Agentes móviles inteligentes aplicados al diseño y desarrollo de servicios de datos en entornos inalámbricos y distribuidos

José Alberto Royo Ratia

Tesis Doctoral
Universidad de Zaragoza

Agentes Móviles Inteligentes Aplicados al Diseño y Desarrollo de Servicios de Datos en Entornos Inalámbricos y Distribuidos

José Alberto Royo Ratia

Tesis Doctoral

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

Abril 2009

A mis padres,
ellos me enseñaron las cosas importantes de la vida.

Agradecimientos

Este trabajo no podría haber sido realizado sin la inestimable ayuda de mi director, Eduardo Mena. He tenido la suerte de conocerle y trabajar con él; durante los últimos años, tanto en el trabajo como fuera de él. El resto de los miembros del grupo de Sistemas de Información Distribuidos (SID) en la Universidad de Zaragoza y el de Bases de Datos Interoperantes (BDI) de la Universidad del País Vasco me ayudaron discutiendo muchos de los temas tratados en la Tesis, especialmente Eduardo Mena y Arantza Illarramendi.

También, querría agradecer a Yolanda Villate el precioso tiempo que me ha dedicado tanto para tratar temas profesionales como de otros ámbitos. Del mismo modo mencionar a Sergio Ilarri, compañero de laboratorio durante toda mi época de becario ya que comenzamos la dura andanza del doctorado juntos. Finalmente, agradecer también al resto de los miembros del grupo los ánimos que me han dado en los tiempos difíciles.

No puedo olvidarme de mencionar aquellas personas que han trabajado conmigo en los últimos años. Querría agradecerle a Ignacio Gracia, Victor Pérez y Daniel Fanjul su inestimable ayuda en la implementación del prototipo, y a Javier Campos y Fernando Tricas por su ayuda en el estudio y diseño de funciones de probabilidad utilizadas en la tesis. Finalmente y no por ello menos importante, también querría destacar la inestimable ayuda de Nikola Mitrovic y Jorge Bernad en temas relacionados con interfaces de usuario adaptativos y la estimación del grado de sinonimia entre términos de distintas ontologías, respectivamente.

Además debo agradecer la ayuda de todas aquellas personas que me han ayudado sin ánimo de hacerlo, como por ejemplo los usuarios que realizaron las pruebas de los distintos prototipos desarrollados.

Finalmente, he recibido una inconmensurable ayuda de Isabel, que me ha dado toda clase de apoyo y me ha animado a terminar mi tesis doctoral a pesar de la nueva orientación de mi carrera profesional. Ella nunca perdió la confianza que había depositado en mi, confió en mi incluso cuando yo no creía en mi mismo. No puedo olvidarme de mencionar a mis padres, Antonio y María José, que siempre han estado, están y estarán a mi lado, tanto en los momentos buenos como en los difíciles.

Índice general

Capítulo 1

Introducción

El trabajo presentado en esta tesis se encuadra dentro del estudio y diseño de sistemas multiagente e inteligentes, en entornos inalámbricos y distribuidos. El auge producido en los últimos años, tanto de los entornos de trabajo distribuidos como de los inalámbricos, y sobre todo de la combinación de ambos, ha motivado la búsqueda de nuevos paradigmas de computación con el objetivo de permitir una adaptación flexible del software desarrollado para estos contextos y sus especiales características. Debido al gran auge de los entornos inalámbricos y distribuidos se están desarrollando nuevos paradigmas de computación que puedan facilitar la adaptación del software a este tipo de contextos. Uno de estos nuevos paradigmas consiste en la utilización de agentes software, móviles y/o inteligentes; este paradigma permite el desarrollo de software que puede autoadaptarse a entornos con características tales como: 1) potencia computacional reducida, 2) comunicaciones inestables, de bajas prestaciones y elevado coste (características inherentes de las redes inalámbricas), y 3) necesidades de acceso a datos remotos o distribuidos de forma eficiente. Por lo tanto, debido a las características intrínsecas de los nuevos entornos inalámbricos y distribuidos, resulta de especial interés la capacidad de los agentes software para reaccionar y adaptarse autónomamente al estado del entorno, permitiendo así adaptar su actuación a los diferentes estados de la red, el consumo de energía, el contexto concreto de ejecución, el comportamiento del usuario, y las características técnicas del dispositivo en el que se ejecuten. La principal contribución de esta tesis consistirá *en el estudio y análisis de las distintas propiedades que poseen inherentemente los sistemas multiagente, y los beneficios que puede proporcionar la tecnología de agentes móviles inteligentes en el diseño y desarrollo de arquitecturas software, en especial para aquellas destinadas a utilizarse en entornos distribuidos e inalámbricos.*

Comenzaremos por establecer las definiciones y terminología, relativas a la tecnología de agentes software, que utilizaremos a lo largo de esta tesis. Un *agente software* [IK96, PS98, SSPE04a] se ha definido como un programa que se ejecuta en un cierto contexto de ejecución o *place*[1] [MBB+98] creado usando un *sistema de agentes*. Un sistema de agentes [MBB+98] es una plataforma que puede crear interpretar, ejecutar y transferir agentes. Un agente posee las siguientes propiedades principales: 1) *Autonomía*, posee el control sobre sus propias ac-

[1] Utilizamos el vocablo inglés debido a su amplia aceptación en contextos técnicos de habla hispana.

ciones; 2) *Finalidad*, gestiona una agenda de objetivos; 3) *Cooperación*, un agente es capaz de comunicarse con otros agentes; 4) *Aprendizaje*, cambia su comportamiento de acuerdo a su experiencia previa; 5) *Movilidad*, puede viajar de un ordenador a otro (en realidad, de un *place* a otro); 6) *Reactividad*, percibe ó reacciona ante los cambios de su entorno; y 7) *Persistencia*, para poder guardar su estado, interrumpir su ejecución y continuar la misma más adelante. En algunos contextos concretos no tienen porqué concurrir todas estas propiedades necesariamente.

Para alcanzar los objetivos propuestos en esta tesis, se estudiarán y verificarán las ventajas proporcionadas por la programación basada en la tecnología de agentes mediante la aplicación de este paradigma de diseño a distintos ámbitos y problemas: 1) un servicio de recuperación de software, 2) un servicio de bibliotecas digitales [RW98] para publicaciones de investigación, y 3) tres servicios que tienen una arquitectura dinámica, que dependerá de las capacidades de visualización, ubicación, y recursos de almacenamiento y procesado de datos disponibles en el dispositivo del usuario. Cada uno de estos contextos de ejecución será presentado en detalle en capítulos posteriores, indicando las ventajas que proporcionan las arquitecturas basadas en sistemas multiagente en su diseño.

Inicialmente se diseñara un caso de estudio complejo que permitirá mostrar muchas de las bondades de los sistemas multiagente [IK96, PS98, Mil99, MBB+98] en entornos heterogéneos, distribuidos e inalámbricos. Este sistema, el Servicio de Recuperación de Software (SRS), estará constituido por un conjunto de agentes inteligentes que ayudarán a usuarios inexpertos en la búsqueda, descarga e instalación de software en su dispositivo móvil. El catálogo de software mostrado al usuario, definido mediante una ontología [Gru92], permitirá detallar y describir semánticamente los tipos de software disponibles, los programas y sus características (nombre, descripción, tamaño, versión, sistema operativo, etc.). La arquitectura del SRS se basará en varios agentes que: 1) autónomamente extraerán conocimiento de la web, 2) unificarán automáticamente el conocimiento y representación de los datos extraídos de la web, 3) adaptarán su comportamiento para ayudar al usuario en la tarea de buscar en el catálogo de software, y 4) seleccionarán automática, e inteligentemente, los métodos y tiempo de comunicación consumidos en la ejecución de las distintas tareas del sistema reduciendo con ello la utilización y coste de las comunicaciones inalámbricas. Este caso de estudio permitirá mostrar las bondades de las arquitecturas basadas en agentes inteligentes en la creación de ontologías en un contexto concreto de aplicación, como es el acceso a depósitos de software. La creación de dichas ontologías será realizada automáticamente, por los agentes desplegados por la arquitectura del servicio, extrayendo conocimiento de un conjunto de sitios web que contendrán información semi-estructurada. Además de la extracción de los datos, los agentes que participarán en la arquitectura del sistema, analizarán la información obtenida y la unificarán, definiendo un conjunto de categorías de software con múltiples propiedades y distintas piezas de software.

En segundo lugar, se estudiarán las ventajas proporcionadas por los agentes móviles inteligentes en sistemas de recuperación de información en entornos heterogéneos, distribuidos e inalámbricos, mediante un nuevo caso de estudio, encuadrado dentro del contexto de las bibliotecas digitales. Se demostrará como un conjunto de agentes pueden colaborar para realizar una tarea, y obteniendo mejores prestaciones que siguiendo aproximaciones más clásicas basadas en una arquitectura cliente/servidor. Del mismo modo, se mostrará como di-

chos agentes pueden realizar diversos tratamientos de la información analizada, por ejemplo modificándola o completando información a partir de datos obtenidos en distintos depósitos de datos. Además, se mostrará como los agentes pueden tomar decisiones por el usuario ayudándole en el tratamiento y procesado de la información bibliográfica almacenada en distintos depósitos de datos.

En tercer lugar, se presentará el tercer caso de estudio, mediante tres nuevos casos particulares de utilización de agentes, basados en la generación del Interfaz Gráfico de Usuario (GUI[2]) dependiendo de las capacidades gráficas del dispositivo, la localización de dispositivos en entornos cerrados y el diseño de sistemas de información globales. De esta forma, se analizarán las ventajas proporcionadas por los sistemas multiagente en el desarrollo de sistemas de de acceso a datos inalámbricos en entornos dinámicos, donde las características del entorno (dispositivos, recursos, capacidad de procesamiento, etc.) pueden variar frecuentemente en el transcurso del tiempo. Uno de los principales beneficios obtenidos mediante el diseño de arquitecturas basadas en agentes móviles en contextos que cambian dinámicamente, se traduce en la posibilidad de minimizar la infraestructura subyacente que debe estar disponible en el dispositivo móvil del usuario. Gracias al uso de agentes en el diseño de la arquitectura del sistema, la infraestructura podrá ser creada y desplegada dinámicamente, atendiendo a las necesidades del momento. Además, la utilización de agentes móviles en este contexto de aplicación ayudará a minimizar el software que se requiere tener instalado en el dispositivo del usuario, ya que los agentes que viajarán al dispositivo del usuario bajo demanda, le proporcionarán los servicios que requiera en cada momento, pudiendo incluso ser personalizados atendiendo a la ubicación del usuario.

Los casos de estudio seleccionados para el trabajo de esta tesis: Servicio de Recuperación de Software, bibliotecas digitales, GUIs adaptativos, servicios dependientes de la localización y sistemas de información globales, presentan las siguientes características en común:

- *Requieren acceso a datos distribuidos y heterogéneos*. Nos permitirán comprobar como la utilización de una arquitectura basada en agentes inteligentes permite minimizar el impacto de estos factores debido a que facilitan el acceso a las fuentes de datos, de forma autónoma y en paralelo.

- *Adaptación de arquitecturas heterogéneas*. Las arquitecturas basadas en agentes móviles permitirán realizar un balanceo de la carga de ejecución de los servicios proporcionados, trasladando la ejecución a dispositivos u ordenadores con más recursos. Además, debido a que las plataformas de agentes móviles están implementadas utilizando Java [Mic06a] como lenguaje de programación, el mismo agente podrá ejecutarse en dispositivos con arquitecturas diferentes.

- *Heterogeneidad de usuarios y contextos de ejecución*. Veremos como las arquitecturas basadas en agentes facilitarán la adaptación automática a los distintos mecanismos de comunicaciones utilizables por los diferentes servicios. Del mismo modo, gracias a la inteligencia de los distintos agentes, el servicio que se presta a los usuarios podrá adaptarse a sus preferencias personales, a las características de sus dispositivos y a los datos ubicados en las diferentes fuentes de datos utilizadas.

[2] *Graphical User Interface*. Utilizamos la abreviatura del vocablo inglés debido a su amplia aceptación en contextos técnicos de habla hispana.

Por consiguiente, los casos de estudio seleccionados para el trabajo desarrollado en esta tesis, plantean la resolución de un conjunto de problemas que pertenecen a ámbitos diferentes de aplicación pero que plantean una problemática común, que puede afrontarse mediante el diseño de soluciones basadas en una arquitectura distribuida centrada en el uso de agentes móviles inteligentes.

Finalmente, conviene mencionar que el estudio de la gestión de datos en entornos inalámbricos también ha sido abordado por otros grupos de investigación [Inf09, Sam09, Dis09, Lar09, Adv09]. La aproximación adoptada en esta tesis, está en sintonía con las técnicas propuestas por otros grupos, tales como el uso de agentes (móviles) inteligentes, y el uso de ontologías e Interfaces Gráficas de Usuario adaptativas. Sin embargo, esta tesis *aporta un conjunto de nuevas arquitecturas para servicios de datos, que son altamente escalables y adaptables a cualquier entorno (inalámbrico o no), y a cualquier tipo de dispositivo empleado por el usuario, permitiendo de esta forma demostrar, de forma generalizable, las ventajas que proporcionan los sistemas multiagente para el diseño y desarrollo de servicios de acceso (distribuido/remoto) a datos*.

1.1. Motivación

En la actualidad, tanto el uso de las diversas redes de comunicaciones, como el uso de sistemas de acceso a datos distribuidos, son críticos y tienen una importancia creciente, tanto para la empresa privada como para los organismos públicos. En este nuevo milenio es de vital importancia la capacidad para buscar y descargar información (datos), de forma eficiente y rápida, desde cualquier lugar y en cualquier momento. Debido a esto, se están proponiendo nuevas arquitecturas y paradigmas de computación basados en agentes inteligentes [KSCP04, GSS06, FTS+03, IKT04, PS04, SHD+06, PRU06], que permiten el desarrollo de sistemas que se adaptan automáticamente a entornos inalámbricos y distribuidos.

El modelo de comunicaciones remotas clásico está basado en la llamada a procedimientos remotos[3]. En este modelo un sistema cliente solicita un servicio (Figura 1.1, fase A) a un servidor remoto. A continuación, el servidor procesa la petición y obtiene los resultados (Fase B). Por último, el servidor envía los resultados al cliente que los solicitó (Fase C). El principal problema de la Llamada a Procedimientos Remotos radica en que fue diseñado para entornos de ejecución distribuidos sobre redes cableadas, es decir, redes rápidas y sin apenas errores de transmisión. Por lo tanto, en el diseño de estas arquitecturas no se consideró que pudiese haber errores de conexión frecuentes en las comunicaciones, bajas velocidades de transferencia, o incluso elevados costes de facturación por el uso de las comunicaciones, etc. Características todas ellas que deben tenerse en cuenta como habituales en el contexto de las comunicaciones inalámbricas.

Gracias a la autonomía de los agentes, el modelo anterior puede modificarse de forma que, en la Fase A de la Figura 1.1 un agente cliente viaja desde el dispositivo cliente al dispositivo en el que se esté ejecutando un agente servidor. Una vez allí, el agente cliente puede solicitar la prestación del servicio mediante una invocación local del servicio. El agente cliente esperará los resultados permaneciendo en el servidor (Fase B). Finalmente, el agente

[3] *Remote Procedure Calling (RPC).*

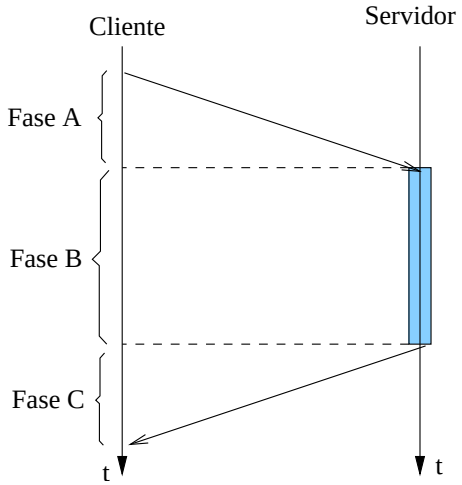


Figura 1.1: Comunicaciones en una arquitectura basada en llamadas a procedimientos remotos

cliente trasladará su ejecución de nuevo al dispositivo donde fue creado (Fase C), donde una vez obtenidos los resultados, continuará con sus tareas.

Un aspecto importante a destacar, es que las arquitecturas cliente/servidor requieren que la conexión de red esté abierta y disponible durante toda la ejecución de la llamada remota. Sin embargo, en las arquitecturas basadas en agentes inteligentes, solamente se requiere que la red esté accesible cuando es utilizada por el agente (Fases A y C). Es decir, podrán producirse fallos en la red, o la desactivación de la misma, durante la ejecución del servicio (Fase B) sin que esto afecte al uso del servicio ni retrase la obtención de resultados para el usuario. A continuación, describiremos el impacto que puede tener un fallo en la red , cuando se produce en alguna de las fases mostradas en la Figura 1.1, tanto en el caso de una aproximación basada en RPC como en el caso de una basada en agentes:

- *Fallo en la Fase A*: En este caso el fallo de red se produce en el transcurso del envío de los parámetros del cliente al servidor ó durante el viaje del agente cliente al dispositivo en el que se ejecuta el servidor. El impacto producido por el fallo implicará en ambos casos volver a iniciar la ejecución del servicio desde el principio, por lo tanto tiene aproximadamente el mismo coste en ambas aproximaciones.

- *Fallo en la Fase B*: En el caso de arquitecturas basadas en una aproximación cliente/servidor, un fallo o interrupción de las comunicaciones durante la fase B, supondrá que deberá volver a iniciarse la ejecución desde el principio (Fase A). Sin embargo, un error de comunicaciones durante esta fase no afectará en absoluto a la aproximación basada en agentes, debido a que no se requiere que la conexión con la red de comunicaciones permanezca abierta. El agente cliente invoca el servicio del agente servidor localmente.

- *Fallo en la Fase C*: En este caso un fallo producido en las comunicaciones provocará un impacto mucho mayor en las prestaciones en la arquitectura cliente/servidor que en las

dos fases mostradas anteriormente. En el caso de arquitecturas basadas en una aproximación cliente/servidor se deberá volver a iniciar la ejecución desde el principio (Fase A). Sin embargo, en el caso de una aproximación basada en agentes, únicamente se deberá repetir el proceso de movimiento del agente cliente (Fase C), desde el servidor hasta el dispositivo del usuario, dado que los resultados de la ejecución del servicio ya estaban en posesión de dicho agente.

En conclusión, las arquitecturas basadas en agentes móviles permitirán el diseño y desarrollo de sistemas robustos que se adaptan fácilmente a las limitaciones de los entornos inalámbricos. La capacidad de adaptación de los sistemas multiagente a entornos inalámbricos se fundamenta en la posibilidad de disminuir el impacto producido por los fallos de red.

Otra de las ventajas de la utilización de los sistemas basados en agentes, en especial cuando éstos son agentes móviles, es que se puede trasladar la ejecución de los distintos servicios al ordenador/dispositivo más adecuado, evitando sobrecargas, cuellos de botella, y el consumo de recursos. Por ejemplo, supongamos que un usuario móvil se conecta desde su PDA[4] para solicitar que se le avise cuando la cotización de un determinado valor bursátil varíe más de un 5 %. Si el usuario ha enviado un agente a otro ordenador de la red fija para que analice las variaciones de la bolsa de valores, podrá apagar su dispositivo móvil. Este agente, podrá realizar un análisis de las cotizaciones desde la red fija, sin consumir recursos de red inalámbrica, y avisando a su usuario cuando éste vuelva a conectarse, tomando mientras tanto sus propias decisiones de forma autónoma y contactando con el usuario solo en caso necesario. De esta forma, en este ejemplo, no solo se posibilita el ahorro de comunicaciones y de la energía necesaria para ellas, también se permite analizar de forma más frecuente los cambios producidos en de la bolsa de valores.

1.2. Entornos de ejecución inalámbricos

Nuestro estudio se centrará en demostrar las ventajas proporcionadas por los sistemas multiagente para diseñar y desarrollar servicios de datos en entornos inalámbricos[5] y distribuidos. Las arquitecturas diseñadas hoy en día para este tipo de entornos, deben considerar que los usuarios requieren poder acceder en cualquier momento y lugar a la información que poseen en su ordenador personal o en la web. Por ejemplo, desde un aeropuerto o desde la oficina de un cliente, del mismo modo que si estuviesen en su despacho. Además, el diseño de estas arquitecturas debe tener en cuenta tanto las restricciones del dispositivo del usuario, como las de los medios de comunicación utilizados.

1.2.1. Conexión a la red en cualquier momento y lugar

En los últimos años se ha producido un cambio crucial en el uso común de Internet. No solo se ha producido una popularización sin precedentes del acceso a Internet, sino que

[4] Siglas que se corresponden con el vocablo inglés *Personal Digital Assistant*, conocido en castellano como Asistente Personal Digital. A lo largo de este documento nos referiremos a este tipo de dispositivos con sus siglas en inglés.

[5] Estas ventajas también lo son en contextos de ejecución con redes cableadas, aunque en menor medida.

mientras que hasta hace poco los usuarios sólo podían disponer de conexión a Internet en el trabajo, y tal vez en el hogar, los usuarios actuales demandan también su disponibilidad en cualquier momento y desde cualquier lugar. Este cambio ha venido provocado por el desarrollo de nuevas tecnologías inalámbricas que nos permiten mantener la conectividad sin necesidad de cables, cambio mayoritariamente posibilitado por las comunicaciones inalámbricas de corto alcance basadas en radio-frecuencia como Bluetooth [Mor02b, Blu03] o WiFi [GG02, Bre97].

Debido a este auge, la investigación y estudio de nuevas arquitecturas y sistemas basados en tecnologías inalámbricas ha cobrado especial importancia. Los esfuerzos se centran en ayudar a los usuarios a trabajar en este tipo de entornos, y lleva a los investigadores a proponer nuevos dispositivos, tecnologías y arquitecturas para estos contextos de ejecución. Por ejemplo, redes inalámbricas más potentes, rápidas y estables, interfaces gráficos que se adapten a las capacidades de los dispositivos (tamaño de pantalla, almacenamiento, potencia consumida, etc.).

Entornos inalámbricos

Una de las acciones más frecuentes de los usuarios actualmente es conectar su computador o dispositivo a la red para realizar todo tipo de acciones, tales como leer el correo electrónico, obtener un nuevo software que le permita realizar una determinada tarea, buscar una referencia bibliográfica, localizar la farmacia que tenga más próxima utilizando un callejero, etc.

Sin embargo, esta tarea tan habitual plantea problemas importantes cuando la realiza desde un dispositivo inalámbrico. La problemática radica principalmente en la heterogeneidad de los dispositivos inalámbricos existentes. En general, un dispositivo inalámbrico se define como un dispositivo electrónico que puede conectarse a una red inalámbrica, como por ejemplo un PDA (como el mostrado en la Figura 1.2), un ordenador portátil, etc. Esto implica que las características de los distintos dispositivos pueden ser muy dispares. Las principales características de las redes inalámbricas que son utilizadas por este tipo de dispositivos son:

1. Elevado coste económico: los medios de comunicación inalámbricos más importantes (de cobertura global) están gestionados por operadores de telecomunicaciones que imponen tarifas altas por sus servicios. El usuario puede ser facturado según dos unidades: por tiempo (GSM [ETSIE93, Rah93, MPH92]) o por cantidad de datos transferidos (GPRS [And01, ETSIE98] o UMTS [KNL+01, HT04]). Las redes inalámbricas de área amplia[6] como GSM, GPRS y UMTS son de pago. Sin embargo, las redes inalámbricas de área local[7] son instaladas por empresas, administraciones públicas ó usuarios y son gratuitas. Ejemplos de WLAN son las redes Bluetooth [Mor02b, Blu03] o IEEE 802.11 [Bre97, GG02] (ampliamente conocidas como WiFi).

2. Bajo ancho de banda [CM06, ETSIE98]: habitualmente las redes inalámbricas son más lentas que las redes cableadas. Por ejemplo, la velocidad de transmisión de datos en GPRS varía desde 8 kbits/segundo hasta 20 kbits/segundo, en WiFi desde 5.5

[6] *Wireless Wide Area Network (WWAN).*
[7] *Wireless Local Area Network (WLAN).*

Figura 1.2: Asistente Digital Personal o PDA

Mbits/segundo hasta 54 Mbits/segundo. Sin embargo, en las redes cableadas se alcanzan velocidades de 1024 Mbits/segundo. Este hecho se debe a la mayor cantidad de interferencias que sufren las redes inalámbricas debidas al medio de transmisión y a las limitaciones de potencia de emisión que tienen los dispositivos inalámbricos.

3. Inestabilidad [Raa04]: cuando se usan conexiones inalámbricas también debe tenerse en cuenta que la probabilidad de sufrir desconexiones es mucho mayor que en las redes cableadas, debido a la pobre calidad de la transmisión.

Dispositivos inalámbricos

Otro factor determinante que debe ser considerado en este tipo de entornos son las limitadas capacidades de los dispositivos móviles. Las principales restricciones son:

1. Pequeño tamaño y resolución de la pantalla [EVP00]: esta restricción requiere el uso o estudio de tecnologías que puedan adaptar automáticamente la visualización de una aplicación al dispositivo en el que se esté ejecutando.

2. Consumo de energía [Zac03, FS04, VBH03]: los dispositivos móviles tiene una batería que les permite mantenerse en funcionamiento durante un periodo de tiempo muy limitado. Por lo tanto, en el software desarrollado para ser ejecutado en entornos inalámbricos debe tenerse presente evitando la utilización abusiva de la energía disponible en la batería del dispositivo. Por ejemplo, pueden emplearse políticas de desconexión automática de las comunicaciones cuando éstas no estén siendo utilizadas.

3. Bajas prestaciones de los dispositivos inalámbricos [VV00]: un aumento de las prestaciones suele implicar un aumento de la energía consumida por el dispositivo. Esto

implica que en los dispositivos dispositivos inalámbricos puede considerarse la posibilidad de trasladar la ejecución de los programas del dispositivo móvil del usuario a un computador de la red cableada, permitiendo así desconectar incluso el dispositivo móvil, y recuperar los resultados en futuras conexiones.

4. Menor capacidad de almacenamiento [VV00]: en general los dispositivos inalámbricos son más pequeños que los cableados. Esto implica una menor capacidad de almacenamiento de datos en los dispositivos inalámbricos y que el precio por byte del mismo sea mayor.

En consecuencia, las características de los dispositivos inalámbricos y las restricciones impuestas por el entorno inalámbrico requieren el estudio de nuevas arquitecturas que minimicen estos problemas. Comprobaremos además que dichas arquitecturas proporcionarán beneficios a los futuros usuarios de las mismas.

1.2.2. Sistemas multiagente en entornos inalámbricos

A continuación detallaremos qué son y por qué se propone realizar diseños basados en sistemas multiagente (SMA) [Woo02, SZ96, Syc89, DL01, Wei99] y no en un único agente. La tecnología de sistemas multiagente ha generado muchísimas expectativas en los últimos años, principalmente debido a su promesa de un nuevo paradigma que permite una nueva conceptualización, diseño e implementación de sistemas software. Esta promesa es especialmente atractiva en la creación y diseño de software destinado a ser ejecutado en entornos distribuidos, heterogéneos y abiertos, como Internet.

Centrándonos en el diseño y la efectividad de funcionamiento de los SMA, aparece un conjunto de problemas e incógnitas que deberán ser estudiados durante los próximos años por la comunidad científica del área de inteligencia artificial distribuida. Estos estudios deberán basarse en la búsqueda de soluciones a los problemas derivados de las características inherentes de los SMA:

1. Cada agente tiene información incompleta o capacidades para resolver parte de un problema y, por lo tanto, tiene una visión parcial y limitada del mismo.

2. No hay un control global del sistema, sino que cada uno de los agentes controlará una parte del mismo.

3. Los datos están descentralizados. Por lo tanto, cada uno de los agentes que constituye el sistema realizará un procesamiento de la información del depósito de datos al que tiene acceso y posteriormente deberán coordinarse para integrar toda la información disponible.

4. La comunicación entre los distintos agentes se realizará de forma asíncrona. Esta característica facilitará la ejecución en entornos inalámbricos ya que las probabilidades de desconexión entre los distintos nodos es mayor que en una red cableada.

La motivación del creciente interés de la investigación en SMA es debida a varios factores: en primer lugar, permiten resolver problemas que serían demasiado grandes para ser

centralizados en un único agente, de esta forma se evitan la introducción de cuellos de botella y puntos críticos. En segundo lugar, facilitan soluciones a problemas mediante su división y asignación a una sociedad de componentes/agentes que interactúan autónomamente colaborando en la resolución del problema.

1.3. Motivación de los casos de estudio

En esta tesis demostraremos las ventajas de la utilización de la tecnología de agentes móviles inteligentes mediante el análisis de varios contextos contextos de aplicación con diferentes características y requerimientos. A continuación presentaremos los tres casos de estudio considerados y sus respectivas características.

Los casos de estudio seleccionados no se han propuesto por el dominio de aplicación en el que se enmarcan sino por sus características como arquitecturas basadas en agente. En primer lugar, se estudiarán una arquitectura multiagente compleja, que facilité la búsqueda, descarga e instalación de software, donde varios agentes inteligentes colaboran en la realización de una tarea. En segundo lugar, se demostrarán las ventajas de las arquitecturas basada en un único agente, mediante el análisis de una aproximación en el contexto de bibliotecas digitales. En tercer y último lugar, se detallarán varios casos de arquitecturas basadas en agentes que serán creadas dinámicamente dependiendo de los recursos disponibles, permitiendo la adaptación del servicio de datos a la las características de visualización, ubicación o capacidad de almacenamiento del dispositivo del usuario.

1.3.1. Servicio de Recuperación de Software (SRS)

Una de las tareas más frecuentes para los usuarios de ordenadores es obtener nuevo software para sus equipos, que pueda aumentar las capacidades y funcionalidades de sus ordenadores, o dispositivos electrónicos de cualquier otro tipo. No obstante, diferentes clases de usuarios necesitan diferentes tipos de software. Por ejemplo, los usuarios inexpertos pueden estar interesados en programas de entretenimiento, tales como videojuegos y reproductores de DVD. Por el contrario, otros usuarios podrían estar interesados en gestores de bases de datos, procesadores de texto, hojas de cálculo o utilidades de copias de seguridad, entre otros muchos tipos de software. Otras clases de software pueden resultar de interés para un amplio espectro de usuarios, tales como software de antivirus, navegadores Web, etc. Actualmente, el procedimiento más común para obtener dicho software es visitar alguno de los muchos sitios web que contienen software libre y versiones de demostración (Tucows [Tuc06], CNET Shareware.com [CNE06a], CNET Download.com [CNE06b]), juegos (CNET Gamescenter.com [CNE06c]), software relacionado con Java (Gamelan [Ear06]) o muchos otros). Sin embargo, este procedimiento presenta problemas para muchos usuarios, ya que deben:

1. *Conocer los diferentes programas que pueden satisfacer sus necesidades*, no sólo sus nombres, sino también dónde pueden encontrarlos en el inmenso espacio que es la Web. Los sitios web son movidos y rediseñados haciendo que los enlaces a dichos sitios se queden obsoletos muy frecuentemente.

2. *Conocer las características de sus dispositivos*, para poder así seleccionar la versión más apropiada del software que necesitan. Esta tarea implica tener un conocimiento técnico acerca de su sistema (CPU, sistema operativo y versión, espacio de almacenamiento disponible, memoria RAM, etc), y del software previamente instalado (para poder optar por obtener una versión completa o simplemente una actualización).

3. *Conocer el nuevo software y las nuevas versiones del software de su interés*. Aunque algunos programas comerciales alertan a los usuarios acerca de nuevas versiones, los usuarios necesitan estar atentos a la web si quieren ser informados acerca de nuevos tipos de software que podrían resultar de su interés.

Todos estos problemas se hacen incluso más evidentes e importantes en el caso de los usuarios de dispositivos inalámbricos [FDL07] porque muchos de ellos son inexpertos (por ejemplo, se está popularizando el uso de PDAs entre los profesionales no especializados en informática) y por lo tanto su conocimiento acerca del software es limitado, debiendo ser ayudados en el proceso de búsqueda por el sistema (los agentes). Además, el uso de medios inalámbricos [VV00] debería ser minimizado durante el proceso de selección del software debido a su alto coste e inestabilidad [Raa04]. Además, los sistemas tipo *Tucows* muestran la misma información a todos los usuarios y, por lo tanto, no tratan de minimizar el tamaño de los datos transmitidos.

Por consiguiente, en esta tesis verificaremos mediante este caso de estudio las ventajas que puede proporcionar la utilización de la tecnología de agentes en este contexto, diseñando un procedimiento alternativo para la búsqueda, descarga, instalación y actualización de software para el dispositivo del usuario. El sistema propuesto se denomina *Servicio de Recuperación de Software (SRS)*, y estará basado en el uso de un catálogo de software y de la tecnología de agentes, lo que permitirá a los usuarios encontrar y descargar software de forma: 1) fácil: porque los agentes harán transparente para los usuarios las características técnicas de sus dispositivos, su localización, y el método de acceso a distintos almacenes de software remotos; 2) eficiente: porque los agentes [MBB+98] optimizarán el uso de los medios inalámbricos personalizando los catálogos de software que el usuario visualizará; y 3) adaptable: porque los agentes tendrán en cuenta el estado de la red, podrán predecir el comportamiento del usuario, y aprenderán de sus propios errores.

1.3.2. Bibliotecas digitales

Una biblioteca digital es una colección de información que es almacenada y accedida electrónicamente [RW98]. La información almacenada en la biblioteca puede tener un tema común a todos los datos, por ejemplo en la biblioteca digital con las publicaciones y referencias de un grupo de investigación, o contener información sobre distintos temas [BSBK01]. También hay que destacar que se pueden combinar distintas bibliotecas digitales [PCWGM98] para ser accedidas mediante un interfaz común. El propósito de una biblioteca digital es proporcionar una gestión centralizada para el acceso a la información de un tema particular.

El número de sistemas de búsqueda y recuperación de información [Sch97] aumenta cada día, aunque la mayor parte de las aproximaciones existentes siguen técnicas basadas en la arquitectura cliente/servidor [FFF+00]. Sin embargo, uno de los inconvenientes de estas

arquitecturas es su dependencia intrínseca de la fiabilidad de la red, característica con la que no se puede contar en el caso de las redes inalámbricas. En esta tesis hemos seleccionado el caso de una biblioteca digital, para estudiar y mostrar las ventajas de las arquitecturas basadas en agentes móviles inteligentes, sin embargo los resultados pueden extrapolarse a cualquier otro ámbito. La aproximación diseñada y desarrollada se adaptará automáticamente a contextos de ejecución cableados e inalámbricos debido a que se fundamentará en una arquitectura basada en agentes móviles inteligentes.

Concretamente, la aproximación propuesta en esta tesis, se basará en el uso de agentes móviles en el contexto concreto de bibliotecas digitales destinadas a recoger y gestionar las publicaciones de un grupo de investigación. Este caso, permitirá analizar diferentes depósitos de datos [Pal07a], y además nos permitirá estudiar como facilitar el mantenimiento de la consistencia de sistemas de publicaciones distribuidos. Del mismo modo, tendrá la ventaja de optimizar el uso de la red, gracias a la utilización de la tecnología de agentes móviles, y será robusta frente a las desconexiones del cliente, característica también frecuente en los entornos inalámbricos. Además, la técnica de detección de inconsistencias utilizado por los agentes móviles, adaptará el mecanismo de comparación de referencias bibliográficas al contenido de los almacenes de datos.

Debido al contexto en el que se encuentran las bibliotecas digitales de publicaciones de investigación, el sistema propuesto no se limitará a la introducción de las referencias bibliográficas cuando se crea el almacén de bibliografía, sino que las referencias podrán sufrir constantes cambios (debido tanto a la actualización de referencias existentes, como a la inserción de nuevas referencias). Se estudiarán distintos métodos para realizar estas tareas, siempre facilitando que las distintas tareas puedan ser realizadas por agentes autónomos o por usuarios inexpertos. Posteriormente mostraremos que esta característica se verá simplificada tanto por el uso de agentes inteligentes como de servicios web [JC02, STK01].

Finalmente, conviene resaltar que con la biblioteca digital que se va a diseñar utilizando la tecnología de agentes, se demostrarán las ventajas proporcionadas por los sistemas multiagente en entornos de acceso a datos heterogéneos, distribuidos e inalámbricos. Las principales ventajas serán: adaptabilidad al contexto de ejecución y al tipo de depósitos de datos, reducción del tráfico de red y robustez frente a redes inestables, inalámbricas o cableadas. Además, se demostrará la capacidad inherente de los agentes para tomar decisiones por el usuario de forma autónoma, en este caso mediante la detección inconsistencias en una base de datos bibliográfica.

1.3.3. Servicios de acceso a datos en entornos de ejecución dinámicos

La proliferación de dispositivos de computación móvil y de redes de área local inalámbricas [And01] están desencadenando un creciente interés en el desarrollo de servicios y sistemas que permitan su despliegue de forma dinámica, minimicen el consumo de recursos, e incrementen la robustez de las aproximaciones tradicionales frente a desconexiones. Por todo ello, se busca que el diseño de servicios de datos para estos contextos de ejecución permita: 1) su adaptación automática a las características del dispositivo del usuario (por ejemplo, mediante la utilización de interfaces de usuario que se adapten a las características de visualización impuestas por el dispositivo); 2) su personalización dependiendo de la ubicación del

usuario (tal como se hace en aplicaciones basadas en la localización); y 3) un filtrado inteligente basado en la semántica de la información visualizada (recurriendo al descubrimiento automatizado del significado de las preguntas formuladas por el usuario mediante un conjunto de palabras clave). Conviene destacar que todas estas características son imprescindibles en el diseño de sistemas inalámbricos donde los recursos disponibles en cada momento sufren frecuentes alteraciones, o sufren fluctuaciones dependiendo tanto de las características del dispositivo del usuario como de la ubicación en la que se encuentre. En capítulos posteriores se demostrará como el paradigma de los agentes móviles facilitará la adaptación de los servicios de datos a contextos dinámicos y cambiantes como son los entornos de ejecución inalámbricos.

En primer lugar, se demostrarán las ventajas que puede proporcionar una arquitectura basada en agentes móviles e inteligentes, para adaptar el interfaz gráfico de usuario a las características del dispositivo en el que se esté ejecutando. Debido a que los distintos dispositivos móviles tienen capacidades de procesamiento, almacenamiento y visualización muy dispares, será necesario prestar especial cuidado en el diseño del GUI de aplicaciones para estos dispositivos. De esta forma, para cada dispositivo utilizado para acceder a un servicio de acceso a datos generarán distintos GUIs, desde un GUI basado en WML a uno que utilice Java Swing. Actualmente se diseña un interfaz gráfico de usuario para cada dispositivo, lo que supone un elevado coste económico y de personal en el desarrollo de aplicaciones para este tipo de dispositivos. Con la aproximación propuesta en esta tesis, mediante una especificación del GUI para el servicio, un agente especializado podrá modificar en tiempo de ejecución el interfaz gráfico generado para ser mostrado al usuario, adaptándolo a las características del dispositivo en el que se esté ejecutando el servicio de acceso a datos concreto. Esto permitirá reducir el tiempo de diseño y desarrollo de los servicios, así como el número de desarrolladores implicados en el proceso de creación.

En segundo lugar, estudiaremos las ventajas que facilitará el diseño de aplicaciones basadas en agentes para adaptar los servicios de acceso a datos a la ubicación en la que se encuentre el usuario en un momento concreto. No obstante, comprobaremos que para facilitar el diseño de servicios basados en la localización primeramente habrá que proporcionar una infraestructura que permita calcular la posición en la que se encuentra el usuario. El mecanismo utilizado para calcular la posición de un dispositivo móvil dependerá de la tecnología de comunicaciones utilizada. En este contexto nos centraremos en estudiar y diseñar una arquitectura basada en agentes que permitirá la localización de dispositivos móviles que empleen conexión WiFi (802.11b/g) [Bre97, GG02] en el interior de edificios. Conviene destacar que los sistemas GPS no pueden ser utilizados dentro de edificios, porque para localizar un dispositivo al menos deben estar accesibles —visualmente— cuatro satélites. Las tecnologías de comunicaciones de área amplia, como GSM [Rah93, MPH92] o GPRS [And01, ETSIE98], también pueden proveer a algunos usuarios con cierta información de localización, pero normalmente solo pueden aportar la identificación de la celda de cobertura inalámbrica en la que se encuentra el usuario. Por consiguiente, la arquitectura propuesta en esta tesis ofrecerá funcionalidades que los actuales sistemas de posicionamiento global no pueden aún ofrecer. Como se detallará en capítulos posteriores, la aproximación propuesta para el diseño de servicios basados en la localización, constará de dos fases: 1) *Fase de creación de los mapas de potencia* [BP00, NDA01]: permitirá establecer la potencia con la que se recibe la señal

emitida por un dispositivo, desde las distintas estaciones base que le dan cobertura, en las distintas posiciones geográficas; y 2) *Fase de localización*: utilizará los mapas de potencia construidos en la fase anterior para localizar el dispositivo del usuario.

En tercer lugar, se estudiarán las ventajas que pueden obtenerse de la aplicación de sistemas multiagente al diseño de sistemas de información globales. Para verificar las ventajas del uso de agentes en este contexto, se diseñará un servicio enmarcado dentro del contexto de la Web Semántica. Como se detallará en capítulos posteriores, los agentes inteligentes de la arquitectura del servicio facilitarán el filtrado de la información mostrada al usuario, siguiendo un proceso que tomará como parámetro de entrada un conjunto de palabras clave. Posteriormente, un conjunto de agentes inteligentes desambiguarán el conjunto de palabras clave dado, y devolverán al usuario los datos relacionados semánticamente con su pregunta que se encuentren disponibles en la Web, no los relacionados sintácticamente, tal y como hacen los buscadores actuales [Goo06]. Igualmente, se mostrará como, gracias al uso de los agentes, la arquitectura propuesta podrá adaptarse a fluctuaciones en la cantidad de recursos disponibles en cada momento, y a las prestaciones de la conexión de red utilizada por el usuario. Es decir, gracias a la capacidad de movimiento de los agentes, la arquitectura propuesta podrá trasladar los procesos de cálculo al lugar más idóneo, todo ello en tiempo de ejecución, evitando así cuellos de botella en el servicio.

1.4. Estructura de la tesis

A continuación vamos a resumir los temas tratados en cada uno de los restantes capítulos de esta tesis.

En el capítulo 2 se describen los distintos tipos de redes de comunicaciones usados en entornos inalámbricos, tanto de área global como de área local. A continuación, se detallan las distintas aproximaciones utilizadas para diseñar arquitecturas software en contextos inalámbricos y distribuidos, haciendo especial hincapié en los mecanismos utilizados por los agentes para especificar y gestionar el conocimiento asociado a su inteligencia y al acceso de depósitos de datos.

En el capítulo 3 se presenta qué son los agentes software y cuáles son las propiedades que poseen. Posteriormente se realiza una comparación teórica entre las arquitecturas cliente/servidor y las arquitecturas basadas en agentes inteligentes, mostrando las ventajas e inconvenientes que presentan en el diseño de servicios de datos en entornos distribuidos e inalámbricos. A continuación, se detallan las características de las plataformas de agentes móviles más utilizadas.

En el capítulo 4 se describe el caso de estudio del Servicio de Recuperación de Software (SRS) para usuarios inexpertos basado en la tecnología de agentes inteligentes dirigidos por el conocimiento. En este capítulo se detalla cómo los agentes pueden adaptar automáticamente los catálogos de software mostrados al usuario dependiendo del comportamiento del mismo, del estado de la red, y del software buscado por el usuario.

En el capítulo 5 presentaremos cómo, dentro del SRS, dos agentes inteligentes construyen una ontología de software (SoftOnt) de forma automática y autónoma, que será utilizada por el sistema para almacenar la descripción semántica de los almacenes de software.

En el capítulo 6 se muestran las ventajas de la utilización de la tecnología de agentes en el contexto de las Bibliotecas Digitales. La tecnología de agentes en este contexto ayuda a solucionar problemas de integración de distintos depósitos bibliográficos con información inconsistente de forma semiautomática. La solución a los problemas anteriores se consigue gracias a las propiedades de movilidad, autonomía e inteligencia de los agentes.

En el capítulo 7 se describirán otros servicios inalámbricos de acceso a datos que despliegan dinámicamente sus arquitecturas basadas en agentes en tiempo de ejecución, permitiendo minimizar los recursos consumidos y aumentar la robustez del servicio frente a cambios en el contexto de ejecución. Los tres servicios estudiados y diseñados en este capítulo son: 1) la generación indirecta de interfaces de usuario; 2) el diseño y desarrollo de servicios basados en la localización; y 3)la reducción de la información mostrada al usuario en un proceso de búsqueda guiado por la semántica. El diseño de todos estos servicios se fundamenta en un conjunto de agentes especializados: en la generación de GUIs, en el cálculo de la posición del usuario (para su posterior reutilización), y en el tratamiento de palabras clave introducidas por el usuario (para generar una pregunta no ambigua expresada en Lógica Descriptiva).

En el capítulo 8 se presenta el estado del arte de la investigación en sistemas distribuidos en entornos inalámbricos y el uso de agentes en dichos sistemas. Destacando las ventajas e inconvenientes de las arquitecturas basadas en agentes inteligentes frente a otras aproximaciones para el diseño de servicios de acceso a datos, centrándonos especialmente en los casos de estudio diseñados en esta tesis.

Finalmente, en el capítulo 9 se presentarán las conclusiones del trabajo desarrollado y las posibles líneas de ampliación.

Capítulo 2

Contexto tecnológico

En este capítulo se presentarán los aspectos tecnológicos que nos permitirán posteriormente desarrollar en profundidad las ventajas proporcionadas por los sistemas multiagente en entornos inalámbricos y distribuidos. Primeramente, se detallarán cuestiones relativas a los protocolos de red utilizados en entornos inalámbricos, tanto en redes de área global (GSM [ETSIE93], GPRS [ETSIE98] y/o UMTS [HT04]) como en redes de área local (Bluetooth [Mor02b], WiFi [GG02], WIMAX [Ohr05] y HomeRF [Pal07b]. Nos centraremos en la arquitectura y características que influyen en el desarrollo de servicios de acceso a datos para cada uno de los tipos de red inalámbrica descrita .

A continuación, describiremos los distintos tipos de arquitecturas software que se utilizan en contextos distribuidos para el diseño e implementación de servicios de acceso a datos y las aproximaciones tecnológicas disponibles para su desarrollo. Posteriormente detallaremos las distintas aproximaciones que permitirán ejecutar una aplicación en un cliente que accede a un servicio remoto sin necesidad de instalarse localmente, evitando el consumo de recursos. Esta aproximación es muy interesante en los entornos inalámbricos debido a que los dispositivos móviles tienen un espacio de almacenamiento mucho más limitado que los equipos utilizados en las redes cableadas.

Finalmente, detallaremos las distintas aproximaciones que permiten la descripción semántica de depósitos de datos heterogéneos y distribuidos, permitiendo el diseño de sistemas de información distribuidos que pueden ser accedidos tanto por seres humanos como por agentes inteligentes, haciendo posible el análisis automático de los datos.

2.1. Tecnologías de comunicaciones inalámbricas de largo alcance

En esta sección nos vamos a centrar, en las redes móviles celulares de largo alcance, que han sido instaladas por compañías de telefonía debido al elevado coste de instalación. Debido a esto, algunas compañías de telefonía decidieron el protocolo de comunicaciones que se puede utilizar en sus redes provocando una gran heterogeneidad en los protocolos de comu-

nicaciones utilizados, dificultando la interoperabilidad entre las mismas. Como mostramos a continuación, en una etapa posterior del diseño de los protocolos de comunicaciones en redes inalámbricas de largo alcance se procedió a estandarizar y homogeneizar los protocolos utilizados por la redes celulares permitiendo una mayor movilidad de los usuarios.

En Europa la primera red inalámbrica digital basada en radio frecuencia de largo alcance que se desplegó fue GSM (*Global System for Mobile communications*) [ETSIE93] que proporciona velocidades de acceso de 1 Kbs. En otras zonas del planeta se utilizan otros protocolos de comunicaciones, por ejemplo, iDEN [FDL08d], red propietaria utilizada en Estados Unidos y Canadá, en América del Norte también se utilizan otros protocolos como D-AMPS [FDL08c] o cdmaOne [FDL08e] (también es utilizado en algunas partes de Asia), y PDC [FDL08g], usado exclusivamente en Japón.

Posteriormente, se implantó una red GPRS (*General Packet Radio Service*) [ETSIE98] de acceso a IP móvil[1] sobre la infraestructura de la red GSM. Últimamente, se ha producido el despliegue de la red de tercera generación denominada UMTS (*Universal Mobile Telecommunication System*) [KNL+01] que proporciona servicios de acceso a datos de banda ancha de forma inalámbrica.

Este tipo de redes de comunicaciones permite acceder a servicios de datos en cualquier momento y lugar. Es decir, el usuario puede conectarse a Internet desde un dispositivo que puede cambiar de lugar al mismo tiempo que sigue conectado, por ejemplo permitiendo proporcionándole servicios de localización. Sin embargo, las redes inalámbricas de largo alcance tienen las siguientes desventajas: un el elevado coste de transmisión de datos y son más inestables y lentas que las redes cableadas.

2.1.1. GSM

A principios de los ochenta en Europa había nueve sistemas de comunicación inalámbrica por lo que comenzó un proceso de estandarización que proporcionó las bases para la definición de GSM [ETSIE93, Rah93, MPH92], su arquitectura puede verse en la Figura 2.1. En esta arquitectura, la estación móvil compuesta por componentes móviles (habitualmente, un teléfono móvil) y un módulo de identificación del cliente[2] (SIM). Además, todos los usuarios almacenan la información del dispositivo móvil en la SIM (o una parte de la información), como por ejemplo, una clave secreta para autenticarse y buscar los datos almacenados en la agenda. Esto facilita que los usuarios puedan cambiar de dispositivo móvil cambiando la tarjeta SIM de uno a otro.

Por lo tanto, debido a que los usuarios puede cambiar el dispositivo que utilizan fácilmente, las nuevas arquitecturas software que se diseñen para este tipo de entornos deben adaptarse fácilmente a las características (sistema operativo, capacidades de procesamiento disponibles, etc.) de cualquier dispositivo.

Del mismo modo, las arquitectura GSM proporciona al dispositivo móvil una comunicación directa con las antenas GSM a través de un interfaz inalámbrico, estas antenas proveen con cobertura a los dispositivos móviles, pero únicamente proporcionan la capa física de acceso.

[1] La red GPRS tiene un ancho de banda de entre 10 Kb/s y 50 Kb/s.
[2] *System Identity Module.*

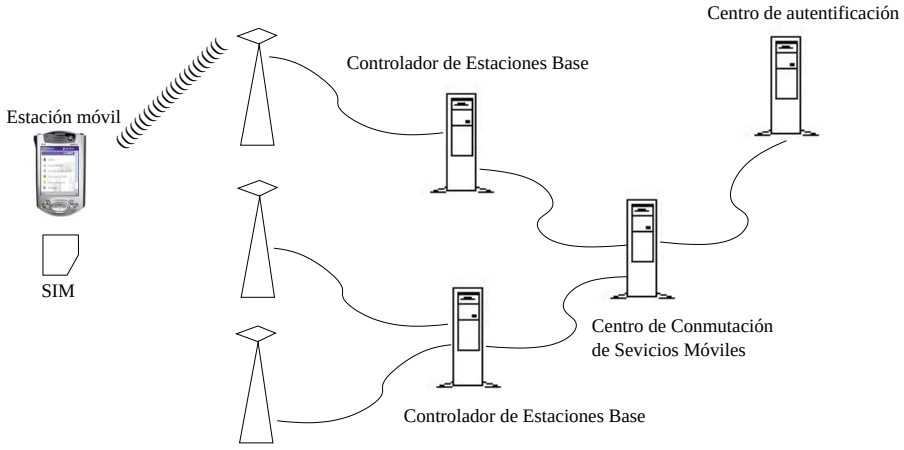


Figura 2.1: Arquitectura de una red GSM

El Centro de conmutación de servicios móviles (*Mobile Services Switching Center (MSC)*) proporciona los servicios que permiten la movilidad de los usuarios. Es decir, permite que los usuarios accedan al sistema y actúa como puerta de enlace para los mismos conectándolos con la red cableada. Este componente pide continuamente al dispositivo información para que se autentique y la compara con la almacenada en el centro de autenticación, para verificar si un dispositivo y usuario puede acceder a la red. Además, para saber que dispositivos y usuarios se encuentran bajo el área de cobertura de un MSC, se almacena qué usuarios hay bajo su área de cobertura y en qué posición se encuentran. Por consiguiente, cuando un usuario cambia de área de cobertura la información almacenada en el MSC debe ser actualizada.

Por consiguiente, el protocolo de comunicaciones hace transparente a los usuarios y agentes el cambio de la estación base que le proporciona cobertura. Por ejemplo, proporcionando los servicios de encaminamiento necesarios a los agentes que viajen desde el dispositivo móvil del usuario a la red fija o en sentido inverso, haciendo disponibles estos servicios independientemente de la estación base que dé cobertura al usuario.

También hay que considerar que hay tres bandas principales definidas para las comunicaciones GSM, 900 MHz, 1800 MHz y 1900 MHz, la primera de las mismas es la más ampliamente instalada, la segunda ha sido adoptada en Europa y la tercera en Estados Unidos. Debido a esto, los dispositivos móviles deberán encapsular las diferentes frecuencias utilizadas en GSM para que un servicio de datos pueda ser accedido desde redes GSM que utilicen distinta frecuencia, o un agente que fue creado en una red GSM con una frecuencia, pueda seguir proporcionando sus servicios a un usuario en una red GSM que utilice otra frecuencia.

Del mismo modo cada una de estas tres bandas se divide en distintos canales, que serán utilizados por los dispositivos a los que les da cobertura una estación base produciéndose interferencias en las comunicaciones y provocando errores de red. Por consiguiente, las aplicaciones desarrolladas para proporcionar servicios de datos en entornos inalámbricos deberán considerar la inestabilidad de la red.

2.1.2. GPRS

La introducción de GPRS (*General Packet Radio Service*) [ETSIE98, And01] facilita el comienzo de la transmisión de datos por conmutación de paquetes, permitiendo a los usuarios la transmisión de datos de forma inalámbrica. Es decir, esta tecnología no solamente permitirá la transmisión de voz sino también el envío y recepción de fax, la utilización del correo electrónico o aplicaciones basadas en internet, en definitiva, el acceso a servicios de datos inalámbricos.

Debido a que GPRS se basa en una tecnología de conmutación de paquetes, el usuario será facturado por la cantidad de información que envíe a través de la conexión inalámbrica y no por el tiempo que esté conectado, tal como se hace en GSM. Es decir, el usuario puede tener conectado su dispositivo móvil a la red de forma continuada pero solamente tendrá un coste para él cuando envíe o reciba información. Debido a esto GPRS ha sido diseñado para ser compatible con X.25 y con las redes basadas en IP [Com00]. Por lo tanto, un dispositivo móvil en una red GPRS tendrá una dirección IP que le identificará dentro de dicha red y, por consiguiente, programas de correo electrónico o navegadores web podrán ser utilizados sin la necesidad de intermediarios que realicen el encaminamiento de los datos[3].

Para soportar este tipo de servicios se modifica y amplia la arquitectura de GSM, tal y como puede verse en la Figura 2.2. A la arquitectura de GSM se añade el nodo que da soporte a servicios GPRS (*Serving GPRS Support Node*), y la puerta de enlace GPRS que permite encaminar el tráfico GPRS hacia una red externa.

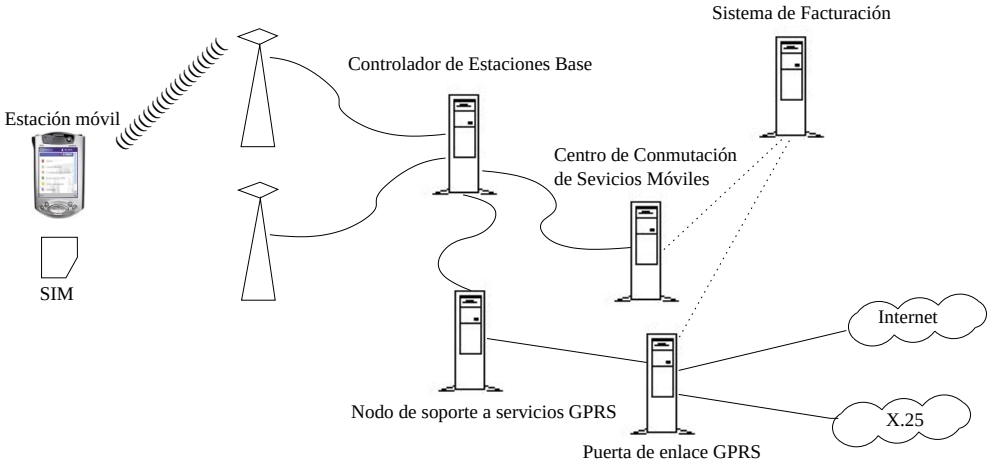


Figura 2.2: Arquitectura de una red GPRS

Una de las ventajas de GPRS es que puede conseguir tasas de transferencia de 14.4 Kbps utilizando un único canal de la red GSM. Sin embargo, como puede multiplexarse la utilización de hasta 8 canales de comunicación pueden alcanzarse tasas de 115 Kbps, dando lugar a una red inalámbrica que es más rápida que las redes cableadas basadas módems analógicos.

[3] A diferencia de la aproximación seguida en las redes GSM donde los dispositivos se identificaban mediante un SIM y el encaminamiento de datos lo realizaba el centro de conmutación de servicios móviles.

Las dos principales ventajas de GPRS para el diseño y desarrollo de servicios de acceso a datos son: 1) el usuario puede estar conectado el tiempo que desee sin que se le facture más siempre que no transmita datos utilizando el enlace inalámbrico, se factura por volumen de datos transmitidos y no por tiempo de conexión, 2) se aumenta la velocidad de la red, aumentando el espectro de servicios que pueden ser proporcionados a los usuarios. Del mismo modo, las arquitecturas basadas en agentes son una aproximación perfecta para este contexto debido a que permiten reducir el tráfico de red y son robustas frente a desconexiones.

Finalmente destacar la importancia de GPRS para los proveedores de servicios inalámbricos, debido a que es un paso entre las comunicaciones inalámbricas de segunda generación (2G), como es el protocolo GSM, y las de tercera generación (3G) como UMTS [KNL+01, HT04]. Por esta razón GPRS es conocido como generación 2.5G.

2.1.3. UMTS

Las sucesivas definiciones que dieron lugar al estándar UMTS (*Universal Mobile Telecommunication System*) [KNL+01, HT04] ilustran la evolución que está teniendo lugar en las arquitecturas de transporte de voz y datos. UMTS utiliza una red de acceso denominada UTRAN (*UMTS Terrestrial Radio Access Network*) para dar conectividad desde los teléfonos móviles al núcleo de la red. Esta conectividad se da a través de UTRAN, tanto para comunicaciones de voz como de datos. En el diseño inicial de UMTS, el núcleo de la red tiene dos dominios separados: conmutación de circuitos para voz y conmutación de paquetes para datos. Las versiones más recientes que se han desplegado, han extendido el uso del protocolo IP en el núcleo de la red y en la UTRAN. Así la versión 4 de UMTS realizada por el 3GPP (*3rd Generation Partnership Project*) permite sustituir la parte de conmutación de circuitos por una arquitectura de voz sobre IP.

UMTS permite velocidades de transmisión de hasta 1920 kbits/s (y no 2 Mbits/s como se dice habitualmente, incrementando la velocidad del protocolo en un 6.25 %), aunque actualmente los usuarios solamente pueden utilizar ratios de transferencia de hasta 384 kbits/s (en Japón se está experimentando con ratios de hasta 3 Mbits/s). Sin embargo, los ratios de transferencia actuales están muy por encima de las comunicaciones GSM. Por otra parte, desde el comienzo de 2006, las redes UMTS de Japón están siendo actualizadas con HSDPA (*High Speed Downlink Packet Access*) [HT06] que alcanza ratios de transferencia de hasta 14.4 Mbits/s para descargas; este estándar está siendo conocido como la generación 3.5G. Actualmente se está trabajando en mejorar los ratios de subida (es decir, las comunicaciones desde el dispositivo móvil a la estación base que le da cobertura).

Hasta el momento actual hemos descrito las tasas de transferencia que se pueden obtener mediante la utilización de UMTS, pero no hemos detallado qué servicios nos proporciona. Una de las ventajas que proporciona UMTS es que permite distintas calidades de servicio dependiendo de cuál sea la finalidad de la comunicación. Esto da lugar a los siguientes tipos de tráfico:

1. Tráfico conversacional: este tipo de tráfico facilita el diseño de servicios para voz, telefonía y videojuegos.

2. Tráfico de flujos de datos (*Streaming*): permitiendo el desarrollo de servicios multimedia o video bajo demanda.

3. Tráfico interactivo: dando lugar a servicios de navegación, juegos en red, acceso a bases de datos, etc.

4. Tráfico en segundo plano: proporcionando la infraestructura necesaria a servicios que utilizan el ancho de banda de forma puntual pero que necesitan que la red esté accesible en todo momento. Por ejemplo, servicios de correo electrónico, mensajes cortos o SMS, descargas de software, etc.

Como se ha visto hemos detallado los tipos de tráfico más relevantes y algunos de los servicios que pueden estar asociados a los mismos, del mismo modo hay que destacar que son muchos más los servicios accesibles y disponibles. Todo este tipo de servicios deben diseñarse mediante un paradigma de computación basado en agentes móviles por las siguientes razones: 1) tienen una naturaleza inherentemente distribuida, como las arquitecturas basadas en agentes; y 2) están basados en la utilización de una red inalámbrica que es inestable y cara, por lo tanto el diseño de los servicios de datos proporcionados debe minimizar la utilización del enlace inalámbrico y ser robusta a posibles errores, factores inherentes de las arquitecturas basadas en agentes inteligentes.

2.2. Tecnologías de comunicaciones inalámbricas de corto alcance

En este apartado vamos a comentar brevemente las características de varios protocolos de comunicaciones inalámbricas de corto alcance, basados en radiofrecuencia, existentes y ampliamente utilizados en la actualidad, como son Bluetooth [Mor02b], WiFi [Bre97] y Home RF [Pal07b]. En la tabla 2.1 se muestra un resumen de las características de las redes que vamos a describir en las siguientes subsecciones. Indicando si cada uno de los protocolos de comunicaciones puede utilizar cifrado, cuál es la distancia máxima a la que pueden transmitir, su arquitectura de red y el tipo de redes que se forman —redes de área local (LAN) y redes personales (PAN[4])—.

Protocolo	**Cifrado**	**Distancia**	**Arquitectura**	**Utilización**
Bluetooth	Siempre	150m.	maestro/esclavo	PAN
WiFi	Opcional	100m	puntos de acceso	LAN
WIMAX	Opcional	5 km.	puntos de acceso	LAN
HomeRF	No	50m.	cliente/servidor o *peer-to-peer*	PAN

Tabla 2.1: Tecnologías de comunicaciones inalámbricas de corto alcance

[4]Los acrónimos LAN y PAN están asociados a los vocablos ingleses que identifican: *Local Area Network* y *Personal Area Network*, respectivamente.

Las dos alternativas tecnológicas de comunicaciones inalámbricas basadas en radiofrecuencia más ampliamente utilizadas son: Bluetooth y WiFi.

2.2.1. Bluetooth

Bluetooth [Mor02b] es un protocolo de comunicaciones inalámbrico basado en radio frecuencia con tasas de transferencia de hasta 721 kbps y tiene un alcance que varía entre 10 y 150 metros. Está diseñado para consumir la menor potencia posible debido a que ajusta la misma dependiendo de la distancia entre emisor y receptor. Además proporciona servicios de alto nivel para descubrimiento de dispositivos Bluetooth activos.

Bluetooth se considera el protocolo más indicado[5] para comunicaciones de muy corto alcance y además proporciona servicios de descubrimiento de dispositivos a los que conectarse. La capacidad de descubrimiento de dispositivos permite facilitar el desarrollo de servicios orientados a usuarios, debido a que la configuración de la topología de la red es gestionada automáticamente. Existen dos variantes posibles, para las configuraciones maestro o esclavo:

- Las *piconets*: en las que hay un nodo maestro y el resto son esclavos (Ver Figura 2.3.a).
- Las *scatternets*: formadas por la superposición de las áreas de cobertura de varias piconets (Ver Figura 2.3.b).

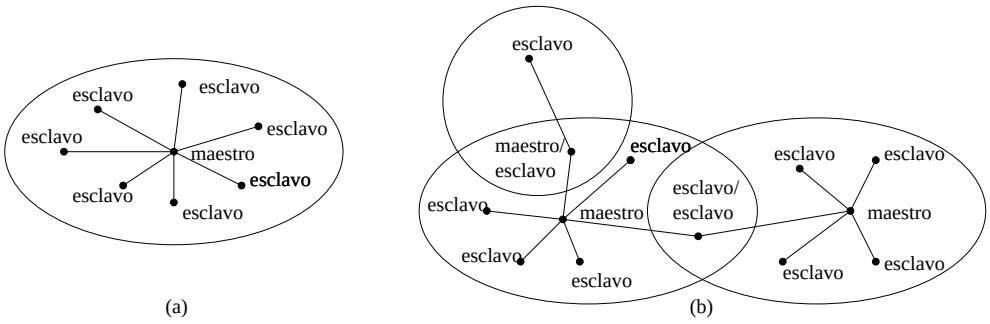


Figura 2.3: Topologías en redes Bluetooth: (a) piconet y (b) scatternet

Bluetooth está definido a modo de pila, similar al esquema ISO/OSI. Como se muestra en la Figura 2.4, esta pila queda dividida por la interfaz HCI en dos partes: la parte inferior que habitualmente es incorporada por hardware dentro de los dispositivos Bluetooth y la parte superior que se implementa por software. En algunos dispositivos empotrados, la parte superior puede estar implementada dentro del propio chip, pero no es lo habitual.
La funcionalidad de cada uno de los niveles que permiten la creación de una conexión TCP-IP es la siguiente[6]:

[5] Sustituyendo a las comunicaciones basadas en infrarrojos por su robustez y seguridad.

[6] No se definen el resto de niveles debido a que las plataformas de agentes están basadas en el protocolo de comunicaciones TCP-IP y no pueden utilizar el resto de servicios de la pila Bluetooth quedando fuera del ámbito de esta tesis.

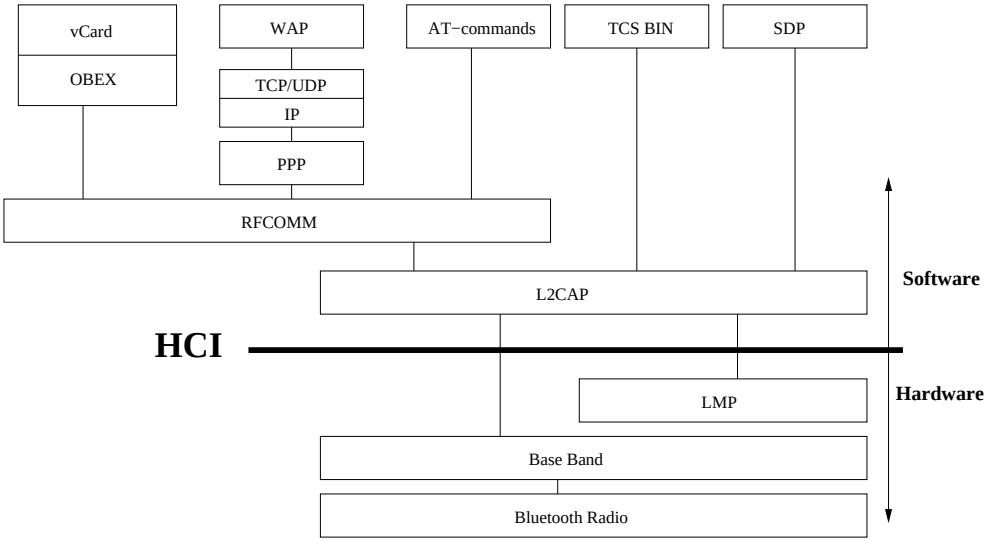


Figura 2.4: Pila Bluetooth dividida por el interfaz HCI

1. *Bluetooth radio*: es el nivel encargado de la emisión-recepción de la señal de radio. Trabaja en el rango de frecuencias de la banda ISM (Médico-Científica Internacional) en 2.45Ghz.

2. *Baseband*: es el encargado de la sincronización, control de flujo, transmisión de la información, corrección de errores, división lógica de canales y seguridad. Permite la formación de dos tipos de conexiones: 1) asíncronas (ACL[7]): utiliza conmutación de paquetes como protocolo de comunicaciones y se usan para la transmisión de datos; y 2) síncronas (SCO[8]): el protocolo de comunicaciones utilizado es por conmutación de circuitos.

3. *Protocolo de control de enlace (LMP):* este nivel traduce los comandos de niveles superiores a los niveles inferiores de la pila.

4. *Host Controller Interface (HCI):* es la interfaz que une el dispositivo Bluetooth con el PDA, ordenador, o teléfono móvil.

5. *Link Layer Control and Adaptation Layer Protocol (L2CAP)*: esta capa toma los datos de los niveles superiores y se los pasa a las capas inferiores. De esta forma permite: 1) multiplexación de varias capas superiores (incluso protocolos diferentes) sobre un mismo enlace, 2) segmentación y ensamblaje de paquetes de gran tamaño en paquetes del tamaño de las capas inferiores y 3) calidad de servicio (QoS) para capas superiores.

[7] *Asychronous Connectionless*.
[8] *Synchronous Connection-Oriented.*

6. *Service Discovery Protocol (SDP)*: protocolo para descubrir qué servicios tienen disponibles los dispositivos Bluetooth próximos.

7. *RFCOMM*: es un protocolo que implementa una emulación de puertos serie RS232 sobre un canal L2CAP. Permite configurar una conexión punto a punto (PPP) sobre la que se establecerá un enlace TCP-IP.

En esta sección se han definido las funcionalidades que ofertan cada una de las capas de la pila Bluetooth, una implementación de la misma para Linux es Bluez [Blu03]. Su meta es hacer disponible una implementación de las especificaciones del estándar inalámbrico Bluetooth para Linux. Desde 2006 Bluez proporciona soporte a todas las capas del protocolo Bluetooth. Fue desarrollado inicialmente por *Qualcomm* [Que08], y esta disponible en el kernel de Linux en la versión 2.4.6 y posteriores. En los siguientes apartados vamos a definir como se puede establecer un enlace TCP-IP sobre un dispositivo Bluetooth.

TCP-IP sobre Bluetooth

En las secciones previas hemos descrito Bluetooth como una pila de protocolos de comunicaciones inalámbricas, permitiendo únicamente la conexión de un dispositivo móvil con otros. Sin embargo, para diseñar una arquitectura basada en agentes que utilice una red Bluetooth debemos integrar esos dispositivos móviles con las redes ya existentes, sean fijas o móviles. Para ello tenemos que utilizar un protocolo estándar a todas estas redes (TCP-IP) que se ejecute sobre la pila que ya tenemos definida. La utilización de TCP-IP le aporta grandes ventajas a los dispositivos móviles, debido a que permite acceder a cualquier red del mundo a través de su enlace inalámbrico.

A continuación vamos a describir las ventajas proporcionadas por la interconexión de una red Bluetooth con una red TCP-IP. Las características más importantes de estas redes son:

- Utilizan TCP-IP estándar, no una versión modificada como 802.11, con lo cual se integra perfectamente en las redes ya existentes. Permitiendo el diseño y desarrollo de arquitecturas basadas en agentes que utilizan la infraestructura de las redes Bluetooth del mismo modo que si fuese cualquier otra red TCP-IP.

- Permite la formación de redes ad hoc. Estas redes se crean y se destruyen dinámicamente y no tienen necesidad de ninguna infraestructura previa, en modo de estaciones base, para su creación. Bastan dos dispositivos móviles que se conecten entre sí para su formación. Esto resulta muy útil para transferir datos entre un dispositivo y otro cuando se encuentran en el mismo área de cobertura.

- Las redes creadas pueden ser completamente heterogéneas. La compatibilidad entre distintas máquinas y distintos sistemas operativos es total.

Hay cuatro modos principales de configurar IP sobre la pila de protocolos Bluetooth: 1) IP sobre L2CAP, 2) PPP sin RFCOMM, 3) PPP sobre RFCOMM y 4) *Bluetooth Network Encapsulation Protocol* (BNEP). Los dos primeros no tienen mucha importancia debido a que el acceso a los mismos se realiza mediante código nativo y no permite la configuración de

una red TCP-IP. Por lo tanto los dos modos más comunes son los dos últimos, que ahora describiremos más detalladamente.

Para el establecimiento de una conexión *PPP sobre RFCOMM* es necesario configurar sobre el nivel RFCOMM una conexión punto a punto (PPP) y sobre ella establecer los niveles de red habituales, IP y TCP. En la Figura 2.5 se puede ver claramente como se comunican entre sí los diferentes niveles de la pila.

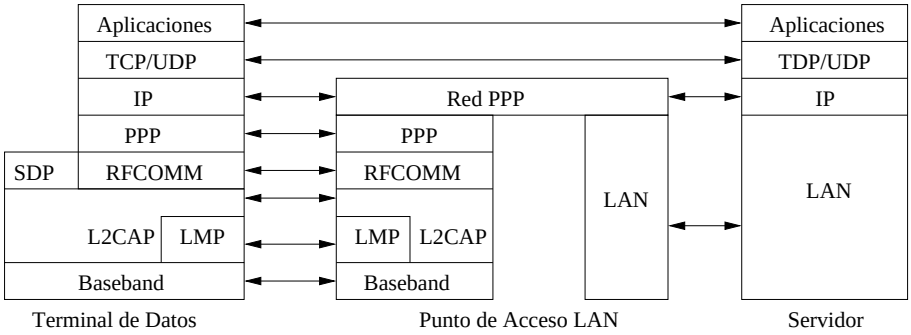


Figura 2.5: Pila de una conexión PPP sobre RFCOMM

El método *Bluetooth Network Encapsulation Protocol (BNEP)* es más simple que RFCOMM. En BNEP directamente se transmiten los paquetes IP dentro de los paquetes L2CAP con una cabecera que identifica los paquetes BNEP. Se puede utilizar para la formación de redes *ad hoc* debido a su simplicidad a la hora de establecer la comunicación. En la Figura 2.6 se observa como queda la pila del protocolo.

