty>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:rx>
    <core:in rdf:resource="#cabinet_1"/>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:ry>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.01</core:height>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:full>
  </dso:Dish>
  <dso:Bottle rdf:ID="beerBottle_1">
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:in rdf:resource="#refrigerator"/>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:containsSubstance rdf:resource="#Beer"/>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:posz
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
```

```
    >0.0</core:posz>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
  </dso:Bottle>
  <dso:Bottle rdf:ID="orangeBottle_4">
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
    <core:posx
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:containsSubstance rdf:resource="#Orange"/>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:in rdf:resource="#refrigerator"/>
    <core:posy
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:height
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
  </dso:Bottle>
  <dso:Bottle rdf:ID="cokeBottle_2">
    <core:filterType
    rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ALL</core:filterType>
    <core:containsSubstance rdf:resource="#Coke"/>
    <core:posy rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posy>
    <core:rx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:rx>
    <core:yaw rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:yaw>
    <core:height
```

```
    rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.2</core:height>
    <core:posx rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posx>
    <core:in rdf:resource="#refrigerator"/>
    <core:full
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >true</core:full>
    <core:max rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:max>
    <core:empty
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:empty>
    <core:ry rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.015</core:ry>
    <core:posz rdf:datatype="http://www.w3.org/2001/XMLSchema#float"
    >0.0</core:posz>
    <core:num rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >1</core:num>
    <core:objectFree
    rdf:datatype="http://www.w3.org/2001/XMLSchema#boolean"
    >false</core:objectFree>
  </dso:Bottle>
</rdf:RDF>
```