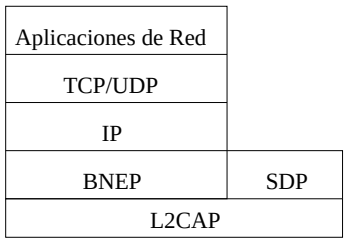


Figura 2.6: Pila de una conexión BNEP

Se pueden dar dos tipos de escenarios:

- *Network Access Point* (NAP): un dispositivo actúa como puente para conectar una piconet y una red cableada.
- *Group ad hoc Network* (GN): un dispositivo conecta uno o más dispositivos Bluetooth en una piconet. Estas redes no permiten la comunicación con el exterior. En este tipo de configuración cada usuario puede tener su propia red de área local[9].

[9] Personal Area Network User (PANU).

La utilización de los protocolos BNEP y RFCOMM permiten la creación de redes TCP-IP basadas en Bluetooth permitiendo el diseño y desarrollo de servicios de datos en entornos distribuidos e inalámbricos. Del mismo modo, nos facilita el diseño de los servicios mediante arquitecturas basadas en agentes inteligentes permitiendo minimizar el impacto de la velocidad de red disponible en estos entornos y aumentar la robustez de las aplicaciones diseñadas.

2.2.2. WiFi

En esta sección vamos a centrarnos en el protocolo de comunicaciones 802.11b [Bre97] aunque actualmente se utilizan protocolos que proporcionan unas mayores prestaciones como el 802.11g [GG02].

A continuación pasaremos a detallar la pila del protocolo 802.11, aunque no detallaremos todos los niveles, sino solamente los que han sido modificados respecto a una conexión cableada. Es decir, solamente vamos a comentar en detalle el nivel físico y de enlace tal y como puede apreciarse en la Figura 2.7.

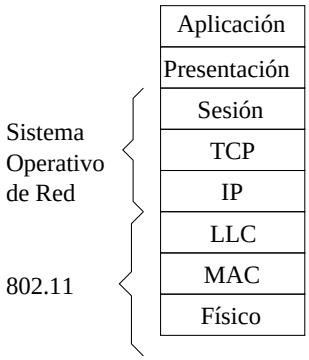


Figura 2.7: Pila del protocolo 802.11 (WiFi)

La redes 802.11 operan a nivel físico en un rango de frecuencias que tiene un ancho de 2.4Ghz, este rango es dividido en distintos canales que pueden utilizarse para la transmisión y de esta forma evitar solapamientos de la señal. Debido al rango de frecuencias utilizado pueden producirse interferencias con las señales emitidas por dispositivos que funcionen en la misma frecuencia, como teléfonos inalámbricos, hornos microondas y otros dispositivos. Estas interferencias producen frecuentes errores de red, haciendo necesario que el diseño de aplicaciones en este tipo de contextos deba considerar los problemas de red. Por lo tanto, la utilización de arquitecturas basadas en agentes en este tipo de entornos facilitará la creación de servicios de datos robustos frente a problemas de red en entornos inalámbricos inestables como en el que nos encontramos.

El nivel de enlace se divide en dos capas la LLC (*Logical Link Control*) y la MAC (*Media Access Control*). La capa LLC al igual que las redes LAN utiliza direcciones de 48 bits haciendo muy simple la transmisión de datos de una red inalámbrica a una cableada. Esta característica permite la utilización de las distintas plataformas de agentes inteligentes disponibles sin necesidad de ser modificadas debido a que tanto las redes WiFi como las cableadas

se basan en el protocolo TCP-IP. Sin embargo la capa MAC es única en las redes inalámbricas y ha sido diseñada para soportar múltiples usuarios en un medio común.

Este tipo de redes se esta haciendo muy popular debido a su amplia implantación en oficinas y hogares, ó para proporcionar acceso inalámbrico en lugares públicos, como parques, campus, bibliotecas, o tiendas. Sin embargo, tiene el inconveniente, de que la velocidad de acceso y las interferencias sufridas por el usuario dependen de la distancia al punto de acceso (a mayor distancia menor velocidad de red) y del número de usuarios que estén utilizando la red (a mayor número de usuarios mayor cantidad de interferencias y menor velocidad de red).

WIMAX

WIMAX (*Worldwide Interoperability for Microwave Access*) [Ohr05] promueve la estandarización y la interoperabilidad dentro del estándar IEEE 802.16, también conocido como *WirelessMAN*. La finalidad de este estándar es proporcionar una alternativa al problema de la "última milla"[10] frente a las alternativas del cable o ADSL.

Hay que destacar que la forma de operar de WiFi y WIMAX es totalmente distinta. En WiFi el controlador de acceso al medio utiliza un acceso por contención, es decir, todas las estaciones móviles que quieren transmitir datos a través de un punto de acceso compiten entre sí por utilizarlo. Esto puede provocar que los nodos que se encuentren alejados del punto de acceso sean descartados cuando los nodos cercanos al punto de acceso intentan transmitir datos. Esta característica además dificulta la implantación de servicios que requieren políticas de calidad de servicio para su funcionamiento.

Sin embargo, la capa de acceso al medio de WIMAX permite que los dispositivos solamente tengan que competir cuando se conectan a la red y no para realizar transmisiones de datos. Esto se debe a que los puntos de acceso de la red tienen definida una política de acceso que permite que todas las estaciones transmitan, sin ser descartadas las que están más alejadas del punto de acceso. Es decir, el punto de acceso decide qué nodo transmite en cada momento. El tiempo que un nodo puede estar transmitiendo datos depende del número de estaciones que hay conectadas en ese momento. Además, el algoritmo utilizado por este protocolo de comunicaciones permite balancear la carga y establecer políticas de calidad de servicio.

Por lo tanto, cuando se diseña un servicio de acceso a datos en una red WIMAX este no debe considerar que haya problemas de transmisión debidos a que el punto de acceso que le da cobertura al dispositivo de usuario esté alejado. Sin embargo, se mantienen los posibles problemas debidos a que el usuario se encuentra en un entorno inalámbrico, es decir, se está utilizando una red que puede sufrir interferencias y desconexiones frecuentes. Por consiguiente, los servicios de datos diseñados para ser utilizados en este tipo de entornos de ejecución deberán considerar estas características y proporcionarles solución. Una aproximación basada en agentes inteligentes facilita el diseño y desarrollo de aplicaciones distribuidas en contextos con redes inestables e inalámbricas como son las redes WIMAX.

[10] Conexión de hilo de cobre realizada entre la centralita de telefonía y el hogar del abonado.

2.2.3. HomeRF

El protocolo de acceso inalámbrico compartido HomeRF [Pal07b], *HomeRF Shared Wireless Access Protocol (SWAP)*, ha sido diseñado para transmitir datos y voz dentro de los edificios. Una nueva clase de dispositivos móviles que utilizan los PC's e Internet han sido desarrollados con las ventajas de HomeRF. En telecomunicaciones, los cable-módem y el xDSL son utilizados para cubrir la 'última milla". En cambio, HomeRF solamente puede ser utilizado en entornos muy reducidos, por ejemplo hasta 50 metros. HomeRF utiliza la infraestructura de comunicaciones existente entre los PC's e internet, TCP-IP. Además, se han desarrollado estándares para transmisión de información telefónica.

El protocolo de acceso inalámbrico compartido HomeRF puede funcionar tanto con redes ad hoc en modo *peer-to-peer* o en modo gestionado, es decir, bajo el control de un punto de acceso. Este tipo de redes están formadas por tres tipos de dispositivos:

- Puntos de control.
- Dispositivos de voz síncronos.
- Dispositivos de datos asíncronos.

Utilizando un protocolo basado en la contención, el punto de acceso puede realizar transferencias de datos desde y hacia los dispositivos. Se utiliza un modo de transferencia basado en una arquitectura cliente/servidor entre el punto de acceso y los dispositivos de voz, sin embargo la comunicación entre el punto de acceso y los dispositivos de datos es *peer-to-peer*. La pila de comunicaciones de este protocolo se muestra en la Figura 2.8.

La principal diferencia entre HomeRF y Bluetooth es el tipo de redes que pueden constituirse, mientras que en HomeRF se forman redes cliente/servidor y *peer-to-peer* en una red Bluetooth solamente pueden establecerse relaciones de tipo maestro/esclavo (formando redes en estrella). Además, Bluetooth fue diseñado para posibilitar la comunicación entre dispositivos móviles y sus periféricos, posteriormente se amplió su utilización a impresoras, teclados y otro tipo de dispositivos.

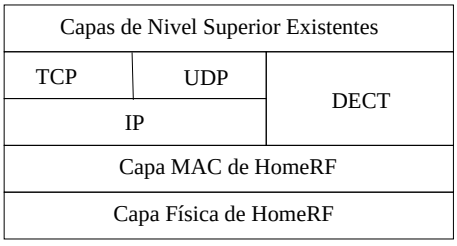


Figura 2.8: Pila de una conexión HomeRF

La capa MAC ha sido optimizada y desarrollada para entornos de ejecución con un área de cobertura máxima de un edificio permitiendo la transmisión de voz y datos.

2.3. Programación en entornos distribuidos

En esta sección vamos a presentar las distintas aproximaciones que pueden seguirse para diseñar una arquitectura software que permita el acceso a un conjunto de datos en un entorno distribuido. En este contexto hay que considerar que tanto los distintos depósitos de datos, como los distintos módulos que están implicados para permitir el acceso a los datos, pueden estar distribuidos en una red. Asimismo, hay que considerar la conectividad que utiliza el usuario, es decir, si estamos ante un entorno cableado o inalámbrico. Debido a la creciente utilización de los sistemas de comunicaciones inalámbricos y sus peculiares características (redes inestables, lentas, con un alto coste económico, etc...), el diseño y desarrollo de sistemas software debe considerar la red que utiliza el usuario para adaptar su comportamiento a la misma y obtener un aumento de las prestaciones y una reducción de costes (económicos y temporales).

2.3.1. Llamada a procedimientos remotos

La aproximación más clásica para el desarrollo de aplicaciones distribuidas que encapsula la utilización de la red en el acceso a servicios remotos es la llamada a procedimientos remotos [Sri95], también conocido como RPC (*Remote Procedure Calling*). Anteriormente, las aplicaciones distribuidas se desarrollaban mediante la utilización de *sockets*, un conjunto de librerías escritas en C [FDL08b] que permiten a dos procesos comunicarse a través de la red.

En la Figura 2.9 puede verse un ejemplo de este tipo de arquitectura donde distintos clientes envían una petición a un único servidor. El servidor procesará cada una de las peticiones y devolverá los resultados a los clientes.

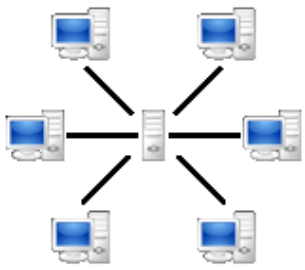


Figura 2.9: Arquitecturas cliente/servidor

RPC es un paradigma muy popular para el diseño e implementación del modelo cliente/servidor en el diseño de aplicaciones distribuidas. Una invocación a un procedimiento remoto es iniciada por el cliente mediante el envío de un mensaje/llamada a un servicio de un

servidor remoto conocido, para ejecutar un procedimiento específico utilizando los parámetros necesarios. La ejecución de este procedimiento devolverá unos resultados al cliente y el cliente podrá continuar su ejecución .

Existen implementaciones como CORBA [OMG93] o RMI [Mic06b] que permiten el desarrollo de aplicaciones basadas en RPC. Ambas aproximaciones permiten la realización de servicios de acceso a datos distribuidos. En ambos casos el diseñador utiliza un conjunto de interfaces para el desarrollo de su aplicación y el *middleware* utilizado se encarga de permitir y encapsular el protocolo de comunicaciones.

Una diferencia importante entre la invocación de un procedimiento remoto y una llamada local, es que la llamada a un procedimiento remoto puede fallar debido a errores en la red de comunicaciones. Por lo tanto, el diseño de la aplicación debe considerar este tipo de problemas, especialmente en contextos inalámbricos debido a que son mucho más comunes que en las redes cableadas. Del mismo modo, se debe considerar que el cliente puede invocar un servicio de un servidor cuando éste esté atendiendo la petición de otro cliente.

A continuación, enumeraremos las ventajas de las arquitecturas cliente/servidor, habitualmente basadas en RPC:

- Una arquitectura cliente/servidor permite dividir la solución de un problema de un sistema y distribuirlo en distintos ordenadores independientes que se comunican a través de la red. Esto proporciona otra ventaja adicional: una mayor facilidad en su mantenimiento. Por ejemplo, es posible reemplazar, reparar, actualizar, ó incluso cambiar la ejecución del servidor a otro ordenador sin modificar ni afectar al comportamiento de los clientes.

- Todos los datos son almacenados en los servidores, que generalmente tienen un mayor control de seguridad que los clientes. Los servidores pueden controlar mejor el acceso y los recursos, para garantizar que solo los clientes con unos permisos adecuados accedan y modifiquen los datos.

- Existen distintas tecnologías cliente/servidor disponibles para el diseño de aplicaciones mediante un interfaz amigable, que además permiten definir políticas de seguridad.

Sin embargo, este tipo de aproximación tiene las siguientes desventajas:

- El protocolo de comunicaciones habitualmente suele ser propietario por lo que depende de la implementación, dificultando la interoperabilidad. En el caso de desarrollar una arquitectura mediante CORBA, el protocolo de comunicaciones no es propietario, pero presenta el problema de que no se utilizan puertos estándar[11]; por lo tanto, las políticas de seguridad de la empresa deben ser modificadas para permitir la utilización de este tipo de aproximaciones.

- La utilización de un mecanismo de comunicaciones síncrono, hace que las aplicaciones diseñadas utilizando esta arquitectura sean poco robustas en redes inestables como las inalámbricas.

[11] Como los utilizados por HTTP, FTP, SMTP o POP3, habitualmente habilitados para permitir comunicaciones entre la intranet de la empresa e Internet.

- Se basan en un servidor para cada uno de los servicios desarrollados, por consiguiente un elevado número de procesos cliente que realicen distintas peticiones de servicio de forma concurrente saturará al servidor. La saturación del servidor se puede producir porque no se tiene la suficiente capacidad de procesamiento o porque no se puede transmitir la suficiente información a través de la red.

Como se ha visto las arquitecturas basadas en llamadas a procedimientos remotos solucionan algunos de los problemas debidos al diseño de servicios de acceso a datos distribuidos. Sin embargo, no dan solución a algunos de los problemas debidos al uso de redes inalámbricas, como por ejemplo el diseño de servicios de datos en redes inestables, lentas y caras. Haciéndose necesario el diseño de nuevas aproximaciones.

2.3.2. Sistemas *Peer-to-Peer*

Como hemos mostrado anteriormente las aproximaciones cliente servidor se pueden ver saturadas y producirse fallos si el numero de clientes que intentan acceder a un conjunto de servidores es muy elevado. Debido a esto aparecieron las arquitecturas *peer-to-peer* (P2P) [ATS04] que permiten que los distintos nodos puedan actuar, tanto como clientes como servidores. Es decir, cuando un usuario busca un dato en este tipo de redes primeramente pregunta qué nodos los tienen y posteriormente lo obtiene independientemente de quien se lo proporcione. Tal y como puede apreciarse en la Figura 2.10 todos los nodos pueden establecer una conexión entre sí evitando que se sature el sistema.

Figura 2.10: Arquitecturas P2P

También hay que considerar que un sistema P2P utiliza distintas rutas para realizar las conexiones de red entre los nodos que lo constituyen, como consecuencia de esto el ancho de banda de la red es igual a la suma de los anchos de banda de cada uno de los nodos que constituyen la red. Sin embargo, siguiendo una aproximación basada en una arquitectura cliente/servidor, en la que existe un reducido número de servidores y muchos clientes que se conectan, el ancho de banda disponible es menor. Las redes P2P son típicamente utilizadas para conectar un gran número de nodos entre sí y formar una red ad hoc. Este tipo de redes son

muy útiles para muchos propósitos como compartir el contenido de ficheros que contienen video, audio, datos o cualquier otro formato digital en el que un elevado número de usuarios quieren acceder a un mismo conjunto de recursos; también son muy útiles para compartir datos en tiempo real, como por ejemplo tráfico telefónico.

El primer sistema P2P ampliamente utilizado fue el servicio de noticias Usenet [FDL08h], en el que distintos nodos se comunicaban entre sí para propagar los artículos por la red. Sin embargo, el servicio de noticias también actuaba en modo cliente/servidor cuando los usuarios individuales accedían a los servidores para leer los artículos enviados. Otro de los primeros sistemas P2P ampliamente utilizado fue el servicio de correo electrónico en el que los distintos agentes de transferencia de correo constituyen una red P2P, mientras que los agentes de usuario se conectan directamente con su servidor.

Uno de los beneficios obtenidos debido a las arquitecturas P2P es que todos los nodos obtienen y proporcionan recursos, incluyéndose entre los mismos el ancho de banda, espacio de almacenamiento y potencia de cálculo. Por lo tanto, cuantos más nodos constituyen el sistema, mayor es la capacidad total del mismo. Esta característica no se verifica en los sistemas con una arquitectura cliente/servidor, en este caso añadir más clientes podría implicar la aparición de cuellos de botella.

Por otra parte, la naturaleza distribuida de las redes P2P también incrementa la robustez en caso de fallos debido a la replicación de los datos en múltiples nodos del sistema, y —en un sistema P2P puro– los nodos permiten encontrar los datos sin la necesidad de un servidor centralizado.

2.3.3. Servicios web

De acuerdo a [BHM+04], *"Un servicio web es un sistema software diseñado para soportar la interoperabilidad máquina-máquina a través de una red. Tiene un interfaz descrito en un formato que puede ser procesado por una máquina (concretamente WSDL [CGMW02]). Otro sistema interactúa con el servicio web tal y como se específica en su descripción utilizando mensajes SOAP [STK01], habitualmente utilizando el protocolo HTTP con una serialización XML [McL01] junto con otros estándares relacionados con la web"*.

Los servicios web proveen a los diseñadores de aplicaciones con un conjunto de estándares que permiten la interoperación entre distintas aplicaciones, permitiendo su ejecución en distintas plataformas y entornos de ejecución. Las arquitecturas basadas en servicios web proveen un modelo conceptual y un contexto de ejecución para los servicios web, junto con las relaciones entre los distintos componentes de estas arquitecturas. Algunas de las características de los servicios web son:

- *Facilidad de Interoperación*: Los servicios web, tal y como se muestra en su definición permiten ser utilizados sin necesidad de saber cómo han sido implementados. Además, las arquitecturas basadas en servicios web también tienen la ventaja de que permiten la composición de servicios web. Esta característica permite la colaboración para el desarrollo de servicios entre distintas entidades en la administración pública y entre distintas empresas en el ámbito privado.

  Además, hay que destacar la facilidad de interoperabilidad entre los distintos servicios

web, facilitada por la utilización de XML como lenguaje de serialización. Es decir los datos utilizados por los distintos servicios web son trasmitidos en formato XML.

- *Servicios de búsqueda*: Los servicios web habitualmente son registrados en servicios de directorio que permiten su búsqueda para su posterior utilización. Por ejemplo, una empresa puede hacer pública la descripción de todos los servicios, mediante el estándar UDDI [OAS08], con los que provee a sus clientes y éstos utilizar un servicio de directorio para localizar el servicio que más se ajusta a sus necesidades.

- *Comunicación basada en HTTP*: Los servicios web utilizan HTTP como protocolo de comunicaciones evitando problemas (por ejemplo, problemas de interoperabilidad, modificación de las políticas de filtrado de los cortafuegos, etc.) de arquitecturas anteriores para el diseño y desarrollo de sistemas distribuidos, como CORBA [OMG93]. Mediante la utilización de servicios web las reglas de filtrado de comunicaciones se limitan a la autorización del protocolo HTTP, es decir, no hay que habilitar ningún nuevo protocolo de comunicaciones adicional en las empresas[12]. Habitualmente dicho mecanismo de comunicaciones está permitido porque la mayoría de las organizaciones disponen de un sitio web.

La arquitectura de servicios web solamente describe: 1) las características mínimas que son comunes a todos los servicios web, y 2) no impone restricciones en como combinar un conjunto de éstos. El estándar de desarrollo de servicios web describe qué características pueden tener los servicios web, pero no tienen por qué darse todas las características en un único servicio web.

Como se ha visto, los servicios web solventan los problemas de interoperabilidad que pudiesen tener las arquitecturas basadas en RPC, sin embargo siguen teniendo el resto de problemas que tienen las arquitecturas cliente/servidor debido a que utilizan llamadas a procedimientos remotos.

2.3.4. Sistemas multiagente

Como se ha mostrado anteriormente, el diseño y desarrollo de sistemas multiagente permite facilitar la implantación y ejecución de aplicaciones distribuidas que facilitan el acceso a datos.

Actualmente existen distintas plataformas, como son Voyager [Obj99], Grasshopper [BM99], Zeus [Tho01], Jade [Lab06], SPRINGS [ITM06], etc., para el desarrollo de aplicaciones basadas en agentes. La mayor parte de estas plataformas permiten el desarrollo de aplicaciones basadas en agentes móviles, aunque la metodología de desarrollo varía entre unas y otras. Además, también varían las prestaciones y los servicios que proporcionan las distintas plataformas.

Por otra parte, con la llegada de los agentes móviles existe una nueva posibilidad. Los agentes móviles son módulos inteligentes y autónomos que se mueven por sí mismos de un ordenador a otro, llevando consigo su estado y continuando su ejecución en el ordenador

[12] Modificando la configuración del cortafuegos utilizado en la corporación.

remoto. Debido a la utilización de arquitecturas basadas en agentes (móviles o no) se obtienen una serie de ventajas relacionadas con el acceso a información remota:

- *Encapsulan el protocolo de comunicaciones.* El agente móvil sabe cómo moverse por sí mismo de un lugar a otro; no hay código de transferencia de información entre ordenadores. Es decir, el programador ejecuta un método que toma como parámetros el nombre DNS [Moc87a, Moc87b] o la IP del ordenador en el que se va a ejecutar el método del agente. Además de este parámetro también pueden utilizarse otros parámetros necesarios para la ejecución del método.

- *Son asíncronos.* No necesitan comunicaciones síncronas para trabajar, dada su naturaleza autónoma, aunque por supuesto pueden sincronizarse con otros módulos o agentes.

- *Permiten reducir el uso de la red.* En el caso de los agentes móviles, cuando el agente ha llegado al ordenador destino, la conexión a red ya no se necesitará hasta que el agente necesite viajar de nuevo o comunicarse remotamente; el agente terminará el trabajo en ese ordenador y, cuando se restablezcan las comunicaciones, podrá continuar con el trabajo en otro ordenador. De esta manera, los procesos cliente no deben preocuparse de fallos o cortes de la conexión de red.

- *Interacción local.* En lugar de realizar llamadas a procedimientos remotos pueden viajar al ordenador adecuado e interactuar localmente con el sistema servidor.

- *Adaptabilidad al contexto.* Pueden comportarse de forma diferente en situaciones distintas, adaptándose a los recursos disponibles.

Todas estas características les hace muy interesantes para el diseño de aplicaciones en entornos inalámbricos, o donde simplemente se quiere reducir el uso de la red. Sin embargo, la principal ventaja de utilizar agentes inteligentes frente a otras aproximaciones, como RPC/servicios web o P2P, es que los agentes pueden de forma autónoma estudiar su contexto de ejecución y modificar su comportamiento permitiendo que la arquitectura basada en agentes se comporte como una arquitectura RPC o una P2P en los momentos más adecuados, maximizando las ventajas de cada aproximación y minimizando sus inconvenientes. Además hay que destacar que los agentes permiten desplazar la ejecución (característica no disponible en los sistemas basados en arquitecturas RPC o P2P) de los servicios al lugar más adecuado, permitiendo de esta forma optimizar su ejecución y la forma en la que se consumen los recursos.

2.4. Ontologías: creando y explotando el conocimiento

Primeramente definiremos qué es una ontología, para posteriormente describir cuáles son las ventajas de su utilización y los lenguajes utilizados en su creación. Según T. Gruber, *"una ontología se define como la especificación de una conceptualización"* [Gru92]. En un principio las ontologías comenzaron a especificarse en lenguajes como KIF [GF92, ANS06] u Ontilingua [Gru93, SU06]. Sin embargo, los lenguajes de especificación de ontologías han

evolucionado para favorecer su utilización en la Web y permitir compartir más eficientemente el conocimiento.

En el contexto en el que se encuadra la tesis, una ontología se define como un conjunto de primitivas que permiten modelar un dominio de discurso [Gru07]. Las primitivas son clases (o conjuntos), atributos (o propiedades) y relaciones entre los miembros de las clases. La definición de las distintas primitivas que pueden utilizarse para especificar una ontología incluyen una explicación de su significado y un conjunto de restricciones acerca de su utilización. En el contexto de los sistemas de bases de datos una ontología puede ser vista como un nivel de abstracción del modelo de datos, análogo al modelo relacional o jerárquico. Sin embargo la utilización de ontologías en este contexto está destinada a modelar el conocimiento disponible acerca de los individuos, sus atributos y las relaciones existentes entre distintos individuos.

Lo importante no es cómo se define una ontología sino para qué sirve. Las ontologías desde un comienzo han sido diseñadas con el propósito de permitir compartir y reutilizar el conocimiento. Por ejemplo, cuando definimos un conjunto de ontologías que son utilizadas por distintos agentes para comunicarse acerca de un dominio de discurso, decimos que los agentes están asignados a la ontología si sus acciones son consistentes con las definiciones de la ontología.

También hay que considerar que las ontologías habitualmente son especificadas mediante un conjunto de lenguajes que permiten la abstracción de las estructuras de datos y las distintas estrategias de implementación que pueden ser utilizadas; en la práctica, los lenguajes utilizados para la especificación de ontologías tienen una potencia expresiva más cercana a la lógica de predicados de primer orden que a los lenguajes utilizados en la definición de los modelos conceptuales de bases de datos. Por esta razón, se dice que las ontologías están en un nivel semántico, mientras que los esquemas de bases de datos son modelos que pueden integrar bases de datos heterogéneas, permitiendo la interoperación en sistemas distribuidos. Es decir, las ontologías permiten la definición de servicios basados en el conocimiento, los cuales proporcionan una descripción de sí mismos que puede ser analizada y comprendida por agentes y por seres humanos.

El problema actual es que se dispone de acceso a muchos depósitos de datos a través de internet (software, servicios web, páginas web, documentos multimedia, etc...), pero no se dispone de información semántica acerca de los mismos. También existe el problema de tener disponibles un conjunto de ontologías pero no tener acceso a los depósitos de datos subyacentes. Mediante la utilización de una arquitectura basada en ontologías [DMDH02, Hes06] (ver Figura 2.11) se facilita el acceso a un depósito de datos tanto por usuarios como por agentes software, pero además se puede independizar el acceso de la arquitectura del depósito de datos subyacente.

Es decir, actualmente se encuentran disponibles en Internet muchos datos que pueden ser visualizados y entendidos por los usuarios. En cambio, es necesario ir un paso más lejos; publicar los datos en un lenguaje que sea inteligible para personas y agentes software. Para la publicación de los datos en este tipo de formatos se han definido un conjunto de lenguajes como DAML+OIL [Hor02b] o OWL [AvH03], que especifican el contenido de los documentos y no únicamente su representación gráfica como se hace en HTML. Estos lenguajes permiten almacenar conocimiento que puede ser entendido por los agentes software,

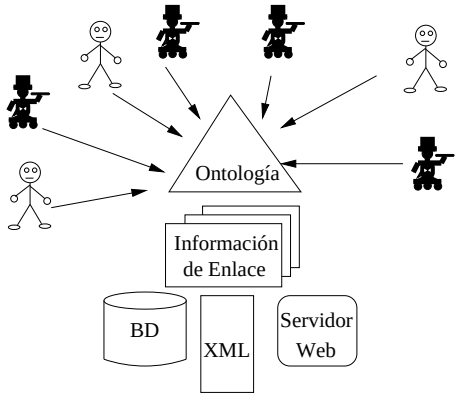


Figura 2.11: Acceso a una arquitectura basada en ontologías

y posteriormente podrá ser utilizado por éstos en razonamientos deductivos o inferencias, obteniendo de esta forma más conocimiento o facilitando ayuda a los usuarios.

2.4.1. Lenguajes de definición de ontologías en la Web

Las ontologías proporcionan a los agentes la capacidad de entender e integrar distintos depósitos de información. Por ejemplo, imaginemos un agente software que planifique la vida social de una persona. Este agente utilizará las preferencias del usuario (tales como la clase de películas que le gustan, el tipo de comida preferido, etc.) y utilizará esta información para planificar las actividades del usuario durante su tiempo de ocio. La tarea de planificar estas actividades depende de la variedad de actividades que pueda ofertar el planificador al usuario y de las necesidades del mismo. Durante el proceso de planificación, el agente tendrá que buscar, clasificar y revisar los servicios disponibles y que más se ajusten a las preferencias del usuario al que presta servicio.

Este tipo de agentes necesitan acceder a ontologías que representen la información que deben procesar, por ejemplo información acerca de restaurantes, hoteles, etc. y ontologías de servicios que representen quién y dónde se proporcionan los servicios. Estas ontologías proporcionan el conocimiento necesario al agente para seleccionar los servicios que más interesan al usuario. Esta información se obtendrá de distintos portales web, webs de reserva de hoteles, etc.

Por consiguiente, el agente software tiene que ser capaz de buscar, analizar y extraer conocimiento de la web. Para que este proceso pueda realizarse, la información disponible en la web debe ser especificada mediante un lenguaje que interpretable por los agentes software. Algunos de estos lenguajes son: RDF [LS01], DAML+OIL [Hor02b] ó OWL [AvH03]. Todos ellos se basan en XML [McL01], de esta forma se favorece: 1) la transmisión de ontologías entre distintos sistemas, y 2) la interoperabilidad, porque la especificación de la ontología es independiente de la plataforma utilizada y del mecanismo de razonamiento utilizado por los agentes software. A continuación veremos los detalles de estos tres lenguajes.

RDF

RDF (*Resource Description Framework*) [LS01] pertenece a la familia de especificaciones del W3C (*World Wide Web Consortium*) [Wor06c], diseñada para representar información en la Web.

RDF se basa en una sintaxis abstracta que representa un modelo de datos mediante un simple grafo y una semántica formal con una notación rigurosamente definida, proporcionando una base para que los agentes puedan razonar sobre el contenido de la Web. El desarrollo de RDF se ha visto motivado por los siguiente usos:

- Permitir la representación de metadatos en la Web, facilitando información acerca de los recursos web disponibles y de los sistemas que los utilizan.
- Facilitar el diseño y desarrollo de aplicaciones que requieren modelos de información abiertos y dinámicos.
- Ayudar en la creación de información que pueda ser procesada autónoma y automáticamente por agentes software, es decir, facilitando el procesado de la información fuera del entorno donde fue creada.
- Permitir la integración de aplicaciones, combinando datos de distintas aplicaciones para generar más información.
- Automatizar el procesamiento de la Web por los agentes software, actualmente la Web es inteligible para los seres humanos. Sin embargo, finalmente la Web se transformará en una red donde distintos agentes cooperen en el procesamiento de recursos RDF proporcionando un lenguaje de comunicación entre agentes.

Debido a las motivaciones que han impulsado el diseño de RDF, este lenguaje verifica las siguientes propiedades: 1) utiliza un modelo de metadatos simple, 2) posee una semántica formal y permite realizar inferencias, 3) utiliza un vocabulario extensible basado en identificadores de recursos válidos universalmente, denominados URIs [McL01], 4) utiliza una sintaxis basada en XML, 5) permite la definición de datos basados en XML schema [Wor01], y 6) permite la definición de cualquier tipo de recurso.

El modelo de metadatos de RDF se basa en la idea de definir recursos mediante ternas, que permiten constituir un grafo. Estas tripletas están constituidas por: 1) un identificador para el recurso (denominado sujeto), 2) un conjunto de relaciones (predicados, habitualmente denominados propiedades) y 3) un conjunto de recursos (objetos) con los que mantiene relación el recurso que se está definiendo. RDF es un modelo abstracto y por lo tanto tiene distintos formatos de serialización, el más utilizado se basa en XML [McL01], otra recomendación del W3C se denomina N3 [FDL08f]. La utilización de un formato basado en XML facilita la interoperación en sistemas heterogéneos y distribuidos. En la Figura 2.12 se muestra un ejemplo en el que se especifica que la abreviatura de la ciudad de Nueva York es NY.

Este mecanismo de descripción de recursos es el componente principal propuesto por el W3C para la Web Semántica: una evolución de la Web actual que permitirá a los agentes almacenar, intercambiar y entender la información disponible en la Web; permitiendo a los

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:terms="http://sid.cps.unizar.es/ciudades/abreviaturas/">
        <rdf:Description rdf:about="urn:x-states:Nueva%20York">
        <terms:alternative>NY</terms:alternative>
        </rdf:Description>
</rdf:RDF>
```

Figura 2.12: Descripción de un recurso en RDF: abreviatura de la ciudad de Nueva York

usuarios obtener información de una forma más eficiente. Sin embargo, han tenido que diseñarse otros lenguajes que permitan incrementar la potencia expresiva de RDF, para permitir realizar un mayor número de inferencias a los agentes de forma autónoma y automática.

Debido a que las ontologías tienen una función clave en la Web Semántica, porque permiten un análisis automatizado de la información por agentes software, se deben extender los lenguajes de marcas (XML [McL01], RDF [LS01],etc.) para permitir la especificación de ontologías. RDFS [DMH+00] (*RDF Schema*) es reconocido como un lenguaje de representación de conocimiento; permitiendo la descripción de clases, propiedades (relaciones binarias), restricciones de rango y dominio (en las propiedades), subclases y subpropiedades en las relaciones.

Sin embargo, debido a que RDFS es un lenguaje poco expresivo deben crearse lenguajes con una capacidad expresiva mucho mayor, permitiendo de esta forma una descripción más detallada de los recursos y que tengan una mayor capacidad de inferencia. Además, estas descripciones deben permitir un razonamiento automatizado, por ejemplo permitiendo la extracción de relaciones entre distintos términos. Estos requisitos fundamentan la creación del lenguaje DAML+OIL, como un lenguaje de especificación de ontologías muy expresivo.

DAML+OIL

DAML+OIL[13] [DAR06], creado en el año 2000, es un lenguaje de especificación de ontologías especialmente diseñado para ser utilizado en la Web. Se basa en estándares Web como XML y RDF, añadiendo las primitivas suficientes para permitir la definición de ontologías que puedan ser utilizadas para razonar sobre las mismas mediante el uso de sistemas basados en el conocimiento, como por ejemplo los sistemas terminológicos [BCM+03], permitiendo de esta forma que los agentes software puedan razonar autónomamente mediante el análisis de la información almacenada en una especificación DAML+OIL[Hor02b]. Las especificaciones que utilizan este lenguaje se basan en una aproximación orientada a objetos que se fundamenta en la definición de clases, propiedades e individuos.

La potencia expresiva del lenguaje viene determinada por los constructores de clases y propiedades que pueden ser utilizados. El lenguaje DAML+OIL posee los siguientes operadores: 1) constructores de clases: intersectionOf, unionOf, complementOf, toClass, hasClass, hasValue, cardinalityQ, minCardinalityQ y maxCardinalityQ, y 2) constructores de axiomas: basados en la lógica de predicados de primer orden [Hal90].

Sin embargo, hay que destacar que se pueden realizar distintas definiciones de una misma clase en distintas ontologías. Para solucionar este problema DAML+OIL identifica las clases

[13]DARPA Agent markup Language+Ontology Inference Layer

y propiedades de una ontología mediante un identificador único denominado URI (*Uniform Resource Identifier*). Permitiendo de esta forma reutilizar las clases y propiedades de otras ontologías en la creación de una nueva, permitiendo reutilizar el conocimiento ya adquirido. Tanto si la creación de la nueva ontología la realiza una persona o un agente inteligente.

OWL

OWL [AvH03] son las siglas de *Web Ontology Language* (lenguaje de ontologías para la Web). OWL nace en 2003 como una evolución de DAML+OIL, por lo que también está basado en RDF, y por consiguiente en XML. Este lenguaje se ha diseñado para ser utilizado en aplicaciones que necesiten procesar información y razonar sobre la misma, en lugar de únicamente presentarla al usuario. OWL facilita la interpretación del contenido de la Web por las máquinas (agentes software). Por lo tanto, OWL facilita una mayor integración de la Web y los agentes software debido a que permite realizar una descripción formal de la semántica de la información disponible permitiendo su análisis automatizado.
OWL tiene las siguientes ventajas respecto a DAML+OIL:

1. Incrementa el número de operadores disponibles: en OWL además de utilizar todos los operadores disponibles en DAML+OIL, se pueden definir los siguientes tipos de propiedades: transitivas, simétricas, biyectivas e inversas. Además se puede definir un nuevo tipo de propiedades en las que su rango es un conjunto de literales (tipos definidos) y su dominio es una clase. Sin embargo, en DAML+OIL solamente se podían definir propiedades en las que el dominio y el rango fuese una clase. Por consiguiente, la utilización de OWL permite aumentar la capacidad de razonamiento de los agentes respecto a la capacidad que tendrían si utilizasen DAML+OIL para describir las bases de conocimiento utilizadas en sus razonamientos.

2. Facilita la integración de ontologías: debido a que las ontologías pueden tener un tamaño muy grande hay que poner especial atención en que los lenguajes utilizados para su especificación permitan dividirlas en distintos módulos y posteriormente realizar una integración de los mismos. Del mismo modo, debido a que las ontologías se encuentran disponibles en la Web, distintas aplicaciones o agentes software pueden utilizarlas pudiendo realizar razonamientos. Sin embargo, la situación puede complicarse más cuando un agente necesite utilizar más de una ontología al mismo tiempo. Por esta razón, OWL proporciona un conjunto de relaciones que permiten integrar clases de distintas ontologías. Las relaciones proporcionadas por OWL son: de equivalencia de clases y relaciones entre términos (son las mismas que pueden realizarse entre términos de una misma ontología). También se pueden establecer este tipo de relaciones entre propiedades e individuos.

3. Mecanismo de control de versiones: para adaptar las ontologías al dinamismo de la Web. Este mecanismo de control de versiones permite que las relaciones entre clases y propiedades de una ontología o varias pueda variar en el tiempo y que se puedan controlar los cambios. Mediante este mecanismo los agentes software pueden actualizar el conocimiento utilizado para sus razonamientos, mediante la actualización de la ontología que utilizan para extraer conocimiento de forma autónoma y automática.

4. Distintas expresividades para diferentes necesidades: El lenguaje OWL está dividido en tres sublenguajes dependiendo de la expresividad proporcionada. El sublenguaje *OWL Lite* solamente permite la definición de jerarquías y la especificación de restricciones muy simples. Debido a esto, aplicaciones que necesiten un mayor poder expresivo como la capacidad para definir relaciones complejas, manteniendo la capacidad de razonamiento dentro de una complejidad de cálculo computable, deben utilizar el lenguaje más expresivo, como *OWL DL*. Sin embargo, también pueden darse situaciones en las que se requiera la mayor capacidad expresiva posible independientemente del coste computacional del proceso de razonamiento, en tales situaciones debe utilizarse *OWL Full*.

A continuación detallaremos las distintas expresividades que proporciona OWL para realizar la representación de conocimiento, dependiendo de la expresividad que necesite el ingeniero del conocimiento en cada situación.

1. *OWL Lite*: da soporte a aquellos usuarios que necesitan una jerarquía de clasificación y un conjunto de restricciones simples. Por ejemplo, da soporte a restricciones de cardinalidad, pero permitiendo únicamente los valores 0 y 1. La expresividad de este lenguaje es $\mathcal{SHIF}$ [BCM$^+$03]. Su potencia descriptiva es similar a RDF.

2. *OWL DL*: permite dar soporte a aquellos usuarios que quieren una expresividad máxima para especificar las relaciones entre los conceptos de una ontología, manteniendo la completitud computacional de los razonamientos realizados por el sistema terminológico [BCM$^+$03]. Es decir, todo proceso de razonamiento terminará en un tiempo finito y todos los razonamientos que puedan expresarse utilizando este lenguaje podrán ser calculados. *OWL DL* incluye a *OWL Lite*. La expresividad de este lenguaje es $\mathcal{SHOIN}$ [BCM$^+$03].

   Aunque *OWL DL* y *OWL Full* utilizan el mismo vocabulario, *OWL DL* tiene algunas restricciones:

   - Separación de tipos: una clase no puede ser al mismo tiempo un individuo y/o una propiedad, y una propiedad no puede ser simultáneamente un individuo y/o una clase. Esto implica que las restricciones no se pueden aplicar a elementos del lenguaje de OWL en sí mismos (característica que se permite en *OWL Full*).
   - Propiedades de objetos y propiedades de tipos primitivos: son los dos únicos tipos de propiedades que pueden definirse en *OWL DL*. Es decir, solamente se pueden definir relaciones entre: 1) instancias de clases y datos primitivos (propiedades de tipos primitivos), y 2) dos instancias de dos clases.
   - Clases complejas: solamente se permiten en *OWL Full*; permitiendo la descripción de clases arbitrariamente. Por ejemplo, se permite la definición de clases por enumeración. También se permite realizar restricciones sobre propiedades y combinaciones mediante álgebra de Bool [FDL08a].

3. *OWL Full*: está destinado a aquellos usuarios que quieren una expresividad máxima, es decir no imponen ningún tipo de restricción en los constructores utilizados. Sin

embargo, la gran expresividad implica que los algoritmos de razonamiento utilizados por este lenguaje pueden dar lugar a algoritmos con un coste exponencial. Además, proporciona mecanismos para la extensión del propio lenguaje, es decir, proporciona mecanismos para la definición de nuevos constructores y axiomas. *OWL Full* incluye a *OWL DL*.

Como se ha dicho anteriormente, *OWL DL* y *OWL Lite* pueden ser representados como una base de conocimiento que puede ser utilizada por un sistema terminológico, en cambio *OWL Full* no cumple esta aseveración. Por lo tanto, para que los procesos de razonamiento automático realizados por los agentes se puedan realizar con un coste temporal y de recursos no exponencial en todas las situaciones no se debe utilizar *OWL Full*.

Como se ha visto, OWL es una herramienta fundamental dentro del contexto de la Web Semántica [BLHL01]. OWL se ha diseñado específicamente para permitir dar servicio tanto a la visualización de contenido web para que sea inteligible por humanos, como para que el contenido de la web sea procesado por agentes software (componentes software). Además como OWL está escrito en XML puede intercambiarse fácilmente entre diferentes sistemas y lenguajes de aplicación. Una de las principales ventajas de OWL es que proporciona un contexto a la industria que permite el intercambio de información facilitando su integración. Sin embargo tienen un inconveniente: es un lenguaje complejo.

2.4.2. Sistemas basados en Lógica Descriptiva

Para la definición de ontologías se utilizaban los lenguajes facilitados por los sistemas basados en lógica descriptiva o sistemas terminológicos [BCM+03]. Estos sistemas permiten razonar con los conceptos y roles de las ontologías. En OWL los conceptos y roles son denominados clases y propiedades, respectivamente. Debido a que *OWL DL* y *OWL Lite* se basan en los operadores disponibles en Lógica Descriptiva, las ontologías creadas mediante los constructores proporcionados por estos lenguajes pueden ser analizadas por agentes software mediante la utilización de un sistema terminológico.

Los sistemas terminológicos [BBMAR89, DLNS96, CGLN99] son sistemas de representación del conocimiento basados en Lógica Descriptiva. Además, éstos se conocen con otros nombres, como sistemas basados en KL-ONE [BS85] (ahí tienen su origen), o sistemas basados en Lógica Descriptiva[14]. Hay un número importante de sistemas terminológicos, por ejemplo KANDOR [PS84], CANDIDE [BGN89], CLASSIC [BBMAR89, KLP97], LOOM [Mac88], RACER [HM01], FaCT [Hor02c], BACK [vLNPS87], y FaCT++ [TH06]. Los sistemas terminológicos se han utilizado en distintos campos como son el diseño conceptual, el procesamiento del lenguaje natural y las aplicaciones que tratan de explotar datos existentes en distintas fuentes de información. Hay que destacar que se distinguen dos componentes fundamentales en un sistema terminológico:

1. *Componente terminológico*: permite describir términos, que pueden ser conceptos (conjuntos de objetos que cumplen unas cierta condiciones) o roles (relaciones binarias entre conceptos o entre conceptos y tipos simples).

[14] *Description Logics*(DL) en inglés.

2. *Componente asercional*: nos da la posibilidad de crear instancias para los conceptos.

Estableciendo un símil con los sistemas de gestión de bases de datos, mediante el componente terminológico se describe el esquema de una base de conocimiento y utilizando el componente asercional se crean las instancias (individuos) de la misma. Extrapolando esta conclusión a la Web, mediante las clases y conceptos se describe la información almacenada (las relaciones existentes entre los datos y el significado de estas relaciones) y mediante las instancias de estas clases los recursos disponibles. Por lo tanto, los agentes podrán utilizar un sistema terminológico para analizar las ontologías disponibles en la Web y extraer conocimiento de las mismas de forma automatizada y autónoma.

Por consiguiente, los conceptos son términos que se describen por medio de condiciones que deben cumplir las instancias que agrupan. Cuando estas condiciones son necesarias y suficientes, entonces se dice que el concepto es *definido* y cuando son sólo necesarias entonces se dice que el concepto es *primitivo*. Esta característica de poder crear conceptos definidos es inherente a los sistemas terminológicos y es la que les permite razonar. Debido a que pueden extraerse relaciones transitivas entre los conceptos. Por ejemplo, si sabemos que una persona es un vegetariano estricto sí y sólo sí solamente come plantas y sabemos que una persona es vegetariana sí y sólo sí come plantas o leche. Entonces podemos deducir que un vegetariano estricto es un tipo de vegetariano. Es decir podemos extraer nuevas relaciones de inclusión (relaciones de tipo si A entonces B) a partir de relaciones sí y sólo sí.

Los roles son términos que representan relaciones binarias entre conceptos o bien entre conceptos y tipos simples. Un rol viene descrito por un dominio y un rango, que son el concepto al que está asociado y el tipo de valores que puede tomar, respectivamente. OWL se aprovecha de más de quince años de investigación en sistemas terminológicos debido a que cualquier ontología descrita en *OWL DL* o *OWL Lite* puede automáticamente ser transformada en una sintaxis aceptada por un sistema terminológico. Proporcionando las siguientes ventajas:

1. Semántica bien definida: los sistemas terminológicos tienen una semántica perfectamente definida. Por lo tanto, en el proceso de traducción entre OWL y la sintaxis aceptada por el sistema terminológico utilizado no se produce ninguna perdida de información o semántica.

2. Coste computacional: los sistemas terminológicos proporcionan a *OWL DL* y *OWL Lite* un conjunto de algoritmos con una complejidad polinomial para cualquier razonamiento que quiera realizar un agente mediante una expresión valida en *OWL DL* sobre cualquier ontología. Es decir, se sabe que todos los razonamientos de los agentes expresados en *OWL DL* sobre cualquier ontología podrán ser realizados sin que su coste de procesamiento sea exponencial, ni en consumo de recursos ni en tiempo. Por lo tanto, los sistemas terminológicos proporcionan un conjunto de algoritmos eficientes a los agentes que les permiten razonar autónomamente.

3. Algoritmos de razonamiento: los sistemas basados en lógica descriptiva utilizan un conjunto de algoritmos bien definidos que son conocidos a priori. Por lo tanto, el comportamiento de los agentes que utilicen estos sistemas para razonar es conocido con antelación, independientemente del sistema terminológico utilizado.

La utilización de los sistemas terminológicos para razonar sobre el conocimiento almacenado en las ontologías tiene las siguientes ventajas:

1. La utilización de un sistema terminológico permite que los agentes razonen tanto acerca de las clases (conceptos) y propiedades (roles) que constituyen la ontología; y además, permite extraer conocimiento de los individuos (instancias) que los constituyen.

   Por ejemplo, si se define una ontología como la mostrada en la Figura 2.13 y las siguientes instancias de la clase persona especificada en dicha ontología: 'carlos'con nombre 'carlos'y ocupación 'becario', 'yolanda'con nombre 'yolanda'y ocupación 'agente', 'sergio'con nombre 'sergio'y ocupación 'currante', 'jaroyo'con nombre 'jaroyo'y ocupación 'jefe', y 'edu'con nombre 'edu'y ocupación 'jefazo'. A continuación, se define el proyecto 'Web'con titulo 'Web', en el que todos sus miembros son las personas con nombre 'jaroyo', 'yolanda' o 'edu'. A continuación, el sistema podría descubrir que Web es una instancia de la clase superpro que tiene como jefe a 'jaroyo'.

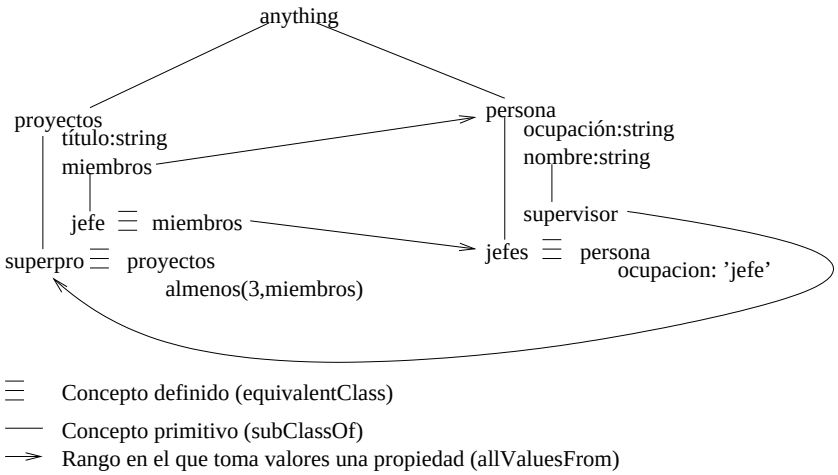


Figura 2.13: Ontología de proyectos

2. Las respuestas que el sistema terminológico devuelve a los agentes indican porqué su pregunta es cierta o falsa. Es decir, no se devuelve una respuesta que indique cierto o falso, sino que además se indica por qué la pregunta formulada por el agente es cierta o falsa.

   Continuando con el ejemplo mostrado en la Figura 2.13, si se pregunta quién es el jefe de al menos un superpro: el sistema responde que la instancia 'jaroyo'de la clase persona, porque es el jefe de un proyecto con al menos tres miembros.

3. La utilización de un lenguaje aceptado por un sistema terminológico permite formular preguntas complejas de forma muy simple. Por ejemplo, si un agente o usuario pregunta por aquellas mujeres que tienen hijas y no hijos, construirá la siguiente expresión:

$$(AND\ woman\ (SOME\ hasChild\ female)\ (NOT\ (SOME\ hasChild\ male)))$$

Sin embargo, debido a que la pregunta formulada por el usuario o por un agente debe ser inteligible para agentes o componentes software, su representación en OWL es la mostrada en la Figura 2.14. Complicando de esta forma que las expresiones en OWL puedan ser inteligibles para las personas, aunque puede diseñarse un agente traductor que facilite dicho proceso.

```
  <owl:ObjectIntersectionOf>
    <owl:OWLClass owl:URI="#woman"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild" />
      <owl:someValuesFrom rdf:resource="#female" />
    </owl:Restriction>
    <owl:complementOf>
      <owl:Restriction>
  <owl:onProperty rdf:resource="#hasChild" />
  <owl:someValuesFrom rdf:resource="#male" />
</owl:Restriction>
    </owl:complementOf>
  </owl:ObjectIntersectionOf>
```

Figura 2.14: Ontología especificada en OWL

4. Además, debido a que los sistemas terminológicos son capaces de realizar simplificaciones, dada una pregunta formulada por el usuario se puede buscar otra que sea semánticamente equivalente pero que sea más simple. Debido a que los sistemas terminológicos pueden calcular la expresión mínima que es equivalente a una dada. Por ejemplo, si un usuario quiere obtener de una base de datos bibliográfica aquellos libros sin ISBN y que tengan más de dos autores, el sistema terminológico podría indicar (sin acceder a los datos) que no hay ningún concepto que satisfaga dichas condiciones ya que es imposible que un libro no tenga ISBN debido a que la ontología asociada al depósito bibliográfico define un libro como una publicación con ISBN. Si se desea mayor información acerca de las optimizaciones que puede realizar un sistema terminológico se recomienda consultar [Goñ95].

Sin embargo, también existe el inconveniente de que la sintaxis utilizada por los sistemas terminológicos es compleja para los humanos y tienen problemas de escalabilidad cuando la ontología utilizada en los razonamientos tiene un tamaño desmesurado (decenas de miles de términos e instancias). Afortunadamente, mediante la utilización de *OWL* las ontologías pueden dividirse en distintos módulos y utilizar para razonar solamente la parte de la ontología que está implicada en los razonamientos solicitados por el agente software.

2.4.3. Arquitecturas distribuidas basadas en ontologías

En el pasado, la investigación en el contexto de bases de datos se centró en entornos relativamente estáticos, representados en la Figura 2.15, que podían ser: 1) centralizados: existe

un único gestor de base de datos que almacena la información de forma centralizada; 2) distribuidos: está constituido por un único gestor de base de datos que almacena la información distribuida en distintos sistemas de almacenamientos conectados mediante una red de comunicaciones; 3) federados: es un sistema de base de datos que integra múltiples sistemas de bases de datos autónomos (que pueden tener distintos esquemas relacionales) y distribuidos en una red proporcionando una visión totalmente integrada y que incluye la visión lógica de todas las bases de datos que integra; 4) interoperables: es una base de datos federada en la que los depósitos de datos que la constituyen no siempre están basados en un sistema gestor de bases de datos sino que permite acceder a cualquier tipo de depósito de datos pudiendo tener cada uno de los mismos distintas capacidades de interrogación [MI01], ó 5) dinámicos: son sistemas gestores de bases de datos en los que las relaciones se especifican al mismo tiempo que la pregunta y la ubicación de los registros relacionados con la pregunta es descubierta en tiempo de ejecución.

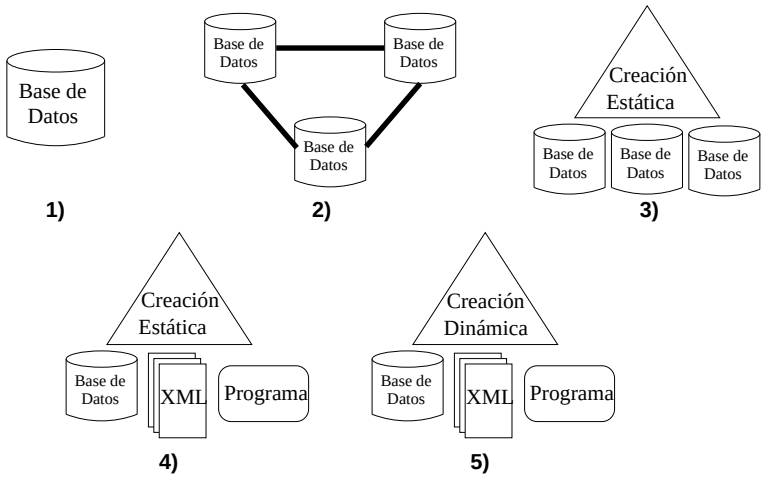


Figura 2.15: Arquitecturas distribuidas basadas en ontologías

En estos entornos de ejecución, se definía una aproximación centralizada o distribuida que gestionaba los datos de una forma consistente. El mecanismo más comúnmente utilizado para enlazar los datos y las operaciones de acceso es mediante un esquema que representa la información disponible en las bases de datos subyacentes, conocido a priori o cuando se producía la integración de distintas bases de datos. Debiendo realizarse la integración de los distintos sistemas de bases de datos de una forma manual y no pudiendo ser realizada de forma automática por un conjunto de agentes inteligentes.

Sin embargo, la aparición de la Web necesita de una aproximación diferente y da lugar a nuevos retos, por ejemplo realizar la integración de distintos depósitos de datos de forma automática. Esta aseveración se fundamenta en la ausencia de un sistema centralizado o federado de bases de datos debido a que cualquiera puede hacer disponible nueva información y en el formato que considere más adecuado, produciendo un incremento exponencial de la información. En los entornos de bases de datos federadas [HM85, SL90, She91] las relaciones

entre el esquema y las bases de datos son preestablecidas; con la aparición de la Web la definición de las relaciones entre los depósitos de datos y el modelo conceptual que representan debe realizarse de forma automática o semi-automática. Por lo tanto, los nuevos sistemas de información deberán poder ser generados dinámicamente por programas/agentes software, independientemente de su localización, estructura, lenguaje de interrogación y semántica de los datos almacenados en los distintos almacenes de datos.

Por lo tanto, con la aparición de la web semántica [BLHL01, IHMT04] debe permitirse no solamente el acceso a bases de datos (centralizadas, distribuidas, federadas o interoperables) sino a cualquier tipo de depósitos de datos, tanto a personas como a agentes inteligentes. Es decir, se debe proporcionar un mecanismo de acceso a cualquier tipo de depósito de datos independientemente de su localización y su estructura [SZ97] (destinado tanto a personas como a agentes software). Debido al gran dinamismo del contexto en el que nos encontramos deberemos adaptar las técnicas utilizadas en el contexto de bases de datos federadas, que permiten la integración de información almacenada en diferentes gestores de bases de datos preexistentes. Es decir, la información de enlace [BGI99] que define la definición de relaciones entre el modelo conceptual de la información almacenada en los depósitos de datos y su representación semántica, deberá generarse de forma automática o semi-automática, por agentes inteligentes. Además, debe considerarse el aumento de la cantidad de información accesible y el acceso desde dispositivos y redes con distintas prestaciones.

Para diseñar un sistema de información que permita la integración de distintos almacenes de datos y que proporcione una descripción semántica de los mismos, permitiendo ser accedido por personas y máquinas, se pueden seguir dos aproximaciones:

- *Visión Local*: Esta aproximación es conocida como *Local as View* [MI01]. Mediante este tipo de arquitecturas el conjunto de depósitos de datos accesibles por el sistema son descritos utilizando distintas ontologías (ver Figura 2.16). El problema de esta aproximación es como traducir una pregunta expresada utilizando las clases y propiedades de una ontología a otra ontología. Para la realización de este proceso existen distintas aproximaciones que pueden considerar traducciones parciales o incluso pérdida de información. Definiendo, pérdida de información como la traducción de una clase por otra o una combinación de clases y propiedades que no es un sinónimo de la primera sino un hiperónimo o hipónimo de la clase que queremos traducir [MI01].

- *Visión Global*. Esta aproximación es conocida como *Global as View* [MI01]. En esta aproximación se desarrolla una única ontología que describe todo el universo del sistema (ver Figura 2.17). Presentando el inconveniente de que se puede crear una ontología demasiado grande para ser manejable. Aunque se pueden aplicar técnicas para reducir su tamaño, tal y como se ha hecho en el servicio de recuperación de software diseñado [MRIG02b], esta aproximación se detallará posteriormente, en el capítulo 4.

Sin embargo, la característica fundamental es que, independientemente de la aproximación que se siga para diseñar y desarrollar un sistema de información que proporcione una descripción semántica de la información que contiene, los agentes software que lo accedan en la mayoría de los casos, tendrán que poder razonar con la información que contiene y relacionarla con otros sistemas de información. Por ejemplo, relacionando la información de

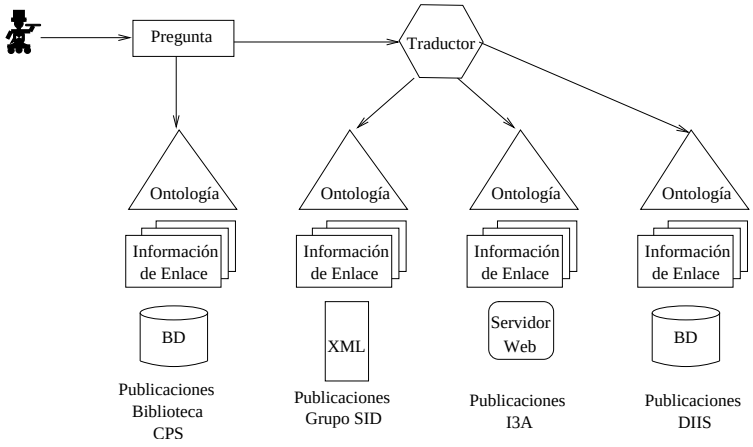


Figura 2.16: Sistema de información basado en varias ontologías locales

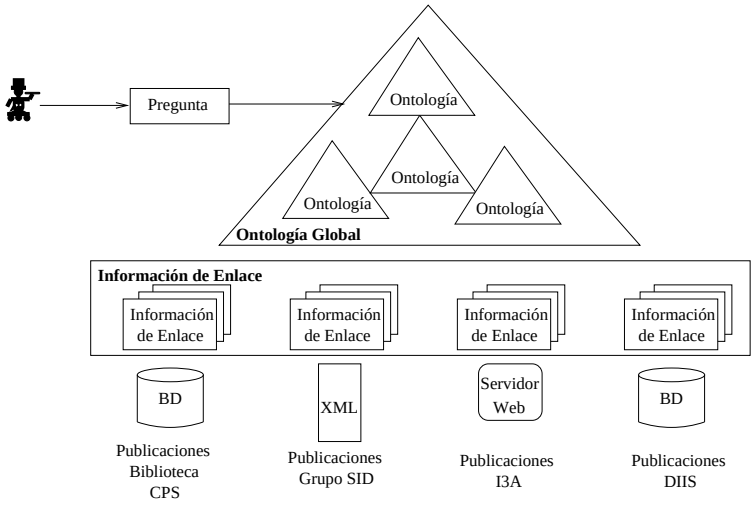


Figura 2.17: Sistema de información basado en una ontología global

una empresa de venta de productos alimenticios con una biblioteca digital que proporcione libros con recetas de cocina. Además, debido a la gran cantidad de información disponible en la Web, este proceso tiene que ser realizado por agentes inteligentes de forma autónoma y automática debido a la imposibilidad de que una persona pueda analizar tanta información y extraer conclusiones sobre la misma en un breve espacio de tiempo.

2.5. Resumen del capítulo

En este capítulo se han presentado las distintas tecnologías que afectan al estudio del diseño de aplicaciones distribuidas en entornos inalámbricos y distribuidos. Este estudio comienza realizando una descripción detallada de los distintos protocolos de comunicaciones inalámbricas basadas en radio frecuencia, tanto de corto como de largo alcance. Destacando una característica común a ambos tipos de redes como es la inestabilidad, provocando frecuentemente la desconexión de los dispositivos de la red. Esta característica, como se ha mostrado anteriormente, hace necesario el estudio de nuevas aproximaciones que permitan resolver este problema.

Posteriormente se ha realizado una descripción pormenorizada de las distintas aproximaciones que pueden seguirse en el diseño de aplicaciones distribuidas, tanto en la arquitectura de las mismas como en la organización de los datos y conocimiento gestionado por éstas.

- Tecnologías de comunicaciones inalámbricas de largo alcance: se ha presentado un estudio de las distintas tecnologías de red utilizadas actualmente en entornos de telefonía móvil. Se han detallado los protocolos de comunicaciones basadas en radio frecuencia de largo alcance como son: GSM, GPRS o UMTS. Destacando que este tipo de redes tiene un elevado coste, independientemente de que el usuario sea facturado por tiempo o por cantidad de información transmitida mediante el enlace inalámbrico.

- Tecnologías de comunicaciones inalámbricas de corto alcance: se ha presentado un estudio de las tecnologías basadas en radio frecuencia, como Bluetooth, WiFi, WIMAX y HomeRF. Este tipo de protocolos de comunicaciones son utilizados en el interior de edificios o zonas reducidas debido a las limitaciones en la potencia de emisión. A diferencia de las redes inalámbricas que utilizan comunicaciones inalámbricas de largo alcance estas redes son gratuitas debido a que suelen ser instaladas por empresas o usuarios domésticos y no por proveedores de servicios. Por lo tanto, el diseño de servicios en este tipo de contextos no debe considerar el coste de facturación sino la optimización de los recursos disponibles.

- Programación en entornos distribuidos: se ha estudiado la problemática del desarrollo de aplicaciones distribuidas y las distintas arquitecturas que pueden ser utilizadas (cliente/servidor, *peer-to-peer*, servicios web y/o sistemas multiagente). También se han detallado las ventajas de los sistemas basados en agentes frente a otras aproximaciones.

- Sistemas de acceso a datos basados en ontologías: en esta sección se han descrito las ventajas que proporcionan las ontologías en el diseño de sistemas de acceso a datos distribuidos. Al igual que se detallan los distintos lenguajes disponibles para la especificación de ontologías, como DAML+OIL y OWL. También se ha descrito, cómo estos lenguajes son utilizados en el diseño de servicios de acceso a datos para especificar semánticamente el contenido de los distintos depósitos de datos utilizados y posteriormente facilitar su acceso.

Capítulo 3

Agentes software inteligentes

En este capítulo presentamos la tecnología de agentes como una aproximación interesante para el diseño de aplicaciones de computación móvil en entornos inalámbricos y de computación distribuida. En primer lugar, describiremos qué es un agente software y cuáles son sus características principales, junto con la importancia de los sistemas multiagente (SMA), en los que distintos agentes, autónomos e inteligentes, colaboran y se coordinan para alcanzar un objetivo.

A continuación, describimos las características de los agentes móviles, junto con los contextos en los que se utiliza la tecnología de agentes. En este punto será necesario detallar la diferencia entre qué es un agente móvil, qué es código móvil, y los distintos tipos de movilidad que pueden tener los agentes. El tipo de movilidad de un agente móvil, como se verá posteriormente, en muchos casos dependerá del lenguaje de programación de la plataforma de agentes móviles que se esté utilizando.

Posteriormente, presentamos los nuevos tipos de arquitecturas que pueden diseñarse gracias a la tecnología de agentes y que se han desarrollado como extensiones a la arquitectura cliente/servidor tradicional, como por ejemplo la arquitectura cliente/interceptor/servidor y otras. Las distintas extensiones que se han realizado de la arquitectura cliente/servidor permiten una fácil adaptación de la misma a entornos inalámbricos. Para verificar la ventajas de las arquitecturas basadas en agentes se presentará un conjunto de medidas de prestaciones que compararán aproximaciones clásicas como cliente/servidor, HTTP, HTTPS, etc. con aproximaciones basadas en agentes. Finalmente se presentarán distintas plataformas de agentes, detallando las características y funcionalidades de las mismas.

3.1. Agentes software y sistemas multiagente: definición y propiedades

Un *agente* se define de forma diferente dependiendo del área de conocimiento en la que nos encontremos, y a veces hay muy poca diferencia entre qué es un agente y qué no [FG96]. A continuación presentaremos distintas definiciones de agente en distintos entornos:

- Según [PN03] se define un agente como "cualquier ente que puede ser visto percibiendo su entorno a través de sensores y realizando efectos que modifiquen dicho entorno".
- En [Mae95], "una agente autónomo se define como un sistema computacional que habita en algunos entornos complejos y dinámicos, siente y actúa autónomamente en dichos entornos, y realizando estas tareas alcanza las metas para las que fue diseñado".
- En [SCS94] definen "una agente como una entidad software persistente dedicada a un propósito específico. La persistencia distingue a los agentes de los procedimientos; los agentes tienen sus propias ideas acerca de cómo llevar a cabo tareas de sus propias agendas de ejecución. Su propósito específico los distingue de aplicaciones multifunción; los agentes típicamente son más pequeños".
- Según [HR95] "los agentes inteligentes realizan tres funciones continuamente: percepción de condiciones dinámicas en su entorno, reaccionan a las condiciones que afectan a su entorno y razonan para interpretar sus percepciones, resolviendo problemas, generando inferencias, y acciones determinadas".
- En [WNRJ95] definen un agente como "un sistema hardware o software que verifica las siguientes propiedades:
  - Autonomía: los agentes operan sin la intervención directa de las personas o de otros agentes, y tienen la capacidad de controlar sus propias acciones y su estado interno.
  - Cooperación: los agentes colaboran con otros agentes (y, posiblemente, con humanos) a través de un lenguaje de comunicación entre agentes.
  - Reactivos: Los agentes perciben su entorno, que puede ser el mundo físico, un usuario mediante la utilización de un GUI, una colección de agentes, Internet, o quizá una combinación de todos éstos, y responden a cambios que se produzcan en el entorno.
  - Finalidad: los agentes no solamente actúan en respuesta a su entorno, sino que también tienen un objetivo a cumplir y realizan las acciones necesarias para alcanzar este objetivo."
- Según [Acr05] un agente es una abstracción de alto nivel que conoce inherentemente cómo realizar una cierta tarea, como por ejemplo, comunicarse con otros agentes, monitorizar el entorno, razonar, tomar decisiones acerca de su estado e incluso moverse a otros ordenadores.

En nuestra opinión, y unificando todas las definiciones presentadas anteriormente, los agentes pueden ser considerados como *componentes software que ejecutan acciones por el usuario* y tienen algunas o todas de las propiedades siguientes:

- *Autonomía* [FG96, TA03, TA04]: capacidad de ejecutar acciones sin la intervención directa de las personas.

- *Sociabilidad* [SS02]: capacidad de pensar sobre sí mismos y sobre otros agentes software (agente altruista o egoísta). Es decir, un agente puede elegir establecer una relación con otro agente para obtener beneficio o para ayudar a otros agentes sin obtener ninguna contraprestación por sus servicios.
- *Reactividad* [MDF$^+$02]: capacidad de responder a cambios en el entorno.
- *Dirigidos hacia una finalidad* [RM00]. Es decir los agentes tienen un objetivo que tratan de alcanzar utilizando todos los medios disponibles a su alcance.
- *Continuidad temporal* [FG96]: persistencia durante periodos largos de tiempo.
- *Aprendizaje/Adaptatividad* [PO02, FG99]: capacidad de aprender del entorno de ejecución y de otros agentes para mejorar sus prestaciones.
- *Raciocinio* [RN03]: poseen mecanismos para la toma de decisiones.
- *Movilidad* [HCK97]: capacidad de migrar a otros ordenadores.
- *Cooperación* [Les99, FTS$^+$03]: interacción con otros agentes para conseguir una meta.
- *Negociación* [Smi77, LMNC04]: por ejemplo para solicitar la ejecución de diferentes subtareas a otros agentes para conseguir su objetivo.

Sin embargo, como se indica en [fRSSiTP98], no todas las propiedades deben estar presentes en todas las clases de agentes, con la excepción de la autonomía, la cual es considerada una propiedad inherente a los agentes software. Comparando un agente con un objeto de ingeniería del software tradicional [BD03], un agente tiene el control sobre sus propias acciones (ejecución) y encapsula no sólo su estado sino también su comportamiento. Sin embargo, en los agentes software inteligentes deben confluir al menos las siguientes tres propiedades: aprendizaje, cooperación y autonomía.

Basándonos en las propiedades mencionadas anteriormente, se pueden realizar distintas clasificaciones de los agentes. Por ejemplo, considerando la movilidad, los agentes son clasificados como estáticos o dinámicos. Para evitar excesivos tipos de agentes, en [Nwa96] se recomienda clasificar los agentes conforme a sus características predominantes (en lugar de considerar las secundarias, como, por ejemplo, agentes de visualización o ayuda): autonomía, aprendizaje y cooperatividad son las propiedades básicas para realizar su clasificación. Atendiendo a estas propiedades, los agentes, tal y como se muestra en la Figura 3.1, se clasifican en: agentes colaborativos, agentes cooperativos, agentes de interfaz y agentes inteligentes. En [WJ95], los agentes que poseen las propiedades de autonomía, sociabilidad, reactividad, tiene un fin y tienen pensamientos[1], son considerados *agentes fuertes*, mientras que los *agentes débiles* no tienen pensamientos. Finalmente, en [M̈98] se presenta una taxonomía basada en las áreas de aplicación de los agentes.

Recientemente, ha aparecido la *Ingeniería del Software Orientada a Agentes* como un nuevo paradigma en el desarrollo de software [LAD04, BGZ04]. Esta metodología trata de

[1] Es decir, poseen pensamientos, deseos, intenciones (*beliefs, desires, intentions —BDI agents—*) [CG01, Bra87], y conocimiento.

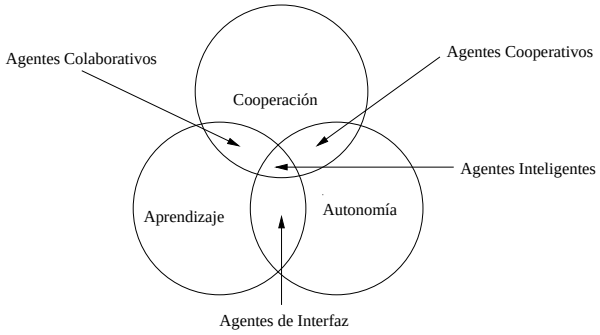


Figura 3.1: Propiedades de los agentes software inteligentes

resolver los problemas descomponiéndolos en distinta subtareas que puedan ser ejecutadas de forma autónoma o mediante cooperación para cumplir el objetivo final; la filosofía detrás de esta aproximación se parece a la estrategia de "divide y vencerás" [BG99], que se popularizó debido a su gran aplicación práctica. Los diseños orientados a agentes son muy atractivos porque ayudan a los diseñadores a resolver problemas complejos mediante la especificación del comportamiento de los agentes. Dentro de la Ingeniería del Software Orientada a Agentes, los investigadores se centran en diferentes problemas como: arquitecturas de agentes, plataformas de agentes, metodologías y lenguajes de modelado, estandarización de diferentes aspectos de la tecnología de agentes (FIPA [fIPA98, FfIPA05] ha sido el proceso de estandarización más importante), distributividad, etc. Desde el punto de vista del diseñador, los elementos más importantes son las plataformas de agentes [HB00, FGR02, Sha05, BBD$^+$06] (por ejemplo, JatLite [JPC00], Zeus [CNNL98], FIPA-OS [PBH00], JADE [BPR01], SPRINGS [ITM06], etc.). Estas plataformas permiten el desarrollo de sistemas basados en agentes y proporcionan herramientas para interoperar con los agentes, en [LAD04] se consideran las plataformas de agentes como el sistema operativo para un ordenador. En la sección 3.3 describimos detalladamente algunas de las plataformas de agentes disponibles actualmente.

Los Sistemas Multiagente (SMA) [SZ96, Syc89, DL01, Woo02, Wei99] son "sistemas software en los que varios agentes semi-autónomos interactúan o trabajan juntos para llevar a cabo un conjunto de tareas o satisfacer una finalidad" [Les95]. El diseño de sistemas multiagente implica considerar el comportamiento cooperativo de los sistemas, además de las capacidades y estructura de los agentes individuales. El uso de múltiples agentes cooperativos [Les99] para resolver problemas complejos ofrece ventajas importantes [TS00, SV00]. En particular, permiten incrementar la escalabilidad, modularidad y robustez de las arquitecturas, al igual que facilitan el procesamiento paralelo en varios ordenadores, incrementan la tolerancia a fallos, y ocultan los detalles de la infraestructura subyacente. Aunque los sistemas multiagente pueden ser beneficiosos incluso en contextos con un único procesador [SV00], han sido ampliamente usados en aplicaciones inherentemente distribuidas, como recuperación de información [BGM$^+$99], diagnóstico de redes [BPW98], y monitorización distribuida de vehículos [HMS$^+$03, IMI06]. Otros contextos donde se han utilizado los sistemas multi-

agente son: en fabricación [SNBN01, BJW04], servicios web [CMMB05] y, en general, en contextos que requieren adaptación y un comportamiento inteligente [ABJ+98, MIRG06]. También se ha sugerido que los agentes pueden ser útiles en la implementación de servicios móviles [KR04, MIRG06].

Además, destacar la *Fundación para los Agentes Físicos Inteligentes*[2] [FfIPA05] formada en 1996 para desarrollar estándares de software para agentes heterogéneos e interactivos y sistemas basados en agentes. Además, para realizar la gestión de los agentes se necesita un lenguaje de comunicación para agentes —*Agent Communication Language* (*ACL*)— que permite comunicarse con una semántica y sintaxis precisas. El lenguaje ACL de FIPA está basado en la idea de que los mensajes que se envían entre los agentes son *actos comunicativos*, debido a que están destinados a realizar una acción específica. Además de FIPA, hay otras especificaciones de agentes como *OAA* (*Open Agent Architecture*) [IAIC06]. Otras propuestas de lenguajes de comunicación son: *KQML* (*Knowledge Query and Manipulation Language*) [FFMM94], *ICL* (*Interagent Communication Language* proposed by OAA), *KIF* (*Knowledge Interchange Format*) [ANS06], y *Ontolingua* [SU06, FLS96].

3.2. Agentes móviles

Los agentes móviles son un paradigma de computación distribuida [SSPE04b]. En esta sección, comentaremos la evolución de los paradigmas distribuidos, desde el cliente/servidor a los agentes móviles, comentando el concepto de código móvil. Posteriormente, describiremos las principales características y escenarios de aplicación de los agentes móviles. Finalmente, detallaremos el concepto de plataforma de agentes móviles y algunos problemas para su estandarización.

3.2.1. Código móvil

En la arquitectura *cliente/servidor* tradicional, un servidor en un cierto ordenador ofrece un conjunto de servicios de interés para distintos usuarios o procesos. Para utilizar dichos servicios se necesita ejecutar las tres siguientes tareas: 1) un cliente desde otro ordenador (o desde el mismo) realiza una petición de ejecución del servicio para interactuar con el servidor, 2) el servidor lleva a cabo la petición del servicio, y 3) el servidor devuelve el resultado o resultados al cliente.

Actualmente han aparecido nuevas aproximaciones, como la *movilidad de código* (capacidad de cambiar dinámicamente los enlaces entre los fragmentos de código y la localización donde éstos son puestos en ejecución [FPV98]). El paradigma de agentes móviles puede ser considerado una evolución del paradigma de evaluación remota y de código bajo demanda:

- *Evaluación Remota* [SG90]. Un componente software conoce cómo invocar un servicio, pero carece de los recursos necesarios para ello, que están disponibles en un sitio remoto. En este escenario: 1) el componente envía el código para ejecutar el servicio a otro componente en un ordenador remoto, 2) el componente remoto lleva a cabo la ejecución del servicio, y 3) devuelve los resultados al componente que realizó la petición.

[2] *Foundation for Intelligent Physical Agents* (*FIPA*).

- *Código bajo demanda*. Un componente en un cierto ordenador tiene suficientes recursos para llevar a cabo la ejecución de un servicio pero no sabe cómo hacerlo. En este caso: 1) pide a otro componente (que se ejecuta en otro ordenador) el código necesario para poder ejecutar el servicio, 2) el componente que recibe la petición proporciona el procedimiento que permitirá la ejecución del servicio, y 3) el componente que realizó la petición ejecuta el procedimiento.

  Un ejemplo muy conocido de este paradigma son los applets Java [SM06]. El código de un applet Java reside en el servidor web. Cuando un cliente realiza una petición de la página web que contiene el enlace al applet, el código del applet es transmitido al cliente vía HTTP. Entonces, el applet es ejecutado localmente en el navegador del usuario.

El paradigma de agentes móviles es una evolución de estos dos paradigmas, debido a que un agente móvil es un componente software que decide por sí mismo, cuándo y dónde moverse para llevar a cabo sus tareas.

3.2.2. Definición y características de los agentes móviles

Los agentes móviles [MBB+98, MDW99, LO99b, HCK97, BR05] son programas que se ejecutan en contextos denominados *places*[3] y que autónomamente pueden viajar de *place* a *place* reanudando su ejecución. Los agentes móviles no están limitados por los límites del ordenador donde se están ejecutando cuando son creados; por consiguiente, pueden moverse libremente entre *places* en diferentes ordenadores. Se puede considerar dos tipos de movilidad de los agentes móviles:

- *Movilidad fuerte* implica que un agente se lleva su código (las clases necesarias para que el agente pueda ejecutarse), los datos del estado (los atributos que representan los pensamientos, deseos e intenciones del agente [RG95]), y el estado de ejecución (incluyendo los hilos de ejecución o *threads*) cuando viaja a otro *place*. Por lo tanto, el agente reanudará su ejecución en el mismo punto (exactamente) donde se paró, en el *place* anterior, para poder viajar.

- *Movilidad débil* [FPV98] implica que el agente lleva consigo su código y datos del estado, pero no su estado de ejecución. Por lo tanto, cuando llegue al nuevo *place* el agente reanudará su ejecución desde el comienzo de la *invocación de un método* específico.

Aunque la movilidad fuerte es más atractiva, su implementación no es posible utilizando la máquina virtual de Java estándar [LY99], y Java es el lenguaje de programación preferido para desarrollar plataformas de agentes móviles debido a su portabilidad. Por consiguiente, el lenguaje Java estándar no permite modificar la pila de ejecución, lo cual es necesario para restaurar el estado de ejecución de un agente móvil que llega a un *place*. Además, se ha probado que la movilidad débil es suficiente en la mayoría de los escenarios [BN01].

[3] Mantenemos la nomenclatura anglosajona debido a que en entornos científicos de habla hispana al contexto de ejecución donde se ejecutan los agentes también se le denomina *place*.

Por consiguiente, la mayoría de las plataformas de agentes móviles proporcionan movilidad débil. Querríamos destacar que Java proporciona técnicas para almacenar el código utilizado previamente en la memoria del sistema como parte de su mecanismo de carga dinámica de clases; por lo tanto, dependiendo de la estrategia de migración implementada, sólo el código que no ha sido cargado anteriormente por la máquina virtual Java será transmitido a través de la red de comunicaciones [BK05].

También destacar que, gracias a la movilidad, los agentes móviles ofrecen muchos beneficios interesantes [LO99b, PMI03, GKCR00]:

1. *Encapsulan el protocolo de comunicaciones*. Como se pueden mover a ordenadores remotos para conseguir sus objetivos, evitan la necesidad de instalar procesos servidores especializados en todas las máquinas que proporcionan acceso a *todos los servicios que necesita el agente*. En su lugar, *un único proceso servidor* (la plataforma de agentes móviles) es necesario, y *muchos agentes distintos* pueden viajar a los diferentes ordenadores que ejecutan las funcionalidades requeridas para llevar a cabo sus objetivos.

2. *Reduce el tráfico de red*, debido a que los agentes móviles pueden viajar a donde se encuentran localizados los datos y acceder a los mismos localmente y realizar un filtrado de los datos que no son necesarios (evitando de esta forma su envío a través de la red). De esta forma, además, se traslada la carga de ejecución al ordenador en el que se encuentran los datos, en lugar de solicitar los datos y posteriormente procesarlos en otro ordenador, mediante este mecanismo pueden ahorrarse muchos recursos cuando deben ser analizadas cantidades ingentes de datos.

3. *Disminuyen la latencia de la red*. Debido a que los agentes pueden moverse de un ordenador a otro se pueden optimizar los tiempos de respuesta de la red (en [RASS99], utilizan el término *programa software que considera la red*). La capacidad de los agentes móviles para reducir la latencia es crítica en sistemas de tiempo real, tales como robots en procesos de fabricación [LO99b]. De esta forma, lo agentes pueden ser enviados de un controlador central a un cierto ordenador para interactuar localmente, evitando múltiples interacciones entre componentes remotos y, por lo tanto, ahorrando comunicaciones remotas [BW04].

4. *Son asíncronos y autónomos*. En las arquitecturas cliente/servidor tradicionales el cliente debe mantener la conexión abierta mientras su petición de servicio está siendo procesada por el servidor. Por lo tanto, si la conexión falla, el cliente tiene que enviar una nueva petición al servidor por segunda vez, que volverá a ser procesada desde el principio. Alternativamente, un agente móvil no necesita mantener abierta una conexión con el ordenador en el que fue creado (o desde el que viajó anteriormente) mientras realiza una tarea. Esto es particularmente importante en redes inalámbricas, debido a que son inestables. De la misma manera, un dispositivo móvil puede enviar un agente móvil a un ordenador de la red cableada y apagarse. El agente se hace independiente del dispositivo que lo creó o desde el que viajó y, por lo tanto, se hace más flexible la forma en la que se pueden llevar a cabo las tareas en un entorno distribuido y heterogéneo, como es la Web, o inestable, como son las redes inalámbricas. Cuando el dispositivo

móvil restablezca la conexión, podrá comunicarse con el agente móvil para obtener los resultados.

5. *Se adaptan dinámicamente a su entorno* [LO99b, ABCB+04, HKGA05]. Los agentes móviles analizan su entorno y reaccionan autónomamente adaptándose a los cambios del mismo. Por ejemplo, pueden viajar a otro ordenador cuando el ordenador en el que se están ejecutando está sobrecargado, permitiendo el balanceo de carga, o si le queda poca batería, en el caso de dispositivos móviles.

6. *Facilitan la integración de sistemas poco acoplados*. Habitualmente, los componentes hardware y software son muy heterogéneos en una red. Los agentes móviles ayudan a superar los problemas de heterogeneidad porque generalmente son independientes del ordenador e independientes del nivel de transporte. Por ejemplo, en [SSEP99] proponen el uso de un agente móvil que configure una conexión JDBC [BEF03] en su *place* de ejecución para acceder a una base de datos remota (para ello el agente viajará al ordenador en el que se encuentra la base de datos y a continuación realizará la conexión de forma local, donde además están disponibles los conectores JDBC apropiados para acceder a dicho gestor de bases de datos); una aproximación sin agentes móviles necesitaría descargar los conectores JDBC apropiados para el gestor de base de datos e interrogar la base de datos remotamente.

7. *Son robustos y tolerantes frente a fallos* [LO99b]. Su capacidad para reaccionar dinámicamente frente a situaciones desfavorables y eventos externos les hace más fácilmente robustos y tolerantes a fallos en entornos distribuidos. Si un ordenador se está apagando, todos los agentes que estén ejecutándose en él podrán ser alertados y tener tiempo para moverse y continuar con la ejecución de sus tareas en otro ordenador de la red, siempre que no necesiten recursos que solamente estén disponibles en dicho ordenador.

Además, también poseen unas *buenas prestaciones* comparados con las tradicionales arquitecturas cliente/servidor [Ade04]. Por ejemplo, en [SSPE04a] evalúan el ahorro introducido por la utilización de agentes móviles cuando se interactúa con bases de datos remotas en entornos inalámbricos. En [GKP+01, RM01a], determinan experimentalmente los beneficios de los agentes móviles en el contexto de la recuperación de información (los documentos son filtrados en un ordenador remoto que almacena los depósitos de datos). Finalmente, evaluamos varias estrategias para descargar ficheros en una red cableada y mostramos cómo los agentes móviles proporcionan las mismas o mejores prestaciones que aproximaciones basadas en CORBA o FTP (ver sección 3.4.4).

3.2.3. Contextos de aplicación

Los agentes móviles se han utilizado en muchos contextos o entornos de ejecución diferentes, como son:

- *Recuperación de información distribuida* [BGM+99, YPHM98, GKP+01, KSCP04]. Los agentes pueden moverse a las fuentes de información remotas para recopilar la

información relevante (por ejemplo, crear índices de búsqueda). Esto reduce la sobrecarga de red y no requiere que el ordenador en el que se creó el agente esté en funcionamiento mientras los agentes realizan sus tareas. En [SDSL99] utilizan agentes móviles para acceder a bases de datos distribuidas, y en [QIC01b, QIC01a] los usan para fusionar datos en una red de sensores.

- *Procesamiento paralelo* [SBMS99, GHCN99, GSS06, FTS+03]. Los agentes móviles pueden ser creados y reenviados a la red para realizar diferentes tareas en paralelo. Además, un agente móvil puede clonarse a sí mismo para paralelizar sus propias tareas.
- *Aplicaciones de monitorización y notificación* [TVP00, Ken00, IKT04, LMN02]. Éste es un uso clásico de los beneficios obtenidos gracias a la naturaleza asíncrona de las comunicaciones utilizadas por los agentes móviles: un agente puede moverse a un sitio remoto para monitorizar localmente los datos y notificar al ordenador que lo creó cuando los datos satisfacen ciertos requisitos. Por ejemplo, esta aproximación ha sido utilizada en sistemas de gestión de redes para medir prestaciones. Otro ejemplo de uso se muestra en [LKR02], donde se presenta el *RTMonitor*, un sistema de gestión de datos en tiempo real para aplicaciones de gestión de tráfico basado en agentes móviles.
- *Asistentes personales* [VM04a, CTC04, PS04]. Un agente móvil ejecuta acciones en nombre de su creador y, por lo tanto, actuando como un "mayordomo personal" [LO99b]. Por ejemplo, un profesor podría enviar a un agente móvil para interactuar con los agentes que representan a sus estudiantes para organizar las tutorías o una corrección de prácticas.
- *Diseminación de información* [ZHS+04, RRP04, BMP+04, SS05a, SHD+06]. El modelo *push* [EFGK03] (que permite el envío de la misma información a muchos clientes) puede ser fácilmente implementado utilizando agentes móviles [XW00]. De la misma forma, los agentes móviles pueden diseminar información, como noticias [Fie98] o actualizaciones de software [BNL02] a través de la red.
- *Comercio electrónico* [VMK98, HL02, WTR05, HL05, Lau05, BAV05, PRU06]. Los agentes móviles puede proporcionar acceso en tiempo real a información remota (por ejemplo, información bursátil) y representar al usuario en una negociación entre agentes. Como ejemplo, *MAgNET* [DNMMS99] es un sistema para venta electrónica a través de la red basado en agentes móviles.
- *Gestión de redes* [KMD02, GDP00, SM98, KX02, SRP04, HM04, GS04]. La heterogeneidad, complejidad y dinamismo de las redes de comunicaciones implica que muchas de las aplicaciones que utilizan la red pueden beneficiarse de la utilización de agentes móviles. Por ejemplo, permiten seguir la ejecución de los recursos disponibles de forma flexible. De forma similar, pueden ayudar a proporcionar servicios personalizados a cada usuario. En [KMM99], utilizan los agentes móviles para encaminar paquetes en una red multi-salto (*multi-hop network*) para dispositivos móviles.
- *Aplicaciones de flujo de trabajo (Workflow) y de trabajo en grupo (Groupware)*. Los agentes móviles puede seguir los pasos especificados en un proceso predefinido y moverse de un sitio a otro para recoger información como podría hacer un usuario. Por

lo tanto, los flujos de trabajo pueden modelarse de forma natural utilizando agentes móviles [WWW00]. De forma similar, pueden ayudar en la realización de trabajos cooperativos [Ram04] proporcionando soporte al flujo de información a través de los trabajadores. A modo de ejemplo, en [FWME00] se presenta *WONDER* — Flujos de trabajo en entornos distribuidos (*Workflow ON Distributed EnviRonment*)—.

- *Aplicaciones relacionadas con la bolsa de valores* [vEHKB02]. Los agentes móviles pueden negociar y buscar servicios que encajen con el perfil y requisitos del usuario, considerando las características del dispositivo del usuario y de la red. En [LO99b], indican los beneficios obtenidos debido al diseño de arquitecturas basadas en agentes móviles en entornos colaborativos donde no todas las partes implicadas son confiables. Del mismo modo, los agentes móviles pueden viajar a un ordenador que se considere seguro, donde tiene lugar la colaboración sin riesgo de que el ordenador que proporciona el contexto de ejecución para los agentes tome parte en la transacción favoreciendo a alguno de ellos.

Finalmente, destacar la idoneidad de los agentes móviles a todo tipo de entornos distribuidos. En [SSPE04a, LTKZ03, Car02, ZSX+04, GKN+96, Hor02a, KCG+02, VM04b] se muestran los beneficios del paradigma de agentes móviles en computación móvil y ambiental, debido a que estos entornos tienen un ancho de banda muy limitado y unas redes muy inestables y los agentes móviles permiten la realización de tareas en modo desconectado. También han sido utilizados en entornos inteligentes [Sat04] y en P2P [LG04, Dun01].

3.3. Plataformas de agentes móviles

Una plataforma de agentes móviles es un entorno que permite la ejecución de agentes y les proporciona diferentes servicios, tales como capacidad de movimiento y comunicación[4]; los dos servicios mencionados están interrelacionados. Además, los agentes móviles deben ser capaces de comunicarse entre ellos, mediante llamadas a procedimientos remotos o mediante paso de mensajes, incluso si se mueven entre distintos ordenadores. Se han definido distintos mecanismos para seguir los movimientos de los agentes; por ejemplo, en [AO99] sugieren tres métodos para localizar los agentes (fuerza bruta, monitorización y redirección de mensajes), y en [MLC98] proponen cuatro métodos (actualizar la localización del agente en la agencia en la que se creó, inscribirse en un registro, buscar en las agencias, redireccionar las llamadas). La *transparencia de localización* es una propiedad muy deseable y se define como la capacidad de comunicarse con un agente móvil independientemente de su localización actual. Algunas plataformas usan la idea de *proxy*, que es una abstracción utilizada para comunicarse con un agente; transparencia de localización [MFS06] significa que el proxy redirige los mensajes al agente correspondiente eficiente e independientemente del lugar en el que se encuentre.

Hay muchas plataformas de agentes móviles disponibles, algunas desarrolladas por grupos de investigación y otras por empresas privadas. Muchas de ellas son independientes de

[4] En [BK05] distinguen: una agencia (entorno de ejecución para agentes), un sistema de agentes móviles (conjunto de agencias en una aplicación de agentes), un sistema de agentes móviles (conjunto de agencias en una aplicación de agentes móviles) y un conjunto de herramientas (toolkit) de agentes (una implementación específica o un producto).

la plataforma y permiten ejecución segura, carga dinámica de clases, múltiples hilos de ejecución y serialización de objetos. Como consecuencia, en la mayor parte de los casos, implementan un mecanismo de movilidad débil. Algunas de las plataformas más populares, ordenadas cronológicamente, son:

- *Aglets* [LO99a, OSP05]. En 1997 fue desarrollada por IBM y posteriormente, en 2000, se liberalizó la implementación pasando a ser mantenida por la comunidad de código abierto. Es probablemente la plataforma de agentes móviles más popular de las desarrolladas hasta el momento. Basándose en Internet, el protocolo de transferencia de agentes (*Agent Transfer Protocol —ATP—*) es un protocolo de nivel de aplicación utilizado para transferir agentes entre ordenadores conectados a la red. Está basado en el paradigma de petición/respuesta entre servicios de agentes y mensajes similares a MIME [The96d, The96c, The96a, The96b, The96e]. El agente servidor es denominado *Tahiti*, y proporciona un interfaz para monitorizar, crear, enviar y eliminar agentes y para configurar los privilegios de acceso del servidor de agentes. La plataforma de agentes ha sido desarrollada independientemente del protocolo de comunicaciones utilizado, de esta forma se considera la utilización de CORBA o RMI como protocolos de comunicaciones.

- *SeMoA* (*Secure Mobile Agents*) [RJS01, Ges05]: desarrollada por *Fraunhofer Gesellschaft* desde 1997, su última versión fue realizada en septiembre de 2007. Se centra en problemas de integración y seguridad en agentes móviles[5]; permitiendo la migración de los agentes móviles utilizando protocolos seguros como SSL [Tho02] o TLS [Tho00]. Hay que destacar que esta plataforma permite transparencia de localización mediante el servicio de nombres ATLAS. Otra de las características de esta plataforma es que permite la ejecución de agentes desarrollados utilizando otras plataformas como Aglets o Jade. También hay que considerar que esta plataforma considera cuestiones como la protección del ordenador que ejecuta el *place* donde se ejecutan los agentes debido a que los agentes, pueden intentar atacar al ordenador en el que se están ejecutando.

- *JADE* (*Java Agent DEvelopment framework*) [BPR01, Til05]: desarrollado por Tilab desde 1999, la última versión fue hecha pública en noviembre de 2008. Es una de las plataformas mas populares y es compatible con FIPA (*FIPA-compliant*), se centra en la comunicación entre agentes, sin embargo la movilidad de los mismos no es una característica básica de la plataforma [PR02]. Debido a que se centra en el total cumplimiento del estándar FIPA, soporta todo tipo de comunicaciones soportadas por el mismo, como XML, Lisp y binarias. Tiene soporte para mensajes basados en RDF [LS01] y contenido XML [McL01]. Da soporte a múltiples *contenedores* que pueden ejecutarse en distintas máquinas formando una única plataforma. También proporciona un servicio de páginas amarillas donde pueden publicarse los distintos servicios proporcionados por los agentes, permitiendo la interconexión de servicios y la subscripción de los distintos agentes a los servicios proporcionados. El lenguaje de programación utilizado en el desarrollo de aplicaciones es Java, facilitando la programación de comportamientos y eventos; además ayuda en la integración con el motor

[5] Los problemas de seguridad con agentes móviles son tratados en [Vig99].

de reglas JESS [FH03, FH07]. La plataforma proporciona soporte para visualizar los datos incluidos en los mensajes de comunicación de los agentes y múltiples protocolos de comunicaciones, permitiendo su desarrollo mediante plug-ins. Esta plataforma proporciona una excelente escalabilidad y velocidad de comunicación entre los agentes.

- *Voyager* [Sof05]: inicialmente fue desarrollado por ObjectSpace en 1997 y actualmente por *Recursion Software*, la última versión se hizo pública en enero de 2009. Es un ORB[6] mejorado con soporte para agentes móviles y ofrece distintas características muy interesantes, como su implementación de la propiedad de transparencia de localización a través de cadenas de proxies que reenvían los mensajes [Mor02a]. Tiene una arquitectura modular que le permite soportar múltiples protocolos de mensajería (IIOP [Gro99], RMI [Mic06b], SOAP [STK01], DCOM [Wal99]), servicios de nombres (basado en RMI [Mic06b], CORBA [OMG93] o JNDI [Mic07]), protocolos de comunicaciones (TCP-IP, SSL, SOCKS) y patrones de comunicaciones (síncronos, unidireccionales —one-way— y asíncronos).

- *Grasshopper* [BM99]: desarrollado por IKV++ en 1999, posteriormente pasó a formar parte de *Enago Mobile*. Esta plataforma implementó por primera vez la especificación de MASIF [MBB+98] y también proporciona un plug-in que permite gestionar mensajes FIPA [FfIPA05]. Permite transparencia de localización mediante el concepto de *región*: la región permite actualizar los *proxies* internamente antes de cada llamada. En una primera versión, el ciclo de vida de los agentes era gestionado mediante un único hilo de ejecución, posteriormente se amplió y pasó a gestionarse mediante varios hilos de ejecución permitiendo que los agentes realizasen diferentes tareas concurrentemente. En esta plataforma el comportamiento de los agentes se diseñaba mediante una máquina de estados y el cambio de un estado a otro se producía mediante el envío de mensaje o mediante el cambio de las condiciones del contexto de ejecución en el que se encontraba el agente. Esta plataforma de agentes móviles dejó de desarrollarse en enero de 2007.

- *NOMADS* [SBB+00b, SBB+00a, Sur06]: desarrollado por la universidad de Florida Este en 2000, la última versión se puso a disposición del público en marzo de 2001. Esta plataforma permite el control dinámico del ancho de banda y la cantidad de recursos utilizados por los agentes móviles (CPU, disco, red), además de permitir la protección frente a ataques de denegación de servicio. Una de las características fundamentales de la plataforma es que permite movilidad fuerte (en cualquier momento) y *movilidad forzada*. Además de trabajar con la migración, permiten comunicarse con los agentes mientras éstos se están moviendo. El mecanismo de comunicaciones utilizado para permitir este tipo de mensajes es *mockets* [MFS06], que encapsula los sockets TCP normales y automáticamente redirecciona la comunicación al ordenador destino. Para permitir la utilización de mecanismos de movilidad fuerte se ha tenido que desarrollar una nueva máquina virtual Java, denominada AromaVM. AromaVM proporciona la capacidad de capturar el estado de ejecución de los distintos hilos de ejecución que constituyen un agente y controlar la cantidad de recursos utilizados por los agentes que

[6] *Object Request Broker*.

se están ejecutando en la plataforma. El mecanismo que permite controlar los recursos consumidos por los agentes que se ejecutan en la plataforma, permite diseñar nuevos procesos de seguridad, aumentando de esta forma el modelo de seguridad proporcionado por Java.

- *Tryllian* [TSB05]: desarrollado por la compañía Tryllian en 2001 bajo licencia LGPL, su última versión es de noviembre de 2005. Está basada en un mecanismo de detección-razonamiento-acción, que permite a los programadores definir comportamientos reactivos y proactivos para los agentes. Tryllian proporciona un modelo de programación basado en tareas. La comunicación entre los agentes se basa en el envío de mensajes, conforme con el estándar FIPA [FfIPA05]. Otra de las características de esta plataforma de agentes móviles es que proporciona mecanismos de persistencia, por ejemplo facilitando la creación de los agentes en caso de fallos del suministro eléctrico o falta de batería (en el caso de dispositivos móviles).

- *Tracy* [BR05, BMG+04, BMS+06]: desarrollada por Peter Braun and Wilhelm R. Rossak en 2004 , la última versión se puso a disposición de los desarrolladores de arquitecturas basadas en Tracy en abril de 2005. Tracy es un sistema de agentes programado en Java 2, que permite la creación y control de agentes móviles, proporcionando un mecanismo de comunicaciones asíncrono por envío de mensajes. Una de las principales características de esta plataforma de agentes móviles es que proporciona mecanismos de migración flexibles (mediante un modelo de movilidad denominado *Kaalong*), permitiendo la migración de partes de código, es decir, no hay que realizar el movimiento de clases completas sino que se puede mover solamente una parte de la clase. Por lo tanto, la gran ventaja obtenida por la utilización de esta plataforma de agentes móviles es la facilidad para adaptar los distintos procesos de comunicación y cooperación entre los agentes debido a la flexibilidad de los mecanismos de comunicaciones proporcionados.

- *SPRINGS* (Scalable PlatfoRm for movINg Software) [ITM06, ITM05]: desarrollada por S. Ilarri y E. Mena en la Universidad de Zaragoza desde 2005. Esta plataforma de agentes proporciona transparencia de localización gracias a su servicio de nombres y a su mecanismo transparente de proxies. Además proporcionan un mecanismo de reintento automático en el protocolo de comunicaciones entre los agentes, permitiendo evitar que se produzcan errores si un agente intenta comunicarse con otro que está viajando en ese momento y en los viajes de los agentes (si fallan). Otra de las características de esta plataforma de agentes móviles es que proporciona un servicio de nombres que permite identificar de forma unívoca a los agentes. Del mismo modo, SPRINGS permite identificar de forma única a un agente independientemente de la máquina en la que se esté ejecutado. Por lo tanto, aunque un agente se mueva de un contexto de ejecución a otro, siempre se podrá establecer una comunicación con éste gracias a un servicio de proxies dinámicos gestionado automáticamente por la plataforma de agentes móviles.

Para una explicación más detallada de la lista de plataformas de agentes móviles se recomienda la lectura de [LJE+01, SRDS01, DS01, Ver98]. Si se quiere ver una comparativa

de las prestaciones de distintas plataformas de agentes se recomienda leer [ITM06, Ila06, TIM07].

Finalmente, mencionar que la especificación de MASIF (*Mobile Agent System Interoperability Facility*) [MBB+98] fue aceptada como estándar de OMG en 1998. Este estándar se centra en definir mecanismos que permitan la interoperación entre distintos sistemas de agentes, proporcionando a los agentes móviles la capacidad de viajar a través de ordenadores con distintas plataformas de agentes móviles. MASIF se centra en las siguientes áreas: 1) gestión de agentes (operaciones estándar para administrar plataformas de agentes móviles), 2) transferencia de agentes (proporcionar una infraestructura común que permita a los agentes moverse entre plataformas), 3) servicio de nombres de agente y plataforma (para permitir a los agentes y plataformas identificarse mutuamente), y 4) sintaxis para tipos y localizaciones de plataformas de agentes (por ejemplo, para que dos plataformas distintas puedan localizarse).

3.4. Agentes software versus cliente/servidor

Antes de que la tecnología de agentes móviles se hiciese realidad, sólo existía una aproximación para el diseño de sistemas distribuidos: la aproximación cliente/servidor, donde un módulo denominado "cliente" invoca servicios remotos de un módulo remoto que acepta dichas llamadas, denominado "servidor". Este procedimiento también se denomina llamada a procedimientos remotos o *Remote Procedure Calling (RPC)*. Existen diferentes protocolos de comunicaciones que implementan esta idea, como RMI [Mic06b] y CORBA [Gro06].

En esta sección vamos a comparar una arquitectura básica, basada en una arquitectura cliente/servidor, que accede a un servidor de bases de datos remoto con distintas modificaciones de una arquitectura basada en agentes móviles que acceden al mismo servidor, para mostrar las ventajas y desventajas de estos diseños. El diseño utilizado en las aproximaciones basadas en agentes móviles puede extrapolarse a cualquier arquitectura basada en agentes móviles. Posteriormente, mostraremos una comparativa de las prestaciones obtenidas utilizando protocolos de comunicaciones como HTTP, HTTPS, FTP, CORBA y agentes móviles.

3.4.1. Arquitectura cliente/servidor

En esta aproximación el usuario accederá al servidor web en el que se encuentra el applet mediante su navegador web y lo pondrá en ejecución. Poner en funcionamiento el applet requiere descargar el driver JDBC necesario para acceder al servidor de base de datos concreto; por lo tanto, si se necesita acceder a distintos gestores de bases de datos se tendrán que descargar distintos drivers.

Por lo tanto, el cliente necesariamente debe descargar (al menos) el código asociado al applet y al del conector JDBC, lo cual puede ser muy costoso en tiempo y recursos si estamos en una red inalámbrica que por definición suelen ser más lenta, cara e inestable que las cableadas.

3.4.2. Arquitectura cliente/agente/servidor

La insuficiencia de las arquitecturas cliente/servidor tradicionales para dar soporte a los entornos inalámbricos ha dado lugar a nuevos modelos de arquitecturas y a nuevos paradigmas de computación como los agentes móviles. En esta sección describiremos algunos modelos basados en agentes que han sido propuestos [PSP00] para entornos de computación móvil inalámbricos.

Las extensiones del modelo cliente/servidor están basadas en agentes estáticos, es decir no viajan por la red durante su ejecución, localizados entre el cliente y el servidor. Estos agentes reducen las restricciones del enlace de comunicaciones inalámbrico optimizando las comunicaciones inalámbricas. Además, la utilización de agentes reduce los recursos consumidos por el cliente, asumiendo parte de la funcionalidad de los clientes. Esta característica se consigue, en mayor o menor grado, dependiendo de la colocación y funcionalidad de los agentes.

Una extensión ampliamente conocida del modelo cliente/servidor es la arquitectura en tres capas cliente/agente/servidor [BBIM93, FGBA96, TSS+97, ZD97b, ZD97a], mostrada en la Figura 3.2, en la que se introduce un agente en la red fija. Dicho agente puede ser utilizado de muchas formas y con distintas funciones. La más radical de todas, el agente se usa como un proxy para realizar todas las comunicaciones desde el dispositivo móvil con los servidores de la red fija. En este caso todas las comunicaciones desde y hacia el dispositivo móvil pasan a través del agente. En el extremo opuesto se encuentra la siguiente situación, el agente es utilizado por un único servicio o aplicación, como por ejemplo, en la navegación web [HSL99, HL96] o acceso a base de datos [Coo97]. Por lo tanto, cualquier petición realizada por el cliente y la respuesta del servidor asociado con la aplicación se comunica a través del agente específicamente creado para ello. En este caso el dispositivo móvil tiene que estar conectado con tantos agentes como servicios estén siendo accedidos.

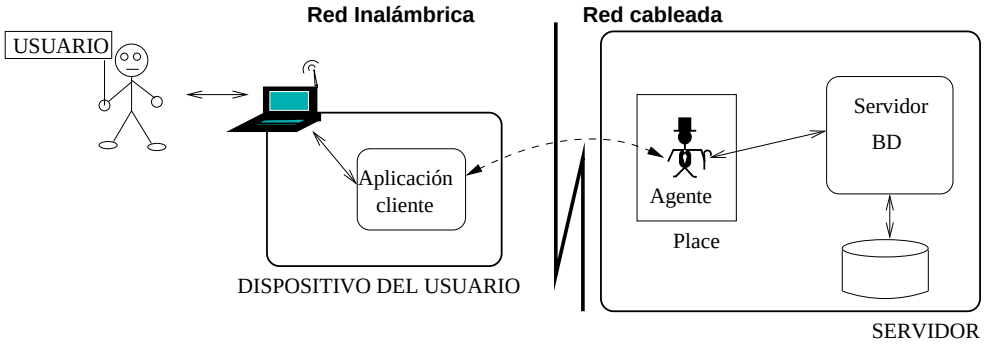


Figura 3.2: Arquitectura cliente/agente/servidor

El agente divide la interacción entre el dispositivo móvil y los servidores en la red fija en dos partes: una entre el cliente y el agente, y la otra entre el agente y el servidor.

En la Figura 3.3 se presenta un caso de ejemplo en el que utilizando la arquitectura cliente/agente/servidor se accede a una base de datos remota. Continuando con el ejemplo presentado en la sección 3.4.1, la diferencia radica en que, en este caso entre la aplicación cliente,

el applet, y la aplicación servidor, el servidor de base de datos, se pone un intermediario, un agente. En el caso de utilizar una arquitectura cliente/servidor, en el mejor de los casos para poner en funcionamiento la aplicación cliente se debía transmitir el código de la página HTML, el código del applet y el código del conector JDBC. Sin embargo, con la nueva arquitectura no se necesita transmitir el código del conector JDBC, aunque tiene que transmitirse el código de las clases del applet, que tiene un tamaño mucho menor. Además, esta aproximación presenta la ventaja de que es más estable frente a desconexiones y permite acceder a distintos gestores de bases de datos; debido a que la carga de las clases correspondientes al conector JDBC la realizará el agente en el servidor, por lo que si el agente viaja a distintos servidores podrá cargar las clases correspondientes a distintos drivers JDBC.

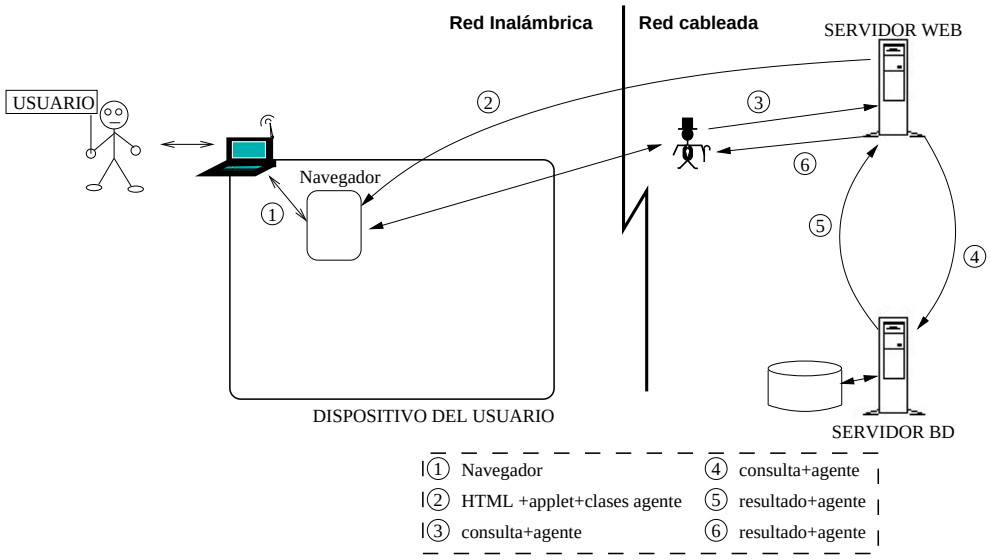


Figura 3.3: Caso de ejemplo cliente/agente/servidor

Además, la aproximación basada en agentes, facilita la concurrencia debido a que permite que el applet cree un agente coordinador directamente en la red fija o que cuando viaje a un ordenador de la red fija (si el agente es creado en el dispositivo móvil del usuario), cree distintos agentes que podrían acceder a un conjunto de servidores de bases de datos remotos, tal y como se muestra en la Figura 3.4. De esta forma se podría realizar un filtrado de los datos en la red fija evitando transmitir elementos duplicados a través del enlace inalámbrico; a mayor duplicidad de datos mejores serán las prestaciones de la aproximación basada en agentes [PSP00].

Sin embargo, aunque las arquitecturas cliente/agente/servidor proporcionan una serie de ventajas (como por ejemplo robustez frente a desconexiones), tienen el inconveniente de que no permiten mantener la ejecución en el cliente durante los periodos de desconexión. El agente puede optimizar solamente las transmisiones de datos del enlace inalámbrico desde la red fija hacia el dispositivo móvil, pero no en el sentido contrario.

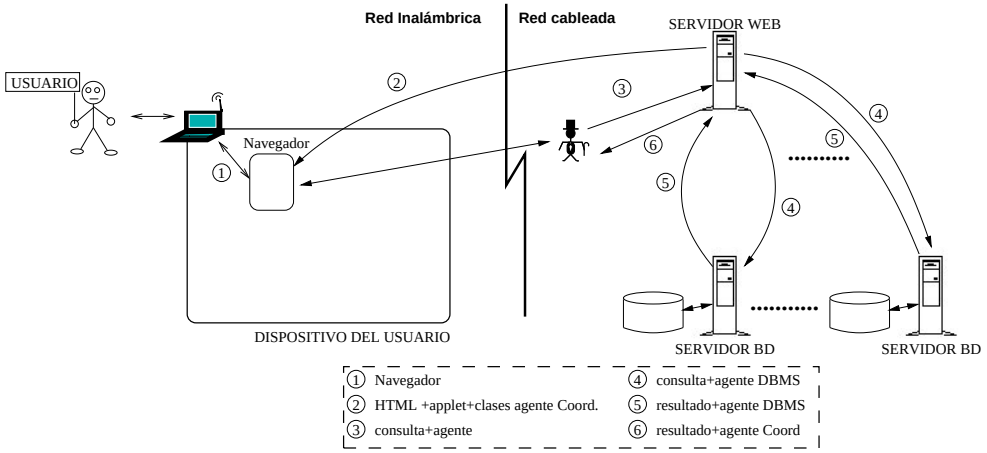


Figura 3.4: Caso de ejemplo cliente/agente/servidor con múltiples gestores de BD

3.4.3. Arquitectura cliente/agente/agente/servidor

Para solucionar los problemas de la arquitectura cliente/agente/servidor, en [HL96, SP97] se propone incorporar a la arquitectura un agente que se ejecutará en el dispositivo móvil del usuario, tal y como puede observarse en la Figura 3.5, que se ejecutará en la red inalámbrica. Este agente en el lado del cliente intercepta las peticiones de éste y de forma conjunta con el agente que se encuentra localizado en el lado del servidor realiza optimizaciones para reducir las transmisiones de datos sobre el enlace inalámbrico, reduciendo la cantidad de datos transmitidos y aumentando la robustez frente a desconexiones en la red inalámbrica. Se reduce la cantidad de datos por la misma razón que en las arquitecturas cliente/agente/servidor. Sin embargo, se añade la robustez frente a desconexiones gracias a que los agentes utilizan un protocolo de comunicaciones asíncrono y por consiguiente la red solamente debe estar accesible durante el envío de parámetros y la recepción de resultados, no durante el tiempo de procesamiento de los servicios ejecutados.

Desde el punto de vista del cliente, el agente que se encuentra junto a él actúa como un proxy local con el servidor, debido a que se está ejecutando en el mismo dispositivo que la aplicación cliente. Del mismo modo, como los agentes se encuentran localizados en la ruta que seguirán los datos desde el cliente hasta el servidor, esta arquitectura también es conocida como modelo cliente/interceptor/servidor. Este modelo proporciona una distinción y separación de responsabilidades clara entre el agente del cliente y el agente del servidor. El protocolo de comunicaciones utilizado entre los dos agentes puede facilitar la reducción de los datos transmitidos. La existencia de ambos agentes también facilita que la aplicación pueda adaptarse fácilmente, porque los dos agentes pueden dinámicamente dividirse la carga de trabajo basándose en las condiciones del entorno de ejecución. Del mismo modo, esta arquitectura proporciona un incremento de la compatibilidad del sistema debido a la transparencia de cambios (modificación del gestor de la base de datos, del interfaz gráfico de usuario utilizado en el cliente, etc...) en el cliente o en el servidor. Es decir, las aplicaciones legadas

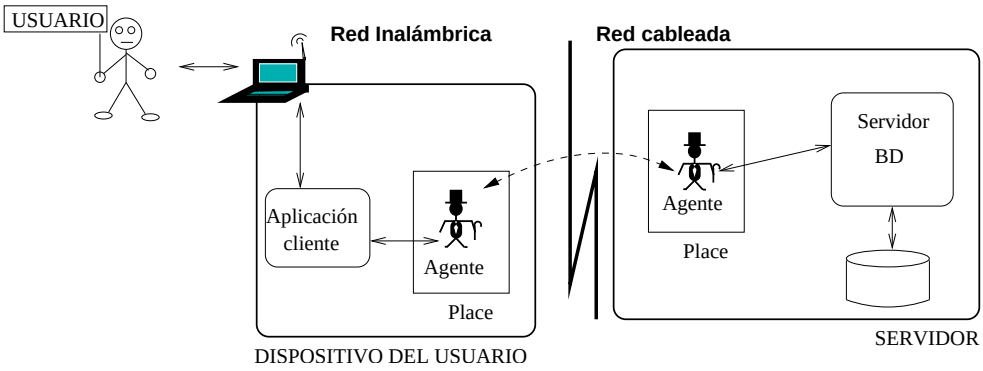


Figura 3.5: Arquitectura cliente/agente/agente/servidor

o existentes pueden seguir ejecutándose como antes de que se introdujese el par de agentes.

3.4.4. Medida de prestaciones

En esta sección vamos a comparar aproximaciones clásicas de transferencia de información con aproximaciones basadas en agentes. En nuestro caso de estudio nos vamos a centrar en la descarga de datos, que bien pueden ser los datos de una consulta a una base de datos como en los ejemplos anteriores; o piezas de software, en el ejemplo mostrado en esta subsección, para aumentar el tamaño de los datos que son transferidos por la red de comunicaciones y, por tanto, el tiempo necesario para su transferencia. Además, este ejemplo también nos permite validar las bondades de las arquitecturas basadas en agentes móviles aplicadas a uno de los casos de estudio diseñados en la tesis. Sin embargo, debemos destacar que puede extrapolarse a cualquier otro proceso de transferencia de datos [EOK04].

Para descargar una pieza de software a un dispositivo móvil utilizando agentes móviles hemos elegido la siguiente aproximación: el software debe ser transferido al dispositivo del usuario mediante un agente móvil en un único viaje. A continuación comparamos esta aproximación con otras no basadas en agentes móviles.

En las diferentes posibilidades consideradas, desde la menos a la más “sofisticada”, se descarga el software desde un servidor remoto al dispositivo del usuario. Las distintas aproximaciones son:

1. *HTTP*: se descarga el software utilizando una conexión HTTP, mediante la creación de un módulo en el dispositivo del usuario que accede a un servidor web remoto. El proceso completo comprende: la petición de descarga de la pieza de software, la descarga y su almacenamiento en el dispositivo del usuario (ver Figura 3.6.a).

2. *CORBA*: se descarga el software invocando un servicio remoto de un servidor CORBA, mediante la creación de un módulo que invoca el servicio del servidor CORBA de forma remota. Este servicio devuelve la pieza de software solicitada como un vector de bytes que es almacenado en un fichero en el disco del dispositivo del usuario (ver Figura 3.6.b).

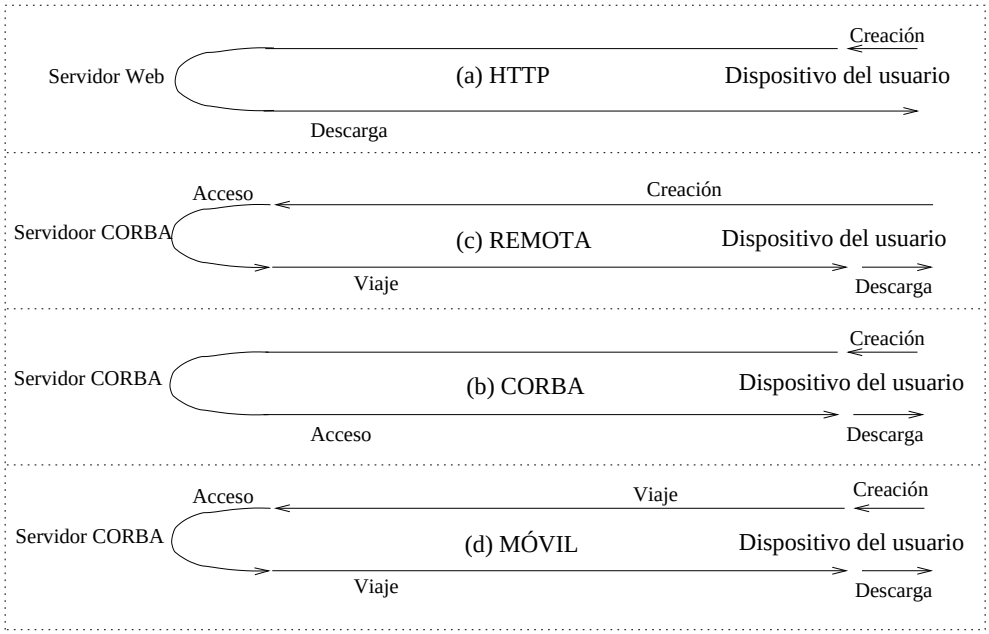


Figura 3.6: Estrategia (a) HTTP, (b) CORBA, (c) Remota, y (d) Móvil

3. *Remota*: un agente móvil *se crea remotamente en el ordenador en el que se encuentra el servidor CORBA* y entonces se realiza la petición del software al servidor CORBA *localmente*, y el agente viaja de vuelta al ordenador del usuario (ver Figura 3.6.c).

4. *Móvil*: un agente móvil es creado en el dispositivo del usuario; posteriormente éste viaja al ordenador en el que se encuentra el servidor CORBA, accede al software utilizando una conexión CORBA, y viaja de vuelta al dispositivo del usuario para finalmente almacenar el software en el disco de dicho dispositivo (ver Figura 3.6.d).

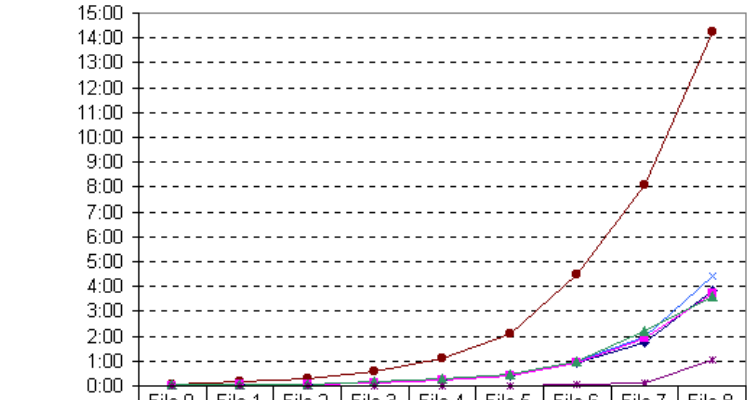
Los test están basados en la descarga de piezas de software disponibles en el servidor (ver Figura 3.7), debemos hacer hincapié en el amplio espectro de tamaños de fichero utilizado. Se han ejecutado las distintas estrategias en 20 ocasiones para cada fichero utilizando una conexión cableada[7]. En la Figura 3.8 mostramos la comparación entre los distintos métodos. En HTTP, los datos son solicitados y guardados en un fichero, byte a byte. En HTTP2, se pide un array de bytes y luego éste es salvado en un fichero, lo cual es espectacularmente rápido pero prácticamente bloquea el ordenador. También hemos realizado algunos test utilizando un cliente FTP (concretamente, ReachOut SuperFTP). En la Figura 3.8 podemos observar *los rangos de tiempos de la descarga de los ficheros*, verificando que los agentes no mejoran pero tampoco empeoran el tiempo de descarga. Sin embargo, se podría considerar una descarga

[7]Para una conexión GSM, las diferencias en tiempo serían mayores, debido a que la velocidad de la red es menor, pero las proporciones serían similares.

que divida el fichero en partes para ficheros grandes, independientemente de la estrategia usada y los resultados serían equivalentes.

File #	Name	Size
0	Hardcopy V1.31	99 KB
1	Shockwave 7	305 KB
2	BulletProof FTP	509 KB
3	Fullshot 99	1.167 KB
4	Windows 95 y2k update	2.210 KB
5	McAfee VirusScan v3.1.6	4.280 KB
6	Voyager 3.1	8.998 KB
7	Netscape Communicator 4.61	15.455 KB
8	X−Wing Alliance Demo	28.476 KB

Figura 3.7: Software transferido por el enlace de red en las pruebas



	File 0	File 1	File 2	File 3	File 4	File 5	File 6	File 7	File 8
HTTP	0:03	0:09	0:16	0:36	1:05	2:06	4:28	8:04	14:15
FTP	0:01	0:02	0:03	0:07	0:13	0:26	0:58	1:58	4:25
CORBA	0:01	0:02	0:03	0:07	0:14	0:26	0:56	1:45	3:51
REMOTA	0:01	0:02	0:03	0:07	0:13	0:26	0:57	1:56	3:42
MÓVIL	0:02	0:03	0:04	0:09	0:16	0:28	1:01	2:14	3:37
HTTP2	0:00	0:00	0:00	0:00	0:00	0:01	0:02	0:08	1:04

Figura 3.8: Comparativa entre distintas técnicas

En esta sección hemos demostrado que los agentes móviles no empeoran el comportamiento de los servicios de acceso a datos cuando éstos utilizan una red cableada frente a aproximaciones clásicas basadas en llamada a procedimientos remotos. Sin embargo, tal y como se muestra en [PSP00, EOK04] los agentes proporcionan grandes reducciones del

tráfico de red además de ser más robustos que las aproximaciones clásicas basadas en cliente/servidor.

3.5. Resumen del capítulo

En este capítulo se ha presentado qué se entiende por agente software, viendo que no se ha llegado a una única definición sino que la comunidad científica ha llegado a un acuerdo en las propiedades que tienen que tener los agentes software. En esta tesis hemos seleccionado la siguiente definición: un *agente software* es un módulo software que se ejecuta en un cierto contexto de ejecución o *place*. Un agente posee las siguientes propiedades principales[8]: 1) *Autonomía*, posee el control sobre sus propias acciones; 2) *Finalidad*, gestiona una agenda de objetivos; 3) *Cooperación*, un agente es capaz de comunicarse con otros agentes; 4) *Aprendizaje*, cambia su comportamiento de acuerdo a su experiencia previa; 5) *Movilidad*, puede viajar de un ordenador a otro (en realidad, de un *place* a otro); 6) *Reactividad*, siente los cambios de su entorno y reacciona ante ellos; y 7) *Persistencia*, para poder interrumpir su ejecución y continuarla más adelante.

No menos importante ha sido destacar la importancia de los sistemas multiagente (SMA). Su realización es muy importante, como se ha explicado anteriormente, para llevar a cabo tareas complejas. Los SMA son una solución de tipo divide y vencerás, es decir, un problema complejo se divide en distintas subtareas que son realizadas por distintos agentes que cooperan y se coordinan para llevarlas a cabo.

A continuación se ha definido qué son los agentes móviles, detallando también las diferencias entre código móvil y agente móvil. Para diferenciar qué es un agente móvil y qué es código móvil nos podemos basar en las características de los agentes móviles que se han presentando anteriormente. En este punto, también se han destacado los distintos tipos de movilidad que pueden tener las plataformas de agentes móviles: movilidad fuerte y movilidad débil. A continuación, se ha mostrado un breve resumen de las distintas plataformas de agentes móviles disponibles actualmente.

Finalmente, se ha mostrado la solución clásica a los problemas de computación distribuida, es decir, la arquitectura cliente/servidor. Sin embargo, debido al gran auge de los entornos inalámbricos se ha visto la necesidad de diseñar nuevas arquitecturas basadas en la utilización de agentes; dando lugar a las arquitecturas cliente/agente/servidor y cliente/interceptor/servidor, más robustas y eficientes en redes inestables como la inalámbrica. Estas arquitecturas mostradas a modo de ejemplo fueron presentadas y diseñadas en [PSP00].

[8] En algunos contextos concretos no tienen porqué concurrir todas las propiedades al mismo tiempo.

Capítulo 4

Aplicación de la tecnología de agentes a un servicio de recuperación de software

En este capítulo estudiaremos las ventajas proporcionadas por las arquitecturas basadas en agentes móviles en el contexto de la búsqueda, descarga e instalación de software en entornos inalámbricos. Entre las principales ventajas proporcionadas por el diseño de una arquitectura basada en un sistema multiagente se encuentran la fácil adaptación al contexto de ejecución, la reducción del tráfico de red y la robustez frente a redes inestables como la inalámbrica. Además, se verificará la capacidad de los agentes para tomar decisiones por el usuario de forma autónoma.

El estudio de este contexto de ejecución se fundamenta en que una de las tareas más frecuentes para los usuarios de dispositivos electrónicos como ordenadores y PDAs, la descarga e instalación de nuevo software. El Servicio de Recuperación de Software (SRS) [MIG00c, MIRG06] es una aplicación basada en una arquitectura de agentes móviles inteligentes que nos permitirá mostrar las ventajas de su utilización frente a una arquitectura cliente/servidor clásica. Hay que destacar que la tediosa tarea de buscar un programa que satisfaga una necesidad es compleja, especialmente cuando el usuario del dispositivo móvil es inexperto y no conoce el nombre del programa buscado, únicamente conoce cuál debe ser su funcionalidad. En este caso el SRS ayudará al usuario en la tarea de buscar, descargar e instalar dicho programa en su dispositivo.

El Servicio de Recuperación de Software se encuentra situado en un servidor de la red cableada que denominaremos *proxy* (debido a que nos servirá de enlace entre la red inalámbrica y la red cableada) y es parte de un sistema más general denominado ANTARCTICA [VGGI98]. El servicio propuesto posee un *place* en el dispositivo móvil del usuario llamado *place del usuario*, y otro localizado en el proxy, llamado *place del software* (según se muestra en la Figura. 4.1). A continuación describiremos brevemente los principales agentes del SRS y como interactúan entre sí.

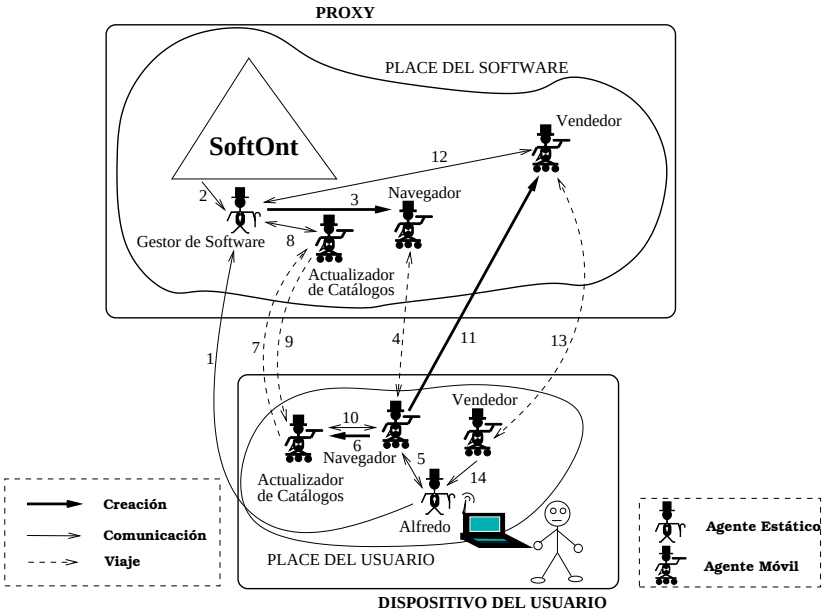


Figura 4.1: Arquitectura del Servicio de Recuperación de Software

1. *El agente Alfredo*: es un eficiente mayordomo que sirve al usuario y es el encargado de almacenar tanta información acerca del dispositivo del usuario, y del propio usuario, como le sea posible. Aunque ayudará al usuario en la ejecución de todos los servicios que haya disponibles en el proxy, nos centraremos en su papel en el Servicio de Recuperación de Software. Es decir, este agente es común a la arquitectura de ANTARCTICA.

2. *El agente Gestor de Software*: crea y proporciona al agente Navegador un catálogo del software disponible, según los requisitos impuestos por Alfredo (siguiendo las instrucciones proporcionadas por el usuario, paso 1 de la Figura 4.1). Por ejemplo, es capaz de obtener metadatos personalizados acerca del software disponible en los depósitos web subyacentes. Para esta tarea, el *agente Gestor de Software* consulta la ontología *SoftOnt* (paso 2) para obtener un catálogo del software disponible que satisfaga las necesidades expresadas por Alfredo (obedeciendo a las restricciones del usuario). Posteriormente, el agente Gestor de Software creará al agente *Navegador* (paso 3) y le asignará el catálogo de software del usuario.

3. *El agente Navegador*: viajará al dispositivo del usuario (paso 4) e interactuará con el usuario (ayudado por el agente Alfredo) para ayudarle a navegar (buscar) por el catálogo de software (paso 5); la nueva información que solicite el usuario será obtenida por el agente *Actualizador de Catálogos*, que es creado por el agente Navegador (paso 6) para actualizar el catálogo de software (paso 10). el proceso de actualización del catálogo de software puede realizarse viajando al place del software o interactuan-

do con el Gestor de Software (pasos 7, 8, y 9). El Agente Actualizador de Catálogos decide autónomamente entre viajar al place del software o solicitar el nuevo catálogo remotamente, estimando el coste de ejecución de ambas aproximaciones. Cuando el proceso de búsqueda termina porque el usuario finalmente ha elegido una pieza de software concreta, el agente *Vendedor* será creado por el agente Navegador (paso 11), que en primer lugar consultará al Gestor de Software acerca del software seleccionado (paso 12). Posteriormente, el agente Vendedor descargará el programa seleccionado al dispositivo del usuario (paso 13), y finalmente realizará las operaciones de comercio electrónico [Sie04] que sean necesarias, tarea que depende de la pieza de software concreta (paso 14[1]).

A continuación pasaremos a detallar las tareas descritas anteriormente y realizadas por cada uno de los agentes que constituyen la arquitectura del servicio propuesto.

4.1. Inicialización: Alfredo, el agente mayordomo

Alfredo es un agente mayordomo eficiente, localizado en el dispositivo del usuario, que sirve al usuario y es el encargado de almacenar tanta información sobre el dispositivo del usuario, y el usuario en sí mismo, como le sea posible. A este agente se le denomina mayordomo porque es el encargado de informar al usuario de los servicios que tiene disponibles y de ayudarle en la ejecución de los mismos. Esta característica es vital en el contexto en el que nos encontramos debido a que estamos estudiando una aplicación que debe ayudar a usuarios inexpertos en la tarea de buscar un programa que satisfaga sus necesidades. Por ejemplo, imaginemos un usuario que desea un reproductor MP3 para su PDA, este usuario puede saber que su dispositivo móvil tiene instalado windows pero no qué versión. La utilización de un agente mayordomo tiene otra gran ventaja, el usuario no necesita conocer detalles técnicos acerca de como debe ejecutarse el SRS, por ejemplo, cuál es el proxy en el que se encuentra ubicado el Gestor de Software. Por ejemplo, supongamos una situación en la cual el usuario desea recuperar una cierta clase de software, pueden presentarse dos casos:

1. El usuario sabe exactamente el programa que necesita, por ejemplo, el reproductor Windows Media Player 10 para Windows Mobile. Así, un usuario experto podría formular directamente la petición, con la ayuda de un GUI, como una lista de las restricciones <*característica, valor*> que describen el software que necesita. En el ejemplo anterior las restricciones introducidas por el usuario serían *[<name, Windows Media Player 10>, <OS, Windows Mobile>]*.

2. El usuario solamente sabe una cierta característica del programa, por ejemplo, su propósito. En este caso, el usuario necesita una cierta clase de catálogo referente al software disponible que satisface las características especificadas, para navegarlo y encontrar el programa deseado, ya que habrá varios programas que cumplan la única restricción indicada. Con la ayuda de un GUI, los usuarios pueden escribir una lista de restricciones para expresar sus necesidades lo mejor que pueda; sin embargo, la información

[1]Los pasos 12, 13 y 14 no se describirán con más detalle debido a que se encuentran fuera del ámbito de esta tesis.

proporcionada puede ser imprecisa. En la Figura 4.2, el usuario no sabe ni el nombre del reproductor MP3 ni la versión concreta de Windows de su dispositivo. Por lo tanto, el SRS también se debe ocupar de las situaciones donde el usuario no sabe ninguna característica del software. El/ella podría haberlo visto en el pasado pero no recordar el nombre ahora, el propósito exacto, etc. Sin embargo, si el sistema presenta un catálogo de software al usuario, el/ella quizá pueda reconocerlo al volver a ver su nombre o descripción.

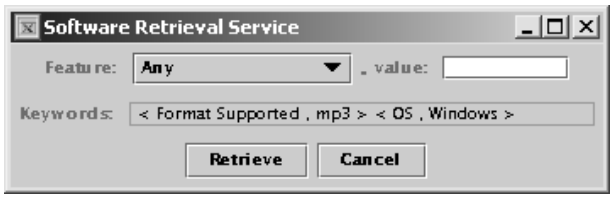


Figura 4.2: GUI de Alfredo para el SRS: solicitando reproductores de MP3 para Windows

Además de las restricciones del usuario, Alfredo puede agregar más restricciones referentes al dispositivo del usuario (por ejemplo, sistema operativo, memoria RAM, tarjeta vídeo, etc.) y ejecuciones anteriores del servicio (e.g. reproductores MP3 anteriormente descargados). De hecho, toda la información proporcionada por el usuario es almacenada por Alfredo. Así, con cada petición del usuario, Alfredo almacena más información sobre el dispositivo del usuario y sobre el propio usuario, convirtiéndose con el tiempo en un verdadero experto en las necesidades software del usuario y de las características de su dispositivo móvil.

Después de que el usuario especifique sus requisitos y éstos sean ampliados por Alfredo, éste envía una petición del catálogo de software al agente *Gestor de Software* que reside en el proxy (su comportamiento se explica en la sección 4.2).

El Conocimiento de Alfredo

El conocimiento gestionado por este agente, es la característica que le hace comportarse inteligentemente ayudando al usuario mediante la adhesión de nuevas restricciones a las impuestas por el usuario y la gestión de la información técnica necesaria para la ejecución del servicio. Por consiguiente, hacemos una distinción general entre el conocimiento general de Alfredo y el conocimiento especializado relacionado con el servicio considerado:

1. Conocimiento independiente del servicio: *conocimiento técnico relacionado con el sistema*, por ejemplo, los componentes hardware disponibles (memoria RAM, tarjeta de sonido y vídeo, CPU, tipo de conexión de red, y otros periférico instalados), la disponibilidad de determinados recursos (memoria libre, tarjetas de memoria, etc), el software actualmente disponible (aplicaciones necesarias para instalar el nuevo software, tal como Java, Flash, etc.), y las preferencias generales del usuario (comunes a todos los servicios, como las características del GUI, programas preferidos, etc). Alfredo obtiene esta información consultando al sistema operativo y estudiando el comportamiento del usuario. Si el sistema operativo no permite recuperar automáticamente ciertos datos,

Alfredo los solicitará al usuario (solamente cuando sea estrictamente necesario) y los almacenará posteriormente.

2. Conocimiento relacionado con el Servicio de Recuperación de Software, pudiendo estar relacionado con características técnicas o con ejecuciones anteriores del servicio. Este tipo de conocimiento se divide en:

   a) *Conocimiento técnico acerca del servicio*, información sobre otros agentes y *places* necesarios para alcanzar el objetivo del servicio, tal como *proxy* utilizado por el servicio, nombres de los agentes, servicios disponibles, etc.

   b) *Trazas de históricos del servicio*, que son una parte muy importante de la "memoria" de Alfredo. La información más importante de estos históricos son las interacciones con el usuario, porque es una propiedad que el sistema quiere minimizar: toda la información proporcionada por el usuario en las peticiones de otros agentes es almacenada.

   c) *Las preferencias del usuario relacionadas con el servicio concreto*, que han sido extraídas de ejecuciones anteriores; por ejemplo, Alfredo puede detectar que siempre que el usuario necesite un reproductor MP3, al final siempre selecciona la última versión de WinAmp. Por lo que directamente el agente Alfredo sugeriría solicitar el WinAmp sin que el usuario lo diga explícitamente. Sin embargo, es importante poner de manifiesto que estas preferencias pueden cambiar en el tiempo, el usuario podría querer instalar diferentes reproductores MP3 porque quiera cambiar de interface de usuario o por cualquier otra razón.

En lo referente a las trazas de ejecuciones anteriores del servicio y a las preferencias del usuario, Alfredo maneja información probabilística de las respuestas del usuario, para hacer frente al problema de diversas respuestas proporcionadas a la misma pregunta en distintas situaciones. Por ejemplo, Alfredo podría almacenar que el usuario seleccionó WinAmp como reproductor MP3 el 85 % de la veces. Como dijimos anteriormente, el usuario puede cambiar de intención/pensamiento debido a cualquier razón. Así, siempre que Alfredo pueda sugerir diversas opciones al usuario, la más probable será seleccionada considerando: 1) el número de veces que fue proporcionada por el usuario, y 2) que las respuestas más recientes pueden encajar mejor con las preferencias actuales del usuario. De todas formas, Alfredo comunicará siempre al usuario la razón de su elección y la decisión final dependerá del usuario.

4.2. Obtención de un catálogo de software: el agente Gestor de Software

Después de recibir una petición de Alfredo (en nombre del usuario), el agente Gestor de Software realiza dos tareas principales: Primero, obtener un catálogo que se corresponda con la petición del usuario mediante la poda de la ontología SoftOnt, y en segundo lugar, crear un agente que viaja al dispositivo del usuario, presenta el catálogo al usuario, y le ayuda a encontrar el software deseado. La contribución principal del agente Gestor de Software es el

algoritmo de poda de ontologías, que permite al agente realizar una selección automática e inteligente de los nodos que verifican las restricciones especificadas por el usuario (como la mayoría de los algoritmos de poda). Hay que destacar que este agente considera dos tipos de nodos en la ontología: los programas, que identifican piezas de software, y las categorías, que identifican tipos de software y pueden incluir otras categorías o programas. Posteriormente, el sistema mostrará al usuario solamente algunos de ellos según: 1) cierto porcentaje (llamado nivel de detalle), seleccionado automáticamente por el sistema considerando el estado de la red y el perfil del usuario, y 2) cierta estrategia de poda que también se selecciona, automática y dinámicamente, por el sistema considerando el comportamiento del usuario. Por lo tanto, a) se evita presentar categorías al usuario y programas software que no se pueden instalar en su dispositivo; b) se evita presentar categorías y piezas de software muy especializados que podrían hacer que usuarios inexpertos gastasen mucho tiempo leyendo el catálogo; y c) reduce al mínimo el coste de las comunicaciones enviando solamente la información interesante.

4.2.1. Pasos seguidos durante el proceso de poda

Los siguientes pasos son ejecutados por el Gestor de Software pare realizar la poda de la ontología SoftOnt:

1. *Seleccionar el nodo a podar*. Para el primer catálogo será el nodo raíz de SoftOnt, la primera vez consideramos la ontología entera. En las actualizaciones siguientes del catálogo, se podará un nodo concreto.

2. *Elegir el tipo de nodo*. Si el usuario no especificó ninguna restricción, (no se dispone de ninguna información sobre qué desea), el sistema incluirá solamente categorías en el primer catálogo para ayudar al usuario a elegir primero la clase de software deseado. En otro caso, el catálogo incluirá nodos categorías y nodos programa.

3. *Podar la ontology SoftOnt según las restricciones* que cada nodo debe satisfacer, en caso de que el usuario haya especificado alguna restricción (podar usando restricciones es muy selectivo). Llamemos $Ont_{restric}$ a la ontología después de considerar las restricciones:

$$Ont_{restric} = \begin{cases} podar(SoftOnt, restricciones) & \text{Si } \exists \text{ alguna restricción} \\ SoftOnt & \text{en otro caso} \end{cases}$$

4. *Fijar el nivel de detalle*, es un porcentaje calculado automáticamente que indica la cantidad de datos que deben incluirse en el resultado (el catálogo que será enviado al dispositivo del usuario); por ejemplo, un nivel del detalle de el 30 % indica que solamente el 30 % de los nodos que satisfacen las restricciones se deben incluir en el resultado. El sistema estima la cantidad de información ajustando la información enviada según el valor de la *unidad de facturación*[2]. Esta estimación se basa en el estado actual de

[2]Por *unidades de facturación (UF)* entendemos, en GSM, la unidad de tiempo (por segundos, minutos, etc.) usada para cobrar al usuario su factura de teléfono. En GPRS y UMTS, las UFs se basan en unidades de tamaño (por byte, KB, etc.).

la red (véase sección 4.2.3) y un porcentaje $\%_{incr}$ concreto especificado por Alfredo (detallado en la sección 4.2.2).

5. *Aplicar una estrategia de poda a* $Ont_{restric}$. La estrategia de poda indica *qué* nodos de la ontología serán seleccionados, éste será el criterio de selección. En nuestra aproximación se han diseñado varias estrategias de poda:

   - *Nodos más solicitados*. El agente Gestor de Software actualiza estadísticas globales acerca de la descarga de cada nodo en SoftOnt. Cada vez que un nodo se incluye en el catálogo para algún usuario, se aumenta su cuenta. Así, cuando se utiliza esta estrategia, se seleccionan primero los nodos más solicitados.
   - *Nodos más solicitados por el usuario*. El Gestor de Software también almacena qué nodos se envían a cada usuario (estadísticas del usuario). Así, usando esta estrategia el Gestor de Software selecciona primero los nodos que ese usuario solicitó un mayor número de veces en el pasado.
   - *Estrategia proporcional*. Los nodos hermanos[3] tienen la misma prioridad. Es decir, cuando un nodo se debe podar según un cierto nivel del detalle igual a $n\,\%$, todos los descendientes inmediatos de tal nodo se podarán usando un nivel del detalle de $n\,\%$ y así recursivamente. Esta estrategia es muy útil cuando el sistema no tiene ninguna idea de lo que está buscando el usuario, porque todas las ramas tienen la misma prioridad.
   - *Estrategia del más pesado*. Los nodos con más programas o categorías subyacentes de software se seleccionan primero. Esta estrategia se basa en la idea de que el usuario podría buscar un software bajo categorías muy pobladas (por ejemplo, juegos).

   En la Figura 4.3.a, mostramos mediante un ejemplo los distintos resultados obtenidos después de aplicar la estrategia proporcional (Figura 4.3.c) y la del más pesado (Figura 4.3.b) a la misma ontología.

   Para el primer catálogo, se selecciona la estrategia de poda proporcional porque poda los nodos hermanos proporcionalmente (se da la misma importancia a todos los nodos hermanos), lo cual es una buena idea para el primer catálogo. En las actualizaciones futuras del catálogo, el sistema selecciona automáticamente la estrategia más conveniente. En la sección 4.3.2 se pueden encontrar más detalles sobre las estrategias de poda diseñadas. Llamemos Ont_{podada} al resultado de esta tarea de poda de $Ont_{restric}$.

6. *Obtener una respuesta incremental*. El primer catálogo será Ont_{podada}. Por otra parte, para evitar enviar los datos que están ya en el dispositivo del usuario, el Gestor de Software almacena *las identificaciones* de los nodos enviados a cada uno de los usuarios[4] ($nodos_{usuario_k}$) y, en las actualizaciones futuras del catálogo, elimina esos nodos del catálogo obtenido. Así, solamente se envía la *nueva* información.

[3] Nodos hermanos son aquellos que *al menos* tienen un nodo padre inmediato común (hiperónimo).

[4] Esta información es almacenada por el agente Navegador en el dispositivo del usuario; cuando el usuario se traslada a otro proxy (*handoff*) el Navegador comunica al nuevo Gestor de Software cuáles son los nodos que han sido recuperados anteriormente.

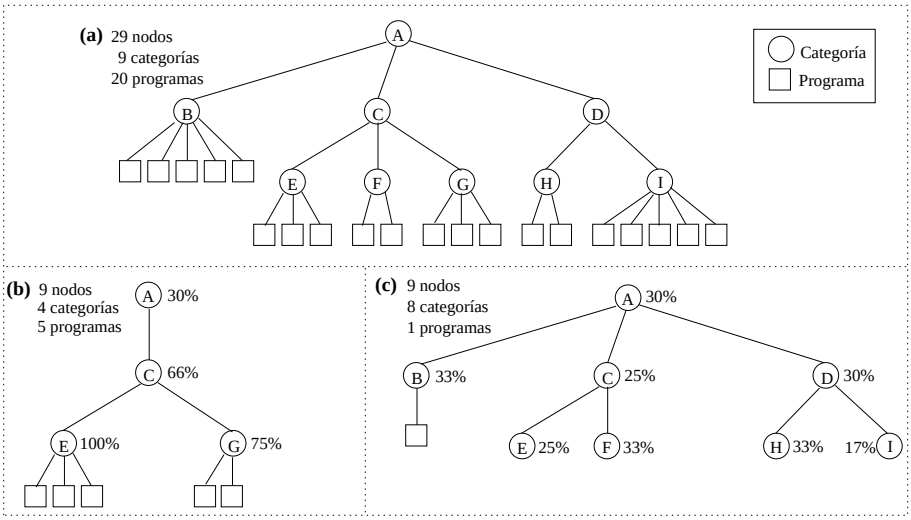


Figura 4.3: Estrategia de poda proporcional versus estrategia del más pesado

7. *Comprimir el catálogo obtenido*. La información obtenida en el paso anterior se puede comprimir para reducir el uso de la red cuando un catálogo es enviado al dispositivo del usuario, y por tanto, se reduce el coste.

A continuación detallaremos los pasos más importantes seguidos en este proceso de creación de catálogos de software personalizados.

4.2.2. Selección del nivel de detalle más apropiado

Como hemos explicado anteriormente, el nivel del detalle es un porcentaje que indica la cantidad de datos que se deben incluir en el catálogo resultante. para seleccionar el nivel del detalle se consideran parámetros como el perfil del usuario y el estado de la red. Por lo tanto, se siguen los pasos siguientes:

1. *Obtener el tiempo necesario para hacer un refinamiento remoto*, considerando el tamaño de $Ont_{restric}$ y los nodos enviados anteriormente al usuario, $nodos_{usuario_k}$, porque solamente se envían respuestas incrementales al usuario:

$$nodos_{usuario_k} = \{nodo_i \in SoftOnt \mid nodo_i \text{ en el dispositivo del } usuario_k\}$$

$$t_{necesario} = t_{reintento}\left(\frac{\left\lceil\frac{|Ont_{restric}|\times porcentaje}{100}\right\rceil - |nodos_{usuario_k}|}{velocidadRed_{ping}}\right)$$

donde $|x|$ es el tamaño en bytes del conjunto x, $porcentaje$ es $\%_{incr}$ (para la primera poda) o el nivel de detalle previo[5] (en podas posteriores), $velocidadRed_{ping}$ es

[5]Nivel de detalle con el que se podó ese nodo anteriormente.

la velocidad medida en tiempo de ejecución de la red, y $t_{reintento}(t)$ es el tiempo que resulta después de considerar el número estimado de reintentos para mantener una conexión abierta durante t segundos (véase sección 4.2.3). $\%_{incr}$ es un porcentaje concreto especificado por Alfredo (distintos usuarios podrían tener diferentes incrementos dependiendo de su experiencia, capacidades del dispositivo, tipo de red, etc.). A excepción de la primera poda, $porcentaje$ se aumenta en $\%_{incr}$, cuando se realiza un refinamiento remoto del catálogo de software gestionado por el agente Navegador.

Del mismo modo el agente Actualizador de Catálogos debe considerar dos situaciones especiales:

a) $t_{necesario}$ es muy reducido: Por ejemplo, si $t_{necesario}$ es igual a 20 segundos es razonable hacer que el usuario espere 10 segundos más, aunque este incremento sea del 50 %. En cambio, esto no sería razonable si el tiempo necesario fuese mucho mayor.

b) Método de facturación asignado al usuario: Dependiendo de la tecnología de comunicaciones utilizada por el usuario, éste es facturado en función de distintas unidades de facturación (UF), por ejemplo en GSM en función del tiempo. Debido a esto el agente Actualizador de Catálogos evitará utilizar una nueva unidad de facturación si implica un mínimo incremento del catálogo de software que se va a enviar al dispositivo del usuario.

En la Figura 4.4, mostramos como el porcentaje incrementado se ve modificado dependiendo del tiempo necesario, cuando $\%_{incr} = 20\,\%$ y $UF = 60$ segundos. Del mismo modo mostramos como el agente selecciona incrementos mayores que $\%_{incr}$ cuando $t_{necesario}$ es pequeño, e incrementos menores que $\%_{incr}$ para evitar el consumo de una UF más, hasta que $\%_{incr}$ del tiempo necesario es mayor que una UF; en la Figura 4.4 sucede cuando el tiempo necesario es mayor de 300 segundos (5 minutos) o más. En tal caso, el Agente Gestor del Software elige consumir una UF más para evitar realizar muchos refinamientos remotos con pequeños incrementos del nivel de detalle. En la Figura 4.5 mostramos la función T_{incr} (etiquetada como "Increment") que devuelve el aumento de tiempo correspondiente a un $t_{necesario}$ dado, por lo tanto $t_{final} = t_{necesario} + T_{incr}(t_{necesario})$. Hay que considerar que si el usuario no es facturado en base a UF, T_{incr} siempre se verá incrementado en el tiempo en el porcentaje especificado por Alfredo ($\%_{incr}$).

2. *Obtener el nuevo nivel de detalle,* considerando el tiempo necesario para transmitir el catálogo (t_{final}), la verdadera velocidad de red, el tamaño medio del nodo ($\overline{|nodo|}$), el número de nodos en el dispositivo del usuario ($\#(nodos_{usuario_k})$) y el número de nodos en el catálogo podado usando las restricciones ($\#(Ont_{restric})$):

$$nivel\ de\ detalle = \frac{\left(\frac{t_{final} \times velocidadRed_{real}(t_{final})}{\overline{|nodo|}} + \#(nodos_{usuario_k})\right) \times 100}{\#(Ont_{restric})}$$

donde $velocidadRed_{real}(t)$ es la velocidad de red estimada en tiempo de ejecución considerando la probabilidad de desconexión en una conexión de duración t, su cálculo se realiza mediante el método descrito en la sección 4.2.3.

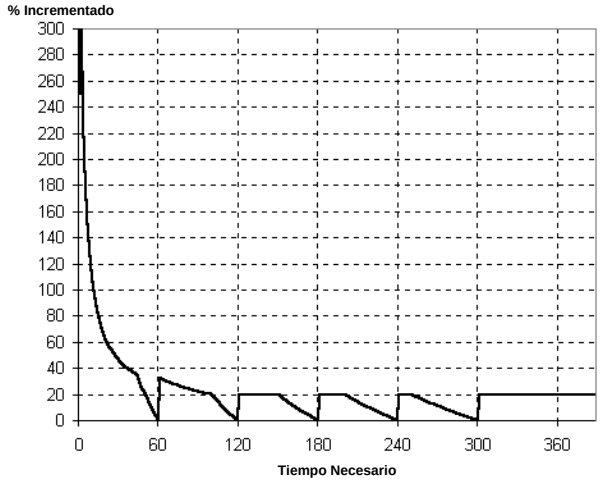


Figura 4.4: Porcentaje incrementado a $t_{necesario}$

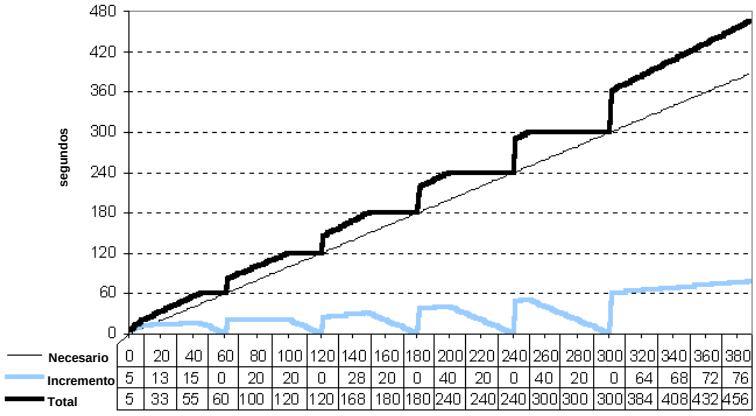


Figura 4.5: T_{incr} ($\%_{incr}$=20 %, UF=60 seg)

La adecuada selección del nivel de detalle del catálogo de software que es descargado al dispositivo del usuario es un factor clave para minimizar el coste económico y temporal del proceso de búsqueda y, como veremos a continuación, este porcentaje se basa en calcular la velocidad real de la red que utilizan los distintos agentes que constituyen la arquitectura para comunicarse.

4.2.3. Estimación del estado de la red

Siempre que un agente en nuestro sistema desee establecer una conexión inalámbrica entre el dispositivo del usuario y el proxy, o viceversa, se estimará la verdadera velocidad

de la red para calcular el tiempo necesario para transmitir cualquier dato [MRIG02a]. La velocidad de la red se mide antes de cada transferencia, la denotamos por $velocidadRed_{ping}$. Sin embargo, además de la velocidad de la red, es muy interesante estimar la fiabilidad del enlace de red entre el dispositivo del usuario y el proxy para enviar datos durante cierto tiempo, es decir, la velocidad de red *real* , $velocidadRed_{real}(t)$. Es decir, estamos interesados en estimar la probabilidad de mantener una conexión de red abierta durante cierto tiempo t. Por lo tanto, debemos considerar las conexiones de red anteriores, debido a que asumimos que el dispositivo del usuario se conectará siempre al proxy utilizando un tipo de red similar. Esta probabilidad se utiliza para estimar en *tiempo de ejecución* el número de reintentos necesarios para enviar completamente una cantidad de datos, $\#reintentos(t)$:

$$p(t) = \begin{cases} \frac{F(t)}{T-S(t)} & T > 0 \\ 0 & T = 0 \end{cases} \qquad \#reintentos(t) = \left\lceil \frac{p(t)}{1-p(t)} \right\rceil$$

$$velocidadRed_{real}(t) = \frac{velocidadRed_{ping}}{1+\#reintentos(t)}$$

donde $p(t)$ es la probabilidad de fallo de la red para conexiones que duran t segundos, $F(t)$ es el número de conexiones pasadas que fallaron antes o exactamente en t segundos, $S(t)$ es el número de conexiones pasadas que no fallaron con una duración igual o menor que t, y T es el número total de conexiones pasadas (de cualquier duración). En la Figura 4.6 podemos observar un ejemplo de cómo la probabilidad $p(t)$ varía debido a nuevas conexiones que fallan o finalizan sin errores.

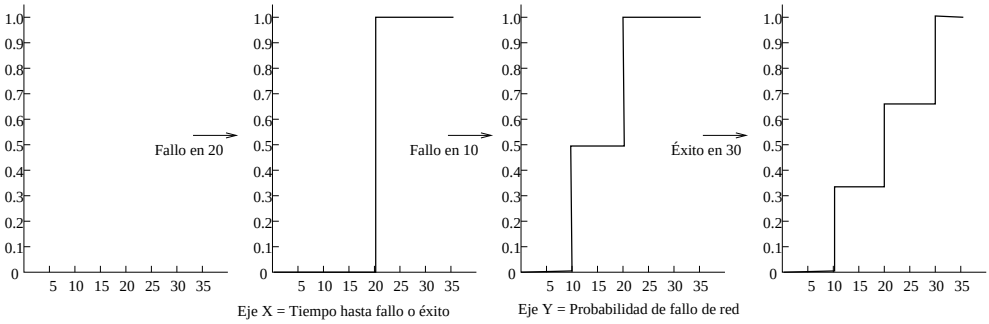


Figura 4.6: $p(t)$, probabilidad de fallo de red

Finalmente definimos $t_{reintento}(t) = t \times (1 + \#reintentos(t))$, es decir, una función que, dada un tiempo t, devuelve el tiempo total necesario (en el peor caso) para mantener una conexión de red abierta durante t segundos, después de considerar el número de reintentos estimado.

4.3. Navegando el catálogo: el agente Navegador

Después de que se obtenga el primer catálogo, el agente Gestor de Software crea un agente Navegador inicializado con dicho catálogo. Este agente especializado viajará al dispositivo del usuario y le ayudará a encontrar el software deseado. El agente Navegador presenta el catálogo como un digrafo acíclico con raíz única (véase la Figura 4.7), donde están las categorías de software. Para ayudar a los usuarios, bajo cada nodo en el catálogo hay una barra que representa gráficamente: 1) cuanta información sobre ese nodo se está mostrando (en gris claro); y 2) cuanta información nueva sobre ese nodo se podría solicitar al Gestor de Software (en gris oscuro). Por ejemplo en la Figura 4.7, respecto al nodo "Java", el Navegador está mostrando el 22 % de toda la información disponible en la ontología (zona clara de la barra bajo la etiqueta "Java"), y el 78 % restante se podrían solicitar remotamente al Gestor de Software (zona oscura de la barra).

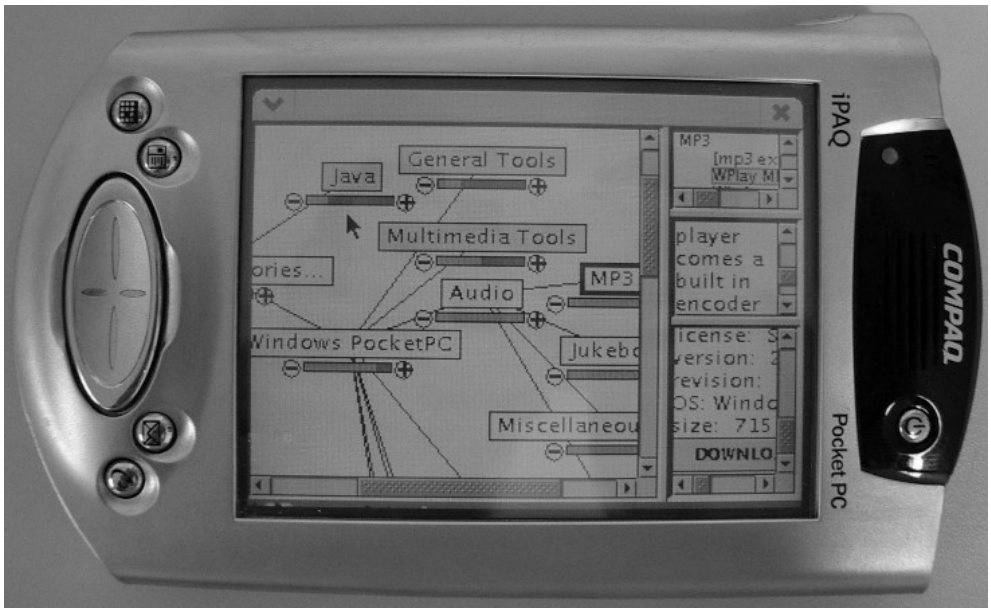


Figura 4.7: Navegando el catálogo: captura de pantalla en un PDA

A continuación explicaremos las diversas acciones que el usuario puede realizar en el catálogo [MRIG02b], debido a que el agente Navegador analiza el comportamiento del usuario para anticiparse a las futuras acciones del usuario, y maneja los refinamientos del catálogo que solicitan nueva información sobre un cierto nodo.

4.3.1. Navegando catálogos: acciones del usuario

Las acciones que puede realizar un usuario cuando navega por un catálogo de software se dividen en dos tipos: remotas y locales. El sistema debe minimizar las acciones remotas

debido a que tienen unos costes temporales y económicos muy elevados. Sin embargo, las acciones locales no tienen coste económico solamente tienen coste temporal y mucho menor que el de las acciones remotas. Sin embargo, también deben minimizarse las locales para reducir al máximo el tiempo que el usuario consume en el proceso de búsqueda.

Las siguientes son las diferentes acciones que un usuario puede realizar después de estudiar el catálogo que se le ha mostrado:

- *Pedir información acerca de un nodo*. Haciendo click en un nodo, el Navegador muestra (en el lado derecho del GUI) todas las características de ese nodo, incluyendo la lista de programas que hay bajo él, si es una categoría de software. Si es un programa, solamente se mostrará la información relativa a dicho programa: una descripción textual, requisitos para su instalación, tamaño del instalador, etc.

- *Abrir/Cerrar un nodo*. Haciendo doble click sobre un nodo, sus descendientes inmediatos son mostrados/ocultados, alternativamente.

- *Podar un cierto nodo*. Haciendo click sobre los símbolos '+'y '-' la barra de un nodo, el usuario tiene la posibilidad de solicitar un mayor o menor nivel de detalle para ese nodo. El usuario puede proporcionar también nuevas restricciones para ese nodo y sus descendientes. Así, diversas acciones pueden ser realizadas:

  - *Solicitar menos detalle de un nodo*, pulsando '-'de la barra, ya que cuando se muestran demasiados descendientes bajo ese nodo, la tarea de encontrar el software deseado se hace demasiado confusa para usuarios inexpertos. Hay que destacar que esta tarea no tiene un coste económico debido a que es realizada localmente en el dispositivo móvil del usuario por el agente Navegador.
  - *Solicitar más detalle de un nodo*, pulsando '+', cuando el usuario puede intuir que el programa deseado podría estar bajo dicho nodo. El Navegador podría tener la información solicitada (no se necesita comunicarse remotamente) o no (el Navegador tendrá que solicitar remotamente esos datos al Gestor de Software en el proxy). El catálogo de software proporcionado al agente Navegador por el agente Gestor de Software o los incrementos de dicho catálogo debidos a refinamientos remotos se optimizan para minimizar el coste de las comunicaciones. Sin embargo, si el agente Navegador mostrase toda la información del catálogo de software que posee al usuario podría llegar a confundirlo, debido a un exceso de información. Debido a esto el agente Navegador gestiona un almacén intermedio permitiéndole resolver ciertos refinamientos localmente. El tamaño de este almacén intermedio será el incremento del tamaño catálogo debido a la optimización de las comunicaciones. Las Secciones 4.3.3 y 4.3.4 detallan cuando y como se realizan los refinamientos locales y remotos, respectivamente.
  - *Añadir nuevas restricciones*, como el usuario puede recordar alguna característica del programa deseado y desear proporcionar una nueva restricción (un nuevo par $<$*restricción, valor*$>$[6]). Como el agente Navegador tiene capacidades de poda,

[6]En nuestro prototipo, las distintas restricciones se unen mediante el operador lógico Y, por lo tanto la adhesión de una nueva restricción implicará un menor tamaño del catálogo de software.

entonces esta tarea puede ser realizada localmente, en el nodo del usuario, sin ninguna conexión remota. La adhesión de estas restricciones se realiza mediante el botón derecho del ratón; en los dispositivos móviles uno de los botones tiene asociada esta funcionalidad.

- *Descargar un programa*, cuando el usuario (afortunadamente) encuentre un software que satisface sus necesidades, pulsa el botón descargar. Para realizar esta tarea el agente Navegador, justo antes de terminar su ejecución, crea remotamente un agente Vendedor en el proxy, que visitará el dispositivo del usuario llevando el programa especificado.

Todas estas acciones realizadas por el usuario deben ser estudiadas por el agente Navegador para minimizar el coste temporal y económico del proceso de búsqueda.

4.3.2. Selección automática de la estrategia de poda: análisis del comportamiento del usuario

Explicamos en la Section 4.2 que los catálogos pueden ser podados siguiendo distintos métodos, que denominamos estrategias de poda. Así la estrategia de poda indica *qué* nodos de la ontología serán seleccionados hasta conseguir el porcentaje indicado por el nivel de detalle solicitado. El resultado de aplicar distintas estrategias de poda es diferente aunque se seleccionen el mismo número de nodos. Por lo tanto, distintas estrategias de poda seleccionan en primer lugar distintas clases de nodos (los más frecuentemente solicitados, aquellos que contienen más programas, etc.).

Gracias a que el agente Navegador estudia el comportamiento del usuario e intenta anticiparse a las futuras acciones del mismo analizando su comportamiento actual y pasado, se minimiza el número de refinamientos del catálogo de software mostrado al usuario, es decir, el número de acciones del usuario necesarias para encontrar el software deseado [MRIG02b]. Así, el agente Navegador almacena los nodos en los cuales el usuario tenía cierto interés en el pasado. Cuando el Navegador detecta que el usuario parece seguir un patrón que se corresponda con los nodos que habrían sido seleccionados por algunas de las estrategias de poda disponibles, esa estrategia de poda será utilizada en la siguiente poda, cuando el usuario requiera un nivel de detalle distinto para ese nodo. Sin embargo, el agente Navegador puede acertar con el cambio de estrategia de poda o no. Cuando el agente acierta después de realizar un cambio de estrategia de poda lo considerará en futuras ejecuciones del servicio y aplicará el cambio de estrategia de poda con mayor anticipación. Por el contrario si la predicción de dicho agente es errónea, éste debe aprender de su error y en futuras ejecuciones del servicio no aplicar la estrategia de cambio de poda hasta que el usuario no haya seguido un determinado patrón de comportamiento durante más tiempo. El cambio de estrategia de poda es realizado por dicho agente cuando el número de nodos visitados consecutivamente por el usuario, coincidiendo con los seleccionados por una estrategia de poda distinta a la aplicada actualmente supera un cierto umbral, gestionado dinámicamente. Es decir, gracias a la inteligencia del agente se ajusta automáticamente el cambio de estrategia de poda, permitiendo de esta forma ayudar al usuario en el proceso de búsqueda.

A modo de ejemplo, en la Figura 4.8 mostramos como un catálogo cambia cuando se utilizan distintas estrategias de poda. Utilizamos la siguiente notación para indicar las diversas

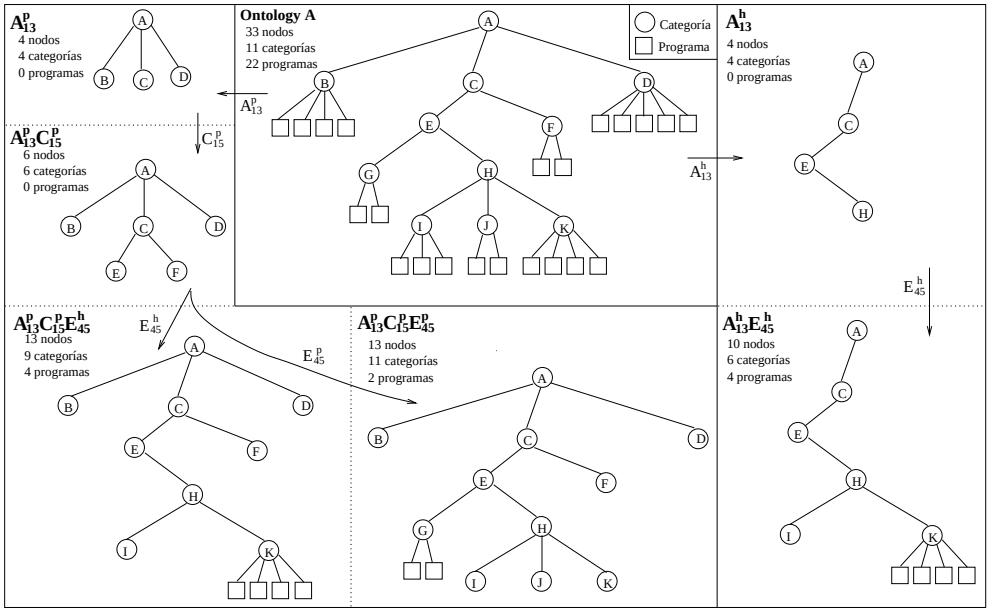


Figura 4.8: Analizando el comportamiento del usuario y seleccionando una nueva estrategia de poda

podas: N_{ndd}^{cad} significa que el nodo N fue podado usando la estrategia de poda cad (abreviaturas: ‘ p’= proporcional, ‘ h’= el más pesado) y un nivel del detalle de ndd %. En este ejemplo se utiliza un umbral igual a tres[7] para cambiar la estrategia de poda, la poda proporcional es utilizada por defecto. En $< A_{13}^{p} >$ mostramos el primer catálogo presentado al usuario (la ontología entera es podada para construir el primer catálogo). Entonces el usuario requiere un nivel de detalle más alto del nodo C; la estrategia de poda sigue siendo igual que en la primera poda, y el resultado es $< A_{13}^{p} C_{15}^{p} >$. Finalmente el usuario requiere un mayor nivel de detalle del nodo E; es la tercera vez que el usuario selecciona la trayectoria que habría sido mostrada si el nodo de la raíz hubiese sido podado usando la estrategia más pesada ($< A_{13}^{h} >$). Por lo tanto, el Navegador selecciona la estrategia de poda más pesada para hacer más fácil al usuario la tarea de encontrar el software deseado, porque el/ella parece estar interesado en nodos con muchos programas; el resultado es $< A_{13}^{p} C_{15}^{p} E_{45}^{h} >$. Con la estrategia proporcional que el resultado sería $< A_{13}^{p} C_{15}^{p} E_{45}^{p} >$. Si el usuario está buscando programas bajo el nodo K, la estrategia más pesada podría haber seleccionado esos nodos en tan sólo dos refinamientos $< A_{13}^{h} E_{45}^{h} >$ del usuario. Sin embargo, seleccionar la estrategia más pesada desde el principio es una opción muy arriesgada, debido a que en primer lugar se seleccionarán aquellas categorías que incluyen un mayor número de programas.

Por lo tanto, el agente Navegador considera los nodos por los que el usuario demuestra algún interés y, siempre que el usuario parezca seguir un patrón reconocido durante cier-

[7]El umbral por defecto debe ser un valor pequeño, tres o cinco, que aumentará cada vez que el agente Navegador detecte un error en su valoración. El sistema almacena un umbral para cada usuario y estrategia de poda.

to tiempo, se seleccionará la estrategia correspondiente. Siempre que el usuario no siga el patrón de la estrategia de poda actual, el agente Navegador seleccionará la estrategia proporcional (que es la menos arriesgada posible) hasta que se siga de nuevo un patrón. El agente Navegador "recuerda" los errores anteriores, y el umbral de una estrategia rechazada se aumenta siempre que el usuario cambie su patrón de comportamiento después de haber sido seleccionada; así el Navegador intenta mejorar sus predicciones (automáticamente).

4.3.3. Tratamiento local de un nuevo refinamiento

Algunas acciones seleccionadas por el usuario son realizadas por el agente Navegador, sin usar recursos de red:

- *Peticiones del usuario para abrir/cerrar un nodo*. El agente Navegador simplemente muestra/oculta los descendientes del nodo, ninguna nueva información es necesaria.
- *Peticiones del usuario de un nivel de detalle menor de un nodo*. Debido a que se basa en mostrar menos información de la que se está visualizando actualmente, tampoco se necesita nueva información y el Navegador tiene capacidades de poda, por consiguiente él mismo podará el nodo seleccionado considerando el nuevo nivel del detalle.
- *Peticiones del usuario de un nivel más alto de detalle de un nodo y bajo el límite del almacén intermedio que posee el agente Navegador*. Una vez más el Navegador tiene ya toda la información necesaria y puede podar el catálogo sin usar la red.

Debemos destacar que, usando las capacidades de poda del Navegador y un almacén intermedio, muchos refinamientos del usuario pueden ser realizados sin usar conexiones de red.

4.3.4. Tratamiento de los refinamientos que implican la utilización de la red

Algunos refinamientos solicitados por el usuario no pueden ser realizados por el propio Navegador: el usuario puede solicitar información que el Navegador no tiene, por lo tanto se necesitará realizar una nueva petición remota al Gestor de Software. Para esa tarea, el Navegador crea el agente Actualizador de Catálogos cuyo objetivo es recuperar del proxy la información necesaria, solicitando al Gestor de Software la poda del nodo objetivo del refinamiento del usuario según el nivel de detalle y las restricciones especificadas. El agente Actualizador de Catálogos podría crearse remotamente en el proxy pero si esa creación remota falla debido a la inestabilidad de la red, el Navegador debería revisar tal tarea, la creación local del agente es más robusta. Crear el agente localmente permite que el Navegador delegue en el Actualizador de Catálogos el manejo de los problemas de comunicaciones debidos a fallos de red. Durante la realización de un refinamiento el agente realiza las siguientes tareas:

- *Creación incremental de catálogos*: Según lo explicado en la sección 4.2, los nuevos catálogos se devuelven de una manera incremental (solamente la nueva información se

transmite para optimizar los costes de comunicación). Así, el Actualizador de Catálogos combina correctamente el anterior catálogo del usuario con la nueva información, y después acaba su ejecución. De esta manera, el Navegador aumenta su conocimiento con cada refinamiento del usuario, haciendo menos frecuente la necesidad de conexiones remotas. Así pues, los refinamientos futuros se pueden atender más rápidamente y evitando el uso de la red.

- Decisión automática entre un refinamiento basado en RPC o en el movimiento del agente: Para actualizar el catálogo de software del usuario, el agente Actualizador de Catálogos puede seguir dos alternativas: 1) una llamada a procedimiento remoto desde el dispositivo del usuario al Gestor de Software localizado en el proxy, o 2) viajar al proxy, y comunicarse con el Gestor de Software localmente, y viajar de nuevo al dispositivo del usuario. Para seleccionar una opción, el Actualizador de Catálogos considera el número de reintentos necesarios para mantener una conexión de red abierta durante cierto tiempo (véase la sección 4.2.3). Definimos $t_{llamadaRemota}$ como el tiempo necesario para actualizar un catálogo usando una llamada remota al Gestor de Software, y $t_{agenteMovil}$ como el tiempo necesario para actualizar un catálogo viajando al proxy, donde $t_{nuevoCat}$ es el tiempo consumido por el Gestor de Software para obtener el catálogo de software solicitado, y $t_{reintento}(s)$ es el tiempo que resulta después de considerar el número estimado de reintentos para mantener una conexión abierta durante s segundos (véase la sección 4.2.3):

$$
\begin{aligned}
t_{llamadaRemota} &= t_{reintento}(t_{llamada} + t_{nuevoCat} + t_{resultado}) \\
t_{agenteMovil} &= t_{reintento}(t_{irAlProxy}) + t_{nuevoCat} + t_{reintento}(t_{volverAlPDA})
\end{aligned}
$$

De las fórmulas mostradas anteriormente podemos extraer la conclusión que si se produce una desconexión, situación muy frecuente en entornos inalámbricos, en una arquitectura basada en llamada a procedimientos remotos deberíamos comenzar el refinamiento del catálogo de software desde el comienzo. En cambio, en una aproximación basada en agentes móviles el impacto es menor, debido a que los errores de red únicamente afectan al agente cuando se está moviendo a través de la red entre el dispositivo del usuario y el proxy. Sin embargo, si no se producen errores de red la aproximación RPC tiene un menor coste, debido a esto el agente actualizador de Catálogos selecciona una estrategia u otra en tiempo de ejecución dependiendo de: 1) la velocidad de la red estimada justo antes de realizar la conexión, 2) la estimación del catálogo de software a transmitir y 3) de la probabilidad de fallo de red.

En lo referente al coste de las conexiones de GPRS y UMTS, debemos destacar que el tiempo pasado en la obtención de un nuevo catálogo no es relevante pues la red no se utiliza (se factura por el tamaño de los datos transmitidos). En tal caso, se realiza una traducción de la formulas presentadas anteriormente, multiplicando los tiempos por la velocidad de la red real estimada en tiempo de ejecución.

4.4. Prestaciones: Tucows versus el Servicio de Recuperación de Software

En esta sección presentamos una comparación entre una arquitectura basada en una arquitectura cliente/servidor, como Tucows, y una arquitectura basada en agentes móviles inteligentes, el SRS. Primeramente, modelamos ambos sistemas para compararlos analíticamente y, posteriormente, mostramos una comparación empírica basada en pruebas realizadas por usuarios de ambos sistemas, relacionando las ventajas e inconvenientes de cada aproximación con el tipo de arquitectura utilizada. Es decir, mostrando cuáles son las ventajas e inconvenientes de utilizar una arquitectura basada en agentes.

4.4.1. Comparación basada en modelos de coste

Primero definimos un modelo para sistemas tipo Tucows (basado en la navegación de páginas HTML), y después definimos un modelo de coste para nuestro sistema y finalmente comparamos ambas aproximaciones analíticamente.

Servicio de recuperación de software basado en la navegación de páginas HTML

La manera más común de obtener nuevo software es navegando páginas HTML organizadas en categorías de depósitos de software (públicos o pagando por descarga). El usuario selecciona una categoría, lee la información, hace click en otro enlace y repite este proceso hasta que finalmente solicita un software. Es importante destacar que en una aproximación basada en la navegación de páginas HTML la red es accedida en cada click, por lo tanto esta aproximación no considera factores muy importantes como la robustez y la optimización de las comunicaciones inalámbricas. A continuación describimos $C_{HTMLnav}$, el coste asociado a la obtención de un software usando la navegación de páginas HTML.

$$C_{HTMLnav} = n \times (C_{nuevaPagina} + C_{leerPagina}) + C_{descarga}$$

donde n es el número de enlaces que el usuario navega hasta encontrar el programa deseado; $C_{nuevaPagina}$ es el coste asociado al acceso de una nueva página HTML; $C_{leerPagina}$ es el coste que corresponde al tiempo consumido por el usuario leyendo la página web para encontrar el programa a descargar o para solicitar una nueva página HTML; $C_{descarga}$ es el coste asociado a la descarga del software seleccionado.

Debemos destacar que en el modelo anterior no distinguimos entre conexiones de red GSM, GPRS y UMTS. En GSM, el coste depende de cuánto tiempo se utiliza la red, por lo tanto $C_X = T_X \times C_{porSec}$, donde T_X es el tiempo consumido en la realización de la tarea X, y C_{porSec} es el coste de la comunicación por segundo. En GPRS y UMTS, el coste depende del tamaño de los datos transferidos a través de la red, por lo tanto $C_X = |X| \times C_{porByte}$, donde $|X|$ es el número de bytes transmitidos a través de la red durante la ejecución de la tarea X, y $C_{porByte}$ es el coste de la comunicación por octeto. Para el usuario, en nuestro contexto, la única diferencia entre usar conexiones GSM o GPRS-UMTS es que, en GSM $C_{leerPagina} = T_{leerPagina} \times C_{porSec}$ (como la conexión de red se mantiene abierta mientras

el usuario lee una nueva página HTML), y en GPRS y UMTS $C_{leerPagina} = 0$ (la red no se utiliza mientras el usuario lee una nueva página HTML).

Así, observando $C_{HTMLnav}$, un sistema basado en la navegación de páginas HTML depende principalmente de: 1) la velocidad de transmisión de la red (tanto usando GSM como GPRS o UMTS), y 2) el número de iteraciones necesarias para seleccionar una pieza de software (es decir, n). El primer problema está fuera del alcance de los desarrolladores de sistemas software, y el segundo parámetro es reducido al mínimo por la mayoría de los depósitos de software clasificando la enorme cantidad de software disponible en su jerarquía de categorías. Sin embargo, en general, los usuarios necesitan varios "clicks", empíricamente podemos decir que: $n \geq 5$ incluso cuando los usuarios conocen perfectamente el depósito de software así como lo que están buscando[8]. Los usuarios inexpertos que no saben la categoría del software, ni incluso el software en sí mismo, podrían hacer click un gran número de veces antes de encontrar el software deseado; algunos podrían incluso perderse en la taxonomía de categorías. Nuestro sistema intenta reducir al mínimo el número de iteraciones anticipando la información necesitada por el usuario.

Nuestra propuesta: el servicio de recuperación de software

El proceso de nuestro Servicio de Recuperación de Software es el siguiente: primero, el usuario recibe la visita de un agente (el Navegador) que ayuda al usuario a seleccionar el software más apropiado navegando un catálogo modificado conforme a los requisitos particulares de ese usuario concreto. El usuario puede solicitar información más detallada hasta que finalmente selecciona una pieza de software. Entonces, un agente nuevo llega al dispositivo del usuario (el Vendedor) con la pieza de software seleccionada. A continuación describimos C_{SRS}, el coste asociado al Servicio de Recuperación de Software:

$$C_{SRS} = C_{navegadorCat} + C_{leerCat} + m \times (C_{refinar} + C_{leerCat}) + C_{vendedorSoft}$$

donde $C_{navegadorCat}$ es el coste asociado a la primera llegada del agente Navegador con el primer catálogo; $C_{leerCat}$ es el coste asociado al tiempo pasado por el usuario para navegar por un catálogo para solicitar un nuevo refinamiento o seleccionar el programa a descargar; m es el número de los refinamientos del catálogo pedidos por el usuario; $C_{refinar}$ es el coste asociado a la petición de un nuevo refinamiento del catálogo; y $C_{vendedorSoft}$ es el coste asociado a la llegada del agente Vendedor con el software seleccionado. Según lo comentado anteriormente, en GPRS y UMTS, $C_{leerCat} = 0$ debido a que no se están transmitiendo datos a través de la red inalámbrica.
Simplificamos la expresión anterior considerando las siguientes propiedades:

1. $C_{navegadorCat} = C_{navegador} + C_{cat}$, es decir, el coste asociado a la llegada del agente Navegador al dispositivo del usuario más el coste asociado al catálogo que lleva.
2. $C_{vendedorSoft} = C_{vendedor} + C_{descargar}$, es decir, el coste asociado a la llegada del agente Vendedor al dispositivo del usuario más el coste asociado a la pieza de software que lleva.

[8]No estamos considerando el uso de motores de búsqueda disponibles en algunos depósitos de software, pues cualquier aproximación puede proporcionarlos.

3. En cuanto al número de los refinamientos (m) pedidos por el usuario, consideramos $m = l + r$, donde l es el número de refinamientos del catálogo que se pueden resolver localmente (por el Navegador), y r es el número de refinamientos del catálogo que implican que el Navegador utilice la red para obtener tal información. Así, $m \times C_{refinar} = l \times C_{refinamientoLocal} + r \times C_{refinamientoRemoto}$.

Por lo tanto, la expresión del coste para el Servicio de Recuperación de Software es la siguiente:

$$\begin{aligned} C_{SRS} = \quad & C_{navegador} + C_{cat} + C_{leerCat} + l \times (C_{refinamientoLocal} + C_{leerCat}) \\ & + r \times (C_{refinamientoRemoto} + C_{leerCat}) + C_{vendedor} + C_{descargar} \end{aligned}$$

Comparación analítica de costes de ambos procedimientos.

Consideramos las suposiciones siguientes entre la aproximación del SRS y la basada en páginas HTML:

- $C_{leerCat} \equiv C_{leerPagina}$ (el tiempo consumido por el usuario para leer un catálogo es similar al que se pasa leyendo una página web con categorías).

- $n \equiv 1 + m$ (la primera transmisión del catálogo y los m refinamientos en el SRS son equivalentes a las n paginas web en la navegación del HTML). Así, $n \equiv 1 + l + r$. Podemos ver en la Figura 4.9 el número de los refinamientos obtenidos en pruebas reales (véase la sección 4.4.2) para descargar diversos programas. Debemos resaltar que, en promedio, los usuarios han hecho 7.1 refinamientos usando Tucows y 7.5 usando el SRS, por lo tanto la aproximación es aceptable.

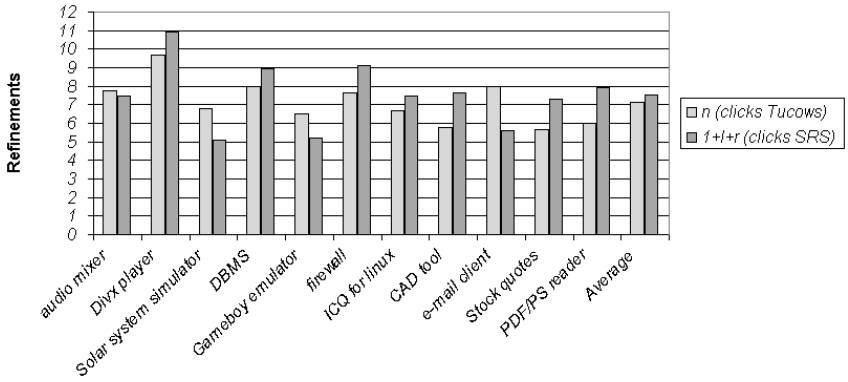


Figura 4.9: Número de refinamientos en Tucows y el SRS en pruebas reales

- $C_{refinamientoRemoto} \equiv C_{nuevaPagina}$, la información transferida a través de la red por el Navegador para realizar un nuevo refinamiento es similar a una página web.

- $C_{refinamientoLocal} \equiv 0$ (es localmente realizado por el Navegador en el dispositivo del usuario sin utilizar la red).

Por lo tanto:

$$C_{SRS} < C_{HTMLnav} \Longleftrightarrow |Navegador| + |catalogo| + |Vendedor| < (l+1) \times |nuevaPagina|$$

Debemos destacar que todos los costes se ven afectados por la velocidad de la red, así que reescribimos las expresiones en términos del tamaño. Podemos observar que, si nos olvidamos del tamaño de los dos agentes (en nuestro prototipo, el tamaño del Navegador es 40K y el tamaño del Vendedor es 8K), las claves del éxito de nuestra aproximación son el tamaño del catálogo llevado inicialmente al dispositivo del usuario y el número de los refinamientos que el Navegador es capaz de realizar sin ayuda externa (l), lo cual también depende directamente del tamaño del catálogo inicial así como de la "inteligencia" de los agentes.

La valoración de la relación entre el tamaño del catálogo y l no es fácil de hacer teóricamente. Por esa razón hemos realizado un conjunto de experimentos explicados en la subsección siguiente, donde el tamaño medio del catálogo inicial llevado al dispositivo del usuario es de 1750 bytes, en media l es igual 4.51 a refinamientos atendidos por el Navegador sin usar la red, y el tamaño medio de la página web al usar Tucows es 20K (el tamaño de la página web más pequeña descargada es 11K), lo cual verifica que el SRS es aproximación mejor.

4.4.2. Evaluación empírica de las prestaciones

En esta sección comparamos mediante pruebas realizadas con usuarios reales el uso del Servicio de Recuperación de Software (SRS) con el uso de Tucows. Los datos han sido obtenidos después de probar ambos métodos de recuperación de software por diversas clases de usuarios finales (47 usuarios en total). Cada usuario recuperó varias piezas de software, primero con el SRS y después con Tucows. La mayoría de los usuarios habían utilizado ya Tucows antes de la prueba.

En la Figura 4.10 demostramos el tiempo total gastado en las distintas tareas (a) cuando usas Tucows y (b) al usar el SRS, en una red cableada; en la Figura 4.11 mostramos los mismos experimentos usando un PDA con red Bluetooth inalámbrica. El Eje X muestra los distintos programas que buscaron los usuarios, y el eje Y muestra el tiempo medio consumido. Debemos resaltar que, aunque usar Tucows puede a veces ser más rápido, con el SRS se reduce el coste debido a que se minimizan las comunicaciones de red (por ejemplo al buscar un DBMS, según se muestra en la Figura 4.10).

En la Figura 4.12 mostramos cómo Tucows y el SRS se comportan desde el punto de vista del tamaño de los datos transmitidos a través de la red. Distinguimos el comportamiento del SRS en cuatro situaciones distintas, que dependen de la inteligencia de los agentes: 1) cuando se considera el estado de la red (véase la sección 4.2.3) y la selección automática de la estrategia de poda (véase la sección 4.3.2); 2) cuando se considera solamente el estado de la red; 3) cuando se considera solamente la selección automática de la estrategia de poda; y 4) cuando ninguno de ambos mecanismos "inteligentes" es utilizado. Como podemos observar, un usuario con una conexión de GPRS o UMTS pagaría menos dinero usando el SRS (cuando

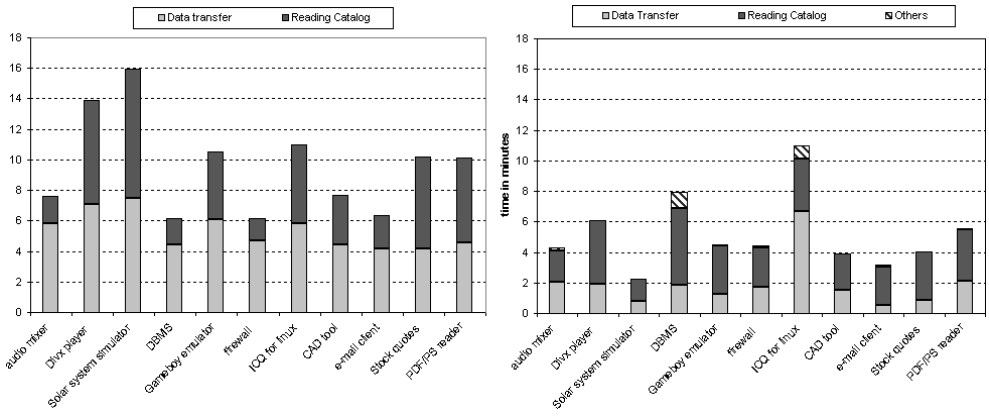


Figura 4.10: Tiempo consumido en las diferentes tareas en (a) Tucows y en el (b) SRS en una red cableada

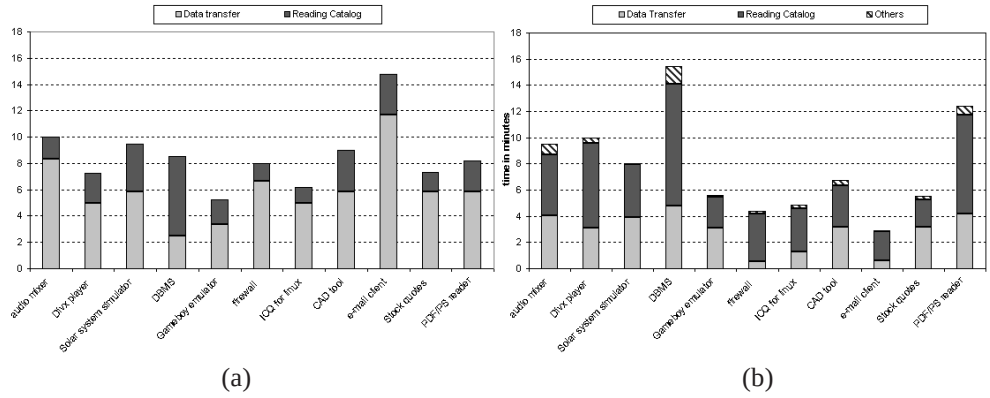
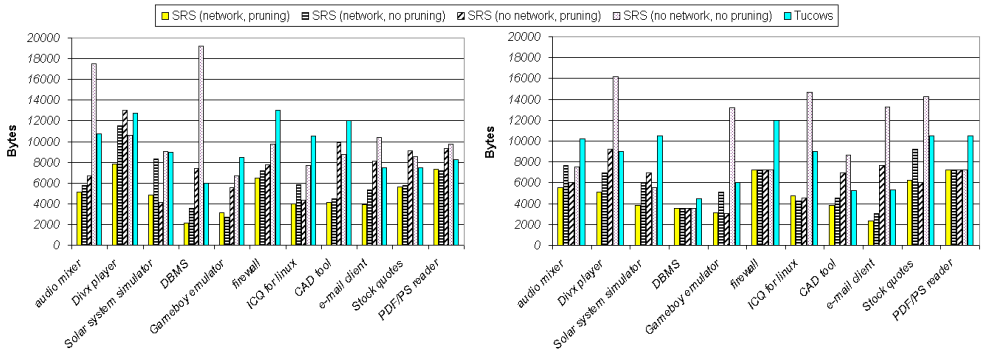


Figura 4.11: Tiempo consumido en las diferentes tareas en (a) Tucows y en el (b) SRS en una red Bluetooth

las opciones que consideran el estado de la red y la estrategia de poda automática están activadas) que usando Tucows; en ese caso, el SRS transfiere menos datos que la aproximación seguida por Tucows en todos los experimentos[9]. Como puede observarse en la Figura 4.12 gracias a la inteligencia de los agentes se minimiza la cantidad de datos transmitidos a través del enlace inalámbrico.

En [MCM03] se puede encontrar una comparación analítica entre el funcionamiento de sistemas tipo Tucows y nuestro SRS; ambos sistemas fueron modelados primero en UML y después usando redes de petri [MCB84, MBC+95]. Como resultado de ese análisis (véase la

[9]El tamaño de la página incluye solo el código del HTML, es decir, no incluimos los gráficos y la publicidad.



(a) (b)

Figura 4.12: Bytes transmitidos en las distintas tareas con una red (a) cableada y una (b) bluetooth

Figura 4.13)[10] se concluyó que el SRS se comporta mejor que Tucows cuando por lo menos el 40 % de los refinamientos se gestionan localmente (sin uso de la red). En la Figura 4.14.a podemos ver que el porcentaje de los refinamientos del usuario manejados localmente por el SRS en las pruebas reales, en una red cableada, considerando el estado de la red y la selección de la estrategia de poda o no, está siempre sobre el 40 %. La Figura 4.14.b demuestra el mismo hecho para una red Bluetooth.

Mediante las pruebas realizadas con usuarios se demuestra que gracias a la utilización de agentes móviles inteligentes se ha diseñado una arquitectura robusta que minimiza el coste de las comunicaciones inalámbricas en un servicio de recuperación de software. Al mismo tiempo que permite ayudar a usuarios inexpertos en el proceso de búsqueda, tarea muy compleja debido a que el usuario puede especificar un conjunto de restricciones ambiguo que debe ser completado por el agente Alfredo.

4.5. Resumen del capítulo

En este capítulo hemos comenzado la descripción de uno de los casos de estudio diseñados en el transcurso de la tesis doctoral. Como se ha mostrado, el Servicio de Recuperación de Software (SRS) desarrollado permite que usuarios inexpertos descarguen e instalen en su dispositivo móvil nuevo software. Además esta tarea tiene las siguientes características:

- *Fácil*, porque con la ayuda de agentes inteligentes, los usuarios pueden navegar localmente la ontología que describe semánticamente el contenido de un conjunto de fuentes de datos que contienen piezas de software, y así pueden seleccionar el software de este catálogo (el servicio hace transparente para los usuarios las características técnicas de sus dispositivos y la localización y método de acceso de varios almacenes de software remotos.

[10] Los datos fueron obtenidos para un escenario con una velocidad real de 1 K/sec (en una red inalámbrica), un tamaño de página web/catálogo de 50 K y un retraso entre los refinamientos del usuario de 60 segundos.

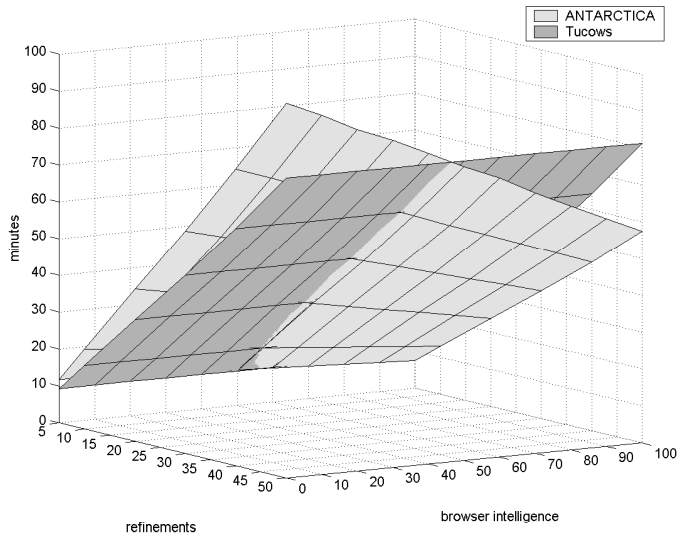
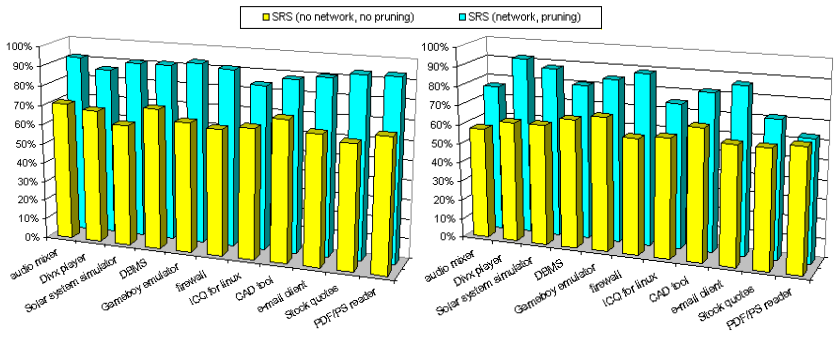


Figura 4.13: Porcentaje de refinamientos del usuario capturados localmente por el SRS calculados analíticamente



(a) (b)

Figura 4.14: Porcentaje de refinamientos del usuario capturados localmente por el SRS en pruebas reales en una red (a) cableada y en una (b) Bluetooth

- *Eficiente*, porque sacando ventaja de las propiedades inherentes de los agentes móviles para tratar las desconexiones y moverse a otros ordenadores, los agentes optimizan el uso de los medios inalámbricos personalizando los catálogos de software que el usuario visualiza.

- *Adaptativa*, porque los agentes consideran el estado de la red (para decidir el método de comunicación y la cantidad de datos a transferir), las acciones llevadas a cabo por el usuario en el pasado (para anticiparse a futuras peticiones), y también aprenden de

sus propios errores (decisiones equivocadas tomadas autónomamente por el agente).

Por consiguiente, con el Servicio de Recuperación de Software se demuestra las ventajas proporcionadas por los sistemas multiagente en entornos inalámbricos. Las principales ventajas son: adaptabilidad al contexto de ejecución y al tipo de usuarios, reducción del tráfico de red y robustez frente a redes inestables como la inalámbrica. Además de mostrar la capacidad inherente de los agentes para tomar decisiones por el usuario de forma autónoma.

En el siguiente capítulo detallaremos, cómo se construye la ontología utilizada por el servicio de recuperación de software.

Capítulo 5

SoftOnt: la ontología de software en el SRS

En este capítulo estudiaremos las ventajas proporcionadas por las arquitecturas basadas en agentes inteligentes en el tedioso y complejo trabajo de la creación e integración de bases de conocimiento u ontologías en un dominio particular, como es el de los depósitos web que almacenan software. Entre las principales ventajas proporcionadas por el diseño de una arquitectura basada en un sistema multiagente en el contexto de la ingeniería del conocimiento se encuentran: 1) la fácil adaptación a dominios particulares; 2) vocabularios reducidos, 3) cambios frecuentes de la estructura de los depósitos de datos utilizados para construir la ontología; y 4) variaciones de la disponibilidad de recursos (memoria, disco, comunicación inalámbrica, etc.) en el tiempo.

Un componente muy importante del Servicio de Recuperación de Software (SRS) es la ontología SoftOnt. Las ontologías, en general, son herramientas de especificación muy interesantes para describir un conjunto de elementos de interés en un dominio particular o concreto [Gru93]. Los términos en la ontología SoftOnt describen programas software y se enlazan a los depósitos de software correspondientes con la información de enlace, que es obtenida y manejada automáticamente por los agentes que constituyen el sistema. La información de enlace almacenada en la ontología permite que a partir de un término de la ontología que representa a un programa se pueda saber a qué depósito de software pertenece y desde donde debe ser descargado e instalado.

Sin embargo, hay que considerar que los depósitos subyacentes de la ontología son actualizados con el paso del tiempo, haciéndose necesaria la automatización del proceso de creación de la ontología SoftOnt. Aplicando una solución de tipo “divide y vencerás”, la creación de SoftOnt se realiza en dos fases: creación de una ontología para cada uno de los depósitos de datos subyacentes (por un agente especializado en la creación automática de ontologías) e integración de las distintas ontologías creadas previamente (por un segundo agente especializado en la integración de bases de conocimiento). La creación automática de una ontología que represente la información almacenada en un depósito de datos es una tarea muy compleja y no puede realizarse automáticamente. Sin embargo, como mostraremos a lo largo

del capítulo, el agente encargado de este proceso en el servicio de recuperación de software, sí que puede llevarla a cabo debido a que el vocabulario utilizado para describir depósitos de software es muy reducido. Por otra parte, la necesidad de integrar las distintas ontologías en una única se fundamenta en presentar al usuario (inexperto) una visión global del software disponible en el sistema. Evitando de esta forma el acceso a distintos depósitos de datos y centralizando el proceso de búsqueda.

Mediante una arquitectura basada en agentes inteligentes se facilita la división y el diseño del proceso de creación de la ontología en dos fases, debido a que cada una de estas tareas será realizada por un agente especializado. Verificándose de esta forma una de las propiedades fundamentales de los agentes como es estar orientados a un fin. Por otra parte, el agente encargado del proceso de creación de las ontologías para cada uno de los depósitos de datos tendrá que ser lo suficientemente inteligente para crear un conjunto de términos y las relaciones semánticas existentes entre los mismos a partir de un conjunto de páginas HTML destinadas a ser utilizadas por usuarios humanos, es decir, el agente deberá extraer conocimiento e información semántica. En una etapa posterior, otro agente será capaz de extraer relaciones a partir de los términos obtenidos en el análisis de los distintos almacenes de software creando una ontología global.

Finalmente, destacar que, como demostraremos a continuación, gracias a la utilización de una arquitectura basada en agentes, los procesos de extracción e integración de conocimiento pueden realizarse de una forma independiente y permitiendo una fácil adaptación en caso de que se modifique la estructura de los distintos depósitos de datos.

5.1. SoftOnt: una ontología de software construida automáticamente

Las ontologías son herramientas de especificación muy importantes para la tediosa tarea de describir conjuntos de términos que tengan algún interés en un dominio particular como son los depósitos de software. Los términos de la ontología están enlazados con los distintos depósitos de software a través de información de enlace [MI01], la cual es gestionada por el sistema propuesto. En esta sección vamos a describir las ventajas e inconvenientes de utilizar una ontología frente a varias y la posibilidad de que la misma sea construida automáticamente. La principal ventaja de la utilización de una única ontología se dirige a facilitar el proceso de búsqueda de software, debido a que el usuario que utiliza el SRS puede ser inexperto. Por otra parte, la automatización del proceso de creación de la ontología por un conjunto de agentes inteligentes, facilita el mantenimiento del sistema, debido a que una actualización manual es demasiado costosa.

5.1.1. Una ontología frente a múltiples ontologías

Cuando las ontologías son utilizadas para describir el contenido de distintas fuentes de datos en un dominio concreto, debemos decidir si trabajaremos con una ontología global integrada o con múltiples ontologías enlazadas por relaciones interontología [MI01]. En general,

las características relevantes que deben ser consideradas antes de elegir entre una de las dos aproximaciones son las siguientes:

- El número de fuentes de datos que deben ser integradas. Un número elevado implicaría una información de enlace compleja en el caso de una ontología global.

- El número de categorías que se necesitará extraer de las distintas fuentes de datos. Un número muy grande implicará trabajar con muchos términos y relaciones entre ellos, en el caso de una ontología global.

- El problema del vocabulario utilizado en las distintas fuentes de datos. La integración de fuentes de datos diseñadas bajo diferentes puntos de vista desembocará en una ontología global con muchos términos debido a la existencia de muchas especializaciones y generalizaciones, y a la ausencia de relaciones de sinonimia entre las fuentes de datos.

Sin embargo, considerando las propiedades mencionadas anteriormente, proponemos la construcción de una única ontología que considere el dominio del software por las siguientes razones:

1. *El número de fuentes de datos es reducido*. En el prototipo desarrollado únicamente se integran unos pocos almacenes de software que permiten el acceso a la mayoría del software (*freerware/shareware*) disponible en la Web. Hay que resaltar que los sitios web más populares tienen la mayor parte del software y de las categorías de software disponibles. De hecho, en la mayoría de las ocasiones los usuarios encuentran la pieza de software que buscan visitando únicamente uno de los sitios web.

2. *El número de categorías no es muy alto*. Aunque diferentes organizaciones pueden desarrollar diferentes clasificaciones de los diferentes tipos de software, los sitios web más grandes comparten un número menor que mil (alrededor de 270 en Download.com y alrededor de 900[1] en Tucows). Únicamente en el caso de varios cientos o miles de tipos de software (que constituirían una ontología enorme) podríamos necesitar una aproximación distribuida.

3. *El problema de la heterogeneidad de vocabulario es limitado*. El dominio de las restricciones impuestas por los distintos tipos de software no permite una gran heterogeneidad con respecto a los nombres utilizados para describir las distintas categorías. Es decir, en los distintos depósitos web se utilizan los mismos vocablos o términos que son sinónimos, pero difícilmente aparecen términos que tengan una relación de homonimia. Este hecho se fundamenta en que no se utilizan todos los significados de las palabras polisémicas sino un único significado. Por ejemplo, con la categoría "Herramientas de red", en el contexto del software solamente te puedes referir a un software que va a ser utilizado en una red de computadores. Despreciando significados de la palabra red como instrumento de pesca.

[1] Después de eliminar los sinónimos internos entre diferentes sistemas operativos.

Así pues, gracias a las características del contexto en el que nos encontramos podemos utilizar una única ontología que represente todo el software disponible en el sistema. Facilitando de esta forma el proceso de búsqueda que realizarán los usuarios inexpertos del SRS, debido a que la pregunta formulada por el usuario se utiliza para acceder a la información de todos los depósitos de software y no debe ser traducida.

5.1.2. Construcción automática versus construcción manual de la ontología

Otra decisión que debe ser considerada cuando se trabaja con ontologías es el proceso de construcción automática frente al proceso de construcción manual de la ontología. Desde hace tiempo, es ampliamente aceptado que la automatización del proceso de federar diferentes fuentes de datos es difícil, debido a la necesidad de gestionar información semántica que no puede ser extraída automáticamente [SGN93], debido a la heterogeneidad sintáctica y semántica sobre todo. Sin embargo, enumeramos a continuación las características de nuestro contexto, principalmente en lo concerniente a los distintos tipos de fuentes de datos involucradas, que permiten una federación automática:

- *Heterogeneidad sintáctica baja*. Para los distintos sitios web que contienen software (como Tucows, Download.com, etc.), necesitaremos diseñar un conjunto de wrappers[2] [HBGM+97, PGGMU95] especializados que extraerán información de páginas HTML o XML. Es importante acentuar que la mayoría de los sitios web públicos ya están organizados en categorías (juegos, redes, entretenimiento, por sistemas operativos, etc.) y por lo tanto, los wrappers podrán aprovecharse de esta característica. Por lo tanto, los agentes encargados del proceso de creación de la ontología analizarán la estructura jerárquica de un sitio web, donde las subcategorías en el sitio web se transformarán en subtérminos (especializaciones) en la ontología (ver Figura 5.1[3]). Además, hay que tener en cuenta que el acceso es relativamente sencillo, debido a que está basado en el protocolo HTTP, por lo que el desarrollo de los wrappers no será una tarea difícil.

- *Baja heterogeneidad semántica*. En el contexto de los depósitos de software se observa, visitando los diferentes sitios web, que la mayoría de ellos tienen muchas categorías en común y que se utiliza un conjunto de términos reducido para clasificar los programas. Por ejemplo, los tipos de categorías de entretenimiento en Tucows (Figura 5.1) son bastante similares a los tipos de juegos en Download.com (Figura 5.2).

Por lo tanto, la integración automática es posible y una excelente solución para la construcción del catálogo de software. Ya que se evita la intervención humana y los agentes encargados del proceso de construcción de la ontología podrán actualizar el software disponible automáticamente con una cierta periodicidad; permitiendo en definitiva incrementar la calidad del software ofertado y del servicio proporcionado.

[2] Es un módulo o agente que proporciona un API para acceder a un depósito de datos.

[3] Las imágenes mostradas en las Figuras 5.1 y 5.2 fueron tomadas en el año 2000, cuando se envió el artículo [MIG00b].

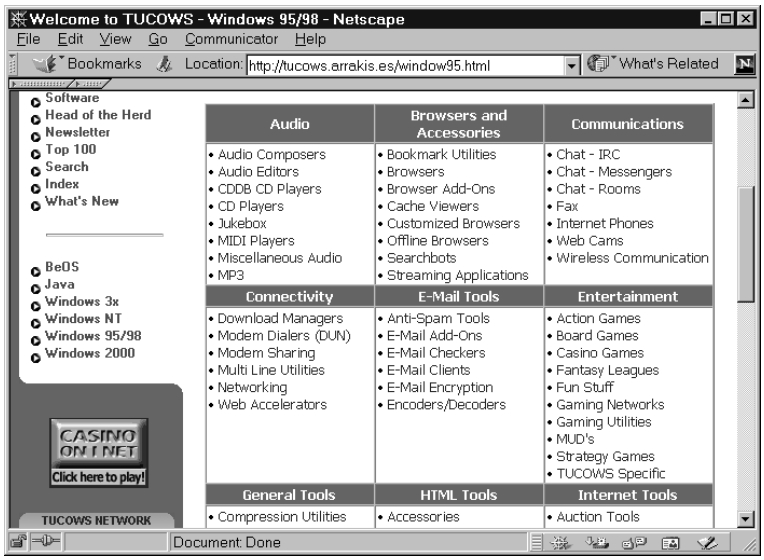


Figura 5.1: Categorías de Windows 95 en Tucows

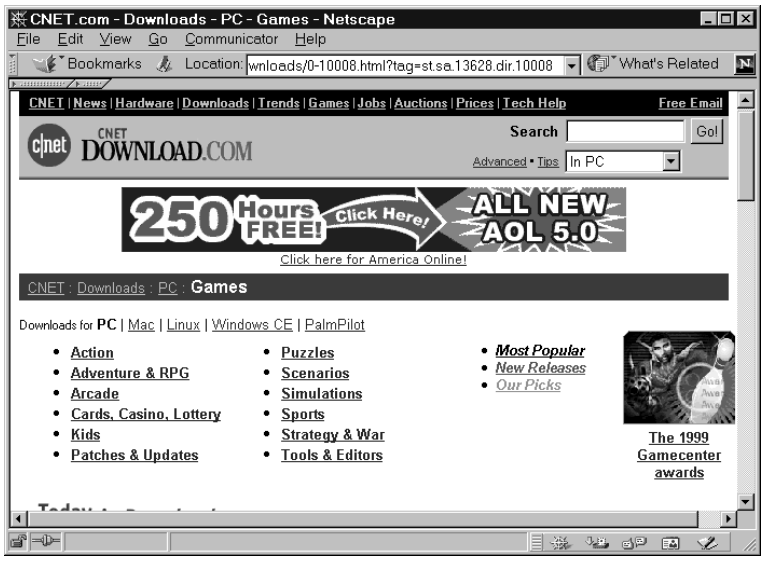


Figura 5.2: Categoría juegos en Download.com

5.2. Construcción automática de SoftOnt

Como se ha comentado anteriormente en este capítulo, es ampliamente aceptado que la automatización del proceso de federar diversas fuentes de datos es difícil debido a la necesi-

dad de manejar la información semántica que no puede ser automáticamente extraída [SGN93]. Sin embargo, como explicamos anteriormente, en el contexto de la clasificación de piezas de software se facilita esta tarea debido a la baja heterogeneidad sintáctica y semántica.

A continuación describiremos los agentes incluidos dentro de la arquitectura del Servicio de Recuperación de Software (SRS) que permiten la creación automática de la ontología utilizada por el servicio. La creación e integración de la ontología utilizada por el SRS es gestionada por dos agentes ubicados en el *place* de adquisición de software (véase Figura 5.3); estos agentes realizan las tareas de construir la ontología SoftOnt [MIG00b] siguiendo los dos pasos clásicos definidos en la literatura especializada de fuentes de datos federadas, traducción [LG90, Dev93] e integración [IHMT04]:

1. El agente *Ingeniero del Conocimiento* aplica técnicas de minería de datos a los depósitos de software (locales o remotos), con la ayuda de *wrappers* [HBGM$^+$97, MI01] especializados, para obtener una descripción semántica (una ontología) para cada depósito de datos.

2. El agente *Integrador* realiza la integración automática de todas las ontologías obtenidas por el Ingeniero del Conocimiento con el objetivo de obtener la ontología integrada, que llamamos SoftOnt. Sin intervención humana, este agente puede actualizarla con cierta asiduidad.

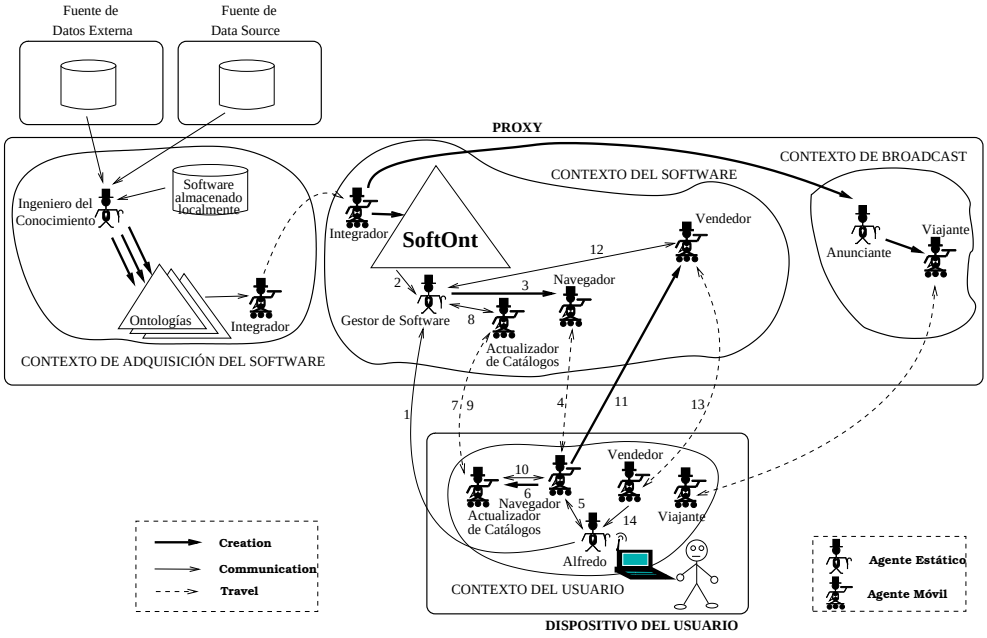


Figura 5.3: Arquitectura extendida del SRS: construyendo SoftOnt

El agente *Informador*, situado en el *place* para hacer broadcast (véase Figura 5.3), es creado por el agente Integrador cuando SoftOnt, la ontología del software, es actualizada. La

meta del agente Informador es informar a los usuarios sobre nuevas versiones del programa disponibles enviando al agente *Vendedor* (que pueden llevar actualizaciones de los programas instalados) a los dispositivos de los usuarios, si esta opción ha sido activada por el usuario.

5.2.1. Proceso de traducción: el agente Ingeniero del Conocimiento

En nuestro contexto, la mayoría de los depósitos subyacentes son sitios web remotos que contienen software, y que se clasifican como depósitos semiestructurados, es decir, no existe un esquema de la información almacenada. Afortunadamente, los sitios web que contienen programas software los clasifican en diversas categorías, y podemos aprovecharnos de esto. Así, nuestra solución para el paso de traducción es la construcción de *wrappers* especializados que tienen acceso a las páginas HTML/XML que componen el sitio web y extraen de ellas las diversas categorías del software (ver Figura 5.4) que se pueden encontrar en el sitio web y la información de los diversos programas (ver Figura 5.5) que puede descargar el usuario. La extracción automática de conocimiento de páginas web es la finalidad de varios trabajos [DH02, PVM+02, IHMT04] y en muchos de los casos se fundamenta en la construcción de *wrappers*.

```
<TD VALIGN="TOP" BGCOLOR="#ffffff">
<FONT FACE="Tahoma" SIZE=2><P>
  <a href="action.html">Action Games</a><br>
  <a href="board.html">Board Games</a><br>
  <a href="casino.html">Casino Games</a><br>
  <a href="fanleague.html">Fantasy Leagues</a><br>
  <a href="fun.html">Fun Stuff</a><br>
  <a href="gnetwork.html">Gaming Networks</a><br>
  <a href="gameutils.html">Gaming Utilities</a><br>
  <a href="mud.html">MUD's</a><br>
  <a href="strat.html">Strategy Games</a><br>
  <a href="tucows.html">TUCOWS Specific</a><br>
<P><a href="preview/281208.html" title="ColorTetris">ColorTetris</a><br>
ColorTetris is a little more than just Tetris <br>
<a href="preview/282680.html" title="4Pinball">4Pinball</a><br>
This pinball game includes scrolling screens...<br>
</FONT</TD>
```

Figura 5.4: Categorías de Entretenimiento de Tucows en HTML (entertainment.html)

El diseño de los *wrappers* debe considerar que las páginas de HTML/XML pueden cambiar. Debido a esto, hemos diseñado un agente Ingeniero del Conocimiento que es capaz de utilizar distintas gramáticas para analizar el contenido de distintos sitios web y extraer conocimiento. Algunos trabajos han desarrollado técnicas para construir fácilmente o adaptar los *wrappers* a depósitos de datos semiestructurados [HBGM+97]. Estos trabajos sugieren el uso de gramáticas libres del contexto para definir la estructura de las fuentes de datos (páginas

```
<FONT FACE="Tahoma" SIZE=2><P><TABLE>
<TR><TD>ColorTetris</TD>
<TR><TD><b>License:</b> Shareware<br></TD>
<TR><TD><b>Price:</b> $4.95<br></TD>
<TR><TD><b>Description:</b>
  ColorTetris is a little more than just Tetris. It also ...<br></TD>
<TR><TD><b>Download:</b>
  <li> <a href="96191.html"><b>SH3</b></a></li>
  <li> <a href="96192.html"><b>ARM</b></a></li>
  <li> <a href="96193.html"><b>MIPS</b></a></li>
    <br></TD>
</TD><TR><TD> <b>Date:</b> Jan 28, 2001<br></TD>
</TD><TR><TD> <b>Size:</b> 621.1KB<br></TD>
</TABLE></FONT>
```

Figura 5.5: Programas de Entretenimiento de Tucows en HTML (preview/281208.html)

HTML/XML, en nuestro caso). En la Figura 5.6 podemos ver la gramática que permite analizar las distintas categorías de software almacenadas en Tucows, y en la Figura 5.7 podemos observar la gramática asociada a un programa.

Mediante la utilización de estas gramáticas, los agentes extraen cierta información de las páginas (categorías de software, en nuestro contexto), tomando una gramática como base. Si sucede un cambio en la estructura de alguna de las páginas HTML/XML, la gramática se debe adaptar a la nueva estructura sintáctica de la página, pero eso no tendría un gran impacto desde el punto de vista de la puesta en práctica. De este modo, un agente construido con esta técnica conserva: 1) el acceso a un sitio web remoto, 2) el uso de técnicas de data mining en sus páginas web y 3) la obtención como resultado de ese proceso de un sistema de categorías y características de los programas (como nombre, descripción, URL, etc) pertenecientes a ellos.

Por otra parte, cuando una ontología describe un sistema de fuentes de datos es necesario definir cierta información de enlace entre los términos de la ontología y los elementos de datos en las fuentes de datos, donde se almacena la información. En nuestro contexto, durante el proceso de minado de los depósitos de software para construir una ontología, toda la información de enlace necesaria también es almacenada. Es decir, para cada categoría de software y cada pieza de software descritos, la ontología también almacena los enlaces (URL's) a los sitios web correspondientes, al software, así como la información descriptiva. Todos estos datos son extraídos por los *wrappers* durante el proceso de traducción.

Como ejemplo, hemos mostrado en las Figuras 5.4 y 5.5 que la información relacionada con las categorías y la información de enlace con otras páginas que contienen software están mezclados con las etiquetas HTML. En las Figuras 5.6 y 5.7, mostramos los esquemas dirigidos por sintaxis que, tomando el HTML anterior como entrada, permiten a los agentes reconocer las categorías y programas. Posteriormente el agente Ingeniero del Conocimiento, ejecutando las acciones entre llaves (en negrita), construye la ontología y la información de

```
<analizarCategoria> ::= <principio> <lista-de-categorias> '<P>'<lista-de-programas> <fin>
        <principio> ::= '<TD VALIGN="TOP" BGCOLOR="#ffffff">'
                        '<FONT FACE="Tahoma" SIZE=2><P>'
<lista-de-categorias> ::= <categoria> <lista-de-categorias>
                        | ϵ
<lista-de-programas> ::= <programa> <lista-de-programas>
                        | ϵ
        <categoria> ::= '<a href=" 'link { url = token.valor } ' ">'
                        nombre-categoria {
                            // Crear una nueva categoría y su información de enlace (url) como subconcepto
                            // de la categoría actual (hipónimo)
                          hiponimo = addNode(hiponimo, token.nombre, url);
                            // Analizar las categorías de la ultima categoría encontrada (seguir su enlace)
                          analizarCategoria(hiponim, hiponim.link) }
                        '</a><br>'
        <programa> ::= '<a href=" 'link { url = token.valor } ' ">'
                        nombre-programa {
                            // Crear un programa y su información de enlace (url) como subconcepto de la
                            // categoría actual (hiponimo)
                          nuevoProg = addNode(hiponimo, token.nombre, url) }
                        '</a><br>'
                        resumen {
                            // Añadiendo el resumen
                          nuevoProg.addRole("resumen", token.valor);
                            // Analizar la pagina web correspondiente al programa (siguiendo su link)
                          analizarPrograma(nuevoProg, nuevoProg.enlace) }
                        '<br>'
            <fin> ::= '</FONT></TD>'
```

Figura 5.6: Extractor basado en gramáticas para categorías de Tucows

enlace de sus términos (véase Figura 5.8): los conceptos representan categorías y piezas de software, y los roles almacenan sus propiedades[4].

Así, el proceso de traducción anterior se repite para otros sitios web, usando diversas gramáticas dependientes del contexto, debido a que los distintos sitios web pueden tener distintas páginas y estructura HTML/XML. Sin embargo, la inteligencia utilizada por el agente para almacenar el conocimiento extraído de los distintos sitios web, no se ve modificada.

[4]Como sistema de representación del conocimiento, nosotros usamos un sistema basado en Lógica de Descripciones [BCM+03]. Por lo tanto, utilizamos los conceptos y los roles de los términos para referirnos a los elementos de datos de la ontología.

```
<analizarPrograma> ::= <principio> <nombre-programa> <lista-de-caracteristicas> <fin>
        <principio> ::= '<FONT FACE="Tahoma" SIZE=2><P><TABLE>'
<nombre-programa> ::= '<TR><TD>ñombre-software '</TD>'
<lista-de-caracteristicas> ::= <caracteristica> <lista-de-caracteristicas>
                             | ϵ
    <caracteristica> ::= '<TR><TD><b>ñombre-caracteristica { rol = token.valor } ':</b>'
                         valor-caracteristica { //Añadir una nueva caracteristica (rol) a un programa
                           addRol(nuevoProg, rol, token.valor) }
                         '<br></TD>'
              <fin> ::= '</TABLE></FONT>'
```

Figura 5.7: Extracto basado en gramáticas para programas de Tucows

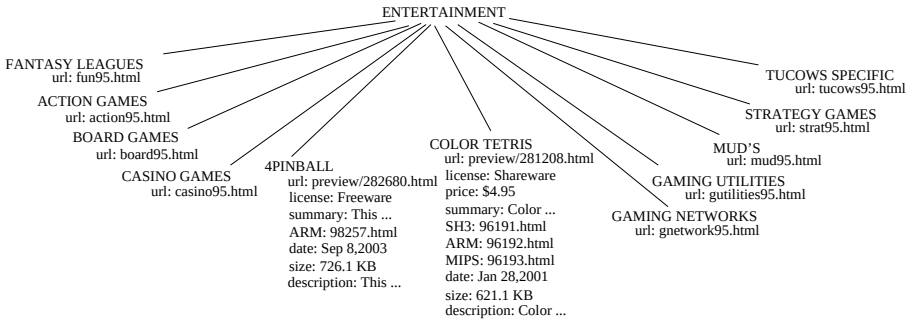


Figura 5.8: Extracto de la ontología SoftOnt con la información de enlace obtenida para las subcategorías y programas de entretenimiento

5.2.2. El agente Integrador

Para realizar la tarea de integrar las diversas ontologías obtenidas por el agente Ingeniero del Conocimiento, hemos diseñado el agente *Integrador*, que está a cargo de crear SoftOnt. Estructuralmente, SoftOnt es un digrafo acíclico con raíz única cuyos nodos internos almacenan la información sobre categorías de software y cuyos nodos sin descendientes (nodos hoja) almacenan la información sobre piezas de software. El Integrador es un agente móvil: después de que se realice la integración se mueve desde el *place* de adquisición de software *al place* del software para poner al día la versión anterior de SoftOnt. El Integrador es también capaz de detectar los cambios ocurridos con respecto a la versión anterior de SoftOnt. Estos cambios se comunican al agente Avisador que alerta a los usuarios interesados sobre ellos (según lo mostrado previamente en la Figura 5.3).

Las ontologías obtenidas a partir de los diversos depósitos de software, durante el paso de la traducción por el agente Ingeniero del Conocimiento, se deben integrar en una única ontología[5], SoftOnt. Como los distintos depósitos pueden haber clasificado su software de

[5]La información de enlace de la ontología integrada puede ser generada automáticamente combinando la información de enlace de las ontologías que se están integrando, como se describe en [BGI99].

diferentes formas, el problema principal que resuelve el agente Integrador es: la extracción de un conjunto de relaciones semánticas entre un conjunto de términos de distintas ontologías. Para resolver este problema el agente Integrador debe resolver los distintos *problemas de vocabulario* que puedan aparecer entre los términos (categorías de software y programas extraídos por el agente Ingeniero del Conocimiento). Es decir, el agente Integrador detecta la existencia de sinónimos, hipónimos o hiperónimos entre términos en diversas ontologías. En un entorno abierto y dinámico, este problema es muy difícil de solucionar aunque se han sugerido algunas aproximaciones, como el sistema OBSERVER [MI01]. En nuestro contexto, el problema del vocabulario no es tan serio debido a la naturaleza restringida del tipo de información almacenado en los depósitos de software: existen muchas clases de software pero, en la mayoría de los depósitos que hemos estudiado, las categorías usadas son muy similares (herramientas de Internet, juegos, audio, vídeo, etc). Por ello, decidimos que el agente Integrador utilizase un tesauro, WordNet [Mil06], para detectar relaciones semánticas y crear nuevo conocimiento. Mecanismos más expresivos pero más complejos, que requieran algún tipo de intervención del usuario, están fuera del alcance de la tesis (algunas proposiciones aparecen en [MIKS00, MBR01, GATTMed]).

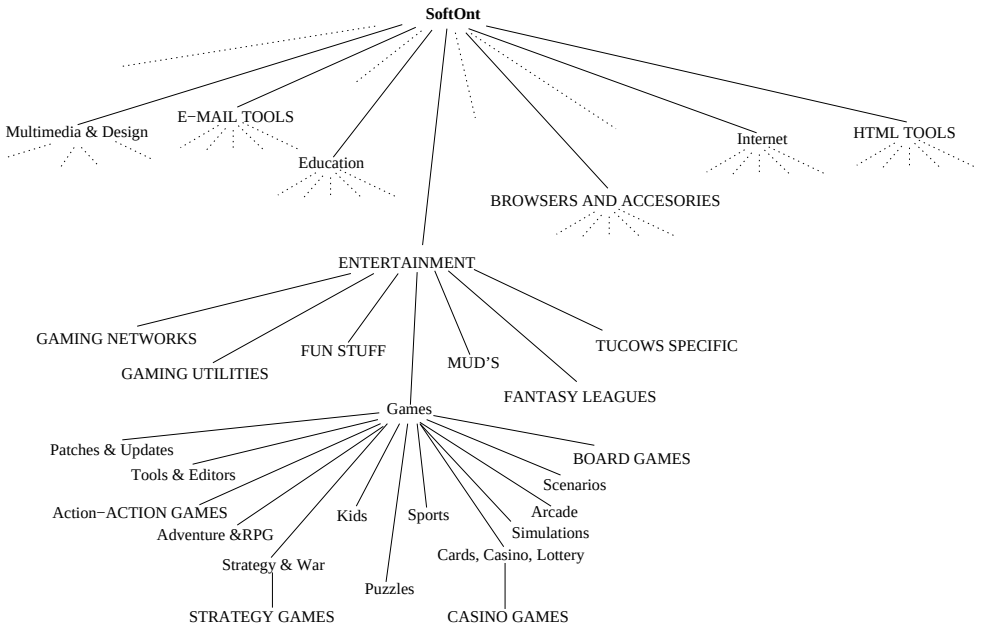


Figura 5.9: Ontología resultante de la integración de Tucows and Download.com

La funcionalidad del agente Integrador, se presenta en la Figura 5.9 (los términos de la ontología asociada al depósito web de Tucows aparecen en mayúsculas mientras que los términos de Download.com aparecen en minúsculas) mostrando un subconjunto del resultado de integrar las ontologías de Tucows [Tuc06] (ver Figura 5.10) y Download.com [CNE06b] (ver Figura 5.11) usando el algoritmo de integración y las características semánticas mostra-

dos en la Figura 5.12. Estas características son descubiertas por el agente Integrador usando WordNet [Mil06] como tesauro; las características adicionales serán analizadas y localizadas por lo agentes con la ayuda de un tesauro especializado en informática.

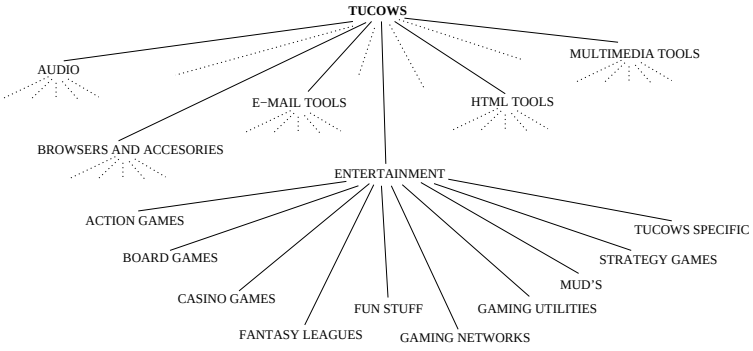


Figura 5.10: Ontologías de Tucows

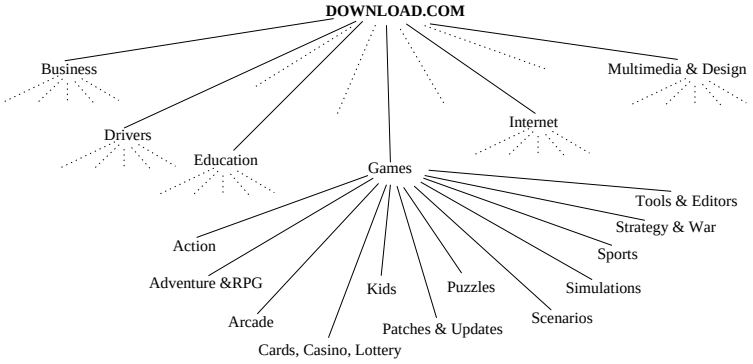


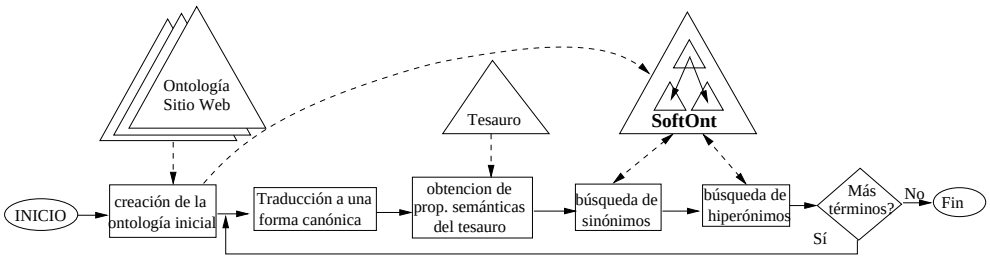
Figura 5.11: Ontologías de Download.com

A continuación explicamos brevemente los pasos del algoritmo de integración:

1. *Creación inicial de la ontología*: Todas las ontologías de los sitios web son unidas por un hiperónimo común, que será la raíz de la ontología integrada.

2. *Traducción a una forma canónica*: El sistema quita las palabras sin significado (preposiciones y artículos) y las raíces de cada término [BYRN99]. Sin embargo, el sistema también considera conjunciones y las separaciones porque, por ejemplo, “Editor Tools” y “Tools & Editors” tienen distinto significado. El resultado es una lista de raíces canónicas.

3. *Conseguir propiedades semánticas del tesauro*: El sistema extrae del tesauro los sinónimos e hiperónimos de cada raíz.



Nuevas relaciones semánticas descubiertas	**Relaciones del tesauro y de las ontologías iniciales**
ENTERTAINMENT $\sqsupseteq$ Games	ENTERTAINMENT $\sqsupseteq$ Games (thesaurus)
ACTION GAMES $\equiv$ Action	ACTION $\equiv$ action $\wedge$ Action $\sqsubseteq$ Games (Download.com) $\wedge$ GAMES $\equiv$ Games
CASINO GAMES $\sqsubseteq$ Cards, Casino, Lottery	CASINO $\sqsubseteq$ Cards, Casino, Lottery (“,” $\equiv \cup$) $\wedge$ Cards, Casino, Lottery $\sqsubseteq$ Games (Download.com) $\wedge$ GAMES $\equiv$ Games
STRATEGY GAMES $\sqsubseteq$ Strategy & War	STRATEGY $\sqsubseteq$ Strategy & War (“&” $\equiv \cup$) $\wedge$ Strategy & War $\sqsubseteq$ Games (Download.com) $\wedge$ GAMES $\equiv$ Games
BOARD GAMES $\sqsubseteq$ Games	GAMES $\equiv$ Games $\wedge$ BOARD GAMES $\sqsubseteq$ GAMES (BOARD es un adjetivo, aunque GAMES no existe en TUCOWS)

Figura 5.12: Tareas realizadas por el agente Integrador y relaciones semánticas descubiertas

4. *Búsqueda de sinónimos*: Cada término en la ontología integrada se compara al resto de los términos. Un término t_1 es un sinónimo de t_2 si para cada palabra en t_1 existe una palabra sinónima en t_2.

5. *Búsqueda de hiperónimos*: Cada término se compara con el resto de los términos. Un término t_1 es un hiperónimo de t_2 si por lo menos una palabra en t_1 es hiperónima de una palabra en t_2, y el resto de las palabras tienen sinónimos en t_2.

6. Los pasos 2 a 5 son repetidos para cada término en la ontología integrada.

El algoritmo anterior tiene un coste $O(n^4)$, donde n es el número de términos de todas las ontologías. El coste es reducido al mínimo buscando los sinónimos en primer lugar (dos términos sinónimos se unen en uno único, por lo tanto el número de nodos se decrementa). Quisiéramos acentuar que las características semánticas detectadas por el agente son insertadas usando un sistema basado en lógica de descripciones, que reclasifica automáticamente toda la ontología integrada según las relaciones semánticas descubiertas.

5.3. Enriquecimiento semántico de la ontología SoftOnt

Semánticamente, la ontología creada automáticamente por el agente Integrador no es tan expresiva como si hubiese sido construida “manualmente” por un ingeniero del conocimiento

humano. De hecho, el resultado depende de la expresividad de los catálogos de los depósitos de software individuales usados en el paso de traducción, y también de la eficacia del proceso de integración (el tesauro y el método para descubrir propiedades semánticas son la clave de este proceso).

Por ejemplo, anteriormente dijimos que existiría un concepto por categoría de software (e.g. *Multimedia Tools*) con algunos roles; sin embargo, pueden presentarse los siguientes problemas:

- *Sinonimia, hiperonimia y hiponimia*: el usuario debe adivinar la palabra más adecuada para hacer la mejor selección. Por ejemplo, si el usuario desea un reproductor de MPEG-IV entonces los programas clasificados como reproductores de DivX no serán seleccionados, aunque los expertos en software multimedia saben que un reproductor de DivX puede decodificar un archivo de MPEG-IV. Es decir, se debe crear una relación de hiperonimia entre el término reproductor DivX y el término reproductor MPEG-IV, debido a que el primero incluye al segundo.

- *Sobrecarga del rol 'descripción'('description')*: cuando se construye la ontología, el texto de la descripción proporcionado por los depósitos de software se almacena en el rol 'descripción'. No obstante, características muy importantes del software se pueden incluir en esa descripción textual (propósito, requisitos, etc), esa información sería muy difícil de extraer del texto automáticamente. Si la ontología se construye manualmente, esa información clave sería almacenada en propiedades individuales que permitirían búsquedas más eficientes.

- *Carencia de la información sobre categorías*: las categorías de software utilizadas para clasificar programas consisten solamente en un nombre y un sistema de subcategorías. Por tanto, la información adicional tal como una descripción y características generales sobre el software bajo tal categoría sería deseable.

Por lo tanto, a pesar de que la ontología generada automáticamente puede ser buena para muchos usuarios (tal y como se demuestra en las pruebas realizadas con usuarios reales), si deseamos proveer a tales usuarios con una vista semánticamente más rica del software disponible, debemos modificar la ontología construida automáticamente. Hay que resaltar el hecho de que la ontología no es actualizada con mucha frecuencia (podría ser actualizada una vez al mes porque los depósitos de software no cambian significativamente a diario).

Para describir la ontología de software SoftOnt, decidimos utilizar OWL [AvH03], un lenguaje de representación de ontología recomendado por el W3C. Esta elección se fundamenta en la aceptación del OWL como un lenguaje de representación de conocimiento en entornos de investigación y empresariales. Del mismo modo OWL es un lenguaje lo suficientemente potente como para describir todos los tipos de relaciones semánticas existentes dentro de la ontología del software; en caso de ser necesario una ampliación de estas primitivas *OWL Full* también proporciona primitivas que permiten la definición de nuevos tipos de restricciones semánticas.

Por otra parte, el proceso del razonamiento realizado por el agente *Gestor de software* es realizado por un sistema basado en Lógica Descriptiva, concretamente RACER [HM01], que

implementa un sistema de cálculo que permite el tratamiento de expresiones complejas basadas en lógica de descripciones [BCM+03] y que cubre todas las primitivas de *OWL DL* que necesitamos. Debido a la utilización OWL y un razonador basado en lógica de descripciones, la pregunta del usuario puede ser simplificada, y su consistencia verificada, automáticamente. Por ejemplo, un usuario que posee un PDA podría especificar la palabra clave *formatSupported=divx* para obtener una cierta aplicación que reproduzca una película de DivX. El agente Alfredo del usuario agregaría la información técnica, tal como que el PDA es, por ejemplo, un iPAQ H3870 206MHz de Compaq con 32 MB Flash ROM, 64 MB Flash y tarjeta de sonido de 16 bit, entre otras características. En este caso, después de recibir la petición correspondiente, el agente Gestor de Software puede obtener (usando un razonador basado en lógica de descripciones) que no existe ningún programa con esas características que se pueda instalar en el PDA especificado porque, según la información almacenada en la ontología sobre los reproductores de DivX, el dispositivo necesita por lo menos una CPU 233Mhz. Por lo tanto, el sistema evita presentar al usuario un catálogo con el software que no funcionará en su dispositivo.

5.4. Resumen del capítulo

En este capítulo hemos detallado el proceso seguido por el agente Ingeniero del Conocimiento y el agente Integrador para la creación de la ontología SoftOnt utilizada por el SRS. El proceso de construcción se divide en dos fases:

- *Proceso de traducción*: en este proceso el agente Ingeniero del Conocimiento analiza la estructura y el contenido de los distintos almacenes de software (locales o remotos) y genera una ontología asociada a cada uno de ellos. Estas ontologías contienen la información de enlace que permite establecer las relaciones necesarias entre los términos que las componen y las piezas de software de los almacenes de datos.

- *Proceso de integración*: en un segundo paso el agente Integrador, mediante la utilización de un tesauro, generará una vista integrada de todos los depósitos de datos que contienen el software que hay disponible en el sistema.

Como se ha mostrado anteriormente el proceso de traducción y su posterior integración se ve facilitado por las jerarquías preestablecidas por los diseñadores de los distintos almacenes de datos. Gracias a esta jerarquía preestablecida por los diseñadores de los depósitos de software, los agentes del sistema pueden considerar qué jerarquías incluyen a otras. Además, el proceso de integración se ve facilitado, debido a que nos encontramos en un contexto concreto y el agente Integrador solamente tiene que establecer relaciones semánticas entre términos que utilizan un vocabulario bien definido.

Gracias a la utilización de la tecnología de agentes, los dos procesos pueden realizarse de forma independiente y permitiendo una fácil adaptación en caso de que se modifique la estructura de los distintos depósitos de datos.

Finalmente, en este capítulo hemos descrito algunos de los problemas del proceso de creación automática de la ontología de software y su posible solución mediante un enrique-

cimiento semántico de la misma, que debe ser realizado manualmente por un ingeniero del conocimiento.

Capítulo 6

Aplicación de la tecnología de agentes al contexto de bibliotecas digitales

El desarrollo de sistemas que permiten la búsqueda y recuperación de información es una de las áreas de mayor interés actualmente. Hasta ahora la mayoría de las aproximaciones se basan en una aproximación cliente/servidor, donde un cliente solicita datos a una serie de servidores. Así el análisis de los datos se realiza en el cliente o dichos datos deben ser enviados al siguiente servidor para realizar un análisis incremental de la consistencia. Ambos métodos requieren una conexión de red continua.

Actualmente se están desarrollando nuevas aproximaciones basadas en agentes software que abren nuevas posibilidades de diseño. En este capítulo proponemos una aproximación basada en agentes móviles inteligentes [IK96, PS98] debido a su capacidad de llevar a cabo tareas especializadas en distintos ordenadores de la red sin necesitad de una conexión continuada con el ordenador del usuario. Concretamente, usaremos los agentes móviles para recolectar datos y analizar las inconsistencias que pueda haber en los distintos almacenes de datos, más concretamente en el contexto de las bibliotecas digitales de publicaciones de investigación. Además se tendrá en cuenta el caso de que puedan producirse errores de red en algún momento del proceso de búsqueda y recuperación de la información. Mostrando las ventajas proporcionadas por una aproximación basada en agentes frente a una aproximación cliente/servidor.

Las ideas aquí presentadas se han aplicado, a modo de escenario de ejemplo, al depósito de publicaciones del grupo de Bases de Datos Interoperantes (BDI) de la Universidad del País Vasco y del grupo de Sistemas de Información Distribuidos (SID) de la Universidad de Zaragoza[1]. En este ejemplo se integrarán mediante una aproximación basada en un agente móvil inteligente un conjunto de referencias bibliográficas almacenadas en ficheros BibTEX localizados en distintos ordenadores. Se propondrá una arquitectura que facilita su posterior

[1] A partir de este momento nos referiremos a la unión de los grupos BDI y SID como BDI-SID.

utilización, tanto por humanos como por agentes inteligentes.

Como mostraremos la utilización de una arquitectura basada en agentes móviles en el diseño de un sistema de análisis y detección de inconsistencias aplicado a un conjunto de depósitos de información distribuido permite obtener las siguientes ventajas: 1) las técnicas de filtrado y detección de inconsistencias extraídas del caso de ejemplo pueden extrapolarse a otros contextos de ejecución, 2) el tratamiento de los datos se hace de forma incremental y local a cada depósito, 3) el análisis de un depósito de datos solamente implica que en el ordenador donde está ubicado se encuentre disponible un place al que puedan trasladar su ejecución los agentes móviles (haciéndose innecesaria la instalación y actualización de ningún software), y 3) la robustez de la arquitectura se fundamenta en las comunicaciones de red asíncronas utilizadas por las plataformas de agentes móviles.

6.1. ¿Qué es una biblioteca digital?

Para diseñar una biblioteca digital y los métodos para poder acceder a la misma primeramente hay que definir qué es una biblioteca digital. Una biblioteca digital [RW98] es una colección de información que es almacenada y accedida electrónicamente. La información almacenada en la biblioteca puede tener un tema común a todos los datos, por ejemplo la biblioteca digital de un grupo de investigación, o contener distintos temas. También hay que destacar que se pueden combinar distintas bibliotecas digitales permitiendo el acceso a las mismas mediante un interfaz común. El propósito de una biblioteca digital es proporcionar una localización centralizada para el acceso a la información de un tema particular. Por consiguiente, una biblioteca digital es una colección de servicios e información, que facilita a los usuarios trabajar con dicha información; la organización y visualización de la información está disponible, directa o indirectamente, mediante medios electrónicos. Sus funciones principales son:

- *Colección de servicios* [SL97, LFP98, Bea05]: Una biblioteca digital es mucho más que una colección de materiales almacenados en sus depósitos de datos. Proporciona una gran variedad de servicios a todos sus usuarios (humanos y máquinas, productores, gestores y consumidores de información). Habitualmente hay una gran variedad de servicios que permiten el manejo de colecciones, servicios de replicación, servicios de búsqueda, etc.

- *Colección de objetos de información* [AS99, YRW+02, LC05]: Los fundamentos de una biblioteca digital debe ser la información que se encuentra almacenada en la misma. Una característica básica de una biblioteca digital es que la información se encuentra organizada en colecciones con un conjunto de operaciones que permiten su gestión. Los tipos de información almacenada varían desde documentos a información multimedia.

- *Dar soporte a los usuarios para que traten la información* [CWW00, THT02]: El objetivo principal de una biblioteca digital es ayudar a los usuarios a satisfacer sus necesidades y requisitos de gestión, acceso, almacenamiento y manipulación de la información almacenada. Además hay que considerar que los usuarios de una biblioteca

digital pueden ser seres humanos o procesos software (agentes software). Debido a esta característica es muy importante que la información y los procesos para el tratamiento de datos almacenados en una biblioteca digital se definan semánticamente, por ejemplo mediante la utilización de ontologías [WA97, WBD99, SS05b].

- *Organización y visualización de la información* [TEDBT99, THT04]: La clave para manejar colecciones de forma efectiva es implementar una organización estructural pero simple de tal forma que pueda ser presentada a los usuarios/agentes de forma fácil y útil.

- *Disponibilidad directa o indirecta* [SH05]: La información que contiene una biblioteca digital puede ser accedida directamente, en el caso de acceder a objetos digitales, o indirectamente, es decir, especificando un método para poder acceder al objeto buscado. Por ejemplo, dando las instrucciones necesarias para acceder a un material no digitalizado.

- *Disponible en formato digital* [FM98, GGMPW02]: Aunque los objetos en algunas ocasiones no pueden ser digitalizados, pueden ser representados de forma digital.

Es decir, una biblioteca digital debe facilitar el acceso a colecciones de publicaciones, bien en formato digital o no, y proporcionar mecanismos que permitan a los agentes software y a los seres humanos acceder a dicha información.

6.2. Escenario de ejemplo: biblioteca digital de los grupos BDI y SID

Para ilustrar la utilidad de los agentes móviles en sistemas de información bibliográficos que deben realizar el análisis de información distribuida en distintos depósitos de datos nos basaremos en un ejemplo de aplicación: el depósito de publicaciones del grupo de Bases de Datos Interoperantes (BDI) de la Universidad del País Vasco y del grupo de Sistemas de Información Distribuidos (SID) de la Universidad de Zaragoza. A continuación describiremos algunas de las características más relevantes de dicho depósito de datos (ver la Figura 6.1).

Nos gustaría destacar que no entraremos aquí a analizar lo acertado de las decisiones que dicho grupo ha ido tomando a lo largo de los años respecto a la gestión de las publicaciones, sino que simplemente describiremos un escenario real que nos sirva como ejemplo para la extracción de conclusiones a cerca de las arquitecturas basadas en agentes móviles inteligentes en el contexto de las bibliotecas digitales. Las características de la biblioteca digital del grupo de investigación son:

- *Tipos de publicaciones*. El trabajo con publicaciones del grupo BDI-SID, al igual que otros muchos grupos de investigación, se reduce a dos tareas: 1) almacenamiento de referencias BibTEX que se usan para elaborar artículos de investigación, y 2) gestión de un sitio web donde se hayan disponibles las referencias y ficheros PostScript de las publicaciones de los miembros del grupo.

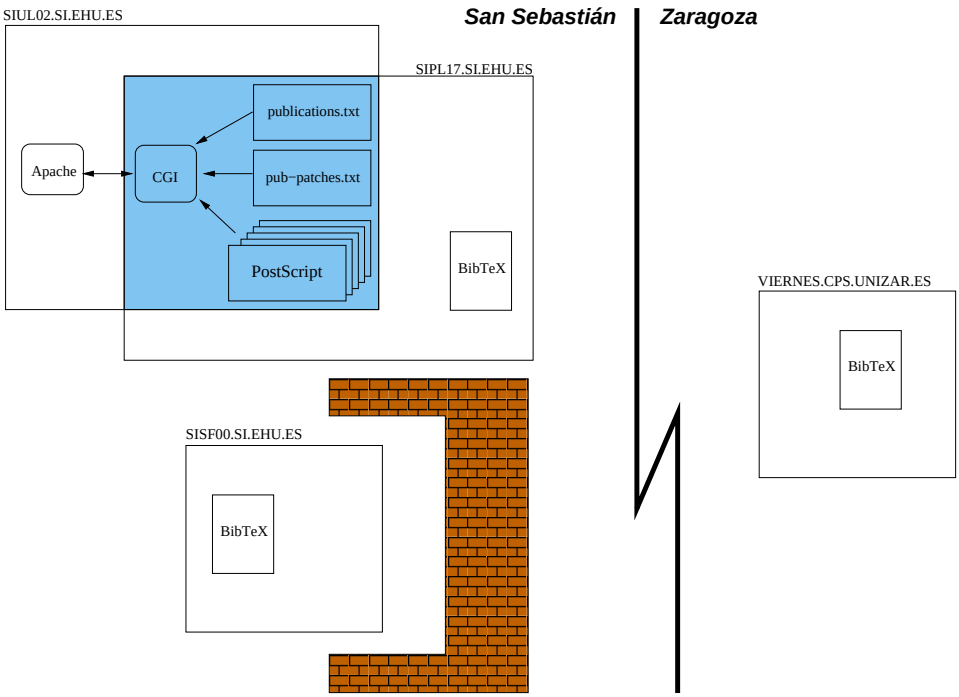


Figura 6.1: Gestión de las publicaciones en el grupo BDI-SID

- *Distribución en San Sebastián*. Desde la formación del grupo BDI-SID, el ordenador denominado sisf00 ha sido utilizado para la elaboración de artículos. Sin embargo, en los últimos años también se ha venido utilizando un ordenador propio del grupo, denominado sipl17. Además, el servidor web del grupo, instalado en el ordenador denominado siul02, ofrece acceso a las publicaciones del grupo así como a los ficheros PostScript de las mismas. Por razones de seguridad y facilidad de uso, parte del disco de siul02 es visible desde sipl17.

- *Distribución entre universidades*. El hecho de que los miembros del grupo BDI-SID trabajen en la Universidad del País Vasco y que los del grupo SID en la Universidad de Zaragoza complica aún más la elaboración de publicaciones debido a dos razones: 1) el uso de herramientas gráficas, como nedit o xfig para la edición de artículos hace que haya que trabajar con ficheros locales debido a la baja velocidad de la red, y 2) la información de sisf00 es inalcanzable desde Zaragoza, debido a que sisf00 se encuentra tras un cortafuegos. Por tanto, los miembros del grupo BDI-SID en Zaragoza trabajan en un ordenador local, llamado viernes.

Por tanto, podemos ver en la Figura 6.1 como el sistema de referencias bibliográficas del grupo BDI-SID está basado en un conjunto de ficheros BibTEX y de otros tipos que se en-

cuentran en ordenadores pertenecientes a ambas universidades, y por lo tanto está distribuido y contiene un alto grado de redundancia.

6.2.1. Publicaciones en la Web

Al igual que la mayoría de los grupos de investigación, el grupo BDI-SID desarrolló un servicio web para facilitar el acceso a las publicaciones del grupo a aquellas personas interesadas. En lugar de diseñar una serie de páginas HTML estáticas, se optó por desarrollar un sistema que, a partir de un fichero de texto describiendo las publicaciones del grupo, generará las páginas adecuadas en tiempo de ejecución. Para ello se diseñó un formulario web enlazado con un CGI con las siguientes funcionalidades:

1. Recibe los parámetros del formulario indicándole las condiciones que deben cumplir las publicaciones a mostrar.

2. Consulta un fichero de texto[2] llamado "publications.txt" que sirve de catálogo para las publicaciones del grupo BDI-SID. Para cada publicación se almacena un identificador, palabras clave, tipo de publicación, año de publicación, autores, y referencia bibliográfica (un dato en cada línea). Mostramos un ejemplo de entrada en la Figura 6.2.

```
caise97
Active systems, Multidatabase
In proceedings
1997
A. Goñi, A. Illarramendi, E. Mena, J.M. Blanco
"Monitoring the Evolution of Databases in Federated Relational Database Systems"
International CAiSE'97 Workshop on Engineering Federated Database Systems (EFDBS'97), Barcelona (Spain), June 1997.
```

Figura 6.2: Entrada ejemplo de "publications.txt"

3. Verifica si existe, en unos ciertos directorios predeterminados, algún fichero PostScript correspondiente a cada publicación, según cierta nomenclatura. Por ejemplo, el PostScript de la publicación "caise97" deberá estar en un fichero denominado "caise97.ps.gz".

4. Consulta un fichero de texto llamado "pub-patches.txt" que almacena código HTML que hay que incluir en ciertas entradas. Su formato es: identificador de publicación, seguido de cualquier texto HTML. De esta forma podemos añadir cualquier tipo de nota anexa a ciertas publicaciones. Mostramos un ejemplo en la Figura 6.3.

En este sistema web únicamente están accesibles las referencias bibliográficas de las publicaciones en las que alguno de los autores es o ha sido miembro del grupo BDI o SID.

Por motivos de seguridad, y dado que tanto sipl17 como siul02 están fuera del cortafuegos, siul02 (que es donde reside el servidor web del grupo BDI-SID) sólo puede ser accedido

[2]El CGI se desarrolló en el año 1995, luego no se utilizó XML para describir las publicaciones.

```
caise97
<P> <A HREF="http://siul02.si.ehu.es/~jirgbdat/PUBLICATIONS/OTHERS/sigmodefdbs97.ps.gz">EFBDS report </A>
published in the December 1997 issue of ACM SIGMOD Record.
```

Figura 6.3: Ejemplo de añadido HTML en "pub-patches.txt"

por el administrador del sistema. Por tanto, y para facilitar tanto la actualización del fichero con las publicaciones del grupo como la adición de nuevos ficheros PostScript, parte del disco de siul02 ha sido exportado a sipl17 (ver en la Figura 6.1 el área sombreada entre sipl17 y siul02). De esta forma los miembros del grupo BDI-SID pueden trabajar solamente en sipl17 y hacer disponibles ficheros en la Web sin ni tan siquiera tener que hacer FTP a siul02.

Con el sistema descrito los miembros del grupo BDI-SID pueden describir sus nuevas publicaciones en un formato sencillo y hacer disponible el PostScript correspondiente simplemente dejando el fichero con el nombre adecuado en uno de los directorios definidos para ello. El CGI (mostrado en las Figuras 6.4 y 6.5) se encargará tanto de seleccionar las publicaciones que cumplen las condiciones indicadas en el formulario de consulta, así como de mostrarlas adecuadamente en formato HTML.

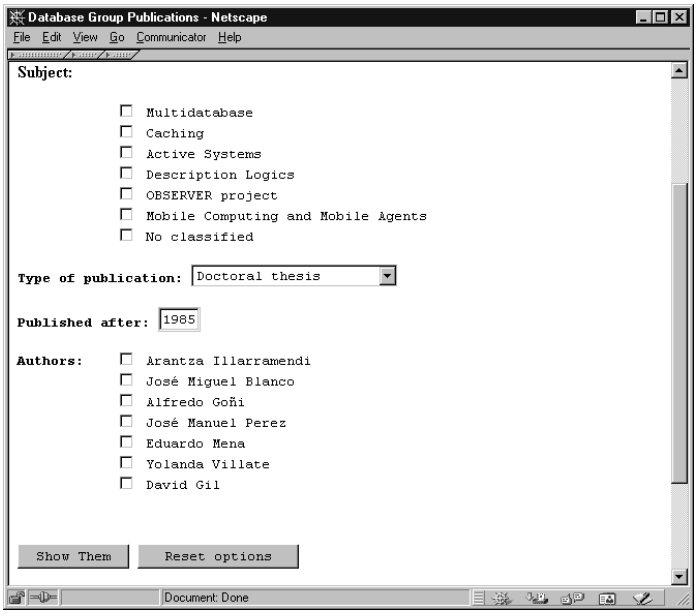


Figura 6.4: Formulario de consulta del CGI

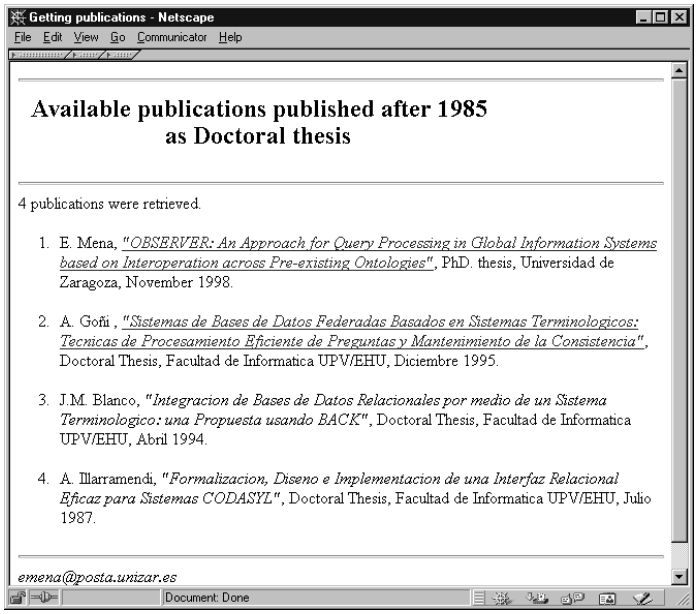


Figura 6.5: Resultados del CGI

6.2.2. Problemas y objetivos a conseguir

Este sistema, a pesar de ofrecer el servicio esperado desde 1995, tiene un conjunto de limitaciones que pasamos a detallar a continuación:

- *Inconsistencia entre ficheros BibTEX*, debido a que algunas citas bibliográficas se describen únicamente en uno de los tres ficheros. O lo que a veces es peor, la misma publicación se describe en distintos ficheros BibTEX pero de forma distinta (distinto identificador, datos incompletos o no actualizados).

- *Duplicidad de datos entre ficheros BibTEX y "publications.txt"*. Los ficheros BibTEX se utilizan únicamente para su referencia desde fuentes LATEX mientras que "publications.txt" se utiliza para interrogar al depósito de publicaciones en la Web y obtener las páginas de respuesta dinámicamente. Sin embargo, las publicaciones del grupo BDI-SID aparecen tanto en "publications.txt" como en alguno de los ficheros BibTEX.

- *Opciones de consulta estáticas*. El formulario de consulta se diseñó como una página HTML estática, es decir las palabras clave, autores y tipo de publicaciones sobre las que se puede consultar está predeterminado. Si cambian los miembros del grupo BDI-SID o se publican artículos sobre nuevos temas se debe modificar manualmente la página HTML para poder preguntar sobre ellos.

- *Consulta únicamente sobre publicaciones del grupo BDI-SID*. Sería interesante poder realizar consultas sobre todo el fondo bibliográfico del grupo, incluido en los ficheros

BibTEX. Aún así no se desea perder la posibilidad de restringirse sólo a las publicaciones del grupo.

Por tanto, los objetivos principales de la aplicación propuesta son los siguientes: 1) *Integración de todos los datos sobre publicaciones*, con la idea de poder preguntar por cualquier publicación disponible en el depósito de publicaciones del grupo BDI-SID; 2) *Análisis de inconsistencias inteligente* entre la información de los distintos depósitos de datos distribuidos; y 3) *Posibilidad de actualizar los distintos depósitos de datos* para mantener la consistencia. Y todo ello sin obligar a cambiar el modo de trabajo de los miembros del grupo BDI-SID ni la funcionalidad de los ordenadores que habitualmente usan.

6.3. Arquitectura del sistema: acceso a datos usando agentes móviles

Para satisfacer los requerimientos indicados hemos diseñado un nuevo sistema, basado en la arquitectura del anterior. De cara a facilitar el acceso a todas las referencias bibliográficas del grupo BDI-SID, almacenaremos la información integrada de todos los ficheros BibTEX en una base de datos relacional. Describimos a continuación las principales aplicaciones (ver Figura 6.6):

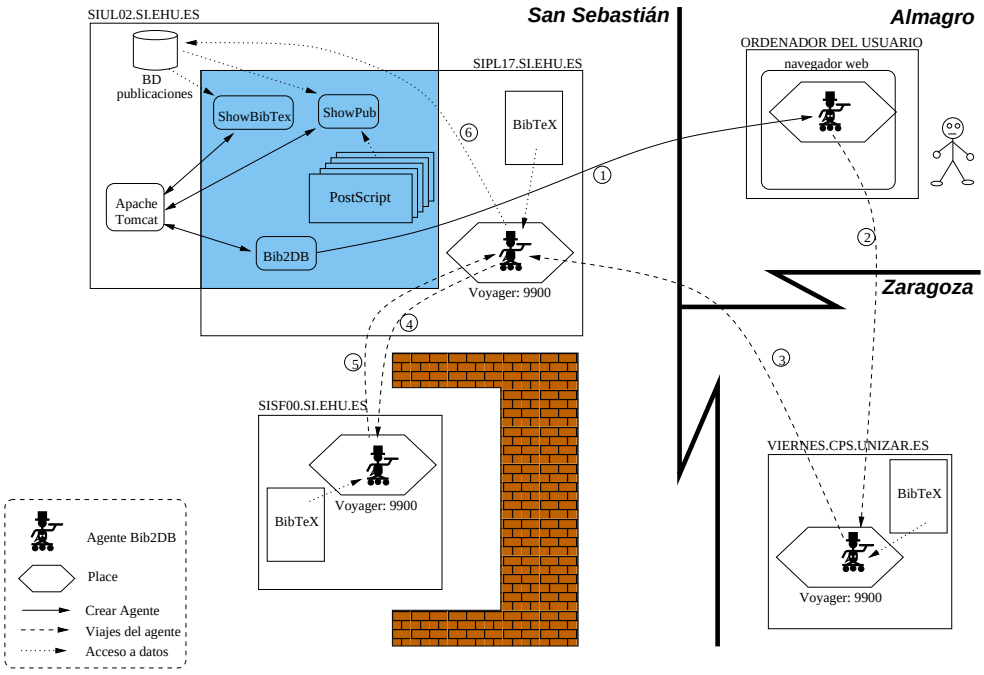


Figura 6.6: Arquitectura de QBIB

- *Bib2DB*. Su objetivo es recopilar referencias bibliográficas distribuidas en varios ficheros BibTEX. Mediante un formulario se definen los ficheros BibTEX que hay que analizar (ver Figura 6.7). Al lanzar la aplicación se ejecutará en el navegador del usuario un applet con dos objetivos iniciales(ver Figura 6.6, flechas numeradas): 1) crear un agente móvil[3] que se encargará de recopilar, analizar y almacenar en una BD las referencias bibliográficas, y 2) mostrar al usuario las tareas que va desarrollando el agente móvil (ver Figura 6.8). Dicho agente irá viajando a los ordenadores indicados y accediendo e integrando la información residente en los distintos ficheros BibTEX. En la Figura 6.6 se han enumerado los pasos correspondientes a una ejecución de ejemplo donde se debe viajar a viernes, sipl17 y sisf00 (con sipl17 como proxy). Finalmente el agente, antes de finalizar su ejecución y tras solicitar la aprobación del usuario, almacena la información integrada en una BD (en siul02, en el escenario de ejemplo) donde queda a disposición de otras aplicaciones. En la sección 6.4 describimos este proceso con más detalle.

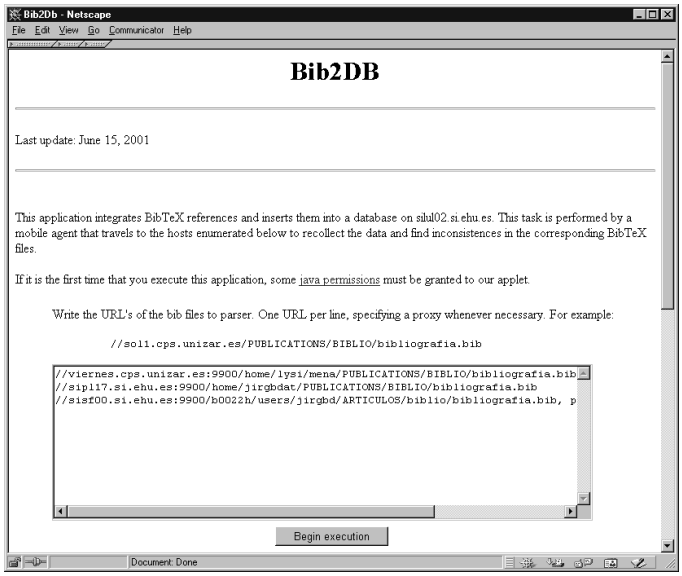


Figura 6.7: Bib2DB: Inicialización del Applet para informar al usuario

Debemos hacer un especial hincapié en las ventajas obtenidas gracias a la utilización de una aproximación basada en agentes. Si se utilizase una aproximación cliente/servidor debería existir un servidor en cada uno de los distintos almacenes de datos, en el caso de la aproximación basada en agentes no es así (solamente debe existir un place que puede ser compartido con otros servicios). Del mismo modo gracias a la utilización de una aproximación basada en agentes se puede filtrar la información de forma local en cada uno de los depósitos de datos, evitando su transmisión a través de la red. Finalmente,

[3]Para ello el applet creará un place en el navegador web del usuario.

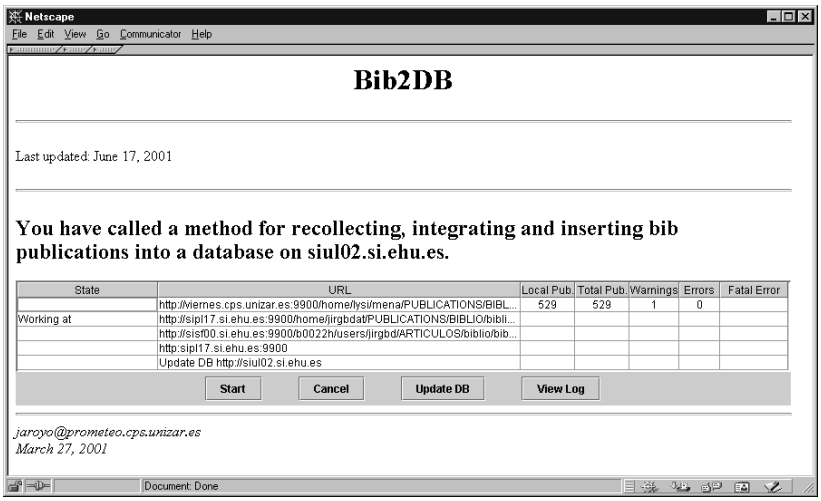


Figura 6.8: Bib2DB: Applet para informar al usuario

destacar que gracias a la utilización de un sistema de agentes móviles se permite la posibilidad de que el agente Bib2DB pueda clonarse a sí mismo para realizar el análisis de más de un fichero bibliográfico de forma concurrente.

- *ShowPub*. Es la aplicación que permite realizar preguntas sobre las publicaciones. Se compone de un JSP [Hal00] que genera dinámicamente el formulario de condiciones. Las posibles palabras clave y los tipos de publicaciones son obtenidos de la BD de publicaciones, los miembros (actuales y pasados) del grupo BDI-SID son obtenidos tras analizar en tiempo de ejecución las páginas web del grupo donde se enumeran los mismos. De esta forma, el formulario se adapta automáticamente a los cambios en la BD de publicaciones, así como a la lista de miembros de las páginas web del grupo. Como novedad también se permite preguntar por cualquier otra palabra clave o autor, así como restringir la búsqueda considerando solamente las publicaciones del grupo; en caso contrario, se considerarán todas las referencias de la BD de publicaciones. Tras rellenar el formulario, se invoca a otro JSP que realiza básicamente la misma labor que el CGI del sistema anterior pero consultando la BD de publicaciones. También permite obtener el registro BibTEX para cada publicación mostrada. (ver Figura 6.9).

- *ShowBibTex*. El objetivo es mostrar en formato BibTEX todas las referencias bibliográficas almacenadas en la BD de publicaciones. Esta información puede utilizarse para reemplazar cualquiera de los ficheros BibTEX del sistema, ya que contiene la información integrada de todos ellos.

En el caso de haber seguido una aproximación RPC se sobrecargaría la máquina que contiene los servicios web, que en el caso de ejemplo posee unas prestaciones menores que el resto. En cambio, mediante el uso de agentes móviles puede observarse que la carga del sistema se distribuye entre las distintas máquinas.

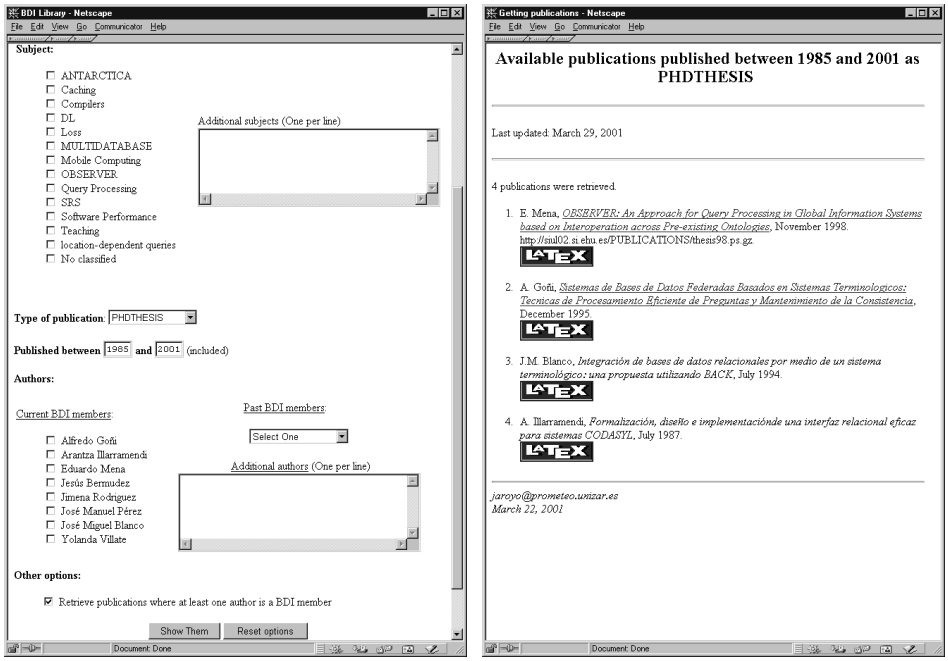


Figura 6.9: ShowPub: formulario y respuesta

Como puede observarse se ha logrado que el grado de independencia entre las distintas aplicaciones que integran el sistema sea máximo. La aplicación Bib2DB únicamente depende de las máquinas que contienen los datos que deben ser analizados para poder construir la biblioteca digital del grupo BDI-SID. En el caso de ShowPub y ShowBibTex, únicamente requieren de la máquina que posee el servidor web y el sistema gestor de bases de datos, es decir, en el escenario de ejemplo el sistema estará totalmente accesible para consulta en caso de que únicamente esté accesible siul02. La utilización de la BD, además de proporcionar robustez al sistema, permite un mayor poder de interrogación que la que poseen los distintos depósitos de datos por separado: ahora podemos usar SQL para consultar las publicaciones.

6.4. El agente móvil Bib2DB

En la Figura 6.10 mostramos el algoritmo que sigue el agente móvil que recopila información bibliográfica. Como ya hemos comentado, el agente es creado por el applet que se ejecuta tras rellenar el formulario donde se indican los ficheros BibTEX a analizar[4]. A partir de ese momento, el agente comenzará a viajar a los ordenadores correspondientes, accediendo y analizando los ficheros indicados.

[4]La labor de recopilación de referencias se dispara al solicitarlo el usuario pero con muy poco esfuerzo se podría incorporar a un sistema automático que lanzara el agente cuando alguno de los ficheros BibTEX ha sido actualizado.

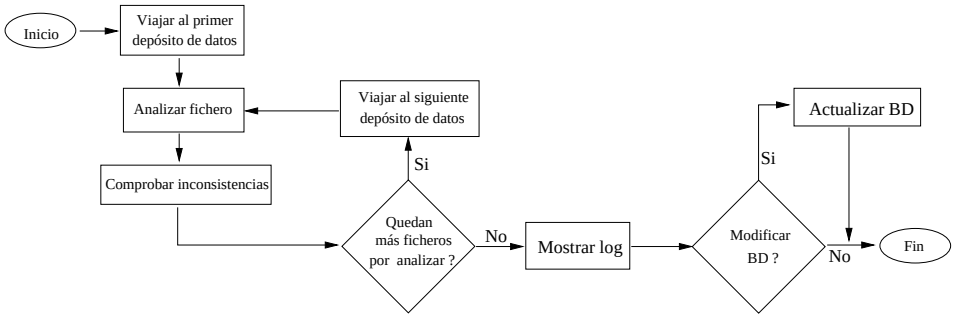


Figura 6.10: Algoritmo del agente móvil

El agente móvil implementado juega el papel de *wrapper* de ficheros BibTEX en un sistema de información distribuido. Además se le han añadido las funcionalidades de comunicarse con un applet para informar al usuario del estado de su agenda, y de la capacidad de crear una base de datos de publicaciones. Comentamos a continuación esas tres funcionalidades.

6.4.1. Acceso distribuido a ficheros BibTEX

El agente móvil es capaz de viajar a los ordenadores que contienen los ficheros indicados por el usuario y realizar la integración de la información accedida de una manera inteligente. A continuación describimos brevemente algunas de las características exhibidas en esta fase:

- *Gestión de proxies*: algunos ficheros pueden encontrarse en ordenadores que no son accesibles desde cualquier lugar. En esos casos se puede especificar el punto de acceso (o proxy) a dichos ficheros. Así, el agente primero viajará al proxy, luego al ordenador que contiene el fichero, lo leerá, volverá al proxy, realizará el análisis de consistencia[5], y continuará con el siguiente fichero.

- *Insensibilidad a desconexiones*: el agente continuará con su trabajo incluso si pierde la comunicación con el usuario temporalmente.

- *Insensibilidad a fallos en la red*: en el caso de no poder viajar a algún ordenador, el agente tiene asignada una cierta política de reintento, que podría variar según la situación.

- *Análisis de inconsistencias*: el agente es capaz de detectar las diferencias existentes entre dos publicaciones que tienen el mismo identificador o las similitudes entre dos referencias que podrían ser la misma. En tales casos se registrará un mensaje de error o de aviso, dependiendo del caso.

[5]La integración de la nueva información se realiza en el proxy para permitir la comunicación con el usuario en caso de encontrar alguna incidencia.

Hay que tener presente que el agente debe ser lo suficientemente inteligente para analizar textos LaTeXy poder modificar los caracteres especiales que utiliza dicho formato [Lam94]. En el caso de que el tipo de sistema al que se tuviese que acceder fuese distinto serviría el mismo tipo de aproximación mediante la utilización de agentes móviles, solamente habría que cambiar la parte de acceso a ficheros. Por ejemplo, con sustituir el wrapper propiamente dicho se podrían acceder ficheros MARC o bases de datos de publicaciones; el agente móvil invocaría al wrapper que fuese necesario en cada caso. Para una mayor información sobre la construcción de wrappers consultar [MI01, PGGMU95].

6.4.2. Comunicación con el usuario

Durante su ejecución el agente móvil informará al applet de cualquier incidencia encontrada (aunque continuará su tarea) así como de los objetivos parciales que vaya logrando. Para ello el applet ha sido diseñado como un servidor CORBA [OMG93] que acepta invocaciones remotas.

En el caso de no poder comunicar al applet del usuario los objetivos ya conseguidos o las incidencias encontradas, el agente continuará su ejecución hasta que llegue algún momento en que requiera el permiso del usuario para continuar. Entonces volverá a aplicar la política de reintento hasta que finalmente pueda comunicarse con el usuario. Es decir, el agente haciendo uso de su autonomía en la toma de decisiones puede continuar su ejecución salvo cuando necesita una autorización explícita del usuario.

Tras integrar los datos de todas las publicaciones encontradas, el agente móvil avisará al usuario (a través del applet), le permitirá ver el fichero de log que contendrá las posibles incidencias encontradas, y le solicitará si debe actualizar la BD o no. Para evitar problemas de seguridad, el agente solicita una clave para que la BD no pueda ser actualizada por personas no autorizadas.

6.4.3. Ventajas de los agentes móviles para el análisis de bibliografía

Las ventajas obtenidas del diseño de una arquitectura basada en agentes móviles en el contexto del análisis de depósitos bibliográficos puede extrapolarse al análisis y detección de inconsistencias en almacenes de datos distribuidos. Las principales ventajas debidas al diseño de una arquitectura basada en agentes móviles son:

1. La utilización de una aproximación cliente/servidor implica un servidor en cada uno de los distintos almacenes de datos: gracias a la utilización de una aproximación basada en agentes solamente debe existir un place que puede ser compartido con otros servicios. Es decir, para todos los servicios que hay disponibles en un servidor solamente implica la ejecución de un único place.

2. El filtrado y análisis de la consistencia de la información almacenada en los distintos depósitos de datos puede realizarse de forma local en cada uno de los servidores en los que se ubican los datos. Evitando su transmisión a través de la red.

3. El análisis de la información de los distintos depósitos de datos puede realizarse de forma concurrente. Es decir, el agente (en nuestro caso de ejemplo el agente Bib2DB)

puede clonarse a sí mismo tantas veces como sea necesario, incrementando el grado de paralelismo. Este aumento del paralelismo incrementará el volumen de información transmitido a través de la red, debido a que no se detectarán los duplicados existentes en dos depósitos de datos distintos hasta que los distintos agentes colaboren entre sí para unificar la información disponible.

En esta sección hemos mostrado las principales ventajas proporcionadas por los agentes en el filtrado y detección de inconsistencias en depósitos de datos distribuidos.

6.5. Inteligencia del agente Bib2DB: detección de inconsistencias

Los depósitos de referencias bibliográficas son utilizados de forma concurrente por un gran número de personas, tanto para consulta como para actualización. Del mismo modo, una tarea común en estos contextos es integrar varios depósitos de referencias bibliográficas.

En consecuencia, se precisa algún mecanismo que evite que se introduzca más de una referencia a la misma publicación. El identificador que se asigna a las referencias en el formato BibTEX es a todas luces insuficiente porque funciona solo a modo de puntero para no tener que introducir todos los datos de la referencia directamente en el fichero TEX. Lo que verdaderamente identifica a una referencia es el conjunto de datos asociados.

El mecanismo de detección de inconsistencias, utilizado por el agente Bib2DB, puede aplicarse de forma preventiva (para detectar que una referencia que se pretende insertar se refiere a la misma publicación que otra ya existente) o de forma correctiva (para examinar la lista de referencias actualmente almacenadas y detectar posibles inconsistencias y duplicados). La acción a realizar cuando se detecta que dos referencias son la misma es independiente del método de detección utilizado. En particular, se mostrará un aviso al usuario incluyendo las entradas BibTEX correspondientes y en el sistema quedará finalmente almacenada una sola referencia de acuerdo con los siguientes criterios:

- *Completitud*: la referencia resultante contendrá todos los campos incluidos en ambas referencias de entrada. Por ejemplo, si en una referencia se incluye información referente al número de páginas y en la otra no, es deseable guardar esta información en la referencia que finalmente se almacene.
- *Resolución de inconsistencias*: cuando las dos referencias tienen distintos valores asociados al mismo campo, se genera un mensaje de error y se solicita ayuda al usuario para que solucione el problema. Por ejemplo, si consideramos que dos referencias identifican a la misma publicación y tienen distinto número de autores (o aparecen en distinto orden), se le pedirá al usuario que verifique e introduzca la información correcta.

A continuación presentamos tres métodos que difieren en el criterio utilizado para determinar si dos referencias son iguales[6]. El primero es un método sencillo que utiliza sim-

[6] Antes de realizar la comparación de las referencias bibliográficas se realiza una normalización [BYRN99] de las mismas (eliminación de mayúsculas, espacios redundantes, caracteres especiales).

plemente un conjunto de reglas de comparación predefinidas. Los dos siguientes utilizan el algoritmo de Levenshtein para obtener las distancias de edición de las cadenas de caracteres correspondientes a los distintos campos BibTEX[7], y se diferencian únicamente en la manera en que dichas distancias se combinan para tomar una decisión final con respecto a la similitud de las referencias: utilizando redes neuronales o calculando una media ponderada adaptativa con pesos asignados a cada campo.

6.5.1. Comparador basado en reglas

En el tratamiento de inconsistencias se sigue un algoritmo incremental donde cada referencia nueva se compara con las anteriormente obtenidas e integradas. Este análisis se realiza tanto dentro de un mismo fichero BibTEX como entre los distintos ficheros analizados. Lo primero que se verifica es si se trata de la misma publicación o no (se considera que dos referencias son la misma si tienen el mismo identificador BibTEX). A continuación enumeramos las posibles incidencias que se pueden encontrar:

- *Con el mismo identificador*. En este caso se realizan las siguientes verificaciones de inconsistencias:
  - *Un campo aparece en una de las referencias bibliográficas y en la otra no*. La referencia almacenada en la base de datos será la que contiene a dicho campo. Se mostrará un mensaje de aviso al usuario.
  - *El mismo campo con valores distintos*. Se distinguen dos casos, aunque de cualquier forma nos quedaremos con la primera referencia encontrada:
    - *Mismo valor sin considerar mayúsculas y minúsculas*. Se genera un aviso.
    - *Distinto valor*. Este es el único caso en el que se genera un mensaje de error al ser una inconsistencia grave (aunque se continúa con el análisis).
- *Con distinto identificador*. Aunque dos publicaciones posean distinto identificador, podría tratarse de la misma publicación, ya que los distintos usuarios de los diversos depósitos de datos podrían haber asignado identificadores diferentes a la misma referencia. Por tanto se ha decidido que dos referencias bibliográficas podrían ser la misma si son del mismo tipo y poseen el mismo título y autores, a pesar de que el identificador sea distinto. En dicho caso se muestra un aviso.

Junto a cada uno de los mensajes de aviso o error aparecerán los registros BibTEX implicados en el mensaje, la URL correspondiente al fichero que actualmente se está analizado y las URLs de los ficheros analizados anteriormente.

Este método presenta notables deficiencias, como su incapacidad para detectar dos campos con distinto valor pero que contienen la misma información; por ejemplo, el contenido de uno de ellos podría incluir errores tipográficos, información incompleta, o emplear abreviaturas. Además, no se puede adaptar automáticamente a distintos contextos en los que se utilicen campos BibTEX no estándar, ya que no se utilizarían en la comparación. Las dos

[7] Incluyendo los campos personalizados, que influirán mínimamente debido a su escasa aparición.

aproximaciones que se presentan a continuación pretenden solventar estos inconvenientes. Este método no detecta ninguna inconsistencia en el depósito de datos utilizado para realizar las pruebas, aunque como veremos posteriormente si que existen varias inconsistencias.

6.5.2. Comparadores basados en el algoritmo de Levenshtein

En esta sección describimos dos aproximaciones para la detección de inconsistencias mediante el cálculo del grado de similitud de dos referencias bibliográficas: si el grado de similitud calculado supera un cierto umbral (α) se considera que ambas referencias son la misma.

Estas aproximaciones se basan en el algoritmo de Levenshtein [Lev66], ampliamente utilizado para comparación de secuencias [Kru83, RY98], que calcula la distancia de edición entre dos cadenas de entrada (número mínimo de pasos para pasar de una a otra). El algoritmo de Levenshtein se ha venido utilizando en contextos tales como biología computacional (comparación de cadenas de ADN) y para la implementación de correctores ortográficos. En nuestro contexto, la utilizamos entre cada par de campos BibTEX con el mismo nombre de las dos referencias de entrada, obteniendo así una medida de la similitud de los campos. A partir de los grados de similitud de los campos se obtiene un grado de similitud global de acuerdo con una de las dos aproximaciones que se muestran a continuación: utilizando redes neuronales o calculando una media ponderada adaptativa entre los campos.

Cálculo del grado de similitud con redes neuronales

En esta sección proponemos la utilización de redes neuronales [HDB02] para calcular el grado de similitud de dos referencias bibliográficas a partir de las distancias de Levenshtein de sus campos. Consideramos una red neuronal de propagación hacia atrás con una capa de entrada con 29 neuronas[8] que reciben como entrada los grados de similitud de los campos de las dos referencias que se comparan, una capa intermedia con 14 neuronas [9], y una capa de salida con una neurona. Se utiliza una función de transferencia lineal entre neuronas, y la neurona de salida devuelve un número real en el rango $(-\infty, +\infty)$ indicando el grado de similitud global. En nuestra implementación el umbral que determina si dos referencias son o no la misma es igual a 0: si la salida es menor o igual que 0 se consideran distintas, y la misma en caso contrario.

La red neuronal se ha entrenado utilizando aprendizaje supervisado, utilizando para ello un conjunto muestra de 170.000 referencias bibliográficas, que se han obtenido a partir del depósito de datos utilizado (contiene 1.000 referencias aproximadamente). La muestra utilizada para entrenar la red neuronal se ha creado introduciendo variaciones (insertar/eliminar campos, alterar el valor de los campos, etc) de forma aleatoria en las referencias bibliográficas. Para evitar un entrenamiento positivo/negativo de la red se realiza aproximadamente el mismo número de pares que devuelvan un resultado de igualdad que de desigualdad.

Esta aproximación basada en redes neuronales mejora considerablemente a la anterior, dado que es mucho más flexible y proporciona una mayor tasa de aciertos que la que se obtiene

[8] Igual al número de campos considerado de una publicación mediante el comparador basado en redes neuronales.

[9] El número de capas intermedias y de neuronas en dicha capa se ha determinado experimentalmente tras realizar diversas pruebas.

comparando simplemente cadenas de caracteres. La red neuronal descrita se ha implementado utilizando Matlab [GB03] y en las pruebas realizadas se ha obtenido un porcentaje de error del 0.27 %.

Como contrapartida, hay que invertir un esfuerzo inicial en entrenar la red, y un aumento en el número de campos a considerar supone modificar la estructura de la red. Además, el método depende de cómo de representativo sea el conjunto de entrenamiento, lo cual no es en absoluto fácil de predecir.

Cálculo del grado de similitud con media ponderada dinámica

En esta sección comentamos un comparador de publicaciones basado en la distancia de Levenshtein que permite adaptarse automáticamente al depósito de datos y a la información almacenada acerca de las publicaciones.

Para ver si dos referencias bibliográficas, r_1 y r_2, son iguales se calcula una media ponderada de las distancias de Levenshtein entre cada uno de sus campos t. El valor del peso de un campo, $w(t)$, se calcula en cada momento como el cociente entre el número de valores distintos almacenados hasta entonces en el depósito de datos, $\#(t)$, y el número total de referencias bibliográficas, $\#referencias$. Por tanto, la distancia entre dos referencias bibliográficas es:

$$d(r_1, r_2) = \frac{\sum_t \frac{DL(r_1^t, r_2^t)}{w(t)}}{\sum_t w(t)}, t \in \{C(r_1) \cup C(r_2)\}$$

$$w(t) = \frac{\#(t)}{\#referencias}$$

siendo $C(r)$ los campos que tiene una referencia bibliográfica, $DL(r_1^t, r_2^t)$ es la distancia de Levenshtein entre r_1 y r_2 para el campo t.

Este método considera que dos publicaciones son iguales si $d(r_1, r_2)$ es menor que un cierto umbral (α). Experimentalmente hemos determinado que un valor adecuado para α en nuestro contexto es 0.3 (ver Tabla 6.1).

α	Porcentaje de Fallos
0.7	6.24 %
0.5	0.89 %
0.3	0.09 %

Tabla 6.1: Errores detectados según el umbral

La ventaja de utilizar este método es que los pesos de cada uno de los campos se actualizan dinámicamente adaptándose a los datos almacenados. Es decir, la importancia asignada a cada uno de los campos de una referencia bibliográfica varía dependiendo del número de veces que se repite cada uno de los valores de ese campo en el depósito de datos. Por ejemplo, si solamente se almacenan referencias de un único autor entonces ese campo será poco relevante para determinar si dos publicaciones son iguales.

6.6. Extendiendo la arquitectura: digitalización de las publicaciones

En la sección 6.3 se presentó un mecanismo, basado en la tecnología de agentes móviles [MBB+98], para la creación automática de depósitos de publicaciones de investigación a partir de un conjunto de depósitos de referencias distribuidos. Concretamente se proponía la utilización de un agente móvil Bib2DB para la recopilación de información bibliográfica a partir de distintos ficheros BibTEX, visitando uno tras otro los ordenadores que contienen dichos ficheros e integrando los datos incrementalmente, para finalmente almacenarlos en una base de datos relacional. Esta aproximación presentaba las siguientes ventajas:

- *Recopilación de información distribuida*, mediante la utilización de un agente móvil. Este agente viajará al nodo que contiene el primer depósito de datos y, después de que la información haya sido analizada, el agente viajará al siguiente nodo, integrará los nuevos datos con los anteriores, y así sucesivamente hasta haber completado el análisis de todos los nodos.
- *Generación de información web a partir de la base de datos*, permitiendo una gestión automatizada del sitio web de publicaciones del grupo.
- *Detección automática de inconsistencias* entre la información disponible en los depósitos de datos distribuidos. La misma referencia[10] puede haber sido introducida por distintos usuarios con deficiencias en su información.
- *Robustez frente a desconexiones*, el agente móvil es capaz de reaccionar ante los fallos de red que se produzcan, según cierta política de reintentos. Así completará su tarea sin que sea necesaria la intervención del usuario.
- *Integración de la información bibliográfica* en una base de datos con la idea de hacer accesible a otras aplicaciones cualquier publicación disponible en los depósitos BibTEX.

A pesar de estas mejoras, con el tiempo se han detectado otros problemas: 1) *Problemas de actualización*, ya que deben actualizarse todos los ficheros BibTEX que contengan dicha referencia; 2) *Problemas de inserción*, las nuevas publicaciones deben introducirse en alguno de los ficheros BibTEX y debe existir algún mecanismo automático que lance el agente móvil para añadirlas al depósito centralizado[11]; y 3) *Baja interoperabilidad*, el sistema únicamente puede ser accedido mediante las aplicaciones creadas, por lo que su funcionalidad no puede reutilizarse fácilmente desde nuevas aplicaciones.

Por lo tanto hemos mejorado nuestra primera aproximación mediante reestructuración del sistema web para que pueda ser accedido tanto por agentes, mediante un acceso basado en servicios web [JC02], como por personas, mediante un acceso basado en páginas web. La arquitectura propuesta (ver Figura 6.11) presenta los siguientes cuatro niveles:

[10] En nuestro sistema de información entendemos por publicaciones tanto referencias como su documento digital asociado.

[11] Podrían utilizarse mecanismos, como por ejemplo proporcionados por los *EJBs* [RAJF01], pero éstos introducirían una mayor carga en el sistema.

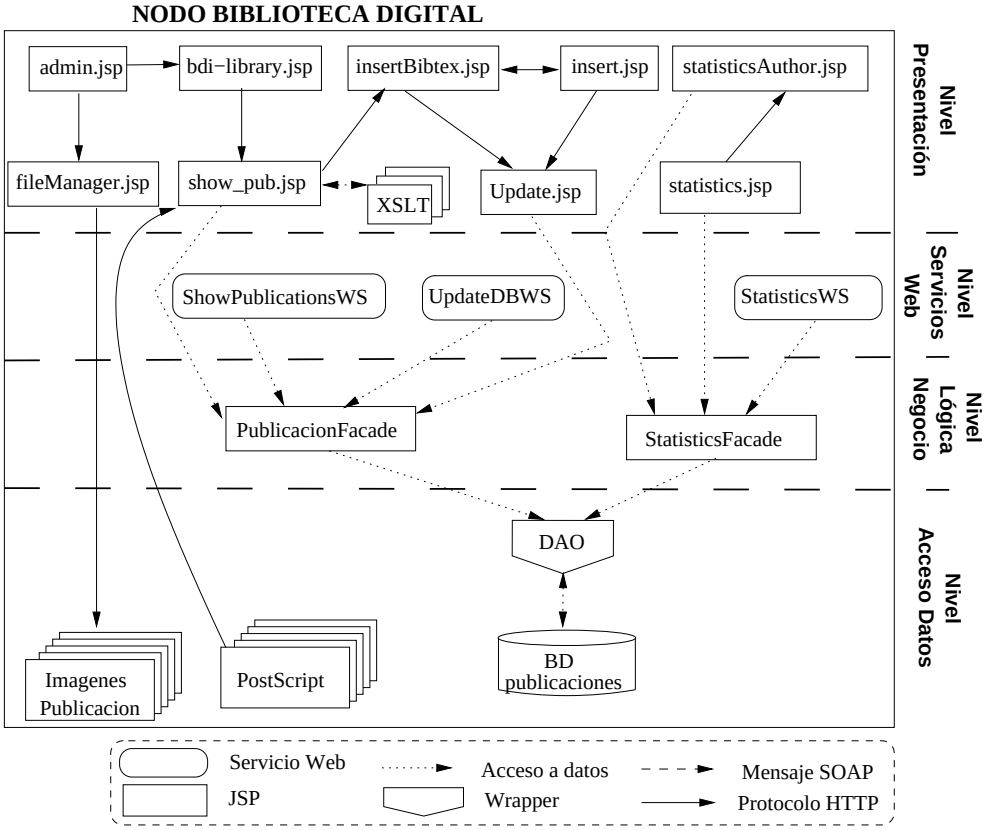


Figura 6.11: Arquitectura del sistema extendido

1. *Presentación*: desarrollada mediante páginas web dinámicas utilizando JSP [Hal00], lo que permite la separación de los datos de su visualización. En esta capa se encuentran servicios de inserción, edición y consulta de datos (secciones 6.6.1 y 6.6.2, respectivamente).

2. *Lógica de servicios web*: desarrollada mediante servicios web para permitir la interacción con otras aplicaciones o agentes, ya que la entrada y salida de esta capa son datos en XML siguiendo el protocolo SOAP (basado en HTTP). Esta capa permite que el sistema sea accedido tanto por seres humanos como por agentes inteligentes.

3. *Lógica de negocio*: Basada en la utilización de patrones de diseño, para facilitar la modificación y futuras ampliaciones de la misma.

4. *Acceso a datos*: es la encargada del acceso a los distintos depósitos de datos. Esta capa esta formada por un conjunto de *wrappers* [HBGM+97] que permiten el acceso a la base de datos de publicaciones y a los ficheros Bib. El wrapper que permite el acceso a

los ficheros BibTEX se encuentra integrado dentro del agente móvil que se presentó en la sección 6.6, la inteligencia de dicho agente se ha reutilizado en los mecanismos de comparación de publicaciones utilizados en la nueva arquitectura. El agente Bib2DB sigue la misma aproximación que en la arquitectura presentada inicialmente.

La arquitectura actual del sistema (ver Figura 6.11) mantiene la compatibilidad con la preexistente y además añade nuevas funcionalidades (destacamos el nombre de los módulos que las implementan):

- *Inserción de nuevas referencias bibliográficas y actualización de las existentes*: La aproximación presentada en la sección 6.3 permite el almacenamiento masivo de referencias bibliográficas en la biblioteca digital, realizado por el agente Bib2DB. Sin embargo carece de un método que permita la inserción de nuevas referencias bibliográficas individualmente y de un mecanismo de actualización de las ya existentes. Del mismo modo mediante el acceso basado en servicios web se permite que la biblioteca digital sea accedida tanto por personas como por agentes inteligentes, para inserción, actualización y consulta de la información relativa a las distintas publicaciones de investigación.

- *Almacenamiento digital de información*: se permite almacenar otra información, tal como el fichero postcript, o imágenes asociadas a la referencia bibliográfica de la publicación (*fileManager.jsp*) que contiene el trabajo investigador (por ejemplo, la portada, contraportada, índice, etc.), como se muestra en la Figura 6.12. Mediante la consulta de esta información un agente especializado en la generación de curriculums podría crear y actualizar los curriculums de los distintos miembros del grupo de investigación.

- *Definición de campos personalizados*: se permite definir campos no estándar para las referencias bibliográficas almacenadas en la base de datos. Dado que se trata de campos opcionales no se almacenan en la misma tabla que los campos estándar; se almacenan en una tabla aparte que tiene tres campos: uno para el identificador de la publicación, otro para el nombre del campo de la referencia bibliográfica y otro para el valor de dicho campo. De este modo evitamos el crecimiento desmesurado del número de columnas. Permitiendo de esta forma que un agente especializado en la recuperación de ratios de impacto (JCR, CORE, etc..) pueda analizar de forma automática la Web e insertar dicha información acerca de las publicaciones disponibles en el sistema.

- *Adaptación de la presentación a distintos formatos*: se utilizan plantillas XSLT para mostrar la salida deseada (*show_pub.jsp y ShowPublicationsWS*). Pudiendo mostrar salidas personalizadas para ser analizadas por agentes inteligentes, por ejemplo salidas basadas en OWL [AvH03].

- *Generación de estadísticas de los datos almacenados*: (*statistics.jsp, statisticsAuthor.jsp y StatisticsWS*) se permite consultar el número y tipo de publicación por autor o por grupo de autores y por año. Puede verse un ejemplo en la Figura 6.13. Estas estadísticas podría ser estudiadas por un agente encargado de la caza de talentos, gracias a que son accesibles mediante un servicio web que proporciona un interfaz basado en XML, concretamente en un subconjunto basado en OWL.

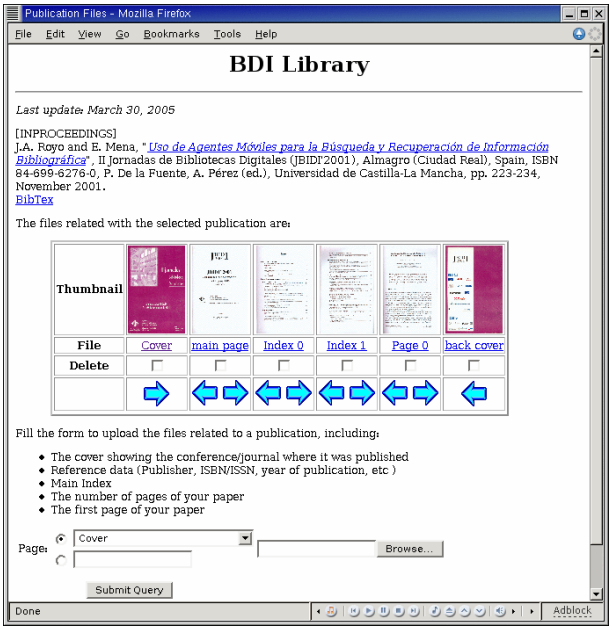


Figura 6.12: Digitalización de publicaciones

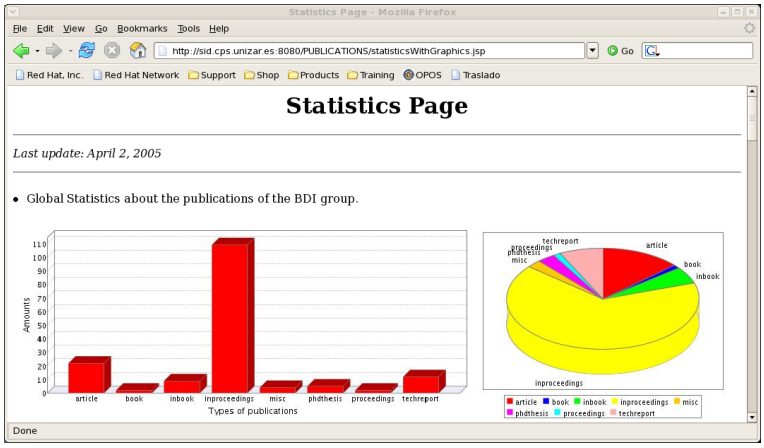


Figura 6.13: Estadísticas

El sistema presentado permite la paginación de los resultados de salida mediante la utilización de los patrones *DAO* e *Iterator* [ACM01], facilitando de esta forma no sólo la interacción con los agentes software sino también con los seres humanos.

6.6.1. Inserción de publicaciones

En esta sección se presenta la forma en la que se introducen las nuevas publicaciones en el sistema. Existen dos mecanismos diferenciados:

- *Inserción masiva* mediante el agente móvil desarrollado en [RM01a], para cuando quieren introducirse un gran número de publicaciones, posiblemente almacenadas en varios ficheros de referencias.

- *Inserción manual* que permite la inserción de publicaciones mediante la utilización de dos formularios web: 1) *Modo registro*[12], que permite la inserción de publicaciones a partir de su registro BibTEX (ver Figura 6.14); y 2) *Modo campo a campo*, para introducir cada uno de los datos asociados a la publicación de forma separada (ver Figura 6.15). Ambos métodos pueden ser utilizados indistintamente debido a un mecanismo automático de traducción que permite pasar de un formulario/modo de inserción a otro.

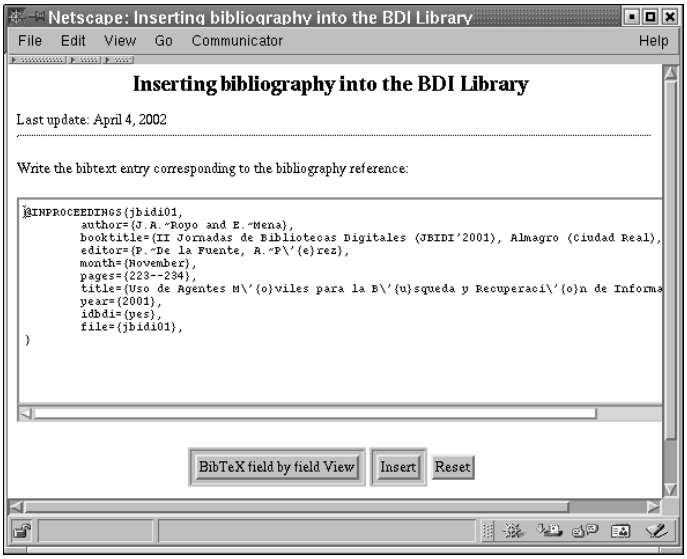


Figura 6.14: Inserción modo registro

Debido a que los distintos servicios de inserción de datos están basados en un conjunto de servicios web, éstos también pueden ser accedidos por agentes inteligentes. Permitiendo de esta forma que la inserción manual pueda ser realizada tanto por seres humanos como por agentes inteligentes.

Durante el proceso de inserción de una nueva publicación, por cualquiera de los métodos descritos anteriormente, se utiliza el sistema de verificación de inconsistencias utilizado por

[12] El modo registro está asociado al JSP “insertBibtex” y el modo campo a campo al JSP “insert”.

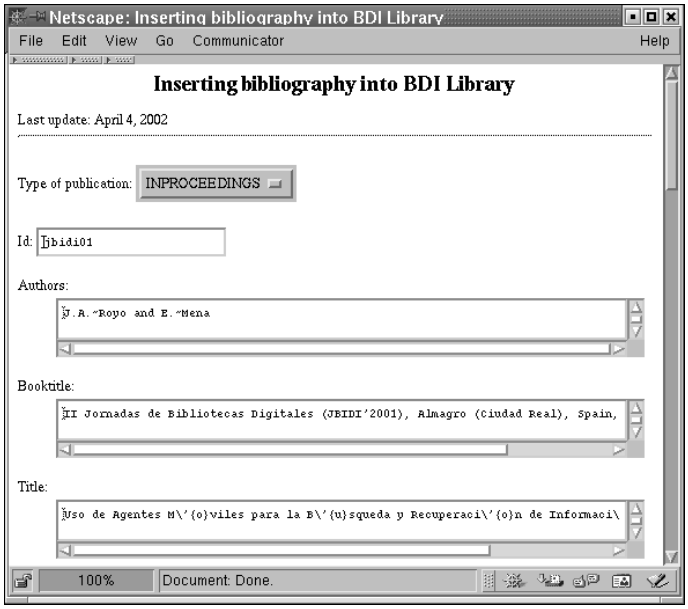


Figura 6.15: Inserción modo campo a campo

el agente Bib2DB, éste se detalla en la sección 6.5.2, que se corresponde con la inteligencia dicho agente. Este es un ejemplo de como reutilizar la inteligencia de los agentes, gracias a que éstos permiten colaborar con otros agentes o módulos software.

6.6.2. Consulta y actualización de publicaciones

En la biblioteca digital desarrollada se permite la realización de búsquedas a través de un formulario construido dinámicamente. En dicho formulario se permite la búsqueda mediante la selección de los temas que tratan, tipo de publicación[13], o autores de las publicaciones (ver Figura 6.16). También es posible imponer restricciones de tipo subcadena a cualquiera de los campos de una publicación. En la Figura 6.17 puede observarse el resultado de la consulta realizada.

Respecto a la actualización de publicaciones, el proceso sería el siguiente: 1) Búsqueda de las publicaciones a modificar (usando el mismo formulario de la Figura 6.16); 2) Selección de la publicación a modificar (en una pantalla similar a la de la Figura 6.17 aparecerá un enlace "Edit" para cada referencia); y 3) Actualización de la publicación, tras pulsar "Edit", mediante formularios similares a los de inserción (Figuras 6.14 y 6.15) que mostrarán los datos actuales de la publicación seleccionada, permitiendo cambiarlos.

Debido a que en esta versión la arquitectura del sistema web se basa en un conjunto de servicios web, éstos pueden ser utilizados por agentes inteligentes para realizar los procesos

[13]Por tipo de publicación nos referimos a si estamos ante un libro, un artículo en revista, etc.

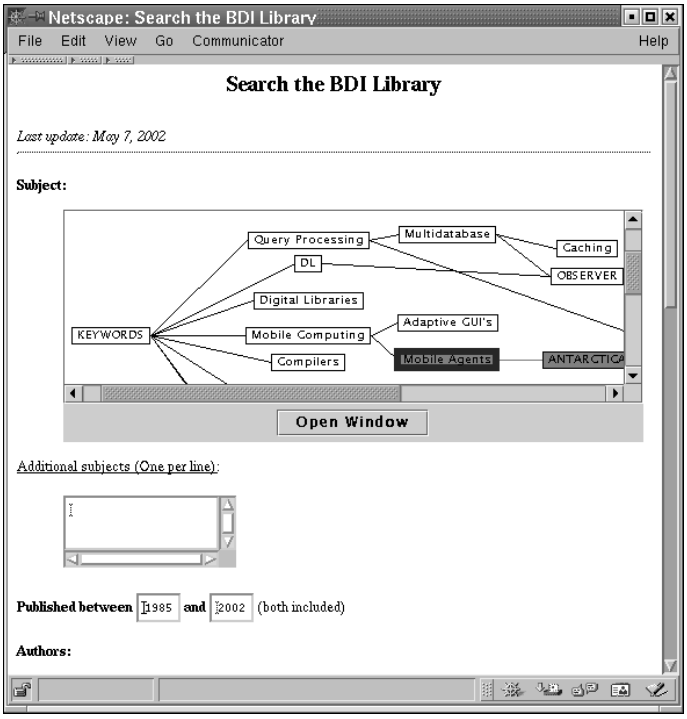


Figura 6.16: Formulario de consulta

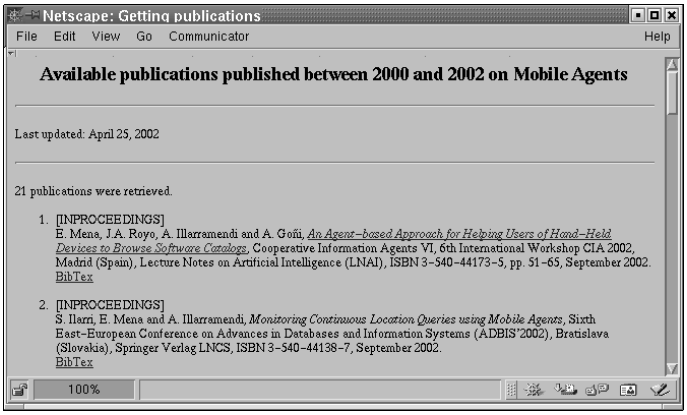


Figura 6.17: Resultado de la búsqueda

de consulta y actualización de publicaciones. Estos procesos de búsqueda realizados por los agentes se ven favorecidos por la utilización de dos ontologías que permiten realizar una clasificación semántica de las publicaciones por temas y por tipos de publicación.

Utilización de ontologías

En [MI01] se describe un problema asociado a la relación entre los posibles valores de las entidades y atributos representadas en un sistema de información. Entre los distintos valores puede haber relaciones de sinonimia, hiponimia e hiperonimia; es decir, puede existir una relación semántica entre los posibles valores de los objetos representados. La representación de este tipo de conocimiento puede realizarse mediante la utilización de ontologías. Además, esta representación es inteligible para agentes inteligentes que utilicen un sistema terminológico [Bor92] para razonar.

Por ejemplo, en el caso de las palabras clave, al buscar las publicaciones referentes a "bases de datos distribuidas" no se incluirán en el resultado aquellas publicaciones clasificadas solamente como de "bases de datos federadas", a pesar de que también tratan sobre cierto tipo de bases de datos distribuidas. Por tanto, los usuarios/agentes deberían especificar *todas* las palabras clave de una publicación al introducir sus datos, recordando la relación semántica entre ellas; si olvidan alguna, los resultados de las búsquedas no serán semánticamente correctos.

Por tanto, para algunos de los campos de una publicación es muy interesante no ofrecer los posibles valores como una lista plana sino como una ontología [Gru92]. La utilización de ontologías además de permitir la representación de conocimiento, permiten almacenarlo de forma organizada. Por todo esto hay que destacar que la utilización de este mecanismo de representación no sirve únicamente para facilitar las operaciones de búsqueda, sino también para organizar los datos que se almacenan en la biblioteca digital. Permitiendo de esta forma que éstos puedan ser consultados y analizados por agentes inteligentes y por seres humanos. En la Figura 6.18 se muestran las ontologías de palabras clave y de tipos de publicaciones manejadas por el grupo de investigación BDI-SID, pero podría utilizarse un tesaurus ya preexistente como WordNet [Mil95].

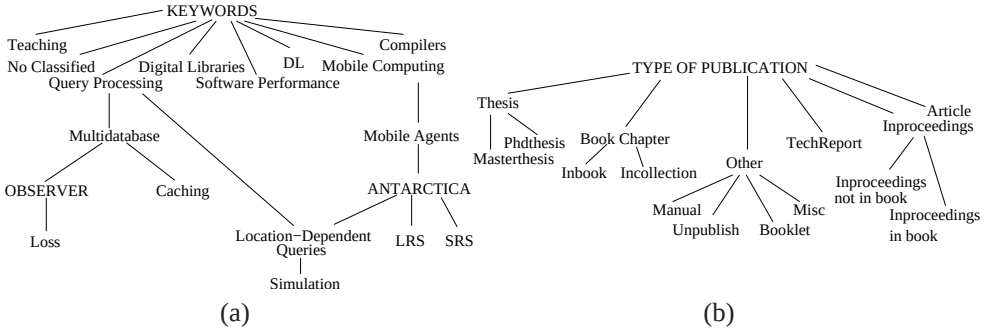


Figura 6.18: Ontologías de palabras clave (a) y de tipos de publicaciones (b)

En nuestro prototipo del sistema, los formularios de inserción, edición y búsqueda incluyen mediante *applets* una ontología de palabras clave (ver Figura 6.16). El sistema incluye la conexión a un sistema terminológico [Bor92] que convertirá automáticamente el conjunto de palabras clave (o tipos de publicación) indicadas en un conjunto mutuamente independiente, lo cual optimizará tanto su almacenamiento como las búsquedas. Por ejemplo, si se introduce

una publicación con “Caching” como única palabra clave, automáticamente será clasificada como una publicación que trata los temas de “Caching”, “Multidatabase” y “Query Processing” (ver Figura 6.18). Estos beneficios se obtienen tanto si el sistema es utilizado por una persona como si es accedido por un conjunto de agentes inteligentes.

6.6.3. Generación automática de bibliografía

Como ya dijimos en la introducción, una biblioteca digital de publicaciones de investigación no debe tener la finalidad única de servir para la búsqueda de documentos sino que también debe ser posible su conexión con los procesadores de texto que se utilicen. En nuestro caso nos centraremos en el editor LaTeX [Lam94], probablemente el más utilizado para la elaboración de artículos de investigación.

La elección de LaTeX se basa en que incluye “bibtex”, una aplicación para generar automáticamente la bibliografía asociada a un artículo tomando como base un fichero de texto en formato BibTeX [Lam94]. Por otra parte, LaTeX no siempre incluye un entorno de trabajo al que añadirle nuevas funcionalidades, como podríamos hacer en un procesador tipo Microsoft Word, sino que muchos de sus usuarios invocan las distintas aplicaciones desde el *prompt* del sistema.

Por tanto hemos desarrollado un agente que permite seleccionar un fichero LaTeX y, tras analizarlo, generar automáticamente el correspondiente fichero de bibliografía en formato BibTeX. Así se hace necesario un cambio de filosofía de trabajo: en vez de preocuparnos de mantener actualizados varios ficheros BibTeX, las referencias se almacenan siempre en un depósito centralizado que mediante un agente software ofrece el servicio de generar cuando queramos el fichero de bibliografía en formato BibTeX correspondiente a un documento LaTeX. De la misma manera podría generarse bibliografía en otros formatos sin ninguna dificultad. Como consecuencia de la instalación de la aplicación en una página web se permite que cualquier persona que lo desee pueda utilizar dicho servicio para generar sus ficheros de bibliografía, mediante la utilización de hojas de estilo para transformar los datos XML al formato deseado.

El análisis de documentos LaTeX se realiza mediante un agente basado en una gramática (ver Figura 6.19) que permite el análisis de los comandos LaTeX identificados por las palabras clave *cite* y *nocite*. El comando *cite* se utiliza para citar una publicación y que su referencia aparezca en la bibliografía del documento; *nocite* se usa para que una referencia aparezca en la bibliografía aunque no sea citada desde el texto. El fichero de bibliografía se actualiza cada vez que se ejecuta esta aplicación.

La inteligencia utilizada por el agente en el análisis de ficheros LaTeX se basa en la detección de los siguientes tipos de errores:

- *Error tipográfico*: cuando el identificador de una cita se encuentra en la base de datos pero no coinciden las mayúsculas y minúsculas, se mostrará un mensaje de aviso (tal y como hace la aplicación “bibtex”).

- *Identificador erróneo*: cuando en el texto LaTeX aparece una cita a un documento que no existe en la base de datos.

```
<documento> ::= <lista_citas> <bibliografia>
              |

<lista_citas> ::= <cita> <lista_citas>
                | <cita>

<cita> ::= "\cite" "{" <lista_referencias> "}"
         | "\nocite" "{" <lista_referencias> "}"

<lista_referencias> ::= cadena "," <lista_referencias>
                      | cadena
                      | *

<bibliografia> ::= "\bibliography" "{" cadena "}"
```

Figura 6.19: Gramática para Ficheros LaTeX

- *Error de seguridad*: cuando no se han dado los permisos Java necesarios para la ejecución del servicio web.
- *Error de conexión a la base de datos*: cuando la base de datos no está accesible por problemas de red.

Este agente podría cooperar con otros agentes permitiendo realizar la traducción de las referencias bibliográficas de formato bibtex a otros formatos, permitiendo de esta forma su utilización para la generación de bibliografía en otros contextos.

6.7. Resumen del capítulo

En este capítulo se ha presentado una arquitectura basada en un agente inteligente que facilita la integración de distintos depósitos bibliográficos. Del mismo modo, la aproximación presentada permite la reutilización de la inteligencia del agente (comparador de referencias) para detectar posibles inconsistencias en contextos de bibliotecas digitales de investigación donde los datos se encuentren almacenados en otro formato, como por ejemplo ficheros MARC [Pie94].

La arquitectura propuesta presenta características deseables para las bibliotecas digitales de publicaciones científicas como: 1) *Alta interoperabilidad* con otros sistemas al haber sido desarrollado como una serie de servicios web; 2) *Unificación del proceso de inserción y búsqueda de publicaciones*, basado en un depósito de datos centralizado; 3) *Gestión de valores jerarquizados*, mediante el uso de ontologías y un sistema terminológico, que facilita la inserción y búsqueda de datos; 4) *Generación automática de bibliografía* en formato BibTeX y XML; y 5) *Minimización del software instalado*, disminuyendo el uso de recursos de los usuarios y evitando el problema de la actualización de versiones, gracias a la utilización de una arquitectura basada en agentes móviles inteligentes y en servicios web.

Como hemos indicado anteriormente el eje central de la arquitectura propuesta es un agente móvil, el agente Bib2DB, que se encarga de recopilar información bibliográfica distribuida por la red, integrándola adecuadamente y generando mensajes sobre posibles in-

consistencias, para finalmente almacenar la información ya procesada en una base de datos relacional que podrá ser consultada por otras aplicaciones.

Cabe destacar que el sistema descrito no debe considerarse como la resolución de un caso particular en el que se analizan un conjunto de ficheros BibTEX, sino que se trata de la aplicación de la tecnología de agentes móviles a un sistema de búsqueda y recuperación de información. Por lo tanto, la inteligencia del agente móvil desarrollado puede ser reutilizada incluso si los depósitos de datos a analizar fuesen una base de datos o una combinación de diversas fuentes de datos con organizaciones distintas. La adhesión de esta funcionalidad solamente implicaría que el agente Bib2DB tuviese disponibles los wrappers necesarios para cada depósito de datos a analizar.

Destacar igualmente que, gracias a la utilización de una arquitectura basada en agentes, el software necesario tanto en el ordenador cliente como en los servidores de información se ha minimizado; con la ventaja adicional de que cualquier actualización en las funcionalidades del sistema no afectará ni al cliente ni a los servidores, sino únicamente al agente. Igualmente, su adecuación a entornos inalámbricos es manifiesta, dada la optimización del tiempo de conexión necesario así como su robustez frente a desconexiones, lo cual permite su utilización desde dispositivos inalámbricos.

Por otra parte, destacar que gracias a la utilización de una aproximación basada en agentes móviles se puede filtrar la información de forma local en cada uno de los servidores que albergan los distintos depósitos de datos, evitando su transmisión a través de la red. Finalmente, resaltar que gracias a la utilización de un sistema de agentes móviles se permite la posibilidad de que el agente Bib2DB pueda clonarse a sí mismo para realizar el análisis de más de un depósito de datos de forma concurrente.

Capítulo 7

Otras aplicaciones de la tecnología de agentes: GUIs adaptativos, servicios basados en la localización y Web Semántica

En este capítulo mostraremos las ventajas proporcionadas por los sistemas multiagente para el diseño de servicios de datos en contextos de ejecución en los que los recursos disponibles se ven modificados dependiendo de las capacidades y ubicación del dispositivo del usuario. Haremos un especial hincapié en la capacidad inherente de las arquitecturas basadas en agentes para adaptarse de forma automática y autónoma a los recursos disponibles en cada momento. Por ejemplo, los agentes móviles permiten trasladar los distintos procesos de cálculo necesarios en un servicio inalámbrico de acceso a datos al lugar más adecuado en cada momento. Otra de las propiedades de las arquitecturas basadas en agentes es que permiten su despliegue de forma dinámica, minimizando el consumo de recursos e incrementando la robustez de las aproximaciones tradicionales frente a desconexiones. Además, demostraremos como las arquitecturas basadas en agentes facilitan el diseño de servicios de datos en contextos de ejecución donde los recursos disponibles cambian de un momento a otro, permitiendo: 1) su adaptación automática al dispositivo del usuario, 2) su personalización dependiendo de la ubicación del usuario, y 3) la minimización de los recursos (disco, capacidad de procesamiento, trasmisión de datos mediante el enlace de comunicaciones inalámbrico) consumidos.

La primera de las arquitecturas diseñadas, en el contexto de los servicios en entornos dinámicos en esta tesis, permite la generación de interfaces adaptativos, que además ayudan a monitorizar el comportamiento del usuario y reutilizar esta información posteriormente. Para afrontar el diseño de esta aproximación será necesario especificar los interfaces gráficos de usuario —*Graphical User Interface* (GUI)—, en nuestro caso esto se realizará utilizando *XUL* [Tut06], una especificación del GUI basada en XML, aproximación utilizada

en [MM02]. Aunque se podría haber utilizado cualquier otro lenguaje de descripción de GUIs. Debemos enfatizar que gracias a los agentes y la especificación del interfaz gráfico de usuario, éstos son capaces de modificar su comportamiento para generar un GUI adaptado a las características del dispositivo, a los recursos disponibles (por ejemplo, dependiendo de estos se podrá generar un GUI WML [WWSV06] o uno que permita una mayor interactividad, como uno basado en Java Swing [Mic06a]) y a las preferencias del usuario. Además, los agentes en los que se basa nuestra aproximación podrán analizar dicha información y modificar el interfaz gráfico ayudando al usuario indicándole la próxima acción, gracias a que pueden monitorizar su comportamiento. Es decir, el GUI se modificará en tiempo de ejecución para sugerir al usuario cuáles son las posibles acciones más probables que puede realizar, dependiendo de las acciones que el usuario realizó en el pasado. Además, se puede modificar la forma en la que se generará el GUI; por ejemplo, si el dispositivo del usuario no permite la instalación de una plataforma de agentes móviles se realizará la generación del GUI en la estación base que le proporciona cobertura al usuario y el agente actuará como proxy durante la ejecución del servicio.

En segundo lugar, detallaremos una arquitectura que facilita el diseño de servicios basados en la localización. Esta arquitectura se fundamenta en un conjunto de agentes que son creados dinámicamente en las estaciones que dan cobertura al usuario y en el dispositivo móvil del mismo. Gracias a la utilización de una arquitectura basada en agentes móviles, se minimiza el software que debe instalar un usuario cuando quiere utilizar un servicio basado en la localización, porque únicamente debe tener en ejecución un *place* al que puedan viajar los agentes móviles que constituyen la arquitectura. Por otro lado y como se mostrará posteriormente, la aproximación propuesta es independiente del mecanismo utilizado para calcular la posición del usuario. Del mismo modo, la arquitectura propuesta en esta tesis se puede adaptar fácilmente a otros protocolos de comunicaciones inalámbricas que proporcionen la potencia con la que son emitidos los mensajes enviados por las estaciones base que dan cobertura al dispositivo móvil del usuario. En nuestro caso nos centramos en el protocolo 802.11 [GG02], más conocido como WiFi. Como mostraremos a continuación, gracias a la utilización de una arquitectura basada en agentes móviles la infraestructura que permite la localización de dispositivos no tiene que estar activa en todo momento, sino que puede crearse en tiempo de ejecución cuando se esté utilizando un servicio basado en la localización.

En tercer y último lugar, se mostrarán las ventajas de los agentes móviles en el diseño de un sistema de información global [MI01] con múltiples ontologías [Gru92], que permite su despliegue en tiempo de ejecución. Gracias a estar basado en una arquitectura de agentes móviles, la aproximación que presentaremos permitirá trasladar la ejecución de los distintos procesos de cálculo al lugar más adecuado, evitando la sobrecarga de los distintos ordenadores en los que se esté ejecutando el servicio. En la arquitectura propuesta varios agentes estudiarán el significado de un conjunto de palabras clave que son introducidas por el usuario para formular una pregunta expresada en Lógica Descriptiva [BCM$^+$03]. Debido a que este proceso es semiautomático, un agente guiará al usuario durante este proceso de traducción. Posteriormente, la pregunta formulada en un lenguaje estructurado será utilizada por otro agente para interrogar distintas bases de conocimiento y devolver al usuario los datos con los que estas ontologías se encuentran enlazadas. Debemos destacar que este agente podría clonarse a si mismo facilitando un análisis en paralelo. Además, gracias a la utilización de

una arquitectura basada en agentes móviles, los distintos procesos de cálculo utilizados para dar respuesta a la pregunta formulada por el usuario pueden ser distribuidos en una red de ordenadores minimizando la sobrecarga del sistema y evitando cuellos de botella, problema principal de un sistema centralizado. Del mismo modo, debido a que las distintas bases de conocimiento y los depósitos de datos subyacentes de éstas puede variar en el tiempo, un diseño basado en agentes móviles permite el despliegue de la arquitectura de forma dinámica dependiendo de los recursos disponibles en cada momento, permitiendo que la arquitectura del sistema se adapte automáticamente tanto a los depósitos de datos como a la capacidad de procesamiento (debida al número de places a los que pueden viajar los agentes) disponible en cada momento. Nuestra aproximación permitirá mostrar las ventajas de la utilización de una arquitectura basada en agentes en un contexto más abierto y heterogéneo que el de la búsqueda de software mostrado anteriormente.

7.1. Generación indirecta de interfaces gráficas de usuario

La computación ambiental (*pervasive computing*) ha proporcionado nuevos retos a los investigadores. Como la necesidad de construir aplicaciones que se puedan ejecutar en cualquier lugar, es decir, aplicaciones diseñadas para adaptarse incluso a los dispositivos más restrictivos y aun así mantener la funcionalidad deseada. Sin embargo, los diferentes tipos de dispositivos tienen distintas potencias de procesamiento, arquitectura y características técnicas. Además, los dispositivos móviles tienen un interfaz de usuario muy restrictivo y es muy importante que las propias aplicaciones/agentes ayuden al usuario a alcanzar sus objetivos de la forma más eficiente posible. Debido a las características de este contexto de ejecución las arquitecturas basadas en agentes móviles son la mejor aproximación porque minimizan las comunicaciones de red utilizadas, son robustos frente a desconexiones, y se adaptan automática y autónomamente a entornos dinámicos y con un número de recursos reducido.

Las soluciones en este contexto se centran principalmente en aplicaciones web con una arquitectura cliente/servidor, creando servicios especializados y centralizados que transforman un tipo de interfaz de usuario en otro. Algunas de las soluciones propuestas se basan en la creación de GUIs distintos para cada tipo de dispositivo, que son visualizadas posteriormente dependiendo del tipo de dispositivo que realiza la solicitud del servicio. Algunos autores, como [MM02], proponen describir los GUIs utilizando XML que posteriormente podrán ser visualizados como un interfaz Java AWT o Swing [Mic06a], o que podrán ser transformados utilizando plantillas XSLT [Wor06b] a algún tipo de representación gráfica de GUIs (HTML, WML, Java Swing, etc). Existen algunos trabajos previos enfocados a la construcción de interfaces de usuario adaptativos que se adaptan a distintos tipos de dispositivos [Tut06, eIML06] y que permiten estudiar el comportamiento del usuario con el fin de poder predecir la siguiente acción más probable y generar un interfaz de usuario más fácil y rápido de usar. Nuestro trabajo se centra en diseñar una arquitectura basada en agentes que permita la generación de interfaces gráficas de usuario adaptándose automáticamente al contexto de ejecución, nuestra arquitectura ha sido utilizada por [MM02, MM03]. Del mismo modo, hay que destacar que la arquitectura propuesta podría ser utilizada por cualquier otro método de generación de interfaces de usuario basado en una especificación del GUI.

Como se ha descrito anteriormente, la finalidad de la arquitectura de agentes diseñada,

es generar interfaces de usuario para dispositivos con diferentes características y estudiar el comportamiento del usuario adaptándose automáticamente a entornos con redes inestables y lentas como las inalámbricas. La utilización de agentes permite facilitar el diseño de aplicaciones evitando diseñar un interfaz de usuario para cada tipo de dispositivo, permitiendo que el software se adapte automáticamente al contexto en el que se esté ejecutando y minimizando los costes debidos a comunicaciones de red. En nuestro caso utilizaremos como caso de ejemplo un servicio de acceso a datos genérico basado en el sistema ADUS [MRM04] — Sistema de interfaces de usuario avanzado (*ADvanced User interface System*)— permitiendo la creación de interfaces adaptativos que facilitan el estudio y análisis de las acciones ejecutadas por el usuario. El sistema ADUS [MRM07] permite la generación de interfaces de usuario indirectos mediante la utilización de agentes especializados, facilitando una adaptación automática al contexto de ejecución.

Del mismo modo, gracias a la utilización de una arquitectura basada en agentes móviles inteligentes [PSP00] nuestra propuesta puede adaptarse fácilmente a distintos contextos de ejecución. Además, una solución basada en agentes móviles proporciona la ventaja adicional de que éstos pueden viajar al dispositivo del usuario y mostrar su GUI al usuario para interactuar con él o ella. Del mismo modo, los agentes móviles pueden ejecutarse en plataformas que soporten distintos tipos de interfaces de usuario o que tengan diferentes capacidades de procesamiento. Los agentes son autónomos, y pueden capturar y tratar errores de red (ordenadores inalcanzables, etc.) autónomamente. También, pueden moverse al dispositivo destino en lugar de tener que acceder al mismo de forma remota. Hay que destacar que los agentes pueden ser enviados a cualquier ordenador que soporte Java, y por otra parte, un agente puede jugar el papel de servidor *proxy* de un dispositivo inalámbrico, como por ejemplo un teléfono móvil o un terminal de acceso web; en este caso se generará un interfaz WML [WWSV06] o HTML [Wor06a], respectivamente. Por consiguiente, la generación del GUI por el agente siempre se hará lo más cerca posible del usuario, es decir, si el dispositivo móvil del usuario acepta Java se hace la transformación en el propio dispositivo y si no en la estación base que le proporciona cobertura. Por el contrario, una solución que no estuviese basada en agentes móviles debería ser instalada en el dispositivo del usuario o ejecutarse remotamente, hecho que no sucede utilizando una arquitectura basada en agentes móviles: solamente se necesita actualizar el agente móvil y éste llevará las nuevas funcionalidades a todos los dispositivos.

7.1.1. Motivación

El sistema de interfaces de usuario avanzado (ADUS), al igual que el Servicio de Recuperación de Software, es parte de un sistema más global denominado ANTARCTICA [GIM+01], su función es proporcionar distintos tipos de servicios inalámbricos a los usuarios de tal forma que aumenten las prestaciones/capacidades de sus dispositivos móviles. La arquitectura de ANTARCTICA (mostrada en la Figura 7.1) se basa en el modelo cliente/interceptor/servidor, e incorpora módulos y agentes en los dispositivos inalámbricos y en los elementos intermediarios del sistema de comunicaciones (también llamados *proxies*), que se encuentran ubicados en la red fija.

En ANTARCTICA, el agente intermediario Alfredo es el responsable de la personalización, descubrimiento de nuevos servicios, y generación de GUIs. Alfredo es un mayordomo

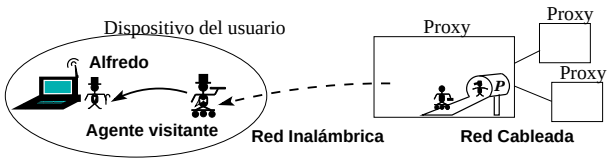


Figura 7.1: Arquitectura del sistema ANTARCTICA

eficiente[1] que sirve al usuario y se encarga de almacenar tanta información, acerca del dispositivo del usuario y del usuario en sí mismo, como le sea posible. Cuando otro agente, el (*agente visitante*), quiere mostrar, solicitar o retornar algún dato al usuario tiene que comunicarse con Alfredo, que crea el interfaz de usuario apropiado conforme a las capacidades de visualización del dispositivo y a las preferencias del usuario (cualquier otro agente desconocerá dicha información). Definimos el *agente de usuario* como aquellos agentes que, de forma similar a Alfredo en ANTARCTICA, son los únicos que interactúan con el usuario porque son los únicos que gestionan el conocimiento acerca de sus preferencias y acerca del dispositivo en el que se ejecutan.

Desde el punto de vista de la generación de GUIs, la visualización de la interfaz de usuario es una tarea compleja [MM02, MM03]; como consecuencia de las distintas características que puede tener el dispositivo del usuario: velocidad del procesador, tamaño de la pantalla, capacidad para mostrar imágenes, tocar sonidos o mostrar películas, etc. Existen varias aproximaciones de diseño que combinan desde unos pocos componentes gráficos (*widgets*), como por ejemplo los GUIs WML, a muchos *widgets*, como Java Swing. Dependiendo de las capacidades del dispositivo, el diseñador debe seleccionar uno u otro componente. Por lo tanto, el desarrollo de GUIs para aplicaciones en contextos con dispositivos heterogéneos tiene los siguientes problemas:

- *Adaptar las interfaces de usuario*: el agente visitante debe adaptar la creación de la interfaz a las preferencias del usuario y a las capacidades del dispositivo. Por ejemplo, el usuario puede preferir visualizar las imágenes como *thumbnails* antes que a tamaño completo. Los agentes visitantes no consideran el contexto del usuario, las capacidades y características del dispositivo, la plasticidad [TC99], etc. Además, las preferencias del usuario pueden cambiar durante la ejecución de una aplicación.

- *Estudiar, analizar y almacenar el comportamiento del usuario*: si el comportamiento del usuario es estudiado y almacenado, entonces el agente de usuario podrá utilizar estos datos de ejecuciones anteriores para, automáticamente, asignar los posibles valores iniciales de los GUIs de las futuras ejecuciones de un servicio siempre que se requiera la misma información, es decir, el agente de usuario completará parcialmente algunos valores de los parámetros tomados por los agentes visitantes. Por ejemplo, siempre que se solicite el nombre del usuario, el agente de usuario podrá completar esta información automáticamente.

[1] Alfredo es un intento de reflejar un papel similar al de un mayordomo en la vida real.

Por lo tanto, la finalidad de la arquitectura presentada en esta sección es la creación de interfaces de usuario adaptativos para *agentes visitantes* utilizando *agentes de usuario* para personalizar dichos interfaces conforme a las preferencias del usuario y a las capacidades del dispositivo móvil, permitiendo el estudio y análisis de las interacciones del usuario con el GUI y optimizando la utilización de las comunicaciones inalámbricas. Debido a la utilización de agentes móviles se facilita la adaptación de la arquitectura propuesta a un contexto heterogéneo, dinámico y distribuido como en el que nos encontramos.

7.1.2. Generación de interfaces de usuario adaptables

En esta sección presentamos y discutimos distintas aproximaciones basadas en agentes que permiten la generación de interfaces de usuario adaptativos, a la vez que se permite analizar el comportamiento del usuario, estudiando qué acciones ejecuta el usuario y qué información proporciona para ejecutar dichas acciones, para posteriormente reutilizar dicha información.

Opción 1: el agente visitante genera el GUI

La primera aproximación se basa en que el agente visitante que llega al dispositivo del usuario solicite al agente de usuario qué recursos hay disponibles, las preferencias del usuario y las capacidades de visualización del dispositivo. Por lo tanto, el agente visitante crea el GUI por sí mismo e interactúa con el usuario directamente.

Esta aproximación resuelve el problema de la generación de GUIs personalizados según las preferencias del usuario, sin embargo, todavía tiene varios problemas:

1. El agente de usuario no puede analizar el comportamiento del usuario debido a que los datos proporcionados al GUI son tratados directamente por el agente visitante.

2. El agente de usuario debe confiar en que el agente visitante visualizará (generará el GUI) conforme a las preferencias del usuario y a las capacidades del dispositivo. Los agentes visitantes pueden ignorar las recomendaciones del agente de usuario y tratar de mostrar su propio GUI[2].

3. Todos los agentes visitantes tienen que conocer el mecanismo de comunicación con el agente de usuario y aplicar el conocimiento que proporciona el agente de usuario (lo cual implica que todos los agentes visitantes tienen que saber generar cualquier tipo de GUI).

Opción 2: El agente usuario genera el GUI y delega la captura de eventos en el agente visitante

En esta aproximación, el agente visitante, después de llegar al dispositivo del usuario, proporciona al agente de usuario una especificación del GUI que desea mostrar al usuario.

[2]En este caso, el tipo de GUI creado por el agente visitante puede que no se pueda generar y visualizar correctamente en el dispositivo.

Posteriormente, el agente de usuario genera el GUI de acuerdo a las preferencias del usuario, las capacidades del dispositivo, y los requisitos del agente visitante, y delega la captura de eventos del GUI al agente visitante.

La especificación del interfaz de usuario puede realizarse en un lenguaje de interfaces de usuario basado en XML, como por ejemplo XUL (*Extensible User-interface Language*) [Tut06]. Esta definición del interfaz posteriormente puede ser adaptada a los requisitos del lenguaje de representación de GUIs (HTML,WML, etc.) por el agente de usuario utilizando transformaciones XSL. La interpretación de XUL en las plataformas que permiten la utilización de Java se realiza utilizando la plataforma jXUL [jXU06] que visualiza XUL utilizando componentes gráficos basados en Java estándar.
Las ventajas de esta aproximación son:

- El agente de usuario garantiza que el GUI del agente visitante se generará correctamente (conforme a las preferencias del usuario y las capacidades técnicas del dispositivo).
- El agente visitante no necesita conocer cómo generar los GUIs en diferentes dispositivos.
- El agente de usuario puede denegar el permiso de generación de GUIs a todos los agentes visitantes [MA02] para evitar la generación directa de GUIs.

Sin embargo, siguiendo esta aproximación, el agente de usuario no puede estudiar y analizar el comportamiento del usuario porque los eventos del GUI son capturados directamente por los agentes visitantes. Por lo tanto, el agente de usuario debe confiar en el agente visitante para extraer la información acerca de la interacción del usuario con el GUI.

Opción 3: Un agente intermediario genera el GUI y captura los eventos

En esta aproximación, en primer lugar, el agente visitante envía su especificación XUL del GUI al agente de usuario, en segundo lugar, el agente de usuario genera el GUI y captura todos los eventos (recibe los datos del usuario), y finalmente, los envía de vuelta al agente visitante.

Esta aproximación tiene todas las ventajas de las aproximaciones presentadas anteriormente. Además, permite que el agente de usuario analice fácil y eficientemente el comportamiento del usuario, debido a que él se encarga de la captura de eventos.

Aunque esta aproximación es interesante, su implementación tiene un problema: el agente de usuario debe atender los distintos servicios que se ejecutan concurrentemente en el dispositivo del usuario y algunas otras tareas, por lo tanto la generación de GUIs, podría sobrecargarlo. Por consiguiente, una aproximación mejor es que el agente de usuario delegue la generación de GUIs adaptativos a un agente especializado, el *agente ADUS*. Del mismo modo, la distribución de los servicios ejecutados en tres agentes (el agente ADUS, el agente de usuario, el agente visitante) nos permitirá conseguir un balanceo de la carga del sistema.

7.1.3. Generación indirecta de interfaces gráficos de usuario

En esta sección describimos en más detalle la arquitectura necesaria para la generación eficiente de GUIs adaptativos. Como se muestra en la Figura 7.2 nuestro sistema está constituido por los siguientes agentes:

- *El agente visitante*: Es un agente móvil que proporciona el servicio solicitado por el usuario y que será ejecutado en el dispositivo del usuario. Este agente es capaz de generar una especificación XUL [Tut06] del GUI que necesita para interactuar con el usuario. Dicha especificación se envía al agente de usuario en el dispositivo del usuario.

- *El agente de usuario*: es un agente especializado en la personalización de servicios adaptándolos a las preferencias del usuario, este agente se encarga de almacenar tanta información acerca del dispositivo del usuario y del usuario en sí mismo, como sea posible. Por ejemplo, conoce: las preferencias de visualización (*look and feel*) del usuario, el tipo de GUI preferido por el usuario o impuesto por el dispositivo del usuario o el sistema operativo. Los principales objetivos de este agente son: 1) hacer de intermediario en la generación de interfaces de usuario, 2) ayudar al usuario a utilizar los distintos servicios de los agentes visitantes, 3) modificar las especificaciones del GUI del agente visitante conforme a las preferencias del usuario, 4) crear un agente ADUS inicializado con las características estáticas del GUI, como las características del dispositivo, y 5) analizar y estudiar las interacciones del usuario para que posteriormente dicha información pueda ser proporcionada al agente ADUS.

- *El agente ADUS*: las principales características de este agente son: 1) adaptar el interfaz de usuario a las preferencias del usuario y a las capacidades de su dispositivo, siguiendo las sugerencias del agente de usuario, 2) generar GUIs para diferentes tipos de dispositivos siguiendo las especificaciones XUL proporcionadas, y 3) capturar los eventos del GUI y comunicarlos al agente visitante y al agente de usuario (este hecho permite que el agente de usuario analice las interacciones del usuario y, posteriormente, ejecute acciones en nombre del usuario o las sugiera). Habrá un agente ADUS por cada agente visitante.

A continuación describiremos el mecanismo de sincronización de los agentes descritos anteriormente utilizando un ejemplo. Los principales pasos de su ejecución son (ver Figura 7.2):

1. *El agente visitante viaja al dispositivo del usuario*: esta acción tiene lugar únicamente en aproximaciones basadas en agentes móviles. Por ejemplo, sería equivalente a la invocación de una aplicación local en una arquitectura cliente/servidor.

2. *El agente visitante solicita la generación de su GUI*: en este paso el agente visitante envía la descripción de su GUI al agente de usuario.

3. *El agente de usuario procesa la especificación del GUI*: el agente de usuario transforma la descripción del GUI para adaptarla a las preferencias del usuario, y crea el agente ADUS, asociado al servicio, inicializado con: 1) la descripción XUL ya transformada

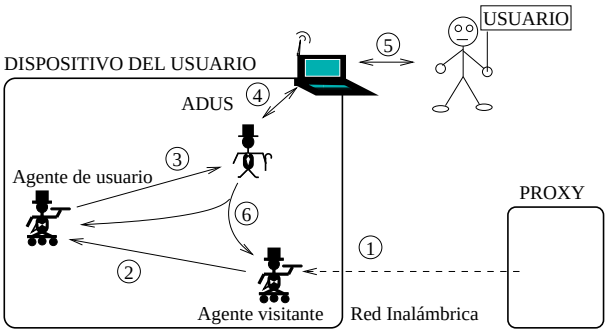


Figura 7.2: Generación indirecta de GUIs

del GUI a generar, y 2) la información estática con las capacidades del dispositivo para poder generar el GUI: resolución de la pantalla, lenguaje de representación del dispositivo del usuario (WML, HTML, Java Swing, etc.), entre otra información.

4. *El agente ADUS genera el GUI*: crea el GUI de acuerdo a la información proporcionada por el agente de usuario (con la información estática del GUI y la información especificada por el servicio). El agente ADUS tiene la capacidad de transformar cualquier descripción XUL en un GUI para dispositivos con distintas características, por ejemplo, un dispositivo WAP o un portátil con un GUI Java.

5. *Interacción con el usuario*: el usuario interactúa con el GUI observando la información visualizada en la pantalla del dispositivo y utilizando los distintos periféricos (teclado, ratón, botones, etc.) para introducir datos o seleccionar una opción entre varias.

6. *El agente ADUS captura los eventos del GUI y los propaga*: las acciones del usuario disparan un conjunto de eventos que son capturados por el agente ADUS. Esta información es enviada a: 1) el agente visitante, que reaccionará a las acciones del usuario dependiendo del servicio que se esté ejecutando, quizás mediante la generación de un nuevo GUI (paso 2), y 2) el agente de usuario, que puede almacenar y analizar la información proporcionada por el usuario para reutilizarla en futuras ejecuciones del servicio. Una de las ventajas de la arquitectura presentada es que ambos mensajes son enviados concurrentemente, por lo tanto se realiza un balanceo de carga.

Por último, destacar la relevancia para el agente de usuario del almacenamiento y posterior análisis de las interacciones del usuario con los agentes visitantes. Conociendo las reacciones del usuario y los datos introducidos en los distintos servicios, el agente de usuario puede almacenar dichos datos localmente y aplicar diferentes técnicas de inteligencia artificial para extraer conocimiento acerca del comportamiento del usuario. Del mismo modo, la personalización de los GUIs puede ser muy útil para el usuario, debido a que el agente de usuario es capaz de almacenar y analizar las interacciones del usuario con el GUI en todas las aplicaciones, podrá modificar automáticamente el tamaño de las fuentes de los servicios ejecutados si detecta que siempre se produce este comportamiento.

7.2. Servicios basados en la localización

El aumento de los dispositivos móviles, las redes de área local y redes ad hoc [And01] están desencadenando un creciente interés en servicios y sistemas dependientes de la localización. Sistemas de localización global, como el GPS (*Global Positioning System [Inc99]*), se están haciendo muy populares entre los conductores y usuarios de PDAs. La explicación de este interés en los sistemas de posicionamiento es que existen *muchas* aplicaciones en diferentes áreas como transporte, agricultura, ingeniería civil, educación, sanidad, etc. Por lo tanto, nuevas técnicas se están desarrollando para calcular la localización de los usuarios/dispositivos en un área global, como el servicio de navegación por satélite europeo (GALILEO) [Adv05], al igual que para redes de área local, con el propósito de dar soporte a aplicaciones basadas en la localización que funcionen en entornos cerrados y abiertos.

La localización del usuario/dispositivo puede no ser muy útil en sí misma, pero puede utilizarse como parámetro de entrada para servicios de datos. Entonces, podemos desarrollar, mapas, mostrar la localización de una persona concreta en un edificio, buscar la oficina de policía más cercana, etc, la lista de servicios es interminable. Por lo tanto, en nuestra sociedad se desea el desarrollo de nuevos servicios dependientes de la localización. Como por ejemplo, nuevas aplicaciones que, basándose en la localización del dispositivo del usuario (obtenida automáticamente por una infraestructura subyacente), provee a los usuarios inalámbricos con servicios personalizados para su localización.

En esta sección se detalla una arquitectura basada en agentes inteligentes [MBB+98, SSPE04a] que obtienen la localización geográfica de dispositivos inalámbricos, como, por ejemplo PDAs, en redes de área local inalámbricas, como redes WiFi[3] [Bre97] o Bluetooth [Mor02b]. La arquitectura diseñada permite proporcionar la infraestructura subyacente para el desarrollo de servicios dependientes de la localización con las siguientes características: 1) minimiza los recursos consumidos para localizar el dispositivo del usuario, 2) no requiere la instalación de ningún software en el dispositivo del usuario, y 3) es robusta frente a fallos de red gracias a la utilización de agentes móviles. Para mostrar las ventajas de la arquitectura propuesta en esta tesis y basada en agentes móviles inteligentes se ha desarrollado un servicio que permite la localización de dispositivos móviles en un edificio y otro que permite reproducir en un sistema de audio distintas composiciones musicales modificando los altavoces activos dependiendo de la ubicación del usuario.

El prototipo implementado se basa en la tecnología WiFi, así el sistema es capaz de localizar dispositivos móviles en el interior de edificios. Debido a que GPS no funciona dentro de edificios, tan pronto como un dispositivo móvil entra en un edificio, el sistema GPS pierde su localización (porque se necesita tener directamente visibles al menos cuatro satélites GPS). Las redes de área amplia, redes WAN, como GSM y GPRS también proporcionan mecanismos de localización para algunos usuarios pero la precisión es a nivel de celda: estos sistemas obtienen en qué celda se encuentra el usuario, el problema es que el tamaño de las celdas puede variar de cientos de metros a varios kilómetros. Por lo tanto, nuestro sistema ofrece funcionalidades que los sistemas actuales de posicionamiento global no proporcionan.

[3] Aunque existen diferentes estándares bajo el nombre de WiFi (IEEE 802.11b, 802.11g,etc), en nuestra aproximación no es necesario hacer ningún tipo de distinción entre los mismos.

7.2.1. Información utilizada para estimar la posición de un dispositivo móvil

En esta sección detallamos la información utilizada para estimar la posición de un dispositivo móvil. Se necesita identificar algún tipo de información que emita o reciba el dispositivo y que dependa de la localización geográfica del mismo. Por consiguiente, se deben considerar distintos parámetros del protocolo de comunicaciones inalámbrico:

- *Potencia de la señal recibida*: esta es la potencia con la que una estación base[4] (BS) recibe las emisiones del dispositivo móvil; se decrementa con la distancia.
- *Ruido en la señal recibida*: el ruido es una medida de la cantidad de interferencias detectadas en la recepción de la señal.
- *Tiempo de transporte de los datos*: es el tiempo transcurrido desde que un paquete de datos es enviado por el dispositivo móvil hasta que es recibido por la estación base que le da cobertura y además también puede ser una medida de la posición del dispositivo: a mayor distancia entre el dispositivo y la estación base, mayor cantidad de tiempo transcurrirá desde que el paquete es transmitido por el dispositivo hasta que éste alcance la estación base.

El tiempo de transporte de datos no sólo varía con la distancia, sino que también depende de otros factores, como la carga de la red. Por esta razón es por lo que muchos investigadores que trabajan en el posicionamiento de dispositivos móviles [BP00, LBM+02, KWW00] seleccionan la potencia de la señal frente al tiempo de transporte. Por lo tanto, en la arquitectura propuesta utilizaremos la potencia de la señal y el ruido detectado para estimar la posición geográfica de los dispositivos móviles, la aproximación sería igualmente válida utilizando cualquier otro mecanismo de triangularización.

WiFi versus Bluetooth

Una vez que hemos decidido el tipo de información que necesitamos para calcular la posición (localización) de un dispositivo, debemos investigar cómo acceder a dicha información. Para obtener la potencia de la señal de los mensajes enviados por un dispositivo móvil tendremos que acceder a la información disponible en la pila del protocolo de comunicaciones, para realizar esta tarea utilizaremos un agente *Sniffer* que nos permitirá capturar y analizar los paquetes de datos transmitidos. Estas dos tareas dependen enteramente de la tecnología inalámbrica utilizada, por lo tanto a continuación resumiremos las conclusiones extraídas de las pruebas realizadas utilizando redes Bluetooth [Mor02b] y WiFi [Bre97]:

- *Bluetooth:* Aunque la pila del protocolo Bluetooth nos proporciona datos acerca de la potencia de la señal, desafortunadamente no es posible utilizar dicha información para hallar la posición de un dispositivo móvil porque en el protocolo Bluetooth la potencia de la señal no varía con la distancia. La explicación es que cuando dos dispositivos

[4] El vocablo anglosajón que identifica a las estaciones base es *Base station*.

Bluetooth se detectan mutuamente, cada uno de ellos automáticamente regulará la potencia de emisión para no gastar más energía de la estrictamente necesaria. Es decir, la potencia de la señal es reajustada automáticamente según la especificación del protocolo Bluetooth para mantener la calidad del servicio [Tsc02] optimizando el uso de energía. Del mismo modo, cuando dos dispositivos Bluetooth se acercan la potencia con la que se recibe la señal es constante (la calidad de servicio se mantiene constante). Además, diferentes tipos de dispositivos Bluetooth en la misma posición pueden necesitar diferente potencia de señal para poder conectarse a la misma estación base, esto depende del tipo de chip Bluetooth que poseen.

- *WiFi:* En nuestras pruebas descubrimos que podíamos localizar dispositivos móviles con tecnología WiFi, utilizando la potencia de la señal recibida por las estaciones base, porque la potencia de la señal cambia cuando la distancia entre el dispositivo del usuario y la estación base también cambia, este hecho se muestra en la Figura 7.3.

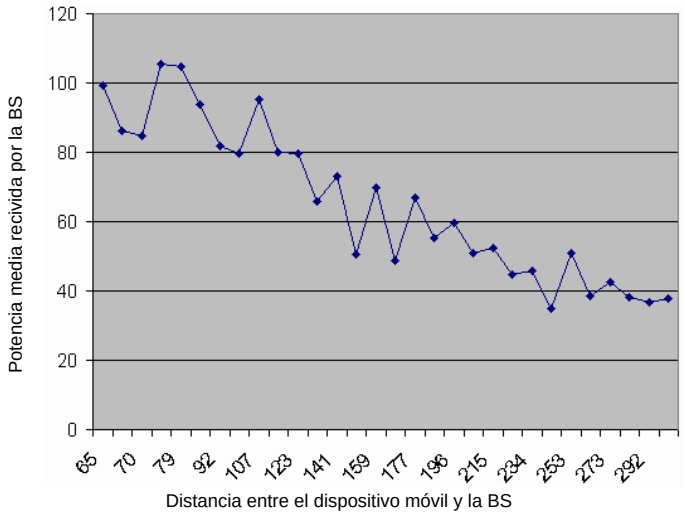


Figura 7.3: Potencia de la señal recibida por una estación base cuando un dispositivo móvil se aleja

Mapas de potencia

Inicialmente podría pensarse que la estimación de la distancia de una estación base al usuario (más concretamente a su dispositivo), utilizando una transformación del valor de la potencia de la señal en una magnitud de distancia, considerando que la potencia de la señal decrece con la distancia, es una transformación directa. Sin embargo, en los entornos cerrados, el número de obstáculos físicos (estáticos o móviles) es tan alto que la definición de tal transformación no es posible. Entre los distintos obstáculos que nos podemos encontrar se encuentran: personas caminando, puertas (abiertas o cerradas), muros, etc.

Una alternativa para posicionar dispositivos móviles es la definición de *mapas de potencia* [BP00, NDA01] que permiten relacionar áreas geográficas con la potencia de la señal

recibida por una estación base (estática) y que es emitida por un dispositivo móvil localizado en ese área. Estas áreas geográficas pueden ser de diferente tamaño y topología, dependiendo de la precisión necesaria. Esta técnica para obtener los valores de la potencia de la señal en un lugar concreto y su posterior análisis es conocido como huellas dactilares (*fingerprint*) [BP00, NDA01]. De esta forma pueden construirse diferentes mapas de potencia para las áreas de cobertura de diferentes estaciones base estáticas; del mismo modo, la estimación de la posición del dispositivo móvil puede ser más precisa mediante la utilización de técnicas de triangularización [BBP00].

7.2.2. Arquitectura del sistema

En esta sección detallamos los elementos básicos del sistema de localización (ver Figura 7.4) y sus principales objetivos:

1. *Dispositivo del usuario*: es el dispositivo móvil conectado mediante una red inalámbrica.

2. *Estación base*: son los ordenadores conectados a la red fija que proporcionan cobertura a los dispositivos móviles dentro de una cierta área geográfica.

3. *Servidor de Localización*: es un ordenador que se encuentra conectado a la red fija que calcula la localización de los dispositivos móviles.

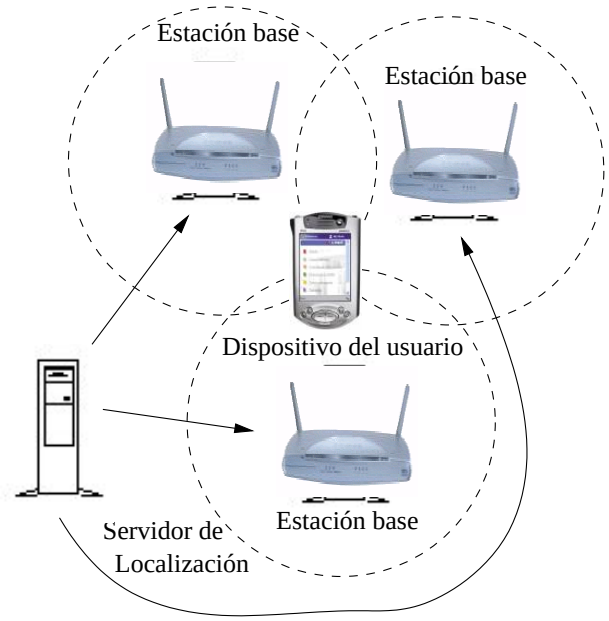


Figura 7.4: Componentes del sistema de localización

El principal objetivo de las estaciones base es proporcionar cobertura a los usuarios inalámbricos, para capturar los paquetes de datos transmitidos por los dispositivos móviles y para extraer de estos paquetes la potencia de la señal y el ruido recibido, para el servicio de posicionamiento. Estos datos son enviados al servidor de localización por todas las estaciones base que detectan el dispositivo móvil del usuario. A continuación, el servidor de localización consulta los mapas de potencia de las distintas estaciones base para realizar una estimación de la localización del dispositivo móvil del usuario.

Los principales agentes como plataforma de agentes móviles, aunque podría utilizarse cualquier otra. de la arquitectura que vamos a presentar y la finalidad de cada uno de ellos es la siguiente:

- *El agente Sniffer*: se encuentra localizado en cada una de las estaciones base, analiza los paquetes de datos enviados por los dispositivos móviles y extrae los parámetros necesarios para estimar su posición geográfica. Este agente nunca deberá ser transmitido por la red inalámbrica de comunicaciones evitando generar tráfico en la misma. Por otra parte, gracias a que este agente tiene un comportamiento pasivo no interfiere en las comunicaciones realizadas entre la estación base y el dispositivo del usuario, solamente monitoriza la información transmitida.

- *El agente Beacon*: este agente móvil puede moverse (bajo petición) al dispositivo del usuario para enviar mensajes/paquetes, también llamados *beacons*, que serán detectados por las estaciones base en el rango de distancia que proporcione la potencia de transmisión de la señal. Hay que destacar que gracias a que este agente tiene la capacidad de trasladar su ejecución no debe estar siempre en funcionamiento en el dispositivo del usuario consumiendo recursos (memoria, espacio de disco, etc.). Por otra parte, debido al reducido tamaño de dicho agente el consumo de comunicaciones inalámbricas debido al viaje del agente desde la estación base al dispositivo móvil del usuario es mínimo.

- *El gestor de la base de datos de localización*: se ejecuta en el servidor de localización, automatiza la generación de los mapas de potencia de cada una de las estaciones base que constituyen la infraestructura, almacena dichos datos en una base de datos y estima en tiempo de ejecución la posición del dispositivo móvil solicitado.

Los servicios de localización necesitan para poder ejecutarse un *cliente de localización*, que es un agente que proporciona un API que puede utilizarse para obtener la posición de un dispositivo móvil concreto, ésta es la interfaz para la arquitectura propuesta (es la interfaz para interactuar con el gestor de la base de datos de localización). El cliente es un agente móvil, por lo tanto puede viajar al dispositivo del usuario cuando se requiere la ejecución de un servicio basado en la localización, además se clonará para facilitar la ejecución en paralelo de distintos servicios basados en la localización.

El agente *Sniffer*

El *sniffer* es un agente especializado en analizar la información de cada uno de los paquetes recibidos por una estación base. Es el único agente que necesita ser modificado si cambia

el protocolo de comunicaciones. Cada paquete de datos transmitido incluye su cabecera de nivel TCP-IP [Com00]. Por ejemplo, el estándar WiFi [Bre97] define que la dirección MAC, que identifica los distintos dispositivos móviles, debe ser almacenada en cada paquete. Sin embargo, también es necesario analizar las cabeceras de nivel físico para extraer la potencia y el ruido de la señal. El formato de estas cabeceras depende del tipo de dispositivo [AVS05], por consiguiente el agente Sniffer debe poseer una inteligencia que le permita distinguir los distintos tipos de dispositivos para de esta forma extraer la potencia de la señal de forma diferente.

Debe existir un agente Sniffer en cada estación base para permitir el desarrollo de servicios dependientes de la localización. Esta aseveración se fundamenta en que cada agente Sniffer se encarga de obtener la potencia y el ruido de los paquetes de datos transmitidos por el dispositivo móvil y detectados por la estación base en la que se está ejecutando. posteriormente el agente Sniffer envía estos parámetros al agente Gestor de la base de datos de localización, que tiene acceso a los mapas de potencia correspondientes a cada estación base. Por lo tanto, la localización del dispositivo móvil se hará mediante la combinación de los diferentes mapas de potencia asociados a cada estación base, mediante la inteligencia del agente Gestor de la base de datos de localización.

Como se ha comentado anteriormente, nuestra aproximación se fundamenta en dos etapas, una primera en la que el agente Sniffer ayuda en la creación de los mapas de potencias, y una segunda en la que el agente Sniffer es utilizado para localizar el dispositivo móvil del usuario.

7.2.3. Creación de los mapas de potencia

En esta sección detallamos como se crean los mapas de potencia y como puede realizarse esta tarea de forma semi-automática.

Un mapa de potencia asociado a una estación base concreta contiene información acerca de la potencia y el ruido de la señal detectados por la estación base y transmitidos por los dispositivos móviles dentro de una cierta área geográfica. Del mismo modo toda el área de cobertura de una estación base es dividida en pequeñas áreas donde la señal detectada tiene características similares (potencia y ruido). Por lo tanto, dada una potencia y ruido concretos de una señal detectada por una estación base, el mapa de potencia nos indicará de qué área geográfica viene dicha señal. Si el mismo dispositivo es detectado por diferentes estaciones base, la información de localización obtenida de sus mapas de potencia puede ser combinada para estimar la localización de ese dispositivo. La base de datos de localización también almacena la localización geográfica de la estación base porque el posicionamiento de los dispositivos se calcula tomando como base la posición de la estación base que los detecta, es decir, en este proceso se obtiene la distancia a la que se encuentra el dispositivo móvil del usuario de la estación base (no su ubicación concreta).

Para crear los mapas de potencia de las estaciones base, el agente Sniffer debe ejecutarse en cada una de las estaciones base que proporcionan cobertura al dispositivo del usuario (que pueden recibir las señales que emite el dispositivo del usuario cuando utiliza la red inalámbrica) para capturar la potencia y el ruido de la señal recibida, que fue emitida por el dispositivo móvil que se quiere localizar. Por lo tanto, para crear los mapas de potencia, el agente Beacon

deberá ejecutarse en el dispositivo móvil del usuario y además necesitaremos una herramienta que nos muestre un mapa del área en que nos encontramos. Esta herramienta nos permite seleccionar manualmente la posición real donde nos encontramos en ese momento (ver Figura 7.5). Estas coordenadas de localización reales son enviadas al gestor de base de datos de localización por el agente Beacon; el agente Sniffer que se encuentra en cada una de las estaciones base detecta la señal correspondiente a la transmisión: cada estación base detecta el dispositivo con diferente potencia y ruido debido a que se encuentra a diferente distancia del dispositivo del usuario. Estos datos de la potencia y ruido de la señal son transmitidos por los sniffers al agente Gestor de la base de datos de localización que los almacenará en la base de datos. Del mismo modo, podemos continuar enviando nuevas señales asociadas a nuevas posiciones reales. De esta forma semiautomática, podemos cubrir toda el área geográfica de interés para nuestros servicios de localización. Destacar, que utilizando este método creamos los mapas de potencia para todas las estaciones base en un sólo paso para nuestra área de interés.

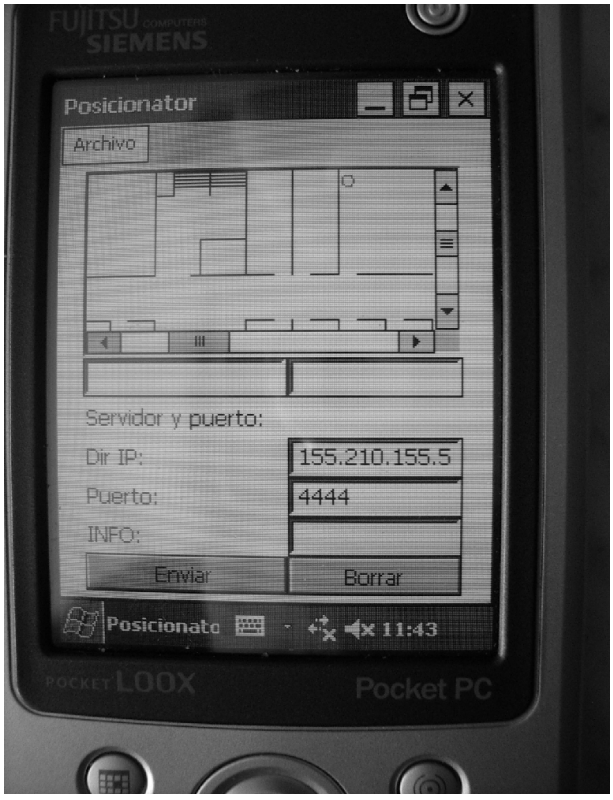


Figura 7.5: Creación de los mapas de potencia: especificando la posición del usuario

7.2.4. Localización del dispositivo del usuario

La localización del dispositivo del usuario se realiza por el gestor de la base de datos de localización utilizando la información previamente calculada en los mapas de potencia. La tarea de localizar un dispositivo comienza con la petición del dispositivo de un usuario concreto por un cliente de localización y finaliza con la respuesta que contendrá la posición del dispositivo del usuario especificado. Esta tarea se divide en los siguientes pasos:

1. *Petición de localización*: se lleva a cabo por un dispositivo móvil o fijo que desea conocer la posición del dispositivo de un usuario concreto. Esta petición es recibida por el agente Gestor de la base de datos de localización (ver la Figura 7.6, paso 1).

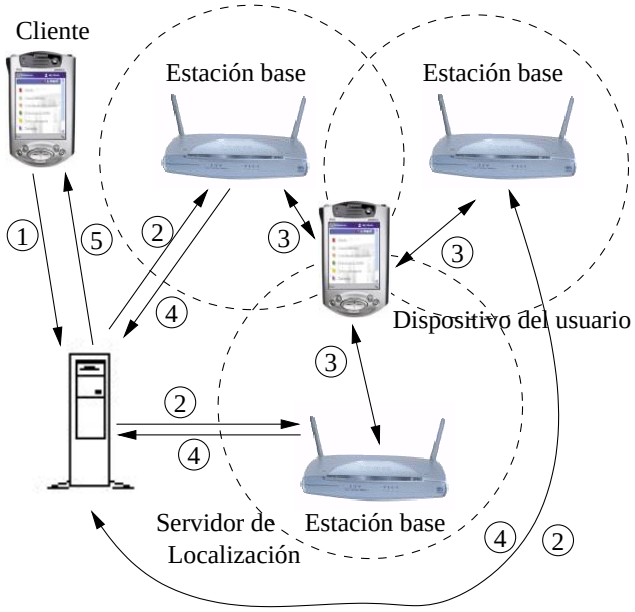


Figura 7.6: Posicionando un dispositivo móvil

2. *Petición de espiar las comunicaciones con el dispositivo del usuario*: el agente Gestor de la base de datos de localización envía un mensaje a cada una de las estaciones base[5] (Figura 7.6, paso 2) y se envía un agente Beacon al dispositivo del usuario que comenzará a enviar mensajes con una cierta frecuencia.

3. *Espionaje de las comunicaciones inalámbricas*: cada agente Sniffer estudiará los datos (beacons) recibidos por su estación base y que provengan del dispositivo del usuario (Figura 7.6, paso 3). A continuación, el agente Sniffer extrae la potencia y ruido de la señal de los mensajes de datos y retransmite dicha información al agente Gestor de la base de datos de localización (Figura 7.6, paso 4).

[5] Si el dispositivo buscado no se encuentra bajo el área de cobertura de una estación base el mensaje es ignorado.

4. *Estimación de la localización del dispositivo del usuario*: el agente Gestor de la base de datos de localización consulta los mapas de potencia de cada una de las estaciones base que han proporcionado medidas de la potencia y ruido detectadas (debido a los paquetes de datos transmitidos por el agente Beacon desde el dispositivo del usuario), y obtiene la posición del dispositivo del usuario. Este dato se devuelve al cliente de localización que realizó la petición (Figura 7.6, Paso 5).

Si al menos tres estaciones base detectan el dispositivo del usuario, el gestor de la base de datos de localización podrá estimar la posición del dispositivo del usuario con suficiente exactitud utilizando técnicas de triangularización [BBP00]; en otro caso, existirá más de una posible localización del dispositivo del usuario.

Recordar que al usar agentes en esta tarea tenemos las siguientes ventajas: 1) se facilita la creación de los distintos agentes en tiempo de ejecución, permitiendo su adaptación automática a los recursos disponibles en cada ubicación; 2) se evita la necesidad de instalar software para permitir la localización en el dispositivo del usuario; y 3) se permite el diseño y desarrollo de nuevos servicios basados en la localización, mediante la implementación de nuevos agentes que colaboren con el agente *Gestor de la base de datos de localización* y el agente *Sniffer*.

7.2.5. Ejemplos de servicios de datos dependientes de la localización

En esta sección detallaremos dos servicios desarrollados para utilizar el mecanismo de localización descrito anteriormente y probar su eficacia. El primero de los servicios puede ser utilizado para localizar usuarios mediante el posicionamiento de sus dispositivos móviles (también posibilita que un usuario pueda recuperar su dispositivo móvil si se lo han robado o lo ha perdido). El segundo de los servicios está íntimamente relacionado con las arquitecturas de computación ambiental inteligentes, debido a que soluciona el problema de que un usuario pueda escuchar las canciones que selecciona independientemente de su localización, es decir, las canciones siguen al usuario allí donde va.

Localización de dispositivos móviles en un edificio

Este servicio muestra cómo localizar en un mapa una lista de personas que pueden ser seleccionadas por el usuario. Este servicio se basa en una relación bidireccional entre los identificadores de los distintos dispositivos móviles (direcciones MAC) y personas del mundo real (sus propietarios), por consiguiente la tarea de localizar a una persona puede reducirse a la localización de su dispositivo móvil y su posterior visualización en un mapa (mostrando el nombre de la persona asociada al dispositivo). La localización de personas puede realizarse bajo demanda o de forma continua. El resultado del prototipo desarrollado se observa en la Figura 7.7.

Para trabajar con usuarios móviles, el sistema utiliza un agente Predictor que es el encargado de filtrar las posiciones que implican cambios desmesurados en el patrón de movimiento. Para suavizar el movimiento de los usuarios mostrados en la pantalla (debido a que los paquetes de datos pueden tardar un tiempo demasiado elevado, o perderse), el agente Predictor estima la posición del dispositivo móvil incluso cuando no se reciben nuevos datos de

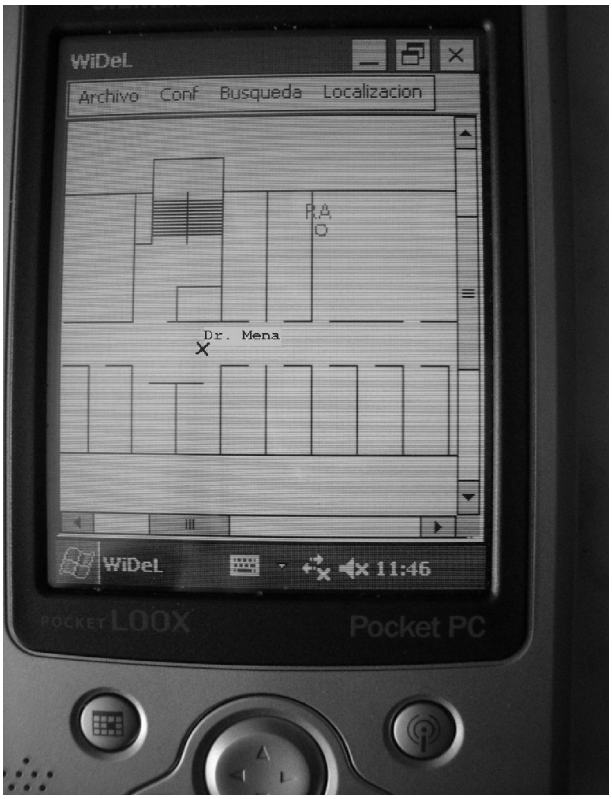


Figura 7.7: Servicio de localización de personas

forma continuada. Por lo tanto, la posición calculada por el agente Predictor se fundamenta en el movimiento de los usuarios en ausencia de datos que permitan calcular su ubicación ayudándose de la trayectoria que habían seguido anteriormente.

Este servicio tiene muchas aplicaciones. Por ejemplo, en un departamento universitario se pueden resolver diferentes tipos de preguntas como "¿dónde está el Dr. Mena?" o "dame el listado de las personas que hay en el aula A3". En ambientes domóticos, puede usarse para localizar a las personas que se encuentran viviendo en una casa o edificio. Permitiendo automatizar el encendido y apagado de luces, equipos de sonido, etc.

Servicio de música dependiente de la localización

Después de estudiar el proyecto Andante [UK04], decidimos crear una aplicación móvil que permita reproducir canciones seleccionadas por el usuario, además la reproducción cambia a un nuevo ordenador con capacidad de reproducir audio automáticamente al mismo tiempo que el usuario cambia de posición. Por lo tanto, esta aplicación: 1) estudia y analiza la posición del dispositivo del usuario, 2) reproduce un fichero de audio en el ordenador

más cercano posible al usuario[6], viajando de un ordenador a otro al mismo tiempo que se mueve el usuario, 3) reproduce la música sin interrupciones no deseadas mientras el agente está viajando, y 4) permite al usuario cambiar el fichero de audio que se va a reproducir.

Para satisfacer los requerimientos comentados anteriormente, este servicio se basa en la tecnología de agentes móviles. A continuación describimos las características de los principales módulos del sistema:

- *Cliente de música móvil*: esta aplicación se ejecuta en el dispositivo del usuario y permite al usuario seleccionar la canción de una lista predefinida. Este cliente está basado en un agente mayordomo que recomienda al usuario qué música escuchar dependiendo de sus gustos musicales.

- *Agente Reproductor*: este agente móvil es creado en el ordenador más cercano al usuario por el cliente de música móvil. Tiene dos objetivos principales; el primero es mantenerse tan próximo como sea posible al dispositivo del usuario, viajando de ordenador a ordenador siempre que sea necesario. Para conseguir este objetivo, el agente Reproductor requiere una actualización continua de la localización del dispositivo del usuario utilizando el cliente de localización propuesto en esta sección. Por consiguiente, el agente Reproductor deberá colaborar con el agente Gestor de la localización. El segundo objetivo es la reproducción de la canción seleccionada en los altavoces del ordenador más próximo al usuario, para alcanzar este objetivo el agente Reproductor se duplicará y trasladará su ejecución a distintos dispositivos.

Destacar que el agente Reproductor no es capaz de reproducir música mientras está viajando a un nuevo ordenador debido a que el usuario se está moviendo. Debido a esto, para conseguir una reproducción continua, el cliente de música móvil crea *dos agentes reproductores*: mientras el primer clon reproduce la música, el segundo analiza la localización del usuario, viajando al ordenador más apropiado cuando es necesario, posteriormente se sincroniza con el primer clon, y continua la reproducción de la canción[7] justo antes de que el primer clon detenga la reproducción y comience a analizar la localización del usuario. Por lo tanto, ambos agentes móviles intercambian sus papeles (reproductor de música y analizador de la localización del usuario) sincronizándose entre sí para evitar interrupciones en la reproducción debidas al movimiento de los agentes. En la Figura 7.8 mostramos la herramienta que analiza el servicio de música móvil para un usuario determinado (mostrando como una x); el ordenador que reproduce la música es mostrado como un pequeño circulo en gris claro.

Se podrían haber adoptado modelos de sincronización más complejos, como por ejemplo, crear un entorno musical envolvente alrededor del usuario (utilizando varios ordenadores para reproducir la música al mismo tiempo). Sin embargo, el propósito de este servicio solamente es mostrar las ventajas de la utilización de una arquitectura basada en agentes para el diseño y desarrollo de servicios dependientes de la localización y no estudiar la creación de un servicio de sonido envolvente.

[6] Asumimos que el dispositivo del usuario no puede reproducir audio con la calidad requerida.

[7] En nuestro prototipo, se utilizan ficheros de audio MIDI debido a su pequeño tamaño. Si los ficheros de música se encuentran en otro formato se podría solicitar a otro agente que tuviese capacidades de conversión entre formatos de audio que realizase dicha conversión.

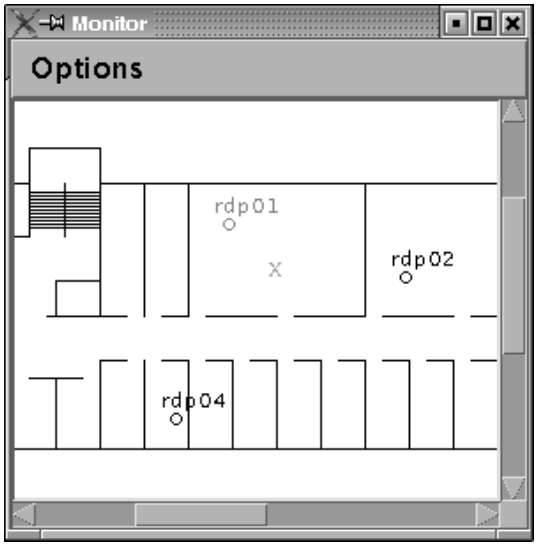


Figura 7.8: Servicio de música móvil

7.3. Arquitecturas basadas en agentes aplicadas a la Web Semántica

En esta sección, nos centraremos en resolver los objetivos de la Web semántica comentados anteriormente aplicando la tecnología de agentes. Nuestra meta es traducir un conjunto de palabras clave en una pregunta semánticamente equivalente expresada en un lenguaje estructurado, en nuestro caso OWL [AvH03], mostrando cuáles son las ventajas que se obtienen debido a la utilización de agentes inteligentes. Posteriormente, esta pregunta podrá ser utilizada por un agente inteligente para interrogar un conjunto de bases de conocimiento enlazadas con sus respectivos depósitos de datos y devolver dicha información al usuario. Debido a que la pregunta generada por los agentes que constituyen el sistema tiene una semántica bien definida no se devolverá al usuario resultados innecesarios o redundantes [TGEM07] como sucede cuando se utilizan los buscadores actuales [Goo06, Yah06].

El proceso de traducción es realizado por un conjunto de agentes especializados que: a) desambiguan la palabras (polisémicas) escritas por el usuario, b) descubren las relaciones semánticas existentes entre las palabras clave del usuario y los términos de un conjunto de ontologías pre-existentes, y c) crean preguntas con una semántica bien definida utilizando los términos de las ontologías que se han enlazado en el paso previo. A continuación, estas preguntas podrán ser procesadas para obtener los datos correspondientes, disponibles en los depósitos de datos subyacentes de las ontologías disponibles. Por lo tanto, las ventajas de nuestra aproximación debidas a la utilización de una arquitectura basada en agentes móviles son: 1) los agentes descubren los posibles significados de las palabras clave del usuario consultando el conocimiento de terceras partes (un tesauro [Mil06] y un conjunto de ontologías

pre-existentes); 2) se permite distribuir la carga de procesamiento en distintos ordenadores optimizando la utilización de la red de comunicaciones; y 3) los agentes utilizan las características deductivas de un razonador basado en Lógica Descriptiva [BCM+03] para facilitar la tarea de traducir las palabras clave del usuario en preguntas con una semántica bien definida.

Además gracias a la aplicación del paradigma de agentes móviles al entorno de los sistemas de información global se puede lograr una adaptación automática a entornos dinámicos y heterogéneos en los que los recursos disponibles en cada momento pueden variar. Esta característica se fundamenta en la facilidad que poseen las arquitecturas basadas en agentes móviles para ser desplegadas en tiempo de ejecución dependiendo de los recursos disponibles en cada momento.

7.3.1. Acceso a datos en la Web Semántica

Una de las tareas más comunes cuando alguien navega por la Web es buscar información acerca de algún tema. La mayoría de los motores de búsqueda actuales toman como entrada una lista de palabras clave (*keywords*), tal y como ha sido probado es una técnica que facilita la realización de búsquedas a todo tipo de usuarios. Sin embargo, estos motores de búsqueda realizan búsquedas sintácticas más que semánticas. Supongamos que un usuario introduce los siguientes keywords "*some publication Book thriller films*" para comprar un libro acerca de películas de cine. Debido a que los motores de búsqueda actuales están basados en técnicas de recuperación de información [BYRN99], obtendrán enlaces a páginas web donde aparezcan las palabras clave introducidas. El principal problema de su aproximación es que no consideran la semántica de las palabras clave, es decir, qué es lo que realmente está buscando el usuario. Esto eliminaría *muchos* de los resultados no deseables que son devueltos acerca de páginas web que describen algún tipo de libro o simplemente no describen ningún libro que trate acerca de películas de terror.

Hace unos pocos años, en el contexto de los sistemas de información globales, se tomó una aproximación diferente [MI01]. Estos sistemas proponían una aproximación para el procesamiento de preguntas basadas en ontologías [Gru93] que describían los depósitos de datos subyacentes. Estos sistemas, a pesar de que realizaban un procesamiento de preguntas dirigido por la semántica, seguían una aproximación basada en una *visión global —global-as-view (GAV)—* o una *visión local —local-asview (LAV)—* [FLM98], sin embargo tenían algunas limitaciones importantes:

1. La información de enlace y las relaciones interontología debían estar disponibles antes de que comenzase el procesamiento de la pregunta (GAV).

2. Cuando algún depósito de datos u ontología se unía/separaba del sistema de información global, la información de enlace y las relaciones semánticas debían ser actualizadas (GAV).

3. Para resolver estos problemas, se utilizó una aproximación LAV desencadenando graves problemas de cálculo y el diseño, además del desarrollo, de sistemas no escalables.

Por lo tanto, la estrategia seguida por estos sistemas no es suficientemente flexible para ser utilizada en un entorno altamente dinámico como la Web. En el que debe crearse dinámicamente la información de enlace y debe poderse generar preguntas formuladas utilizando términos de distintas ontologías. Por consiguiente, la utilización de arquitecturas basadas en agentes móviles permitirá adaptar este tipo de sistemas a redes inestables y caras como las inalámbricas. Además, la utilización de una aproximación basada en agentes facilita la adaptación de la arquitectura propuesta a entornos dinámicos, como por ejemplo la Web, donde los recursos disponibles pueden variar en el tiempo.

La *Web Semántica* "*trata de construir una extensión de la Web actual, en la que a la información se le proporcione un significado bien definido, haciendo posible una mejor cooperación entre ordenadores y personas*" [BLHL01]. Algunos de los objetivos principales de la web semántica son: 1) permitir el procesamiento de preguntas con una semántica bien definida, es decir, que cuando un usuario plantee una pregunta obtenga realmente lo que está buscando, y 2) descubrir automáticamente la información semántica entre las distintas ontologías disponibles en un sistema de información.

7.3.2. Descripción del sistema

Los principales pasos seguidos por un sistema de información global [TGEM07] para traducir un conjunto de palabras clave escritas por el usuario en preguntas expresadas en un lenguaje estructurado con una semántica bien definida son los siguientes (ver Figura 7.9):

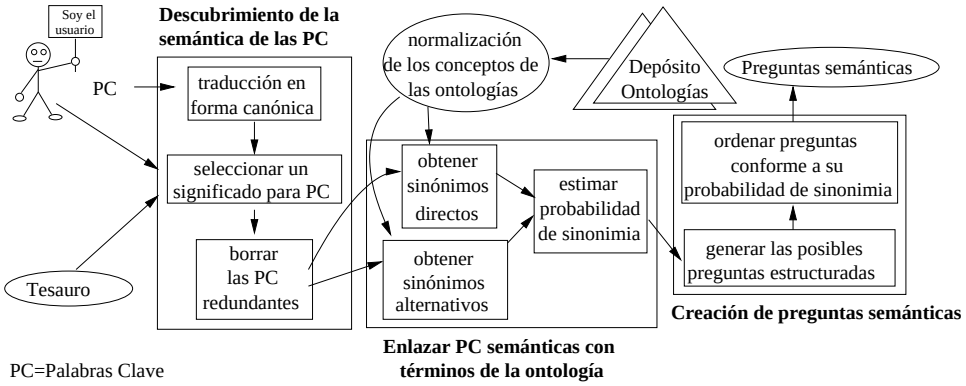


Figura 7.9: Proceso de traducción de las palabras clave del usuario en preguntas semánticas

1. *Descubriendo la semántica de las palabras clave (PC) del usuario*: después de realizar una normalización sintáctica, los agentes obtienen los posibles significados de cada una de las palabras clave del usuario utilizando un tesauro. La salida de esta tarea es una lista de *palabras clave con semántica*, expresando cada palabra clave como su: 1) descripción en lenguaje natural, 2) sinónimos, 3) hipónimos, y 4) hiperónimos. La intervención del usuario puede ser necesaria cuando el sistema se encuentra ante una palabra clave del usuario que sea polisémica, lo cual es altamente frecuente. Después

de seleccionar el significado correcto de cada una de las palabras clave, las palabras clave redundantes son eliminadas.

2. *Enlazando las palabras clave con semántica con los términos de una ontología* los agentes que constituyen el sistema estiman la probabilidad de sinonimia entre una palabra clave con semántica y cada uno de los términos de las ontologías disponibles en el sistema (estas ontologías pueden haberse desarrollado automáticamente o no). El sistema muestrea los hiperónimos e hipónimos entre la palabra clave con semántica y los términos de la ontología para incrementar las prestaciones del algoritmo de enlazado; el tamaño de la muestra es estimado utilizando la distribución normal [MR02].

3. *Creando las preguntas del usuario utilizando un lenguaje estructurado*: los agentes generan preguntas bien definidas utilizando los términos de las ontologías enlazadas con cada palabra clave con semántica. Para cada una de las preguntas generadas se calcula una probabilidad de que se corresponda con la pregunta que hizo el usuario, denominada probabilidad de sinonimia; esta probabilidad se calcula utilizando las probabilidades de las relaciones de enlazado utilizadas en el paso previo. Las preguntas con mayor probabilidad serán ejecutadas en primer lugar.

Gracias a la utilización de una arquitectura basada en agentes móviles las distintas acciones llevadas a cabo por los agentes, pueden ser ejecutados en un entorno distribuido (el usuario podría estar utilizando un dispositivo móvil para interrogar al sistema, y muchos servidores podrían estar almacenando las ontologías y/o los almacenes de datos). Por lo tanto, la utilización de una arquitectura basada en agentes móviles proporciona una fácil adaptación a contextos de ejecución heterogéneos, distribuidos e inalámbricos, robustez frente a desconexiones, y facilita la gestión de distintos tipos de conocimiento, para minimizar y balancear la sobrecarga de las distintas tareas realizadas por los agentes; independientemente del mecanismo de desambiguación utilizado, en nuestro caso nos hemos centrado en el mecanismo de traducción propuesto en [TGEM07]. A continuación, describimos la arquitectura del sistema (mostrada en la Figura 7.10):

- *El agente de usuario*: permite al usuario interrogar (formular preguntas) al sistema y crea el agente Desambiguador que procesará estas preguntas en la red fija. Además, automáticamente personaliza el contenido del GUI a las características del dispositivo (móvil) del usuario [MM02]. Por consiguiente, gracias a la utilización de una arquitectura basada en agentes móviles el resto de las tareas se llevará a cabo fuera del dispositivo del usuario, que tiene unas capacidades de ejecución limitadas, facilitando el ahorro de energía y de recursos en el dispositivo del usuario.

- *El agente móvil Desambiguador*: descubre la semántica de las palabras clave del usuario; en caso necesario interactuará con el usuario (a través del agente de usuario) para ayudarle a elegir entre los distintos significados que puede tener la palabra clave que introdujo al formular la pregunta. Hay un agente Desambiguador por cada conjunto de palabras clave del usuario porque la cantidad de información que analiza es enorme[8].

[8] El tamaño del tesauro utilizado en nuestro prototipo es de 40MB.

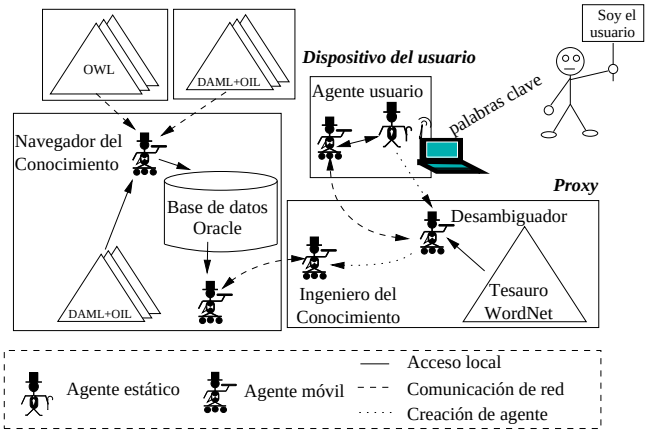


Figura 7.10: Arquitectura del sistema

Mediante la creación de distintos agentes desambiguadores se minimiza la sobrecarga del agente, en cambio en una arquitectura cliente/servidor no podría obtenerse este beneficio. Siempre se ejecuta en el ordenador (*proxy*) que proporciona cobertura al dispositivo móvil del usuario, viajando a un nuevo *proxy* cuando el usuario se mueve, minimizando de esta forma el tiempo de espera que tiene que sufrir el usuario cuando el agente de usuario se comunica con el agente Desambiguador. Del mismo modo, debido a la utilización de una arquitectura basada en agentes se facilita minimizar la utilización del enlace de red inalámbrico gracias a que el agente Desambiguador puede viajar al *proxy* más cercano al usuario.

- *El agente móvil Ingeniero del Conocimiento*[9]: es creado por el agente Desambiguador para descubrir las relaciones entre las palabras clave con semántica y los términos de las ontologías disponibles en el sistema. Además, puede moverse al place donde se encuentra almacenado el conjunto de ontologías analizado. De este modo el agente permite optimizar el acceso al depósito de datos, mediante la realización de accesos locales y no remotos. Finalmente, devolverá la lista de preguntas semánticas, ordenadas decrecientemente, según la probabilidad de sinonimia, respecto a la semántica de las palabras clave del usuario. Gracias a la utilización de una arquitectura basada en agentes inteligentes el proceso puede ser distribuido en distintos servidores evitando la sobrecarga, y gracias a la movilidad de los agentes se permite realizar un análisis local de los depósitos de datos que contienen las bases de conocimiento.

- *El agente móvil Buscador de Conocimiento* descubre los almacenes de ontologías disponibles y crea *wrappers* [HBGM+97, PGGMU95], módulos especializados para acceder a dichos depósitos de datos, para analizar los diferentes lenguajes utilizados en la especificación de las ontologías, como OWL [AvH03] o DAML+OIL [DAR06]. El

[9]Este agente Ingeniero del Conocimiento es distinto que el utilizado en el proceso de creación de SoftOnt en el SRS, sin embargo, su funcionalidad es la misma pero en un contexto con un vocabulario heterogéneo.

análisis de las distintas ontologías se realiza localmente si el agente Buscador de Conocimiento puede viajar donde se encuentra ubicada dicha ontología (característica imposible en arquitecturas de tipo cliente/servidor). Del mismo modo, el análisis puede ser realizado en un entorno distribuido y en paralelo debido a la utilización de agentes móviles (el agente Buscador de Conocimiento tiene la capacidad de clonarse a sí mismo). Las ontologías analizadas han sido almacenadas en una base de datos compartida por todos los agentes Ingenieros del Conocimiento del sistema. Por consiguiente, la búsqueda de fuentes de conocimiento distribuidas se realiza periódicamente e independientemente de las preguntas que son formuladas por el usuario y procesadas por el sistema.

La arquitectura propuesta es independiente del mecanismo utilizado para transformar un conjunto de palabras clave introducidas por el usuario en una pregunta expresada en OWL. Sin embargo, en la arquitectura propuesta gracias a la utilización de una arquitectura basada en agentes se facilita que los componentes (agentes) de la misma estén distribuidos en una red de ordenadores. Además, se permite que los agentes, automática y autónomamente, se adapten a la red de comunicaciones utilizada. Del mismo modo, gracias a la utilización de agentes inteligentes la robustez de la arquitectura propuesta y la facilidad de distribuir los distintos procesos en un conjunto de ordenadores se mantiene independientemente de la inteligencia utilizada por los agentes para extraer el significado de las palabras clave introducidas por el usuario.

7.4. Resumen del capítulo

Como se ha mostrado en este capítulo las arquitecturas basadas en agentes móviles inteligentes permiten una fácil adaptación a cambios en la cantidad (número de dispositivos móviles o fijos) y prestaciones (capacidades de visualización, almacenamiento y procesado de datos) de los recursos disponibles en cada momento. Del mismo modo, facilitan la creación dinámica, en tiempo de ejecución, de un servicio de acceso a datos y de la infraestructura necesaria para proporcionar al usuario cualquier funcionalidad que pueda proveerse mediante un software. A continuación resumiremos la ventajas obtenidas gracias a la utilización de una arquitectura basada en agentes para cada uno de los casos de uso presentados en este capítulo.

En primer lugar hemos presentado una arquitectura basada en agentes móviles para la generación de interfaces gráficos de usuario adaptativos y dinámicos en dispositivos móviles, siguiendo una aproximación indirecta que permite el estudio y análisis de las interacciones del usuario con el GUI. La utilización de agentes inteligentes tiene las siguientes ventajas:

- Facilita una adaptación automática a las características del dispositivo con el menor esfuerzo, debido a que se minimiza el consumo de recursos (procesador, red, disco, etc.).

- Permite una interacción local entre servicios remotos de datos y su GUI; en una arquitectura que no este basada en agentes móviles la comunicación entre el GUI y el servicio siempre necesitaría la red de comunicaciones.

La solución presentada para la generación de interfaces adaptativos es útil para la mayoría de los sistemas que pueden ser ejecutados en dispositivos con distintas características, tanto centralizados como distribuidos.

En segundo lugar, hemos detallado una arquitectura basada en agentes móviles que estima la localización de un dispositivo móvil en una red de área local inalámbrica, facilitando una infraestructura para el diseño y desarrollo de nuevos servicios. El uso de agentes en este contexto tiene las siguientes ventajas:

- Independencia del protocolo de comunicaciones utilizado: es decir, si se utilizase otro protocolo de comunicaciones únicamente se debería modificar el mecanismo utilizado para extraer la potencia de la señal, pero la inteligencia de los distintos agentes no debería modificarse.
- Reducción del software instalado en el dispositivo del usuario: este hecho se fundamenta en que los agentes móviles pueden trasladar su ejecución y proporcionar nuevos servicios al usuario en su dispositivo sin instalar nuevo software.

A continuación, hemos presentado una aproximación basada en agentes que permite mostrar las ventajas de éstos en el diseño de sistemas de información globales. Como hemos demostrado, las ventajas proporcionadas por los agentes se mantienen independientemente de los procesos seguidos para desambiguar las palabras clave introducidas por el usuario, de los mecanismos utilizados por el sistema para descubrir nuevos depósitos de datos y de la forma de acceso a los distintos depósitos de datos. Por ejemplo, podrían utilizarse otros mecanismos de desambiguación como los propuestos en [TGEM07, QHC06, Hes06, LDKG04, PBP05]. Debido a esto, hemos verificado que el uso de agentes en el contexto de los sistemas de información globales proporciona las siguientes ventajas:

- Facilitar la ejecución de las distintas tareas en una red de ordenadores. Gracias a la utilización del paradigma de agentes móviles cuando un agente solicita a otro la ejecución de un servicio, el proceso seguido (para el diseñador) es el mismo independientemente de que los dos agentes se encuentren ejecutándose en el mismo dispositivo o en dos conectados mediante una red de comunicaciones.
- Evitar cuellos de botella gracias a la capacidad de los agentes móviles pueda trasladar su ejecución al dispositivo más adecuado.

Debido a que las ventajas de una arquitectura basada en agentes se mantienen independientemente del proceso de desambiguación utilizado, la arquitectura mostrada en este capítulo han sido utilizadas posteriormente en aproximaciones como la presentada en [TGEM07].

Capítulo 8

Trabajos relacionados

En este capítulo de la tesis vamos a presentar los trabajos relacionados que hemos encontrado. Las distintas aproximaciones se dividen en: 1) comparativas entre la utilización de agentes móviles y llamada a procedimientos remotos, 2) trabajos que avalan la utilización de arquitecturas basadas en agentes inteligentes (al igual que el nuestro); y 3) trabajos relacionados con los distintos casos de estudio detallados a lo largo de la tesis. Resaltar, que en el capítulo 3 se presentaron de forma genérica las ventajas de la utilización de agentes inteligentes en entornos inalámbricos, que son avaladas por nuestra aproximación y por otros trabajos que se detallarán en este capítulo.

En cuanto a los trabajos relacionados vinculados con los contextos de ejecución referentes a los casos de estudio diseñados y desarrollados se detallarán en tres secciones. En primer lugar trataremos los trabajos relacionados con el Servicio de Recuperación de Software desde la siguientes perspectivas: 1) utilización de agentes en inteligencia ambiental, 2) utilización de agentes y ontologías para testear software, 3) utilización de agentes para la reutilización de componentes software, y 4) utilización de agentes y ontologías para buscar y descargar software.

En segundo lugar, nos centraremos en el contexto del uso de agentes inteligentes en bibliotecas digitales y describimos los trabajos relacionados siguiendo la siguiente estructura: 1) generación de bibliografía, 2) recuperación de información en bibliotecas digitales, y 3) comparación de publicaciones.

En tercer y último lugar, compararemos los trabajos relacionados con los servicios de datos en contextos dinámicos y las distintas aproximaciones seguidas en la tesis, centrándonos en los casos de estudio descritos: 1) generación de interfaces de usuario adaptativas, 2) diseño y desarrollo de servicios basados en la localización, y 3) enlazado de palabras clave y ontologías/depósitos de datos. Mostrando las ventajas proporcionadas por el diseño y desarrollo de arquitecturas basadas en agentes inteligentes frente a otras aproximaciones basadas en una arquitectura cliente/servidor.

8.1. Agentes software y sistemas multiagente

En esta subsección analizaremos distintas aproximaciones que comparan la utilización de agentes inteligentes y las arquitecturas cliente/servidor clásicas. Como veremos, las aproximaciones basadas en agentes tienen las siguientes ventajas: encapsulan el protocolo de comunicaciones, reducen el tráfico de red, disminuyen la latencia de red (por ejemplo, en procesos que deben analizar el estado de una aplicación distribuida), son asíncronos y autónomos, se adaptan dinámicamente al contexto, facilitan la integración de sistemas poco acoplados y son robustos frente a fallos de red.

A continuación describiremos distintos métodos que permiten almacenar la inteligencia de los agentes, como mostraremos la mayoría de los trabajos utilizan sistemas de representación del conocimiento basados en ontologías, método seguido también en los casos de estudio presentados en esta tesis.

8.1.1. Comparativas de arquitecturas basadas en agentes software y arquitecturas cliente/servidor

En [GKP+01] se presenta una comparativa entre distintas plataformas de agentes móviles y una aproximación basada en una arquitectura cliente servidor. En el desarrollo de los prototipos se ha utilizado D' Agents [Gra96, GCK+01, GDCR98], NOMADS [SBB+00b, SBB+00a, Sur06], EMAA [MCW00, CMMS00] y KAoS [BDBW97, BSC+01] como plataformas de agentes. Para la realización del prototipo utilizando una arquitectura cliente/servidor se ha utilizando RMI [Mic06b]. Las pruebas realizadas demuestran que los agentes son beneficiosos en entornos con redes lentas e inestables. Sin embargo, en esta aproximación hay que considerar que los agentes desarrollados no poseen inteligencia, sino que únicamente realizan un transporte de datos a través de la red. Por consiguiente, si se añadiese inteligencia a los agentes desarrollados en esta aproximación, los resultados serían más prometedores, tal y como hemos demostrado mediante los casos de estudio presentados en esta tesis.

Del mismo modo, en [SSPE04a] se compara la utilización de arquitecturas basadas en agentes móviles en contextos inalámbricos y cableados (redes de 100Mbits y estables). La aproximación mostrada en dicho artículo verifica algunas de las ventajas proporcionadas por los agentes móviles, como ser robustos frente a desconexiones y permitir el ahorro de comunicaciones. Además, muestra que las arquitecturas basadas en agentes móviles inteligentes facilitan la modificación de las arquitecturas cliente/servidor clásicas, para ayudar en su adaptación a entornos inalámbricos (como los casos de estudio presentados en esta tesis). También, se muestra un ejemplo en el que se accede a un sistema de bases de datos distribuidas y federadas, permitiendo la ejecución paralela de un conjunto de preguntas, permitiendo reducir el tráfico de red inalámbrico mediante un proceso de coordinación entre agentes.

También se han desarrollado estudios teóricos, como el presentado en [SVZP02], que comparan las arquitecturas cliente/servidor y las arquitecturas basadas en agentes móviles. Como modelo de representación de la arquitectura han utilizado Redes de Petri estocásticas generalizadas [MCB84, MBC+95], llegando a la conclusión siguiente: una arquitectura es mejor que la otra dependiendo de las condiciones del contexto de ejecución. Si embargo, el modelo realizado presenta la carencia de que su aproximación no tiene en cuenta factores

como los frecuentes fallos de red que se producen en redes inalámbricas, esta característica es demostrada de forma empírica en nuestro trabajo [MRIG02a]. También hay que considerar que los agentes móviles permiten la reducción de la sobrecarga en los dispositivos móviles debido a que permiten trasladar la ejecución a los ordenadores ubicados en la red fija, con mayores recursos y menores limitaciones de energía.

En [Ade04] se presenta un estudio teórico de las ventajas de las arquitecturas basadas en el paradigma de agentes móviles frente a aproximaciones más clásicas como son las arquitecturas basadas en llamadas a procedimientos remotos. En este caso se modela un aspecto muy importante de la red, la probabilidad de desconexión (característica considerada en los casos de estudio presentados en esta tesis). Los autores modelan la probabilidad de los fallos de red (propiedad no considerada en [SVZP02]) mediante una distribución de una variable aleatoria de Bernoulli con probabilidad de que la conexión se produzca con éxito igual al 80 % y de fallo del 20 %. Tal y como se muestra en las conclusiones, las ventajas que proporcionan las arquitecturas basadas en agentes móviles son superiores a las proporcionadas por las arquitecturas cliente/servidor. Este es un trabajo similar al presentado en esta tesis ya que trata de demostrar las ventajas de las aproximaciones basadas en la tecnología de agentes móviles inteligentes, aunque su trabajo es solamente teórico. Por consiguiente, la diferencia con el trabajo presentado en esta tesis es que nosotros demostramos las ventajas proporcionadas por las arquitecturas basadas en agentes móviles, tanto de forma analítica como empírica.

En esta sección hemos presentado distintos trabajos relacionados que avalan en mayor o menor grado las ventajas debidas a la utilización de agentes inteligentes. A continuación, vamos a describir las distintas aproximaciones utilizadas para almacenar el conocimiento gestionado por los agentes, que les permite ser más eficientes que otras aproximaciones.

8.1.2. Gestión del conocimiento en Sistemas Multiagente

Existen distintos trabajos que abogan por la representación del conocimiento de los agentes mediante la utilización de ontologías [Hen01, SDO00, TCFW05]. Sin embargo, no solamente es importante decidir cómo representar el conocimiento, sino que aun es más importante decidir cómo puede compartirse con otros agentes. Permitiendo de esta forma la cooperación de varios agentes inteligentes para la realización de una tarea.

En [LTE+06] se presenta un proceso que permite coordinar dos agentes independientemente del vocabulario utilizado por cada uno de ellos. No se centran en la definición del protocolo de comunicaciones entre los agentes, como se hace en [fIPA98, FfIPA05, PBH00], sí que tratan de definir un mecanismo para definir el significado de los mensajes intercambiados entre los agentes de un sistema multiagente. La base de este proceso de estandarización es que no todos los agentes utilizan el mismo vocabulario u ontología, dificultando el proceso de comunicación entre agentes. Al igual que en los casos de estudio diseñados en la tesis, la inteligencia de los agentes se encuentra representada en una ontología. Sin embargo, en su aproximación añaden un mecanismo que permite coordinar agentes que utilizan distinto vocabulario enlazando los parámetros del agente que desea invocar un servicio y la especificación del servicio que posee otro agente. Hasta la realización de este trabajo, se produciría siempre o nunca, el enlazado entre los términos de las ontologías que almacenan el conocimiento de los dos agentes. Sin embargo, en [LTE+06] proponen un mecanismo de enlazado

de términos que depende de las preferencias o restricciones que impondrá cada uno de los agentes que intervienen en el proceso de coordinación. Por consiguiente, mientras que nosotros utilizamos las ontologías para gestionar el conocimiento que posee un agente o conjunto de agentes, en su trabajo las ontologías son utilizadas para gestionar el significado de los mensajes intercambiados por los agentes. Sin embargo, ambas aproximaciones abogan por representar el conocimiento manejado por los agentes mediante ontologías.

8.2. Servicios de recuperación de software basados en agentes y ontologías

Según nuestra información, no hay otro trabajo que utilice agentes y ontologías, en un entorno inalámbrico, para ayudar a los usuarios inexpertos de dispositivos móviles en la búsqueda, descarga, e instalación de nuevo software. Por lo tanto, en esta subsección describimos trabajos con distintos objetivos pero relacionados con el nuestro de alguna forma. Es decir, relacionaremos distintas aproximaciones para testear, reutilizar, buscar y recuperar software con la aproximación propuesta en esta tesis.

8.2.1. Uso de agentes/ontologías para testear software

En [HZG03, Zhu04] un entorno basado en agentes software hace uso de una ontología para probar aplicaciones basadas en la web. En su trabajo proponen una arquitectura basada en la tecnología de agentes para construir aplicaciones web asegurando la calidad y proporcionando un entorno de pruebas.

La arquitectura propuesta por los autores consiste en dos tipos de agentes.

- *Agentes de servicio*: que dan soporte a sistemas software desarrollados mediante una estrategia evolutiva. Cumplen los requisitos funcionales para asegurar el desarrollo y la calidad del software, permitiendo la verificación y validación de las funcionalidades.
- *Agentes gestores*: permiten gestionar a los agentes de servicio y se responsabilizan del registro de las funcionalidades de los agentes, de la ordenación de tareas, y del análisis y almacenamiento del estado y comportamiento de los sistemas.

Cada agente de servicio está especializado en realizar una tarea específica y cooperará con otros agentes en la ejecución de un servicio complejo. Estos agentes coexisten con la evolución de las aplicaciones web para analizar las modificaciones realizadas a dichas aplicaciones, para extraer, almacenar y procesar toda la información de las trazas de ejecución del sistema. Estos agentes poseen una ontología que les permite razonar acerca de las funcionalidades del software que debe probarse.

Al igual que en nuestra aproximación abogan por una arquitectura basada en agentes inteligentes. Las diferencias principales con respecto a nuestra aproximación son:

1. Proponen una ontología *ad hoc* mientras que los agentes del servicio de recuperación de software la construyen automáticamente a partir de categorías preexistentes de software.

2. Su ontología se define en XML y definen un número *add-hoc* de relaciones entre los términos básicos de la ontología junto con un mecanismo para razonar con esas relaciones. Sin embargo la ontología utilizada en el SRS se define en OWL y los agentes que la utilizan se benefician de los mecanismos de razonamiento definidos en OWL.

3. Aunque en ambas aproximaciones se identifica un agente *broker* (identificado con nuestro agente Navegador), su objetivo es diferente.

También hay que mencionar que nuestra aproximación para analizar y, posteriormente, reutilizar la información generada por las acciones del usuario, es similar a la descrita en [HZG03, Zhu04]. Sin embargo, la información que proporciona nuestra aproximación es más extensa, debido a que no solamente proporcionamos información acerca de si el software funciona correctamente y con qué grado de prestaciones, sino que además almacenamos información acerca de los parámetros de los distintos servicios ejecutados por el usuario. Posteriormente, esta información es reutilizada por los agentes para ayudar al usuario en la ejecución de dichos servicios, por ejemplo sugiriendo opciones en la ejecución de los distintos servicios.

8.2.2. Uso de agentes/ontologías en la reutilización de componentes

El objetivo de ORES [YKP+01, KL02] (depósito on-line de software empotrado) es facilitar la gestión y recuperación de componentes. Algunas semejanzas entre su trabajo y nuestra aproximación son: 1) el uso de una ontología para organizar elementos software (aunque el proceso para construir la ontología difiere substancialmente), 2) el desarrollo de un algoritmo de poda para dirigir al usuario hacia los resultados deseados, y 3) el uso de palabras claves del usuario para formular sus peticiones. No obstante entre las diferencias principales podemos mencionar que no consideran un entorno de inteligencia ambiental o computación pervasiva[1] ni el uso de la tecnología de agentes como lo hacemos nosotros, aunque proporcionan métodos para la construcción de un tutorial multimedia para los componentes.

El objetivo de ODYSSEY [RBM00] es proporcionar ayuda en el desarrollo y reutilización de componentes en todas las fases de construcción del software. Nuestras aproximaciones coinciden en el uso de módulos especializados que utilizan la información sobre las preferencias del usuario, históricos de búsquedas, caminos de navegación y palabras clave más frecuentes para refinar la búsqueda. Sin embargo, una diferencia sustancial se refiere a la ontología, es decir, abogan por varios mediadores, cada uno asociado a un ontología dependiente del dominio, y el usuario selecciona uno de ellos para trabajar con él. En cambio, los agentes diseñados en nuestro caso de estudio gestionan una ontología global que se relaciona con varios depósitos preexistentes de software; donde los distintos agentes que constituyen la arquitectura colaboran para mostrar al usuario solamente el subconjunto de la ontología que es interesante para él.

Al igual que en nuestra aproximación, AGORA [SHW98] utiliza varios agentes. En el SRS los agentes ayudan al usuario en la búsqueda, descarga e instalación de software y en AGORA facilitan la localización e indexación de la información asociada a distintos componentes. Sin embargo, para buscar los componentes, el usuario debe saber un conjunto de

[1] *Pervasive computing.*

palabras clave y debe proporcionar los nombres de los métodos, características, etc, de los componentes en los que él está interesado. Es decir, no hay una ontología que contenga los nombres de los métodos, de las características, etc, que el usuario pueda navegar. Permitiendo a los agentes que constituyen la arquitectura ayudar en el proceso de búsqueda, tal y como sucede en nuestra aproximación.

A continuación detallaremos cada una de estas aproximaciones y las compararemos con el Servicio de Recuperación de Software (SRS) propuesto en la tesis; mostrando así las ventajas proporcionadas por los sistemas multiagente en el diseño y desarrollo de servicios de acceso a datos en entornos inalámbricos.

ORES: Online Repository for Embedded Software

El objetivo de ORES [YKP+01] (depósito on-line de software empotrado) es facilitar la gestión y recuperación de componentes. Algunas semejanzas entre su trabajo y nuestra aproximación son:

1. *Utilización de una ontología para organizar elementos software*: Aunque el proceso para construir la ontología difiere substancialmente. En su caso solamente han probado el proceso de construcción automática [KL02] de la ontología en un contexto general y no en el ámbito del software, esta aproximación basada en un dominio de aplicación concreto lo proponen como trabajo futuro. En su caso utilizan técnicas de recuperación de información [BYRN99] para pasar a una representación canónica de las palabras utilizadas y generar una jerarquía de documentos agrupados según temas. Para agrupar los documentos se basan en la aparición de distintas palabras en los textos, a continuación asocian a cada uno de los subconjuntos generados un término que los identifica. Para realizar esta tarea utilizan WordNet como tesauro, igual que en nuestra aproximación. La diferencia con nuestra aproximación radica en que nuestros agentes utilizan WordNet para realizar la integración de las distintas ontologías construidas automáticamente para cada uno de los distintos depósitos de datos.

2. *Desarrollo de un algoritmo de poda para dirigir al usuario hacia los resultados deseados*: En ambos casos se han desarrollado algoritmos de poda para dirigir al usuario. En nuestro caso se han desarrollado varios que son seleccionados automáticamente, por los agentes que poseen capacidades de poda, dependiendo del comportamiento del usuario. Por lo tanto, los agentes diseñados se adaptarán mejor al usuario debido a que se analizan las interacciones del usuario con el GUI. Posteriormente, los agentes extraerán conocimiento de esta información y podrán utilizarlo en posteriores ejecuciones, o incluso en acciones posteriores de la misma ejecución del servicio. En ambos casos antes de que los agentes apliquen este mecanismo de poda se puede realizar un filtrado (poda) por palabras clave para reducir el número de categorías y de componentes software que se muestran al usuario.

3. *Uso de palabras claves del usuario para formular sus peticiones*: Una de las ventajas de nuestra aproximación es que parte de las palabras clave utilizadas por el usuario son añadidas por el agente mayordomo; por ejemplo, el usuario puede indicar que el sistema operativo es windows y Alfredo modificará dicha información para indicar

que el servicio está ejecutándose en una máquina con sistema operativo Windows XP. Además, nuestro agente mayordomo podrá sugerir al usuario distintos tipos de software que hayan sido instalados en el sistema con anterioridad pero de los que existan actualizaciones.

No obstante entre las diferencias principales podemos mencionar que no consideran el uso de la tecnología de agentes móviles inteligentes ni un entorno de computación pervasiva como lo hacemos nosotros, aunque proporcionan métodos para la construcción de un tutorial multimedia, permitiendo a los usuarios verificar cómo utilizar los componentes software que hay disponibles.

Otra de las diferencias fundamentales radica en la ontología utilizada, en su caso mantienen distintas versiones de la ontología que almacena los componentes que son integradas en tiempo de ejecución, con el coste implicado por dicho proceso. En cambio en el SRS, los agentes diseñados mantienen una versión integrada de las ontologías creadas para cada uno de los depósitos de software subyacentes que es actualizada periódicamente de forma automática, permitiendo de esta forma una reducción del coste de visualización y de generación de la visión unificada del conjunto de almacenes de software que es utilizado por los distintos agentes que constituyen la arquitectura.

ODYSSEY

El objetivo de ODYSSEY [BWM99, RBM00] es proporcionar ayuda en el desarrollo y reutilización de componentes en todas las fases de construcción del software. Nuestras aproximaciones coinciden en el uso de módulos especializados que utilizan la información sobre las preferencias del usuario, históricos de búsquedas, caminos de navegación y palabras clave más frecuentemente utilizadas para refinar la búsqueda. Sin embargo, gracias a que el SRS se basada en la tecnología de agentes móviles inteligentes puede adaptarse fácilmente a contextos inestables y con velocidades de transmisión limitadas, como son los entornos inalámbricos.

Los principales usuarios de ODYSSEY son ingenieros del software, especialistas en el domino de aplicación a desarrollar e ingenieros especialistas en el dominio de aplicación. Por lo tanto, ODYSSEY no necesita ayudarles a encontrar el componente que desean, sino que únicamente debe proporcionales las herramientas necesarias para que puedan buscarlo y posteriormente utilizarlo. Sin embargo, los agentes que integran el SRS deben ayudar al usuario a encontrar el software que desea, debido a que en algunas ocasiones éste únicamente dispone de una descripción de la tarea que quiere realizar. Por ejemplo, necesito un software para diseñar muebles de cocina, en este caso los agentes que constituyen el sistema intuyen que está buscando una herramienta CAD, pero el usuario no sabe como se denomina el software que está buscando.

Debido a esto en ODYSSEY se almacena información referente a modelos conceptuales, casos de uso, características de modelos, contextos de ejecución, sistemas de análisis de patrones, etc. En cambio en el SRS únicamente se dispone de piezas de software que pueden ser instaladas en el dispositivo móvil del usuario y ser utilizadas para realizar una tarea. Es decir, el usuario en la mayoría de los casos no buscará componentes software que le ayuden en el desarrollo de una aplicación, sino una aplicación completa.

Otra diferencia sustancial se refiere a la ontología, mientras que en ODYSSEY se aboga por varios mediadores, cada uno asociado a una ontología dependiente del dominio, y el usuario selecciona uno de ellos para trabajar con él. En el SRS, los agentes diseñados usan una ontología global que se relaciona con varios depósitos preexistentes de software y muestran al usuario solamente el subconjunto de la ontología que es interesante para él.

Finalmente, hay que destacar que ODYSSEY [dSCB+01] también incorpora un sistema que permite la búsqueda y filtrado de componentes software disponibles en la web, sin embargo la búsqueda de este tipo de componentes no se fundamenta en la utilización de una ontología, sino en una búsqueda sintáctica realizada por el motor de búsqueda Google. Por lo tanto, una de las ventajas de nuestra aproximación es que los agentes que constituyen el SRS analizan la web y crean una ontología automáticamente. Por lo tanto, ODYSSEY solamente permite búsquedas semánticas en sus depósitos de componentes software y búsquedas sintácticas en los depósitos de componentes disponibles en la web. Sin embargo, los agentes integrados en el SRS permiten la búsqueda semántica de piezas de software, tanto en depósitos de software desarrollados ad hoc como en depósitos de software disponibles en la web (mediante la creación automática de una ontología asociada a cada uno de los sitios web analizados, posteriormente las distintas ontologías asociadas a cada uno de los depósitos serán integradas por el agente Integrador).

AGORA

El sistema Agora [SHW98] describe un motor de búsqueda para la descarga de componentes software reutilizables, como JavaBeans o componentes CORBA. Agora utiliza un mecanismo de introspección para registrar los componentes software mediante su interfaz. Como resultado, esta información puede no estar disponible en un cierto momento del tiempo porque el almacén de datos no esté accesible, o la información no pueda ser localizada. En cualquiera de estos casos, la información del API no puede ser correctamente devuelta e indexada y por lo tanto el componente no podrá ser registrado en la base de datos del sistema AGORA. Sin embargo, en nuestro sistema debido a que la ontología de software se encuentra almacenada en el proxy, el agente gestor de software siempre podrá acceder a la misma y el usuario podrá realizar diferentes búsquedas de software. Sin embargo, solamente podrá descargar el software de los distintos sitios web que estén activos en dicho momento. Para poder realizar la descarga del software en todo momento, únicamente habría que cachear todas las piezas de software en el proxy, lo cual únicamente supondría tener suficiente espacio en disco para almacenar todos los depósitos de software. Actualmente, esto no supondría un gran coste debido a que los dispositivos de almacenamiento masivo son muy baratos.

También hay que destacar que el sistema AGORA, solamente indexa applets y componentes CORBA. En el caso de la indexación de applets, su problema radica en que solamente indexa los applets que se han indexado en el motor de búsqueda altavista [OS06], es decir, no posee un motor de búsqueda propio. Del mismo modo, en el caso de la búsqueda e indexación de componentes CORBA, el sistema AGORA tiene el problema de que los usuarios de CORBA no utilizan (habitualmente) un servidor de nombres para registrar los interfaces de los componentes utilizados. Otra de las diferencias de esta aproximación y la propuesta en el SRS es que en nuestro caso se facilita la búsqueda de piezas de software que el usuario puede instalar y no solamente componentes software.

Aunque ambos presenten las diferencias mostradas anteriormente coinciden en que están diseñados mediante una arquitectura basada en agentes; aunque en AGORA [SHW98] el usuario para buscar los componentes, debe saber un conjunto de palabras clave y debe proporcionar los nombres de los métodos, características, etc, de los componentes en los que él está interesado. Además, en AGORA no hay una ontología que contenga los nombres de los métodos, de las características, etc, que el usuario pueda navegar. Por lo tanto, la búsqueda de componentes se reducirá a una búsqueda sintáctica, en nuestro caso las búsquedas del usuario tienen un contenido semántico.

8.2.3. Uso de agentes/ontologías en la búsqueda y recuperación de software

En [IBM06b] explican un mecanismo para la actualización de software en varios clientes remotos conectados con un servidor que aprovecha la capacidad de los agentes móviles para trabajar con desconexiones. Sin embargo, este trabajo se relaciona más con *la tecnología push* que con los servicios creados para asistir a usuarios en la tarea de actualizar el software de sus dispositivos. Mediante JavaWebStart [Sun06] se permite que los usuarios instalen y actualicen software fácilmente; en esta aproximación el usuario tiene que hacer click en un enlace en su navegador web y después la aplicación será descargada e instalada. Sin embargo, Java Web Start no considera los problemas de las redes inalámbricas y ni unos ni otros consideran ayudar a los usuarios a buscar software como en nuestra aproximación. En [DIH00] se centran en la tarea de buscar componentes en grandes, complejas y continuamente crecientes bibliotecas de software. Introducen el concepto de "navegación activa"; donde un agente activo intenta deducir el objetivo de la búsqueda y devuelve los componentes que concuerdan con ese objetivo. En [YF01], se centran en la tarea de buscar componentes software y presentan el concepto del "contexto conocido" facilitando la navegación usando técnicas de establecimiento de correspondencias entre datos. Para decidir qué componentes son relevantes para el usuario, hacen uso de una medida de semejanza. La idea de estas aproximaciones es diferente a la nuestra: intentan ayudar a los desarrolladores de software para anticiparse a sus necesidades (mostrándoles componentes software que pueden ser interesantes para ellos). En nuestro caso, el objetivo de la aproximación seguida permite mostrar las ventajas del diseño de una arquitectura basada en agentes inteligentes en la búsqueda y recuperación de software, ayudando al usuario en dicho proceso y reduciendo el coste de transmisión de datos a través del enlace de comunicaciones inalámbrico.

En resumen, la mayoría de los trabajos relacionados se centran en buscar, indexar y navegar componentes de software reutilizables, aunque en la mayor parte de los trabajos no se considera un entorno inestable e inalámbrico. Debido a esto, las distintas aproximaciones mostradas no consideran la utilización de agentes inteligentes. Además, las distintas aproximaciones mostradas intentan ayudar a los desarrolladores de software que podrían ser expertos en software. El objetivo de nuestra aproximación es diferente, porque los distintos agentes que constituyen la arquitectura ayudan a los usuarios (experimentados o no) en la búsqueda de aplicaciones software. Sin embargo, si una ontología de componentes software fuese definida entonces podría ser integrada con nuestra ontología y esta nueva ontología se podría utilizar para encontrar diversas clases de software (incluyendo componentes software) en un

nivel semántico, es decir, sin la necesidad de saber los detalles técnicos acerca de qué es lo que está buscando el usuario. Por consiguiente, la inteligencia utilizada por los distintos agentes que constituyen la arquitectura del SRS puede ser reutilizada independientemente del dominio en el que se encuadre el proceso de búsqueda.

Distribución de software utilizando agentes móviles

En [IBM06b] explican un mecanismo para la actualización de software en varios clientes remotos conectados con un servidor que aprovecha la capacidad de los agentes móviles para trabajar con desconexiones; destacando las ventajas de la utilización de los agentes para la distribución de software, al igual que en el SRS.

Los autores de este trabajo describen un mecanismo que trata de solucionar los retos debidos a tener que distribuir software a usuarios móviles. Los usuarios móviles desean trabajar en cualquier momento y lugar, pero se añade un gran número de dificultades en la gestión de la distribución de software de una forma eficiente y poco costosa en tiempo:

- Los usuarios de ordenadores móviles (dispositivos móviles) viajan y se desconectan de la red por definición en momentos inesperados. Además, hay que considerar que estos usuarios pueden seguir desconectados durante mucho tiempo.
- Los dispositivos móviles se conectan a la red en distintos lugares.
- Las redes de comunicaciones no siempre son las mismas, el usuario se conecta utilizando un teléfono móvil, una red WiFi o una red de área local cableada.

En la plataforma también consideran la distribución de software mediante la utilización de tres métodos distintos:

- *Normal*: el software se almacena en un proxy hasta que el usuario se lo descarga al dispositivo móvil y lo pone en ejecución. Si expira la fecha de descarga, el software es borrado del proxy y se hace imposible su descarga. Este es el método por defecto.
- *Obligatoria*: ee descarga e instala automáticamente el software, el usuario puede seleccionar o no la instalación de software obligatorio.
- *Oculta*: la descarga y ejecución del software se hace automáticamente sin intervención del usuario. Además, no se muestra ningún mensaje acerca de la instalación del software por pantalla una vez finalizado el proceso de instalación.

Sin embargo, este trabajo se relaciona más con *la tecnología push* que con los servicios creados para asistir a usuarios en la tarea de actualizar el software de sus dispositivos. Es decir, su aproximación está relacionada con la instalación y actualización de software en entornos corporativos. Aunque, su arquitectura no proporciona ayuda al usuario en las tareas de búsqueda de un software que satisfaga sus necesidades y tampoco considera la optimización de las comunicaciones, excepto las ventajas inherentemente proporcionadas por la utilización de la tecnología de agentes móviles. Sin embargo, ambas aproximación tienen en común que están diseñadas utilizando una arquitectura basada en agentes y obtienen beneficios de ello, por ejemplo son robustas frente a desconexiones y minimizan el tráfico de red inalámbrico.

Java Web Start

Java Web Start [Sun06] proporciona un mecanismo lo suficientemente potente para ejecutar aplicaciones Java sin las restricciones impuestas a los applets [SM06], como por ejemplo una hoja de cálculo o un cliente de chat, sin tener que realizar un complicado proceso de instalación. El acceso a Java Web Start se realiza mediante su integración dentro del escritorio del usuario; desde el punto de vista del usuario funcionará como una aplicación nativa. Adicionalmente, Java Web Start sirve a los usuarios para gestionar las actualizaciones de la versión de la máquina virtual instalada en su dispositivo.

Desde el punto de vista de los desarrolladores, el diseño de aplicaciones basadas en Java Web Start se simplifica, debido a que el desarrollo de las aplicaciones es sustancialmente el mismo que implementar una aplicación Java, y la migración de una aplicación legada a Java Web Start es, en muchos casos, trivial. Además, la utilización de esta tecnología permite que los usuarios puedan acceder a la misma mediante una página web que permite su instalación automática.

El desarrollo de aplicaciones basadas en Java Web Start es como el de cualquier aplicación basada en Java pero con las siguientes restricciones:

- Debe estar empaquetada en uno o más ficheros JAR, además la aplicación no podrá acceder a clases, ficheros de propiedades o imágenes fuera del fichero JAR.
- Puede ser modificada para que se ejecute en un entorno seguro, tal y como lo hacen los applets.
- Utilizará el protocolo JNLP (*Java Network Launching Protocol*) [FDL09], permitiendo determinar si la aplicación se ejecuta on-line o no; si interactúa con el escritorio del usuario o trabaja con ficheros del ordenador del usuario.

Hay que tener en cuenta que el desarrollo de una aplicación basada en una arquitectura Java Web Start implica los siguientes pasos: 1) configurar el servidor web, para que acepte el tipo MIME "*application/x-java-jnlp-file*"; 2) crear el fichero jnlp, es un fichero basado en XML que contiene elementos y atributos que indican a Java Web Start cómo ejecutar la aplicación; 3) copiar la aplicación al servidor web, tanto los ficheros JAR como el fichero jnlp; y 4) crear la página web que permite acceder a la aplicación, mediante una etiqueta que cree un enlace al fichero JNLP de la aplicación (* Ejecutar aplicación Java Web Start*).

Asumiendo que Java Web Start está instalado en el ordenador cliente, cuando el usuario hace click en el enlace: 1) se chequea el ordenador del usuario para ver si está instalada la versión adecuada de la máquina virtual, 2) se añade la aplicación al ordenador para que pueda ser ejecutada de forma local y 3) se ejecuta la aplicación. Para los usuarios que no tienen instalado Java Web Start, se puede modificar la página web para que se le pregunte si quiere instalar Java Web Start antes de instalar la aplicación. Es decir, proporciona una funcionalidad similar a los agentes móviles, debido a que la aplicación o agente viaja por la red para dar un servicio al usuario.

Sin embargo, JavaWebStart no considera los problemas de las redes inalámbricas y tampoco ayudar al usuario en la búsqueda del software que necesita para realizar una tarea determinada. Java Web Star únicamente ayuda a instalar un nuevo software, pero el usuario tiene

que saber en qué página web se encuentra. Sin embargo, el SRS, gracias a la utilización de una arquitectura basada en agentes permite minimizar las comunicaciones inalámbricas y que los distintos agentes que constituyen la arquitectura ayuden a los usuarios en el proceso de búsqueda, descarga e instalación de nuevo software.

Agentes software que ayudan en la búsqueda de librerías de software

En [DIH00] se centran en la tarea de buscar componentes en grandes, complejas y continuamente crecientes bibliotecas de software. Para lo cual se ha desarrollado un buscador que analiza el comportamiento del usuario y las anteriores ejecuciones del servicio, permitiendo mostrar al usuario de forma activa los componentes software (librerías software) que tienen más posibilidades de ser utilizadas.

El motor de inferencia desarrollado en [DIH00] se compone de dos partes:

1. *El sistema de inferencia*: que permite la conversión de las acciones llevadas a cabo por el usuario en un conjunto de reglas de lógica de predicados con las que podrá razonar el sistema. Las reglas son utilizadas por el sistema para inferir comportamientos análogos del usuario. Por ejemplo, que impliquen a la misma clase de objetos que está utilizando el usuario. Además, el sistema también incluirá sus propias conclusiones entre las plantillas que pueden ser seleccionadas por el usuario. El resultado de este proceso es un conjunto de componentes o librerías que pueden ser utilizadas por el usuario en un futuro próximo.

2. *El enlazador de plantillas*: se encarga de ordenar las posibles librerías o componentes que han sido obtenidos por el sistema de inferencia, según la probabilidad de ser usados que tengan.

El sistema de inferencia se basa en un sistema de reglas tradicional que permite:

- Encadenamiento de reglas hacia delante mediante conjunciones, disyunciones, negaciones y factores de confianza.
- La utilización de una memoria de trabajo que contiene hechos que representan las acciones de búsqueda del usuario y plantillas.
- La definición de una regla base que contenga otras reglas (para permitir el razonamiento).

Por lo tanto el trabajo presentado en [DIH00], introduce el concepto de “navegación activa”. donde un agente activo intenta deducir el objetivo de la búsqueda y devuelve los componentes que concuerdan con ese objetivo. Al igual que nosotros, esta aproximación trata de ayudar al usuario en un proceso de búsqueda, aunque se enmarca en un contexto diferente. Sin embargo, nosotros hemos diseñado una aproximación basada en agentes móviles que además de ayudar a usuarios inexpertos en el proceso de búsqueda de una pieza de software (no un componente o una librería) también minimizan la utilización de las comunicaciones inalámbricas, gracias a la utilización de una arquitectura basada en agentes inteligentes. Por otra parte, a diferencia de nuestra aproximación en la que la gestión del conocimiento e

inteligencia utilizado por los agentes es representado mediante Lógica Descriptiva, su aproximación utiliza un sistema de inferencia basado en reglas. Del mismo modo, su aproximación tampoco considera cuestiones de tráfico de red o incrementar automáticamente la inteligencia de los componentes que constituyen la arquitectura. Sin embargo, en nuestro caso los agentes aprenden de las decisiones tomadas para mejorar su comportamiento en posteriores ejecuciones del servicio.

Búsqueda dependiente del contexto en depósitos de software de gran tamaño

En [YF01], se centran en la tarea de buscar componentes software y presentan el concepto del "contexto conocido" al navegar usando técnicas que permiten establecer relaciones entre la información de los componentes software. Para decidir qué componentes son relevantes para el usuario, hacen uso de una medida de semejanza de la funcionalidad de los componentes disponibles en un depósito de componentes software y los componentes que ha desarrollado y está desarrollando un ingeniero del software.

Su arquitectura se basa en presentar de forma activa distintos componentes que pueden estar siendo buscados por el desarrollador. Las diferencias entre una búsqueda de un componente, la visualización de componentes software y la visualización de componentes software dependiente del contexto se muestra en la Tabla 8.1. Cuando una aplicación permite realizar búsquedas de componentes software, el propio desarrollador realiza las mismas, por lo que la aplicación no necesariamente tendrá una inteligencia que ayude en la realización de las búsquedas. En el caso de que la aplicación visualice los componentes software tratando de ayudar al usuario, la inteligencia de la aplicación será mayor pero dependerá de un proceso guiado por el usuario. Sin embargo, si la ayuda proporcionada por la aplicación depende del contexto, el sistema deberá ser capaz de estudiar el comportamiento del usuario para recomendarle los componentes que mejor se ajusten a la tarea que está tratando de realizar.

	Ventajas	**Desventajas**
búsqueda	rápida y directa	difícil formular la pregunta correcta, no se realizan búsquedas de forma anticipada
visualización	sobrecarga baja debido a la inteligencia del sistema	no es fácilmente escalable
visualización dependiente del contexto	da soporte al establecimiento automático de relaciones entre componentes	el estudio del contexto es difícil

Tabla 8.1: Comparación de los mecanismos para localizar componentes

En [YF01] se describe un agente software, denominado *CodeBroker*, que permite localizar componentes software. Esta característica se consigue mediante la ejecución como tarea de fondo del agente, que mediante su ejecución dentro del entorno de desarrollo puede estudiar y analizar los componentes que necesita el desarrollador. Mediante el estudio del comportamiento del usuario (mediante técnicas ad hoc) el sistema propondrá al desarrollador: componentes que tengan una funcionalidad que coincida con el contexto del componente

que se está desarrollando actualmente. Para realizar una selección de los componentes que puede necesitar el desarrollador, el agente se basará en el contexto actual y en las ejecuciones pasadas del servicio; del mismo modo que en la arquitectura propuesta en esta tesis para la monitorización del comportamiento y la generación de interfaces adaptativos. La ventaja de nuestra aproximación es que no se utilizan técnicas ad hoc que aunque pueden ser más efectivas para un contexto determinado, no pueden aplicarse de forma generalizada.

El objetivo principal de la aproximación presentada en [YF01] es diferente al nuestro debido a que: 1) intentan ayudar a los desarrolladores de software para anticiparse a sus necesidades (mostrándoles componentes software que pueden ser interesantes para ellos); 2) utiliza un método ad hoc para seleccionar los componentes y analizar el comportamiento del usuario; 3) no realizan un proceso de búsqueda de software basado en una ontología (no se pueden realizar búsquedas semánticas); 4) se trabaja con usuarios experimentados; y 5) no permite la generación de interfaces adaptativos. Por consiguiente aunque la aproximación seguida por [YF01] muestra las ventajas obtenidas debidas a la utilización de agentes inteligentes con usuarios expertos, nuestra aproximación verifica las ventajas de la utilización de agentes móviles inteligentes con usuarios inexpertos o no; además de mostrar las ventajas proporcionadas por las arquitecturas basadas en agentes en el diseño y desarrollo de aplicaciones en contextos inalámbricos.

8.2.4. Agentes que acceden a ontologías en computación pervasiva

En [CFJ03, CFJ04] presentan las características de la ontología COBRA-ONT descrita en OWL, que modela los conceptos básicos de agente, agentes, etc. en ambientes de reunión. Nuestro trabajo coincide con esta aproximación en que el razonamiento con la ontología es realizado en un elemento inmóvil y después desplazado a los dispositivos inalámbricos. En [RMCM03a, RMCM03b], muestran cómo el uso de ontologías puede ayudar a superar tres problemas importantes en el desarrollo y despliegue de servicios en entornos de inteligencia ambiental (descubrimiento e interoperabilidad entre diversas entidades y cuyo conocimiento depende del contexto). Estamos de acuerdo con la mayoría de sus conclusiones, por otra parte nuestro trabajo se enfoca en un dominio específico de aplicación.

CoBrA: Context Broker Architecture

En [CFJ03, CFJ04] presentan las características de la ontología COBRA-ONT descrita en OWL, que modela los conceptos básicos de agente, agentes, etc. en ambientes de reunión. Esta ontología ha sido desarrollada como parte del sistema CoBrA, que es una arquitectura con un agente central que permite: 1) reparto de conocimiento, 2) razonar con el contexto, y 3) protección de datos. Además, crean un motor de inferencia que les permite razonar con la información que tienen disponible en la ontología.

La utilización de ontologías les proporciona las siguientes ventajas, que coinciden con algunas proporcionadas en el SRS:

1. Una ontología común permite compartir conocimiento en un sistema distribuido, abierto y dinámico.

2. Las ontologías mediante una semántica bien definida proporcionan mecanismos a los agentes inteligentes para realizar razonamientos.

3. La utilización de ontologías permite a los agentes no estar diseñados para cumplir un objetivo por sí mismos sino que pueden colaborar.

En nuestra aproximación, además, las ontologías permiten que el usuario pueda formular una pregunta utilizando un conjunto de palabras clave y, mediante la interacción con un agente inteligente, transformarla en otra con una semántica bien definida en un lenguaje estructurado, como puede ser OWL. Además la utilización de ontologías también permite la utilización de razonadores basados en Lógica Descriptiva que pueden servir como en nuestro caso para verificar la consistencia de la pregunta formulada por el usuario y simplificarla. En un contexto más general se permite la simplificación de expresiones en Lógica Descriptiva, en el contexto de CoBrA podría utilizarse para la simplificación del número de reuniones o en la planificación de las mismas.

Nuestra aproximación utiliza un razonador basado en Lógica Descriptiva estándar como es RACER [HM01] o FaCT [Hor02c], en cambio en [CFJ04] se muestra que se ha desarrollado un razonador ad hoc para su contexto. Las ventajas de su aproximación son que pueden realizar mayores optimizaciones pero tienen el inconveniente de que no pueden adaptarse a nuevos contextos, o nuevas funcionalidades del sistema. Por lo tanto, en nuestra opinión es mejor utilizar un razonador que pueda adaptarse a cualquier contexto, debido a que es una solución más general.

Nuestro trabajo coincide con esta aproximación en que el razonamiento con la ontología (en nuestro caso el catálogo del usuario) es realizado en un elemento inmóvil, el proxy, y después desplazado a los dispositivos inalámbricos por un agente. Sin embargo, nuestra aproximación es más general permitiendo mostrar las ventajas de la utilización de ontologías en el diseño y desarrollo de servicios de datos inalámbricos independientemente del contexto de ejecución y de los procesos de razonamiento que deban realizar los agentes.

Gaia: A Middleware Platform for Active Spaces

Gaia [RHC+02] es un metasistema operativo, que trata de dar soporte al desarrollo y ejecución de aplicaciones portables en espacios activos. Gaia es un middleware que coordina entidades software y dispositivos de red heterogéneos localizados en un espacio físico. Gaia permite exportar servicios para preguntar y utilizar los recursos existentes, para acceder y utilizar el entorno, y proporciona un campo de pruebas para el desarrollo de servicios centrados en el usuario; independientemente del tipo de dispositivo, siendo sensibles al entorno y permitiendo el desarrollo de aplicaciones móviles.

La arquitectura de Gaia se basa en un conjunto de servicios CORBA que han tenido que ser modificados para permitir el desacoplamiento de los mismos, es decir, se ha modificado el procedimiento de llamadas síncronas entre distintos componentes de la aplicación para adaptarse a contextos inalámbricos. En nuestro caso debido a que utilizamos una arquitectura basada en agentes móviles inteligentes, esta característica (posesión de un protocolo de comunicaciones asíncrono) es inherente al mecanismo de comunicaciones utilizado por los sistemas multiagente.

En [RMCM03a, RMCM03b], muestran cómo el uso de ontologías puede ayudar a superar tres problemas importantes en el desarrollo y despliegue en entornos de *pervasive computing* (descubrimiento e interoperabilidad entre diversas entidades y cuyo conocimiento depende del contexto). También destacan las ventajas que proporciona la utilización de ontologías para facilitar el almacenamiento del conocimiento de la aplicación de una forma estándar mediante la utilización de DAML o OWL, el lenguaje estándar del W3C para la especificación de ontologías. Como hemos dicho anteriormente la utilización de estos lenguajes de representación del conocimiento proporciona ventajas como: 1) interoperación entre componentes software (independientemente de la utilización de agentes o no), 2) utilización de razonadores basados en lógica descriptiva, 3) verificación de la consistencia de la información almacenada en las ontologías, y 4) capacidad de simplificación automática de preguntas de los usuarios.

Por consiguiente, estamos de acuerdo con la mayoría de sus conclusiones. Sin embargo, nuestro trabajo se enfoca en la extracción de las ventajas proporcionadas por los agentes inteligentes en el diseño y desarrollo de servicios inalámbricos. Debido a esto, en nuestro caso debemos destacar la necesidad de aumentar el dinamismo de las aplicaciones para poder adaptarnos a un mayor número de ontologías, haciendo un especial hincapié en la necesidad de desarrollar nuevas arquitecturas que permitan la generación semi-automática de nuevas ontologías. En nuestra aproximación se han desarrollado un conjunto de agentes inteligentes que no solo permiten la generación automática de ontologías, sino que además permiten su integración en una única ontología. Sin embargo, el proceso de generación e integración esta limitado al contexto de los depósitos de software.

8.3. Tratamiento de la información en bibliotecas digitales

En este apartado vamos a comentar tres tipos de trabajos relacionados, como son la generación de la bibliografía de un documento de investigación, las arquitecturas de recuperación de información (conjuntamente con el análisis de inconsistencias) en el contexto de las referencias bibliográficas y la comparación de referencias bibliográficas.

8.3.1. Generación de bibliografía

La calidad de una fuente de información es frecuentemente cuantificada en términos de la precisión de su contenido y de las referencias bibliográficas a trabajos previos que incluye. Por esta razón, en el campo de la investigación, la tarea de almacenar, actualizar y organizar una base de datos bibliográfica es un punto clave en el trabajo diario. Por esta razón, en este apartado vamos a comparar diferentes aproximaciones con la propuesta en esta tesis.

Los gestores de bibliografía son sistemas software que permiten a sus usuarios insertar citas en posiciones específicas de un documento y generar bajo demanda la lista de referencias bibliográficas apropiadas para ese documento. Por ejemplo, los usuarios de Microsoft Word [Mic02] disponen de herramientas como Procite [Res02b] y End note [Res02a].

En el caso de TEX [UGT02] también existen gestores de referencias bibliográficas como BibTEX que está basado en ficheros `.bib`. Estos son ficheros de texto plano que contienen colecciones de registros con un conjunto de campos predeterminados, aunque el formato

BibTEX permite añadir campos personalizados. En Internet existen multitud de depósitos de datos en este formato aunque como desventaja tenemos que mencionar que no permite el manejo de colecciones, ni la detección de inconsistencias. En la aproximación presentada en esta tesis, la detección de inconsistencias es realizada por el agente inteligente Bib2DB.

En 1997, comenzó el proyecto *Bibword* [Can00] con una doble meta, crear un sistema de gestión de referencias bibliográficas para los usuarios de Word que fuese libre y por otro lado que fuese capaz de reutilizar los ficheros de bibliografía disponibles en formato BibTEX. Se han generados distintas versiones de BibWord, las primeras solamente podían ser utilizadas mediante una base de referencias bibliográficas local. Sin embargo, actualmente se puede acceder a bases de datos de referencias bibliográficas remotas. Además, también se permite el acceso mediante un navegador web.

Todos los sistemas mencionados permiten el almacenamiento de referencias bibliográficas de forma local y, algunos también, remota. Sin embargo, la principal limitación de las herramientas presentadas anteriormente es que no consideran la introducción de duplicados, por lo que el coste de mantenimiento de estas colecciones de referencias bibliográficas es muy elevado. En cambio, como mostraremos posteriormente, nuestra aproximación basada en agentes considera este hecho.

8.3.2. Recuperación de información en bibliotecas digitales

En esta subsección, mostramos distintas aproximaciones que consideran el problema de la integración de distintos depósitos bibliográficos, sin embargo solamente algunas de dichas aproximaciones consideran la utilización de agentes inteligentes para su diseño.

En el proyecto InterBib [Pae06]: Bibliography-Related Services se plantea la unificación dentro de un único depósito de datos de la información contenida en distintos depósitos distribuidos. En esta aproximación se almacena el resultado de la búsqueda en un formato seleccionable por el usuario. Sin embargo no se tratan problemas como detección de inconsistencias o duplicidad de la información contenida en los distintos almacenes de datos. Tareas realizadas por el agente Bib2DB en la aproximación presentada en la tesis.

En la tesis doctoral "*Agentes Móviles en bibliotecas digitales*" [Pér98] se trata el tema de la utilización de agentes móviles para realizar búsquedas en un sistema de bibliotecas digitales teniendo en cuenta la "inteligencia" de los agentes para seleccionar las publicaciones de acuerdo a unas preferencias definidas por el usuario. Consideramos esta aproximación como complementaria a la nuestra puesto que se realiza una búsqueda con propósitos distintos a los nuestros. Aunque, a diferencia de nuestra aproximación, no considera la integración de fuentes de datos distintas.

En [YRG00] proponen construir un conjunto de agentes móviles que colaboren en la integración del gran volumen de datos geográficos disponibles en la biblioteca digital SARA. Se trata pues de una aproximación similar a la nuestra pero aplicada a un contexto distinto.

El Cornell Digital Library Research Group: Architectures and Policies for Distributed Digital Libraries [Lag00] trata el problema del acceso a referencias bibliográficas usando componentes software que se encuentran distribuidos en Internet, siguiendo por tanto una aproximación basada en RPC. En cambio nuestra aproximación basada en agentes móviles permite reducir el tráfico de red y ser robusta frente a desconexiones.

A diferencia de las aproximaciones presentadas, nosotros proponemos una arquitectura basada en agentes que obtiene los siguientes beneficios: proporciona un mecanismo de detección de inconsistencias, permite la distribución de la tarea de detección de inconsistencias mediante la utilización de distintos agentes, y además permite verificar que las ventajas (reducción del tráfico de red, facilidades para el procesamiento paralelo, y posibilidad de distribuir el proceso de detección de inconsistencias) obtenidas debido a la utilización de una arquitectura basada en agentes, son independientes del proceso de detección de inconsistencias utilizado.

8.3.3. Comparación de publicaciones

Se han propuesto diversas heurísticas para medir la similitud entre dos campos de una base de datos [ME96, RY98, Yia97, BM02]. Aunque podríamos aplicar en nuestro trabajo cualquiera de estas aproximaciones, hemos decidido utilizar la distancia de Levenshtein porque es el modelo más ampliamente estudiado y para el que se han desarrollado los algoritmos más eficientes [NR02]. Debemos destacar, que el mecanismo de detección de inconsistencias es la base de la inteligencia del agente Bib2DB; por consiguiente, si se utilizase otro mecanismo de detección de inconsistencias únicamente habría que modificar el módulo asociado a la inteligencia para la detección de inconsistencias de nuestro agente. Manteniéndose las ventajas proporcionadas por las arquitecturas basadas en agentes como son: independencia del protocolo de comunicaciones, robustez frente a desconexiones, etc.

En [BM02] se propone combinar las medidas de similitudes de campos individuales utilizando *support state machines*, pero este método sufre desventajas similares a nuestra aproximación con redes neuronales (coste de entrenamiento).

En [Hyl96] se proponen diversos métodos para agrupar conjuntos de referencias bibliográficas que pueden ser potenciales duplicados. Proponen un algoritmo de comparación del número de palabras y frases no coincidentes. En consecuencia, es de esperar que su método no sea capaz de detectar errores tipográficos.

En [Bea04] se propone un sistema *peer-to-peer* para el intercambio de referencias bibliográficas. Proponen utilizar varias métricas de distancia simultáneamente para mejorar la precisión de la detección de duplicados. Sin embargo, a diferencia de nuestra aproximación, su método no se adapta automáticamente al depósito de datos analizado. Por lo tanto, el agente Bib2DB modifica su comportamiento adaptándose a la información almacenada en los depósitos de datos analizados.

Como en nuestra propuesta inicial de comparador basado en reglas, en [KLK+04] se propone el uso de reglas para detección de duplicados. La novedad es que su objetivo es generar estas reglas automáticamente a partir de coeficientes de similitud obtenidos previamente. Sin embargo, su contexto es distinto (datos biológicos) y por lo que sabemos no ha habido ningún intento de aplicar esas ideas a colecciones de referencias bibliográficas.

Otros trabajos complementarios al nuestro tienen como principal objetivo determinar algoritmos para detectar eficientemente (con menor coste que la solución trivial que es $O(n^2)$) los duplicados en un conjunto preexistente de referencias bibliográficas [HS95, ME97].

Sin embargo, debemos destacar que nuestra aproximación basada en agentes sigue siendo igualmente ventajosa independientemente del mecanismo de detección de inconsistencias

utilizado. Este hecho se fundamenta en que los agentes permiten distribuir el procesamiento y evitar cuellos de botella, además de permitir reducir el tráfico de red y ser robustos frente a desconexiones, debido a que utilizan un protocolo de comunicaciones asíncrono.

8.4. Servicios de acceso a datos en entornos dinámicos

En esta sección vamos a relacionar las arquitecturas basadas en agentes en el contexto de la generación de interfaces de usuario adaptativos, los servicios basados en la localización y la web semántica con otro tipo de aproximaciones, por ejemplo arquitecturas cliente/servidor. Esta comparación nos permitirá mostrar las ventajas y desventajas del paradigma de los agentes inteligentes en el diseño y desarrollo de servicios de datos en entornos de ejecución dinámicos, donde cambian los recursos disponibles en cada momento.

8.4.1. Interfaces de usuario adaptativas

Se presentan distintas aproximaciones que permiten adaptar el interfaz gráfico de usuario a diferentes dispositivos. Básicamente, las aproximaciones se agrupan en dos categorías principales: aplicaciones web y aplicaciones de escritorio clásicas. Mientras que el primer tipo [Cor01, IBM06a] tratan solamente el contenido y las transformaciones en la web, el segundo tipo considera la problemática de definir un interfaz de usuario de forma universal, permitiendo de esta forma que pueda ser mostrado, o interpretado, por distintos tipos de dispositivos o plataformas con distintas capacidades de visualización [PMP03, Sto01, eIML06].

Aplicaciones web e interfaces de usuario adaptativos

Los servidores de aplicaciones están orientados a transformar el contenido web a distintos formatos que pueden ser visualizados por dispositivos móviles (cHTML [Wor06a], WML [WWSV06], etc.). Sin embargo, existen distintas aproximaciones.

Microsoft en su plataforma ".NET" ofrece Formularios Web Móviles[2] [Cor01]. Estos formularios están basados en un conjunto de restricciones de componentes que, en nuestra opinión, no puede ser extendido con nuevos elementos gráficos[3] adicionales. Cada componente es un componente inteligente que transforma su apariencia en función de los recursos disponibles. Además, el control está muy limitado por la familia de lenguajes .NET. Desafortunadamente, las soluciones de Microsoft están solamente disponibles para la plataforma Windows, el número de elementos gráficos es limitado, y las aplicaciones de escritorio no son consideradas, es decir, solamente se consideran aplicaciones con interfaz web. En cambio mediante nuestra aproximación basada en agentes, el GUI puede ser generado independientemente de que se esté ante una aplicación web o no, e independientemente del sistema operativo utilizado por el dispositivo del usuario. Debido a que los agentes móviles son independientes del sistema operativo en el que se están ejecutando.

Otros líderes en el campo de la industria, como IBM, tienen aproximaciones ligeramente diferentes. Por ejemplo, *Transconding Publisher* [IBM06a] de IBM permite la traducción de

[2] *Mobile Web Forms.*
[3] *widgets.*

contenido web a una gran variedad de formatos, consiguiendo posibilidades de personalización de los parámetros de transformación para los distintos tipos de usuarios. Se incluyen algunas características interesantes, tales como transformaciones basadas en Java Script y transformaciones automáticas de imágenes a distintos formatos. Sin embargo, el punto débil de dicha aproximación es que únicamente pueden transformar contenidos web, y de una forma centralizada a diferencia de nuestra aproximación basada en agentes que permite realizar este tipo de transformaciones de forma distribuida. Además nuestra arquitectura basada en agentes móviles considera cualquier tipo de aplicación, no únicamente las aplicaciones web sino también las aplicaciones de escritorio.

Interfaces adaptativos basados en XML

Distintas aproximaciones para definir interfaces de usuario han sido desarrolladas hasta el momento de escribir esta tesis. Sin describir ninguna de ellas en detalle, mencionaremos las distintas aproximaciones: basadas en lenguajes y gramáticas, como por ejemplo, BNF, basadas en eventos, basadas en restricciones, notaciones basadas en las acciones del usuario (en particular las acciones ejecutadas directamente por el usuario) y basadas en componentes gráficos. Sin embargo, la definición de interfaces basados en XML es la más interesante, porque proporciona una gran flexibilidad y facilidad de manipulación. Algunas de las aproximaciones para definición de interfaces basadas en XML son: XUL [Tut06], lenguaje para la definición de interfaces de usuario extensible (*eXtensible User interface Language*, UIML (*User Interface Mark-up Language*) [APB+99, Sto01] y XIML [eIML06].

En nuestro caso hemos adoptado una aproximación similar, es decir los agentes utilizan un lenguaje de definición de interfaces de usuario basado en XML que les permite transformar dicha definición en un GUI concreto, pero realizando la transformación de forma transparente al usuario y en tiempo real dependiendo del dispositivo en el que se está ejecutando la aplicación.

8.4.2. Cálculo de la posición de un dispositivo móvil

Al igual que en la aproximación presentada en esta tesis, existen otros autores que utilizan la potencia de la señal recibida para calcular la posición en la que se encuentra un dispositivo móvil. Por ejemplo, el sistema RADAR [BP00, BBP00] permite calcular la posición de un dispositivo móvil basado en radio frecuencia, su arquitectura facilita la localización y monitorización de los movimientos seguidos por un conjunto de usuarios en un edificio. Su aproximación está basada en la potencia de la señal y en la generación de mapas de potencia. Sin embargo, las arquitecturas basadas en agentes móviles inteligentes, como la nuestra, permiten enviar aplicaciones (como el agente *beacon*) evitando la necesidad de tener activada en todo momento una infraestructura estática en el dispositivo móvil. Por consiguiente, las arquitecturas basadas en agentes móviles, como la presentada en esta tesis, facilitan el diseño y desarrollo de servicios basados en la localización independientemente del mecanismo utilizado para calcular la posición en la que se encuentra un dispositivo móvil.

En [NDA01] detallan distintos métodos para calcular la localización de un dispositivo móvil en el interior de los edificios, su aproximación para localizar los dispositivos móviles de los usuarios se basa en un mapa de potencia al igual que la nuestra. Sin embargo, ellos

correlacionan el mapa de potencias y la potencia de emisión del dispositivo móvil utilizando una red neuronal, mientras que nosotros utilizamos técnicas de triangularización para localizar el dispositivo móvil del usuario, proporcionando resultados más exactos. Sin embargo, si su aproximación o cualquier otra fuese más exacta, entonces la arquitectura propuesta en esta tesis podría utilizarla y se seguirían manteniendo las ventajas debidas a la utilización de una aproximación basada en agentes móviles inteligentes, como por ejemplo, su capacidad para desplegarse en tiempo de ejecución dependiendo de los recursos disponibles en cada momento.

En [LBM+02] se crean automáticamente los mapas de potencia utilizando redes bayesianas. Sin embargo, nosotros hemos detectado que los trabajos que utilizan mapas de potencias creados semi-automáticamente (tomando lecturas reales como base) obtienen resultados más precisos. Debido a esto, la utilización de mapas de potencia es la aproximación utilizada en la arquitectura propuesta en esta tesis.

Sin embargo, hay que resaltar que lo importante de nuestra aproximación no es el procedimiento utilizado para calcular la posición en la que se encuentra el dispositivo del usuario, sino las ventajas obtenidas debido a la utilización de una arquitectura basada en agentes móviles inteligentes. Es decir, nuestra aproximación podría utilizar cualquiera de los métodos propuestos como trabajo relacionado y mantener las ventajas obtenidas de la utilización de una arquitectura basada en agentes: nuestra aproximación permite generar una infraestructura dinámica en tiempo de ejecución que permite localizar dispositivos móviles. Además, la arquitectura propuesta minimiza, la cantidad de recursos (memoria, disco, capacidad de procesamiento, tráfico de red) consumidos en el proceso de localización.

8.4.3. Enlazando palabras clave y ontologías/depósitos de datos

En esta sección describimos un conjunto de aproximaciones que permiten enlazar palabras clave o depósitos de datos con los términos de un conjunto de ontologías. Solamente se incluye un trabajo que utiliza técnicas de recuperación de información (*Information Retrieval*) porque los trabajos basados en esta aproximación indexan un conjunto de fuentes de datos directamente, sin utilizar ontologías u otras técnicas de descripción semántica de los depósitos de datos desarrollados por terceros.

En el proyecto *Seamless Searching of Numeric and Textual Resources* [HM00], se realiza una búsqueda en un conjunto de colecciones que permite descubrir relaciones entre conceptos introducidos por los usuarios y conceptos de un conjunto de fuentes de datos web. Esta aproximación utiliza un diccionario personalizado para desambiguar los conceptos utilizados en las preguntas planteadas al sistema; sin embargo, nuestra aproximación utiliza un tesauro de propósito general, WordNet [Mil06], y el contexto de las palabras clave del usuario.

Una aproximación al problema de generar información de enlace es CUPID [MBR01]. Esta herramienta genera esquemas de enlace sobre un conjunto genérico de depósitos de datos. En Ontobuilder [GATTMed] se describe un modelo que permite establecer relaciones semánticas de forma automática. Las dos aproximaciones previas estudian cómo enlazar depósitos de datos y ontologías, lo cual es complementario a nuestra aproximación; porque nuestros agentes enlazan un conjunto de palabras clave introducidas por el usuario con un conjunto de ontologías, las cuales pueden ser creadas o enlazadas con un almacén de datos

utilizando sus aproximaciones.

Finalmente, GLUE [DMDH02] estudia la probabilidad de enlazar dos conceptos analizando las ontologías disponibles. La principal característica de esta aproximación es que se utiliza un algoritmo de relajación de etiquetas [RHZ76], es decir, el nombre asociado a un nodo es influenciado por las características de sus nodos vecinos en la ontología (esta suposición es común a nuestra aproximación). Sin embargo, esta aproximación no es muy flexible/adaptable porque analiza *todos* los conceptos de la ontología, mientras que nuestra aproximación se basa en el muestreo de los conceptos de la ontología reduciendo el coste del algoritmo en tiempo y en recursos utilizados.

Todos los trabajos relacionados presentados en esta subsección tienen el inconveniente de poseer una arquitectura estática, a diferencia de nuestra arquitectura basada en agentes inteligentes. Debido a esta característica, nuestra arquitectura puede ser creada y replicada en tiempo de ejecución dependiendo de los recursos disponibles en cada momento. Este hecho muestra cómo las arquitecturas basadas en agentes móviles permiten una fácil adaptación al contexto de ejecución y ayudan a su despliegue independientemente de los recursos disponibles.

8.5. Resumen del capítulo

En este capítulo hemos detallado algunos de los trabajos relacionados con la tecnología de agentes móviles inteligentes aplicados al diseño y desarrollo de servicios de datos en entornos inalámbricos (o cableados) en los que los recursos disponibles en cada momento pueden variar.

En primer lugar se han comentado distintas aproximaciones que comparan las arquitecturas basadas en agentes móviles con las arquitecturas clásicas basadas en llamadas a procedimientos remotos (RPC). Se han presentado distintas comparaciones tanto teóricas como empíricas, además las aproximaciones presentadas consideran distintos entornos de ejecución, redes cableadas y redes inalámbricas. También hay que tener en cuenta que algunas de las aproximaciones detalladas consideran los errores de red que pueden producirse cuando se utilizan distintos tipos de comunicaciones. Debido a todo esto, podemos concluir que las arquitecturas basadas en agentes proporcionan ventajas significativas para el diseño y desarrollo de servicios de datos en entornos inalámbricos, además de facilitar la adaptación del sistema al contexto de ejecución y a los recursos disponibles en cada momento.

A continuación, hemos presentado distintas aproximaciones basadas en la utilización de ontologías para representar el conocimiento de los sistemas multiagente. Sin embargo, aunque las ontologías son un mecanismo que permite describir la semántica de los servicios que proporciona un agente y su inteligencia, tienen el inconveniente de que el vocabulario usado en las bases de conocimiento utilizadas por dos sistemas multiagente distintos puede diferir. Por lo tanto, se están empezando a desarrollar nuevas aproximaciones que permiten enlazar los términos de dos ontologías distintas, además estas aproximaciones consideran el contexto del vocabulario utilizado por los agentes.

Posteriormente, se han descrito distintas aproximaciones relacionadas con los casos de estudio utilizados en la tesis para mostrar las bondades de la tecnología de agentes inteligentes en el diseño y desarrollo de servicios de datos en entornos inalámbricos. En el contexto de

los servicios de recuperación de software se han presentado distintos trabajos que describen la utilización de agentes (o arquitecturas cliente/servidor) para: 1) probar componentes software, 2) reutilizar componentes de librerías y 3) aproximaciones que permiten la búsqueda y recuperación de software.

También, se han presentado distintas aproximaciones que permiten la generación de bibliografía, recuperación de información y comparación de referencias bibliográficas en el contexto de las bibliotecas digitales, mostrando las ventajas de las aproximaciones basadas en un sistema multiagente. Concluyendo que las ventajas obtenidas por el diseño de una arquitectura basada en agentes en este contesto se mantienen independientemente del mecanismo de detección de inconsistencia, utilizado por el agente.

Finalmente, se han comparado distintas aproximaciones que permiten el diseño y desarrollo de servicios en entornos dinámicos. Por ello, hemos mostrado las ventajas de las aproximaciones basadas en agentes, relacionando estas arquitecturas con otro tipo de aproximaciones; centrándonos en los tres casos presentados: 1) generación de interfaces de usuario adaptativos, permitiendo el estudio del comportamiento del usuario, 2) creación de una infraestructura para el diseño de servicios basados en la localización, y 3) acceso a un sistema de información global en el que cambian dinámicamente los recursos disponibles.

En todos los trabajos relacionados se han mostrado tanto las carencias como las bondades de las aproximaciones realizadas, comparando aproximaciones basadas en agentes y otro tipo de arquitecturas. Concluyendo que las arquitecturas basadas en agentes permiten su creación en tiempo de ejecución minimizando el consumo de recursos y siendo robustas frente a desconexiones.

Capítulo 9

Conclusiones y trabajo futuro

En este capítulo vamos a resumir cuáles han sido las ventajas que hemos encontrado en la utilización de los agentes móviles inteligentes en nuestro estudio para el diseño y desarrollo de sistemas software para contextos distribuidos e inalámbricos. Incidiremos en los avances logrados en los casos de estudio analizados a lo largo de esta tesis, y las conclusiones a las que hemos llegado, especialmente sobre las ventajas de la utilización del paradigma de agentes móviles frente a aproximaciones más clásicas como las arquitecturas cliente/servidor. Del mismo modo, también detallaremos las ventajas de la utilización de arquitecturas basadas en agentes, independientemente del contexto de aplicación, tal y como se ha mostrado en los capítulos previos.

A continuación, comentaremos cuáles han sido las principales contribuciones y resultados del trabajo desarrollado, y dónde han sido presentados a lo largo del desarrollo de esta tesis doctoral.

Finalmente, discutiremos brevemente los posibles trabajos futuros o ampliaciones que pueden desarrollarse a partir del trabajo realizado, enfocándolo, tanto desde un punto de vista general, ventajas derivadas del uso de agentes inteligentes, como desde un punto de vista particular para cada uno de los distintos casos de estudio estudio y contextos considerados.

9.1. Conclusiones extraídas de la utilización de agentes inteligentes

En esta sección enumeraremos brevemente las ventajas de la utilización de los agentes móviles inteligentes en entornos inalámbricos. Hay que recordar que las ventajas proporcionadas por los sistemas multiagente también se dan en el caso de sistemas que utilicen comunicaciones cableadas, aunque su impacto es significativamente más importante en entornos inalámbricos.

Con el desarrollo de esta tesis, se ha llegado a las siguientes conclusiones sobre el diseño y desarrollo de las arquitecturas basadas en agentes:

1. *Robustez frente a desconexiones*. Las arquitecturas basadas en agentes son inherente-

mente robustas frente a los problemas derivados de la inestabilidad y lentitud de las comunicaciones, características intrínsecas de las redes inalámbricas (en comparación con las redes cableadas).

2. *Adaptabilidad al contexto de ejecución*. Las arquitecturas basadas en agentes permiten ejecutar los servicios que éstos proporcionan en el lugar más adecuado para optimizar el desempeño de su función y la utilización de los recursos disponibles.

3. *Optimización de recursos*. Un contexto de ejecución de agentes software (*place*) de una plataforma de agentes móviles, permite poner en ejecución un número no definido de agentes (que prestan servicios). Esto se debe a que la ejecución de cada uno de los agentes se despliega en uno o más hilos de ejecución, dependiendo de la plataforma de agentes concreta que esté siendo utilizada para su implementación. Esta aproximación es generalmente menos costosa que la basada en arquitecturas cliente/servidor, donde cada servicio es un proceso, que a su vez puede utilizar uno o más hilos de ejecución, lo cual supone consumir mayor número de recursos.

4. *Independencia de la plataforma*. Las arquitecturas basadas en agentes móviles deben ser independientes de la plataforma y del sistema operativo en el que se estén ejecutando, para posibilitar así el movimiento de los agentes entre distintas arquitecturas hardware y software. Esta característica también puede llegar a incorporarse en un diseño cliente/servidor, aunque no está presente de forma intrínseca en este tipo de arquitecturas. Sin embargo, la utilización de aproximaciones basadas en el paradigma de agentes móviles inteligentes son independientes de la arquitectura hardware por definición, tal como se plasma en la propia definición de agente móvil: un agente móvil es un agente software que puede moverse de *place* a *place* (independientemente de la arquitectura hardware en la que se ejecuten cada uno de los *places*).

5. *Reactividad e inteligencia*. Por definición, los agentes deben ser unos componentes software que reaccionen frente a cambios en el entorno, de forma más o menos inteligente. En cambio, otras aproximaciones no poseen estas propiedades “per se”. Por ejemplo, muchas plataformas de agentes pueden permitir un comportamiento que reaccioné o se adapte frente a fallos eléctricos, debido a que almacenan el estado de ejecución de sus agentes en disco [LO99a, OSP05, BPR01, Til05, TSB05] para poder así automatizar la recuperación del sistema; sin embargo, otros agentes inteligentes pueden reaccionar de forma distinta; por ejemplo, un agente móvil podría migrar su ejecución a otro ordenador en previsión de un fallo eléctrico o exceso de carga en el sistema. Además, los agentes que forman parte de un sistema multiagente, tienen que ser lo suficientemente inteligentes para poder cooperar en la realización de una tarea. Incluso, pueden facilitar la incorporación de nuevos agentes al sistema, de tal forma que éste pueda ser ampliado sin realizar cambios en los agentes ya existentes.

6. *Disminución de la latencia de las comunicaciones por red*. Debido a que un agente puede viajar a través de la red y ejecutarse en el ordenador en el que se encuentra un determinado recurso, los agentes móviles permiten aumentar la frecuencia con la que se puede analizar el estado de un sistema distribuido, ejecutando un agente en cada

uno de los ordenadores que posean un recurso del que se desee monitorizar cambios en su estado (realizando así un análisis local). Además, esta aproximación es escalable y permite añadir al estudio el estado de nuevos recursos en los ordenadores que están siendo analizados, sin tener que rediseñar la infraestructura; mientras que en el resto de los ordenadores a monitorizar, únicamente habría que ejecutar una plataforma de agentes móviles que permite la ejecución de los agentes que lo visitarán.

En resumen y con todo ello podemos concluir que los agentes móviles inteligentes facilitan el diseño y desarrollo de aplicaciones distribuidas, debido a que poseen: 1) las ventajas de las aproximaciones clásicas: independencia del protocolo de comunicaciones utilizado, interfaces de comunicación predefinidos, lenguajes de definición de datos que permiten la interacción entre distintas arquitecturas; y 2) añaden algunas nuevas ventajas como: robustez frente a desconexiones, adaptabilidad al contexto de ejecución, optimización de recursos, independencia de la plataforma de ejecución, inteligencia y reactividad para enfrentarse/afrontar de forma autónoma y automática algunos de los problemas que puedan surgir durante la ejecución de los servicios proporcionados por los agentes. Todas estas ventajas se traducen en un aumento de las prestaciones que pueden obtenerse durante la ejecución de servicios distribuidos, e inalámbricos o no. Además los agentes pueden proporcionar una adaptación automática del servicio al contexto de ejecución y al dispositivo del usuario (independizando el diseño y desarrollo, de la posterior ejecución del servicio).

A continuación revisaremos las ventajas encontradas para cada uno de los casos de estudio que se seleccionaron para la realización de esta tesis.

9.1.1. Servicio de recuperación de software

Como se ha mostrado en los capítulos 4 y 5, hemos diseñado en esta tesis un servicio que permite a los usuarios de dispositivos móviles localizar y obtener software, de diferentes depósitos de software disponibles como Tucows, Download.com, etc., de una manera fácil, dirigida y eficiente sin requerir grandes conocimientos por parte del usuario.

1. *Fácil*, porque el servicio permite que los usuarios expresen sus requisitos acerca del software que desean en un nivel semántico, es decir, expresan lo que necesitan pero no cómo obtenerlo (un conjunto de agentes autónomos e inteligentes son los encargados de ayudar al usuario en esta tarea).

2. *Dirigida*, porque el servicio, utiliza agentes especializados que presentan al usuario únicamente las categorías de software relacionadas con sus requisitos (un catálogo adaptado a sus necesidades) y le ayudan a navegar por las categorías hasta que encuentre el software deseado.

3. *Eficiente*, porque aunque el servicio se puede utilizar en cualquier clase de dispositivos, pone un especial énfasis en usuarios móviles, optimizando así el uso de las comunicaciones inalámbricas. En este servicio, los agentes que constituyen el sistema, realizan un estudio y seguimiento tanto de las comunicaciones disponibles como del comportamiento del usuario. Los resultados experimentales obtenidos usando el prototipo desarrollado corroboran su eficiencia al comparar el servicio con otras maneras más clásicas de obtener software, tales como utilizar Tucows.com, u otros mecanismos similares.

Un logro muy importante es el alto grado de autonomía y capacidad de adaptación de los agentes que participan en la arquitectura del sistema: estos agentes analizan el estado de la red, estiman el nivel del detalle más apropiado para cada usuario y situación, observan el comportamiento del usuario, y podan el catálogo de software, además de aprender de sus errores al intentar anticipar las posibles acciones del usuario. Todas estas características han demostrado ser importantes ventajas, como se ha comprobado con la evaluación del funcionamiento de nuestro sistema.

Recopilando, las principales contribuciones del Servicio de Recuperación de Software, soportado por el uso de la tecnología de agentes y el uso de una ontología como catálogo de software, son:

1. El desarrollo de cuatro clases de agentes autónomos, capaces de reaccionar al estado de su entorno para ofrecer/lograr unas mejores prestaciones del servicio, según se ha mostrado en el capítulo 4. Los principales agentes del servicio son: el mayordomo (denominado *Alfredo*), el *Navegador*, el *Gestor de Software* y el *Actualizador de Catálogos*. Sin embargo, el servicio se beneficia también de la utilización del agente *Ingeniero del Conocimiento* y del agente *Integrador*, encargados de la creación automática de la ontología SoftOnt (capítulo 5).

   A continuación enumeraremos únicamente las ventajas de los agentes principales:

   - El agente Alfredo, gestiona dos tipos de conocimiento, independiente del servicio y relacionado con el servicio, además de información probabilística sobre las respuestas del usuario, que utiliza para ayudar y servir al usuario lo mejor que pueda, tal como un mayordomo humano haría. De esta forma el agente mayordomo complementa automáticamente las peticiones del usuario, con restricciones acerca del dispositivo del usuario y las preferencias del mismo, evitando así que el usuario tenga que conocer detalles técnicos sobre su dispositivo.
   - El agente Gestor de Software, permite la obtención de catálogos de software personalizados a las necesidades de cada usuario, de acuerdo con: 1) las restricciones especificadas por el usuario y el agente Alfredo, y 2) la velocidad de la red actual en la que se esté ejecutando el servicio (así una mayor velocidad de red implicará un mayor nivel de detalle en el catálogo de software).
   - El agente Navegador, adapta su comportamiento (aprendiendo de sus errores) al del usuario, con el propósito de anticiparse a las acciones de éste y así ayudarle a encontrar el software deseado en el catálogo mostrado, tan rápido como sea posible.
   - El agente Actualizador de Catálogos, selecciona en tiempo real la mejor estrategia para actualizar el catálogo de software presentado al usuario, considerando el estado actual de la red. Este agente puede optar por comunicarse utilizando una aproximación cliente/servidor, o bien viajar al *place* que posee el catálogo de software, para minimizar el coste de las comunicaciones gracias a su capacidad para realizar un refinamiento del catálogo de software mostrado al usuario.

Este conjunto de agentes optimiza además el uso de las comunicaciones, y reaccionan a los fallos producidos en las comunicaciones autónomamente, aplicando una política de reintentos.

2. Hemos comparado nuestro sistema desde dos puntos de vista diferentes, tanto analítica como empíricamente, con una aproximación más clásica basada en la navegación de sitios web como Tucows.com y similares. Según la comparación analítica mostrada en la sección 4.4.1 podemos extraer la conclusión de que el Sistema de Recuperación de Software consigue unas mayores prestaciones, dependiendo de dos factores: 1) tamaño del catálogo de software, y 2) número de refinamientos locales que es capaz de realizar el agente Navegador sin ayuda externa, lo cuál también depende directamente tanto del tamaño del catálogo inicial como de la "inteligencia" de los agentes. Debido a que esta característica difícilmente puede ser probada analíticamente se ha verificado de forma empírica con los experimentos realizados.

   Como se comentó en la sección 4.4.2, en [MCM03] se realiza una comparación analítica entre el funcionamiento de sistemas tipo Tucows y nuestro SRS. Como resultado del análisis se concluyó que el SRS se comporta mejor que Tucows cuando el número de refinamientos realizados localmente es mayor que un determinado umbral, que depende de la velocidad de la red en la que se este ejecutando el servicio y del tamaño de los agentes. Los tests realizados demuestran que el número de refinamientos locales realizados por el agente navegador siempre es mayor que dicho umbral, independientemente de la red utilizada y del software deseado por el usuario.

   Del mismo modo, se ha verificado que las prestaciones obtenidas por el Servicio de Recuperación de Software son mejores, debido también a que minimiza el uso de la red y ayuda al usuario en la búsqueda del software, gracias a la inteligencia de los distintos agentes que constituyen la arquitectura propuesta.

Finalmente, hemos demostrado también empíricamente cómo los agentes se adaptan autónoma y automáticamente al contexto y comportamiento de usuarios con diferente nivel de experiencia para mejorar las prestaciones del sistema.

9.1.2. Servicios de bibliotecas digitales

Como se ha presentado en el capítulo 6, las ventajas de la utilización de agentes móviles inteligentes en el contexto de las bibliotecas digitales de investigación son:

- *Robustez frente a desconexiones*. Gracias a que la arquitectura de nuestra biblioteca digital está basada en un conjunto de agentes móviles, el servicio de recopilación de referencias bibliográficas es robusto frente a desconexiones, porque los agentes son independientes del entorno que lanzó su ejecución. Esta característica se pudo demostrar en un contexto real de ejecución cuando se estaba probando el sistema puesto que se producían fallos frecuentes en la red. En un contexto tal, la arquitectura desarrollada permitía que el sistema se recuperase autónomamente, ante fallos de red, sin la necesidad de realizar ningún esfuerzo adicional ni requerir la intervención del usuario.

- *Optimización de la transferencia de datos*. Las arquitecturas basadas en agentes permiten la optimización de las comunicaciones, especialmente en casos de recuperación de información. Esta característica es todavía más importante si existen datos duplicados en la información que se debe analizar, hecho muy frecuente en el caso utilizado como ejemplo, las bibliotecas de publicaciones científicas, debido a que todos los miembros del grupo realizan publicaciones juntos, pero las introducen de forma independiente, y no consensuada, en los diferentes ficheros de bibliografía. Otra de las ventajas que nos proporciona una arquitectura basada en agentes, frente a una arquitectura cliente/servidor, es que parte del procesamiento puede realizarse de forma local: por ejemplo, cada uno de los distintos depósitos bibliográficos puede ser analizado en el cliente evitando así saturar el servidor. Es decir, en la secuencia de viajes que el agente realiza entre los distintos ordenadores que almacenan los depósitos de datos de su interés, puede filtrar las referencias duplicadas o añadir información a las referencias incompletas.

- *Adaptabilidad al contexto*. El agente recopilador de referencias bibliográficas adapta su ejecución al contexto gracias al tratamiento automático de las desconexiones. Sin embargo, también se adapta de forma autónoma y automática al conjunto de depósitos de datos que está analizando. Es decir, el agente modifica el comportamiento de su comparador de publicaciones, dependiendo del contenido de los distintos depósitos de datos, tal y como se ha presentado en detalle en el capítulo 6.

- *Inteligencia de los agentes*. En este caso, la inteligencia de nuestro agente se manifiesta mediante la modificación de su comportamiento dependiendo del estado de la red, por ejemplo modificando el tiempo de espera entre reintentos para realizar un viaje. También posee la suficiente inteligencia para decidir autónomamente si dos publicaciones de investigación son la misma, incluso en aquellos casos en los que haya información contradictoria o incompleta en las referencias bibliográficas encontradas en los depósitos de datos analizados. Por consiguiente, gracias a la inteligencia del agente recopilador de referencias bibliográficas, nuestra biblioteca digital permite la integración de depósitos bibliográficos de forma semiautomática.

Como hemos visto, el agente recopilador de referencias bibliográficas posee la mayoría de las cualidades que se asocian con los agentes inteligentes, como son: robustez frente a desconexiones, optimización de las tareas a realizar por el agente, adaptabilidad al contexto, inteligencia, etc.

9.1.3. Servicios de acceso a datos en entornos de ejecución dinámicos

Como se ha presentado en el capítulo 7, en el diseño y desarrollo de servicios de acceso a datos en entornos dinámicos, se debe utilizar un mecanismo para desplegar en tiempo de ejecución la arquitectura que dé soporte a los mismos. Este hecho se fundamenta en que los recursos disponibles en cada momento pueden cambiar, por ejemplo las estaciones base que den cobertura al usuario en un momento concreto dependen de la ubicación exacta del usuario.

Por consiguiente, se ha diseñado un conjunto de arquitecturas basada en agentes móviles que permiten dar soporte a usuarios de dispositivos inalámbricos posibilitando así una

adaptación al contexto de ejecución gracias a que la arquitectura se despliega dinámicamente cuando va a ser utilizada, y minimiza los recursos empleados. La arquitectura propuesta posee las siguientes ventajas:

- *Robustez y optimización de las comunicaciones*. Las arquitecturas propuestas se basan en un conjunto de agentes móviles inteligentes que permiten la utilización de la red de comunicaciones de forma asíncrona, por lo que las distintas aproximaciones basadas en agentes presentadas son capaces de recuperarse autónomamente frente a los errores que puedan producirse en la transmisión. Además, la utilización de la tecnología de agentes ofrece de forma intrínseca mecanismos de comunicación asíncronos, que son de indudable utilidad en entornos distribuidos e inalámbricos, como es el caso de los basados en el uso de comunicaciones WiFi.

  Los mecanismos de comunicación asíncrona entre los agentes también permiten la optimización de las comunicaciones, gracias a que no deben reiniciarse los procesos de comunicación desde el comienzo de la petición de localización realizada por el usuario. Es decir, cuando un agente recibe un mensaje para realizar una tarea, el agente puede continuar con su trabajo a partir de dicho punto independientemente de que se produzcan errores en la red.

- *Creación dinámica de infraestructuras*. En el conjunto de arquitecturas propuestas, la única infraestructura software necesaria para posibilitar su funcionamiento es que el dispositivo móvil posea un *place* y o contexto de ejecución, al que puedan viajar los agentes móviles. En el caso de utilizar una aproximación basada en una arquitectura cliente/servidor, lo más habitual sería tener instalado un servidor por cada uno de los distintos servicios que se requiera ejecutar en el dispositivo móvil, produciéndose así una sobrecarga de trabajo en el dispositivo del usuario. En nuestra aproximación, se permite compartir los agentes utilizados para calcular la posición en la que se encuentra el dispositivo del usuario entre los distintos servicios, o la utilización de un agente mayordomo que permita adaptar la generación de los GUIs de las distintas aplicaciones ejecutadas en el dispositivo móvil, tanto a las preferencias de visualización del usuario, como a las características del dispositivo.

- *Adaptación al contexto*. Cada una de las arquitecturas presentadas se adapta automáticamente a los distintos dispositivos que se encuentren en el área de cobertura, debido a que los agentes modifican su comportamiento, de forma autónoma y automática, adaptándose así al entorno en el que se encuentren. Por ejemplo, cuando la arquitectura de generación de GUIs se utilice en un dispositivo que soporte Java Swing, se creará un GUI basado en los componentes proporcionados por este lenguaje. En cambio, si el dispositivo del usuario únicamente soporta HTML, se creará un GUI más simple utilizando únicamente los componentes gráficos disponibles en HTML. Del mismo modo, en el caso de los servicios basados en la localización, si detectamos un dispositivo que nunca había entrado en el área de cobertura, pero ya teníamos un mapa de potencias generado para otro dispositivo de la misma marca y modelo, podemos utilizar este mapa de potencias para calcular la posición del nuevo dispositivo. No obstante, si no se poseen datos previos para ningún dispositivo de esa marca y modelo, entonces pode-

mos utilizar el mapa de potencias que más se aproxime a la potencia de emisión de dicho dispositivo.

- *Modularidad y facilidad de expansión*. Hemos desarrollado tres arquitecturas basadas en agentes, donde cada uno de los agentes tiene una función especifica y que, colaborando entre sí, proporcionan distintos servicios inalámbricos a los usuarios de dispositivos móviles. El desarrollo de nuevos servicios basados en: 1) *un GUI adaptativo*, implica una colaboración con los agentes diseñados (concretamente con el agente Adus); 2) *la localización*, será necesario utilizar la posición calculada por el agente *Gestor de la Base de Datos de Localización* para otros fines; y 3) *el acceso a nuevas bases de conocimiento*, bastará permitir que sean analizadas por los agentes que constituyen la arquitectura diseñada.

Finalmente, las aproximaciones basadas en agentes proporcionan también otra ventaja, y es que *un único place puede ser utilizado por las distintas aplicaciones basadas en agentes*, evitando de esta forma sobrecargar el dispositivo del usuario. Si se añaden nuevas funcionalidades, por ejemplo mediante la incorporación de nuevos agentes, será necesario instalar nuevo software en el dispositivo del usuario porque los agentes de los nuevos servicios viajarán desde un servidor remoto al *place* existente en el dispositivo del usuario.

9.2. Evaluación de resultados

El trabajo desarrollado en esta tesis ha sido presentado y publicado en varias conferencias y revistas de ámbito nacional e internacional. Siguiendo el mismo criterio que en el resto de esta tesis, esta subsección la estructuraremos de acuerdo al contexto de los distintos casos de estudio y servicios diseñados para validar las ventajas de las aproximaciones basadas en agentes para el diseño y desarrollo de servicios de datos en entornos inalámbricos y distribuidos:

- *Servicio de Recuperación de Software*. En [MIG00c] se presentó una primera aproximación en la que se muestra la arquitectura del sistema, aunque posteriormente se mejoró y presentó mejorada en [MRIG02b]. La mejora de la arquitectura se basó en la incorporación del agente *Actualizador de catálogos* permitiendo así independizar la visualización de los catálogos, de su actualización, y haciendo más robusto el sistema. En [MIG00a] se detallaron los dos tipos de conocimiento que posee el agente *Alfredo*. En [MRIG02a] se presentaron los distintos mecanismos utilizados por los agentes para calcular la velocidad real de la red de comunicaciones (no la velocidad teórica), además de un mecanismo para estimar el número de reintentos que tendrá que realizar un agente cuando intente establecer una comunicación remota. En [MRIG02b], además de incorporar al agente Actualizador de Catálogos a la arquitectura del Servicio de Recuperación de Software, se detallaron los distintos algoritmos de poda que puede seleccionar el agente *Gestor de Software*. Todo lo anterior se recogió en un artículo publicado en la revista *Transactions on Autonomous and Adaptive Systems* [MIRG06], donde además se incorporó también un nuevo mecanismo para la selección automática del nivel de

detalle mostrado al usuario y que contribuye a reducir los costes de ejecución del servicio, avalado por las correspondientes pruebas analíticas y empíricas. Posteriormente, el análisis del SRS ha sido ampliado con nuevas pruebas empíricas que avalan nuestra aproximación, basada en agentes móviles inteligentes [MRM07, MMR08].

- *Servicio de bibliotecas digitales*. Una primera versión del agente Bib2DB se presentó en [RM01a], siendo seleccionada para su publicación en una revista [RM01b]. En esta publicación se mostraron las bondades de las arquitecturas basadas en agentes usadas para la recuperación de información en entornos distribuidos. Posteriormente la arquitectura fue modificada para proporcionar una mayor versatilidad y su intercomunicación con otros sistemas, por lo que la arquitectura se enriqueció con un conjunto de servicios web [RM02] y con técnicas que permitían no solamente la recuperación de referencias bibliográficas, sino que además facilitaban la actualización de las mismas. La actualización de las referencias bibliográficas es una tarea muy común en el tratamiento de publicaciones de investigación, debido a que inicialmente la publicación es aceptada, pero sólo posteriormente se conocen los datos definidos como el ISBN, la fecha de publicación, etc. Del mismo modo, se enriqueció la arquitectura para permitir la realización de búsquedas semánticas mediante la utilización de ontologías. Finalmente, en [RTIM05] se modificó la inteligencia del agente *Bib2DB* dotándolo de un comparador de referencias bibliográficas adaptativo. Este comparador se adapta automáticamente a la información que contienen los depósitos de referencias bibliográficas analizados, mediante la utilización de una media ponderada que actualiza los pesos dinámicamente y dependiendo de la información disponible.

- *Servicios de acceso a datos en contextos de ejecución dinámicos*. Se extendió también el SRS mediante la utilización de interfaces adaptativos [MRM04], que permiten monitorizar el comportamiento del usuario [MRM05] gracias a la utilización de un agente especializado en la generación de interfaces gráficos adaptados al dispositivo concreto del usuario. Por otra parte, se presentaron en dos conferencias de ámbito nacional: un trabajo que estudiaba los mecanismos necesarios para poder ejecutar una plataforma de agentes móviles en un PDA [RMA03], y un segundo trabajo que estudiaba una aproximación para calcular la posición del dispositivo del usuario utilizando la potencia de la señal recibida por las estaciones base que le dan cobertura [RMG05]. En [RMG05], se presentó una arquitectura que facilita el desarrollo de servicios dependientes de la localización mediante la creación dinámica de una infraestructura que permite la localización de dispositivos móviles en el interior de edificios, permitiendo así optimizar la utilización de las comunicaciones y su reutilización para distintos servicios. Además, se presentó la arquitectura de varios servicios (concretamente la localización de dispositivos móviles en un edificio, y un servicio de música que sigue al usuario), permitiendo demostrar las ventajas de la arquitectura y diseño adoptados. Del mismo modo, se diseñó un mecanismo basado en un sistema multiagente, que permite a usuarios inexpertos la utilización de un sistema de información global utilizando un vocabulario heterogéneo y que accede a varias ontologías [RMBI05].

Igualmente, queremos destacar la selección como finalista del servicio SRS en el concurso celebrado por la Cátedra Telefónica para Internet de Nueva Generación de la Universidad

Politécnica de Madrid y titulado "Premio Nuevas Aplicaciones para Internet 2004" (el diploma obtenido se muestra en la Figura 9.1). El Jurado evaluó las propuestas, de acuerdo con los siguientes criterios: originalidad, innovación, viabilidad práctica, aplicabilidad e impacto industrial, y seleccionó como finalistas las aproximaciones mostradas en la Figura 9.2, de entre las 19 propuestas presentadas al concurso. El acto tuvo lugar el día 2 de Diciembre de 2004, en el Salón de Grados de la Escuela Técnica Superior de Ingeniería de Telecomunicaciones de Madrid. Esta selección avala nuestra propuesta como una aproximación válida para un proyecto empresarial dentro del campo de las telecomunicaciones.

Figura 9.1: Diploma obtenido como finalista del Premio Nuevas Aplicaciones para Internet, 2004

9.3. Posibles ampliaciones y trabajo futuro

Para finalizar, en esta sección vamos a presentar algunas mejoras y trabajo futuro que consideramos podrían desarrollarse en el contexto de los sistemas multiagente, independientemente del entorno de ejecución, y en los casos de estudio presentados. Mientras que en el caso de los sistemas multiagente trataremos cuestiones generales, en el caso de los contextos de estudio concretos nos centraremos tanto en la propuesta de nuevas y mejoradas funcionalidades, como en aspectos de diseño.

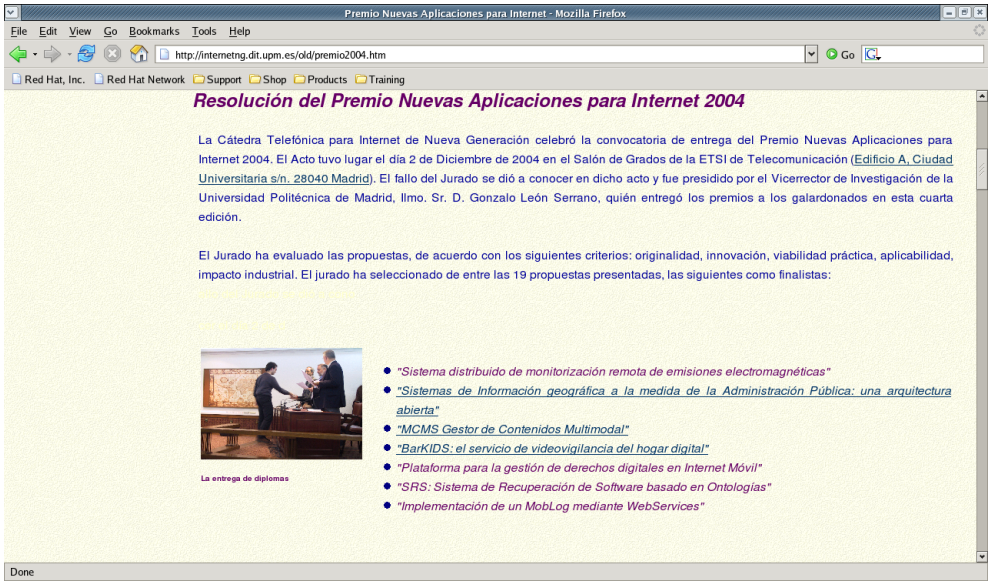


Figura 9.2: Finalistas Premio Nuevas Aplicaciones para Internet 2004

9.3.1. Sistemas multiagente

Las posibles vías de ampliación que podrían ser realizadas en el contexto de los sistemas multiagente, se dirigen a incrementar las posibilidades de cooperación entre los distintos sistemas o servicios, y a aumentar la robustez de las arquitecturas basadas en agentes.

- Lenguajes de comunicación entre agentes. Actualmente se están estudiando nuevas aproximaciones para permitir la interoperabilidad entre agentes [BGIB07, BGIS08]. Sin embargo, consideramos que sería necesario considerar la semántica en los mensajes intercambiados entre los agentes. Esto permitiría detectar y resolver las posibles ambigüedades que pueden aparecer cuando agentes desarrollados por distintas entidades deban colaborar para realizar una tarea.

- Arquitecturas Peer-to-Peer. Otra vía de trabajo futuro consiste en estudiar los beneficios que pudiera aportar la adopción de arquitecturas Peer-to-Peer [ATS04] basadas en agentes [BMM02], permitiendo así el diseño y desarrollo de arquitecturas basadas en agentes móviles inteligentes implementadas utilizando distintas plataformas de agentes, o incluso, la interacción de sistemas basados en agentes desarrollados de forma independiente por distintas organizaciones y/o autores [NSS03]. También cabe destacar que el desarrollo de arquitecturas peer-to-peer puede facilitar el desarrollo de sistemas altamente escalables y dinámicos, proporcionando mayor robustez, y facilitando la replicación de los servicios disponibles [ATS04].

9.3.2. Servicio de recuperación de software

Como líneas de trabajo futuro en el SRS, hemos considerado ampliar las capacidades del SRS para permitir la ejecución de servicios remotos [PMI03], además de en un entorno dinámico con un vocabulario abierto:

- Ejecución de servicios remotos. Mediante esta ampliación, el SRS podrá ofrecer a sus usuarios, no sólo una forma de encontrar y recuperar nuevo software, sino también la posibilidad de solicitar la ejecución remota de tareas [PMI03]. Estas tareas serán realizadas por un conjunto de agentes especializados que cooperarán y adaptarán autónomamente su comportamiento. Por ejemplo, los agentes podrán decidir si una tarea debe ser ejecutada en el dispositivo del usuario o en un servidor de altas prestaciones. Por lo tanto, será necesario extender el Servicio de Recuperación de Software con una ontología de servicios software que facilite la ejecución y búsqueda de los mismos, considerando también las características del dispositivo del usuario y los recursos necesarios para su ejecución. Del mismo modo, deberán diseñarse un conjunto de agentes móviles que ayuden al usuario en la ejecución de los servicios disponibles.

- Adaptación a contextos abiertos. En esta vía de ampliación, pretendemos considerar no únicamente la búsqueda de software, sino ampliarlo a cualquier tipo de búsqueda, considerando para ello un vocabulario abierto, como sucede en los sistemas de información globales [MI01, ACMe$^+$04, ACCM$^+$04]. De esta forma será necesario permitir que el usuario pueda especificar no sólo un conjunto de restricciones sobre un software, sino que pueda preguntar acerca de cualquier tema utilizando el vocabulario apropiado en cada caso. Para esta ampliación se deberá considerar la incorporación en tiempo de ejecución, de nuevos conceptos a la ontología u ontologías gestionadas por el sistema, por consiguiente, deberá basarse en una arquitectura de agentes desplegada dinámicamente. Siguiendo esta aproximación, los agentes descubrirán los depósitos de datos disponibles, especificarán las ontologías para describir e interpretar estos depósitos, y proporcionarán mecanismos de acceso a los datos subyacentes.

9.3.3. Servicio de bibliotecas digitales

En el caso de estudio presentado, se ha considerado la heterogeneidad de los dispositivos de los usuarios que acceden al sistema, y también que los distintos depósitos de datos que contienen las referencias bibliográficas se encuentren distribuidos en general. Sin embargo, también habría que considerar otras cuestiones, tales como:

- Bibliotecas digitales distribuidas. Proponemos estudiar el diseño y desarrollo de una biblioteca digital constituida mediante el acceso a distintas bibliotecas digitales, que a su vez pueden estar distribuidas o no. Consideramos que para facilitar una integración automática de las distintas bibliotecas digitales sería muy interesante diseñar este sistema mediante una arquitectura Peer-to-Peer basada en agentes móviles inteligentes. Del mimo modo, las distintas bibliotecas que vayan a ser integradas, deberán proporcionar un interfaz descrito semánticamente, o diseñarse un mecanismo de generación

automática de interfaces con una descripción semántica asociada para el contexto de bibliotecas digitales.

- Recopilación y clasificación automática de referencias bibliográficas y publicaciones. Consideramos que el sistema podría ser capaz de buscar nuevas publicaciones utilizando técnicas de recuperación de información [BYRN99], basadas en una arquitectura de agentes creada dinámicamente, a medida que se descubran nuevos depósitos bibliográficos. La técnica utilizada para realizar este proceso estaría basada en un conjunto agentes móviles inteligentes, que analizarían localmente (siempre que fuese posible) los distintos depósitos de datos, y posteriormente colaborarían realizando autónomamente una integración automática de la información, evitando así la aparición de información duplicada y de inconsistencias.

Mediante el beneficio obtenido con todas estas posibles ampliaciones, se lograría aumentar la interoperabilidad de la arquitectura propuesta en esta tesis, facilitando así un punto de acceso común tanto a personas como a agentes inteligentes.

9.3.4. Servicios de acceso a datos en entornos de ejecución dinámicos

Como puede observarse, todas las ampliaciones propuestas para el contexto de los servicios de acceso a datos en entornos dinámicos, están orientadas a incrementar la inteligencia y autonomía de los agentes. En este contexto nuestro trabajo futuro se centraría en:

- Descubrimiento automático de las capacidades de visualización de los dispositivos electrónicos. Con el objetivo final de diseñar un sistema, basado en agentes móviles, capaz de descubrir automáticamente el tipo de interfaces gráficos de usuario que puedan desplegarse en un dispositivo móvil. Los agentes deberán considerar tanto las restricciones hardware como las restricciones software a tener en cuenta para la generación de un GUI.

- Aprendizaje automático del agente gestor de la base de datos de localización. En el prototipo desarrollado en esta tesis, el agente gestor de localización utiliza la potencia de la señal de emisión para calcular la posición del dispositivo móvil, aunque esta puede variar dependiendo del dispositivo. Por lo tanto, sería interesante que los agentes que constituyen el sistema pudieran adaptar su comportamiento de forma automática al identificar el tipo de dispositivo ante el que se encuentran.

Cabe destacar que los beneficios derivados de la incorporación de todas estas ampliaciones, se concretarían también en el incremento (o mejora) de la capacidad de aprendizaje de los diferentes agentes que constituyen cada una de las arquitecturas propuestas en esta tesis.

Publicaciones relevantes relacionadas con la tesis

Revistas de ámbito internacional

[MIRG06] E. Mena, A. Illarramendi, J.A. Royo y A. Goñi, A Software Retrieval Service based on Adaptive Knowledge-Driven Agents for Wireless Environments. Transactions on Autonomous and Adaptive Systems (TAAS), 1(1):67-90, ACM, ISSN 1556-4665, September 2006.

Congresos de ámbito internacional

[MRM07] N. Mitrovic, J.A. Royo and E. Mena, Performance Analysis of an Adaptive User Interface System Based on Mobile Agents. Engineering Interactive Systems. Conference on Design Specification and Verification of Interactive Systems. 14th DSV-IS Conference Salamanca (Spain), Springer Verlag Lecture Notes in Computer Science LNCS 2844, ISBN 978-3-540-92697-9, ISSN 0302-9743, March 2007.

[RMBI05] J.A. Royo, E. Mena, J. Bernad and A.Illarramendi. Searching the Web: From Keywords to Semantic Queries. *Third International Conference on Information Technology and Applications (ICITA'05)*, Sydney (Australia), IEEE Computer Society, ISBN 0-7695-2316-1, pp. 244-249, July 2005.

[MRM04] N. Mitrovic, J.A. Royo and E. Mena. ADUS: Indirect Generation of User Interfaces on Wireless Devices. *Fifteenth International Workshop on Database and Expert Systems Applications (DEXA'2004), Seventh International Workshop Mobility in Databases and Distributed Systems (MDDS'2004)*, Zaragoza (Spain), IEEE Computer Society, ISBN 0-7695-2195-9, ISSN 1529-4188, September 2004.

[MRIG02a] E. Mena, J.A. Royo, A. Illarramendi and A. Goñi. Adaptable Software Retrieval Service for Wireless Environments Based on Mobile Agents. *2002 International*

Conference on Wireless Networks (ICWN'02), Las Vegas, USA, CSREA Press, ISBN 1-892512-30-0, pp. 116-124, June 2002.

[MRIG02b] E. Mena, J.A. Royo, A. Illarramendi and A. Goñi. An Agent-based Approach for Helping Users of Hand-Held Devices to Browse Software Catalogs. *Cooperative Information Agents VI, 6th International Workshop CIA 2002*, Madrid (Spain), Lecture Notes on Artificial Intelligence (LNAI), ISBN 3-540-44173-5, pp. 51-65, September 2002.

Capítulos de libro de ámbito internacional

[MMR08] N. Mitrovic, E. Mena and J.A. Royo, Adaptive Interfaces in Mobile Environments, Handbook of Research on User Interface Design and Evaluation for Mobile Technology, J. Lumsden (ed.), Information Science Reference (formerly Idea Group Reference), ISBN 978-1-59904-871-0, pp. 302-317, January 2008.

Congresos de ámbito nacional

[RTIM05] J.A. Royo, R. Trillo, S. Ilarri y E. Mena. Gestión y detección de inconsistencias en colecciones de referencias bibliográficas. *V Jornadas de Bibliotecas Digitales (JBiDi'2005), Granada (Spain)*. Thomson, ISBN 84-9732-453-6, pp 29-36, Septiembre 2005.

[RMG05] J.A. Royo, E. Mena y L.C. Gallego. Locating Users to Develop Location-Based Services in Wireless Local Area Networks. *I Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'2005), Granada (Spain)*. Thomson, ISBN 84-9732-442-0, pp 471-478, Septiembre 2005.

[MRM05] N. Mitrovic, J.A. Royo y E. Mena. Adaptive User Interfaces Based on Mobile Agents: Monitoring the Behavior of Users in a Wireless Environment. *I Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'2005), Granada (Spain)*. Thomson, ISBN 84-9732-442-0, pp 371-378, Septiembre 2005.

[RMA03] J.A. Royo, E. Mena and R. Acero. Agentes Java para Dispositivos Móviles con Conexión Bluetooth. *I Congreso Java Hispano, Madrid (Spain)*, Universidad Carlos III, ISBN 84-688-8080-0, pp. 44-55, Octubre 2003.

[RM02] J.A. Royo and E. Mena. Gestión de Bibliotecas Digitales de Publicaciones de Investigación. *III Jornadas de Bibliotecas Digitales (JBIDI'2002)*, El Escorial (Madrid), Spain, ISBN 84-688-0205-0, José H. Canós, Purificación García (ed.), Universidad Politécnica de Madrid, pp. 77-86, Noviembre 2002.

[RM01a] J.A. Royo and E. Mena. Uso de Agentes Móviles para la Búsqueda y Recuperación de Información Bibliográfica. *II Jornadas de Bibliotecas Digitales (JBIDI'2001)*, Almagro (Ciudad Real), Spain, ISBN 84-699-6276-0, P. De la Fuente, A. Pérez (ed.), Universidad de Castilla-La Mancha, pp. 223-234, Noviembre 2001.

Revista de ámbito nacional

[RM01b] J.A. Royo y E. Mena. Uso de Agentes Móviles para la Búqueda y Recuperación de Información Bibliográfica. *Revista Interamericana de Nuevas Tecnologías de la Información*, ISSN 0122-3356, 6(4):52-63, Organización de los Estados Americanos (OEA), Octubre-Diciembre 2001.

Bibliografía

[ABCB+04] A. Agostini, C. Bettini, N. Cesa-Bianchi, D. Maggiorini, D. Riboni, M. Ruberl, C. Sala, and D. Vitali. Towards highly adaptive services for mobile computing. In *Proceedings of IFIP TC8 Working Conference on Mobile Information Systems (MOBIS)*, pp. 121–134. Springer, 2004.

[ABJ+98] R. Aylett, F. Brazier, N. Jennings, M. Luck, C. Preist, and H. Nwana. Agent systems and applications. *Knowledge Engineering Review*, 13(3):303–308, 1998.

[ACCM+04] K. Aberer, T. Catarci, T. Cudré-Mauroux, T. Dillon, S. Grimm, M. Hacid, A. Illarramendi, M. Jarrar, V. Kashyap, M. Mecella, E. Mena, E.J. Neuhold, A.M. Oukel, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, R. Studer, and O. de Troyer. Emergent semantics systems. In *Semantics of a Networked World. Semantics for Grid Databases (ICSNW04), Paris (France)*, pp. 14–43. Springer-Verlag Lecture Notes in Computer Science (LNCS 2973), ISSN 0302-9743, ISBN 3-540-23609-0, June 2004.

[ACM01] D. Alur, J. Crupi, and D. Malks. *Core J2EE Patterns. Best Practices and Design Strategies*. Prentice Hall, ISBN: 0-13-064884-1, 2001.

[ACMe+04] K. Aberer, P. Cudré-Mauroux, A.M. Ouksel (editors), M. Hacid T. Catarci, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E.J. Neuhold, O. De Troyer, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, and R. Studer. Emergent semantics: Principles and issues. In *Invited paper at 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004), Jeju Island (Korea)*, pp. 25–38. Springer-Verlag Lecture Notes in Computer Science (LNCS 2973), ISSN 0302-9743, ISBN 3-540-21047-4, March 2004.

[Acr05] Inc. Acronymics. Why, when, and where to use software agents, 2005. http://www.agentbuilder.com/Documentation/whyAgents.html, [Last accessed: April 2009].

[Ade04] G. A. Aderounmu. Performance comparison of remote procedure calling and mobile agent approach to control and data transfer in distributed computing environment. *Journal of Network and Computer Applications*, 27(2):113–129, 2004.

[Adv05] Advanced Aviation Technology. GALILEO: European Satellite Navigation System, 2005. http://www.aatl.net/publications/galileo.htm, [Last accessed: April 2009].

[Adv09] Advanced Data Management Tecnologies Laboratory, 2009. http://db.cs.pitt.edu/group/publications [Last accessed: April 2009].

[And01] C. Andersson. *GPRS and 3G Wireless Applications*. Wiley, 2001.

[ANS06] American National Standard. Knowledge Interchange Format, 2006. http://logic.stanford.edu/kif/dpans.html, [Last accessed: April 2009].

[AO99] Y. Aridor and M. Oshima. Infrastructure for Mobile Agents: Requirements and Design. In *Second International Workshop on Mobile Agents (MA'98), Stuttgart, Germany*, pp. 38–49. Springer, 1999.

[APB+99] M. Abrams, C. Phanouriou, A.L. Batongbacal, S.M. William, and J.E. Shuster. UIML: An Appliance-Independent XML User Interface Language. In *WWW8 / Computer Networks 31(11-16): 1695-1708*, 1999.

[AS99] G. Amato and U. Straccia. User Profile Modeling and Applications to Digital Libraries. In *Lecture Notes in Computer Science, ISSN 0302-9743*, volume 1696/1999. Springer Berlin/Heidelberg, 1999.

[ATS04] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[AvH03] G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In S. Staab and R. Studer, editors, *Handbook on Ontologies in Information Systems*. Springer-Verlag, 2003.

[AVS05] AVS. AVS Capture Frame Format, 2004-2005.

[BAV05] R. Buyya, D. Abramson, and S. Venugopal. The Grid Economy. *Proceedings of the IEEE*, 93(3):698–714, March 2005.

[BBD+06] R. Bordini, L. Braubach, M. Dastani, A. El Fallah Seghrouchni, J. Gomez-Sanz, J. Leite, G. O'Hare, A. Pokahr, and A. Ricci. A Survey of Programming Languages and Platforms for Multi-Agent Systems. *Informatica*, 30(1), 2006.

[BBIM93] B. R. Badrinath, Ajay V. Bakre, Tomasz Imielinski, and R. Maramtz. Handling Mobile Clients: A Case for Indirect Interaction. In *Fourth Workshop on Workstation Operating Systems (WWOS'93)*, pp. 91–97, 1993.

[BBMAR89] A. Borgida, R.J. Brachman, D.L. McGuinness, and L.A. Resnick A. Resnick. CLASSIC: A Structural Data Model for Objects. In *Proceedings ACM SIGMOD-89, Portland, Oregon*, pp. 59–67, 1989.

[BBP00] P. Bahl, A. Balachandran, and V. Padmanabhan. Enhancements to the RADAR User Location and Tracking System. Technical report, Microsoft Research Technical Report, February 2000.

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D.Ñardi, and P. Pastel-Scheneider. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press, ISBN 0-521-78176-0, 2003.

[BD03] B. Bruegge and A. Dutoit. *Object-Oriented Software Engineering Using UML, Patterns and Java*. Pearson Education, 2003.

[BDBW97] J. Bradshaw, S. Dutfield, P. Benoit, and J. D. Woolley. *KAoS: Towards an Industrial-Strength Open Agent Architecture*, chapter 17. AAAI Press/The MIT Press, 1997.

[Bea04] J. Broekstra and et al. Bibster - A Semantics-Based Bibliographic Peer-to-Peer System. In *Proceedings of SemPGRID '04, 2nd Workshop on Semantics in Peer-to-Peer and Grid Computing*, pp. 3–22, New York, USA, May 2004.

[Bea05] N. Beagrie. Plenty of Room at the Bottom? Personal Digital Libraries and Collections. *D-Lib Magazine, ISSN 1082-9873*, 11(6), June 2005.

[BEF03] J. Bruce, J. Ellis, and M. Fisher. *JDBC API Tutorial and Reference*. Addison Wesley Professional, 2003.

[BG99] S. Baase and A. Van Gelder. *Computer Algorithms: Introduction to Design and Analysis*. Addison Wesley, 1999.

[BGI99] J.M. Blanco, A. Goñi, and A. Illarramendi. Mapping among Knowledge Bases and Data Repositories: Precise Definition of its Syntax and Semantics. *Information Systems*, 24(4):275–301, 1999.

[BGIB07] J. Bermúdez, A. Goñi, A. Illarramendi, and M.I. Bagües. Interoperation among Agent Based Information Systems through a Communication Acts Ontology. *Information Systems Journal*, 32, 8:1121–1144, December 2007.

[BGIS08] J. Bermúdez, A. Goñi, A. Illarramendi, and S. Santini. Is an OWL Ontology Adequate for Foreign Software Agent Communication? *Applied Ontology*, 2008.

[BGM+99] B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko, and D. Rus. Mobile Agents in Distributed Information Retrieval. In Matthias Klusch, editor, *Intelligent Information Agents*. Springer, 1999.

[BGN89] H. Beck, H. Gala, and S.Ñavathe. Classification as a Query Processing Technique in the CANDIDE Semantic Model. In *Proceedings of the IEEE International Conference on Data Engineering*, 1989.

[BGZ04] F. Bergenti, M. P. Gleizes, and F. Zambonelli. *Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook*. Kluwer Academic Publishers, 2004.

[BHM+04] D. Booth, H. Haas, F. McCabe, E.Ñewcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture, February 2004. http://www.w3.org/TR/ws-arch/#whatis, [Last accessed: April 2009].

[BJW04] S. Bussmann, N. R. J., and M. Wooldridge. *Multiagent Systems for Manufacturing Control - A Design Methodology*. Springer, 2004.

[BK05] P. Braun and S. Kern. Towards Adaptive Migration Techniques for Mobile Agents. In *Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), Utrecht, The Netherlands*, pp. 1239–1240. ACM Press, 2005.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientic American*, May 2001.

[Blu03] BlueZ Project. Official Linux Bluetooth Protocol Stack, 2003. http://bluez.sourceforge.net/, [Last accessed: April 2009].

[BM99] C. Bäumer and T. Magedanz. Grasshopper - A Mobile Agent Platform for Active Telecommunication. In *Intelligent Agents for Telecommunication Applications (IATA'99), Stockholm, Sweden*, pp. 19–32. Springer, 1999.

[BM02] M. Bilenko and R.J. Mooney. Learning to Combine Trained Distance Metrics for Duplicate Detection in Databases. Technical Report Technical Report AI 02-296, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX, February 2002.

[BMG+04] P. Braun, I. Müller, S. Geisenhainer, V. Schau, and W. Rossak. A Service-oriented Software Architecture for Mobile Agent Toolkits. In *11th IEEE International Conference on the Engineering of Computer-Based Systems (ECBS'04), Brno, Czech Republic*, pp. 550–556. IEEE Computer Society, 2004.

[BMM02] O. Babaoglu, H. Meling, and A. Montresor. Anthill: A Framework for the Development of Agent-Based Peer-to-Peer systems. In *22nd International Conference on Distributed Computing Systems, ISSN: 1063-6927, ISBN: 0-7695-1585-1*, pp. 15–22, 2002.

[BMP+04] J. Beaver, N. Morsillo, K. Pruhs, P.K. Chrysanthis, and V. Liberatore. Scalable Dissemination: What's Hot and What's Not. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB '04)*, pp. 31–36, New York, NY, USA, 2004. ACM Press.

[BMS+06] P. Braun, I. Müller, T. Schlegel, S. Kern, and V. Schau5. Tracy: An Extensible Plugin-Oriented Software Architecture for Mobile Agent Toolkits. In *Whitestein Series in Software Agent Technologies and Autonomic Computing, Software Agent-Based Applications, Platforms and Development Kits, ISBN: 978-3-7643-7347-4*, pp. 357–381. Springer-Verlag, 2006.

[BN01] L. Bettini and R. De Nicola. Translating Strong Mobility into Weak Mobility. In *5th International Conference on Mobile Agents (MA'01), Atlanta, Georgia, USA*, pp. 182–197. Springer, 2001.

[BNL02] L. Bettini, R. De Nicola, and M. Loreti. Software update via mobile agent based programming. In *ACM symposium on Applied computing (SAC'02), Madrid, Spain*, pp. 32–36. ACM Press, 2002.

[Bor92] A. Borgida. From Type Systems to Knowledge Representation: Natural Semantics Specifications for Description Logics. *International Journal on Intelligent and Cooperative Information Systems*, 1(1), March 1992.

[BP00] P. Bahl and V.N. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. *Proceedings of IEEE Infocom 2000, Tel-Aviv, Israel*, March 2000.

[BPR01] F. Bellifemine, A. Poggi, and G. Rimassa. JADE: A FIPA2000 Compliant Agent Development Environment. In *Fifth International Conference on Autonomous agents (AGENTS'01), Montreal, Quebec, Canada*, pp. 216–217. ACM Press, 2001.

[BPW98] A. Bieszczad, B. Pagurek, and T. White. Mobile Agents for Network Management. *IEEE Communications Surveys and Tutorials*, 1(1), 1998.

[BR05] P. Braun and W. R. Rossak. *Mobile Agents-Basic Concept, Mobility Models, and the Tracy Toolkit*. Morgan Kaufmann, 2005.

[Bra87] M. Bratman. *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[Bre97] P. Brenner. A Technical Tutorial on the IEEE 802.11 protocol. Research report, BreezeCOM Wireless Communications, 1997.

[BS85] R. Brachman and J. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, Feb 1985.

[BSBK01] A. Blandford, H. Stelmaszewska, and N. Bryan-Kinns. Use of Multiple Digital Libraries: A Case Study. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pp. 179–188, New York, NY, USA, 2001. ACM Press.

[BSC+01] J.M. Bradshaw, N. Suri, A.J. Cañas, R. Davis, K. Ford, R. Hoffman, R. Jeffers, and T. Reichherzer. Terraforming Cyberspace. *Computer*, 34(7):48–56, 2001.

[BW04] D.P. Buse and Q. H. Wu. Mobile Agents for Remote Control of Distributed Systems. *IEEE Transactions on Industrial Electronics*, 51(6):1142– 1149, 2004.

[BWM99] R. Braga, C. Werner, and M. Mattoso. Odyssey: A Reuse Environment based on Domain Models. In *Proceedings of IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSET'99), Richardson, Texas*, pp. 49–57, March 1999.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Adisson-Wesley, ISBN 0-201-39829-X, 1999.

[Can00] J.H. Canos. A Bibliography Manager for Microsoft Word. *ACM Crossroads Special Issue on Windows Programming, ISSN=1528-4980*, 6.4, June 2000. http://www.acm.org/crossroads/xrds6-4/bibword.html, [Last accessed: April 2009].

[Car02] R.S. Cardoso. Mobile Agents: A Key for Effective Pervasive Computing. In *Second Pervasive Computing Workshop of ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'02), Vancouver, British Columbia, Canada*, 2002.

[CFJ03] H. Chen, T. Finin, and A. Joshi. An Ontology for Context-Aware Pervasive Computing Environments. In *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03) Workshop on Information Integration on the Web (IIWeb'03), Acapulco (Mexico)*, August 2003.

[CFJ04] H. Chen, T. Finin, and A. Joshi. An Ontology for Context-Aware Pervasive Computing Environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3):197–207, May 2004.

[CG01] S. Chalmers and P.M.D. Gray. BDI Agents and Constraint Logic. *Journal of Artificial Intelligence and the Simulation of Behaviour, Special Issue on Agent Technology*, 1(1):21–40, 2001.

[CGLN99] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in Expressive Description Logics. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers (North-Holland), Amsterdam, 1999. To appear.

[CGMW02] R. Chinnici, M. Gudgin, J.J. Moreau, and S. Weerawarana. Web Services Description Language (WSDL) Version 1.2, 2002.

[CM06] M. Chao and L. Ming. GPS-GSM Mobile Navigator, February 2006. http://www.circuitcellar.com/library/print/0203/ChaoMing151/, [Last accessed: April 2009].

[CMMB05] L. Cavedon, Z. Maamar, D. Martin, and B. Benatallah. *Extending Web Services Technologies - The Use of Multi-Agent Approaches*. Springer, 2005.

[CMMS00] D. Chacón, J. McCormick, S. McGrath, and C. Stoneking. Rapid Application Development Using Agent Itinerary Patterns. Technical report, Technical Report number. 01-01, Lockheed Martin Advanced Technology Laboratories, March 2000.

[CNE06a] CNET Inc., 2006. http://www.shareware.com,[Last accessed: April 2009].

[CNE06b] CNET Inc., 2006. http://www.download.com, [Last accessed: April 2009].

[CNE06c] CNET Inc., 2006. http://www.gamecenter.com, [Last accessed: April 2009].

[CNNL98] J. C. Collis, D. T. Ndumu, H. S. Nwana, and L. C. Lee. The ZEUS Agent Building Toolkit. *BT Technology Journal*, 16(3):60–68, 1998.

[Com00] D.E. Comer. *Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture (4th Edition) (Hardcover)*. Prentice Hall, January 2000.

[Coo97] Oracle Coorp. Oracle mobile agents technnical product summary, June 1997. http://www.oracle.com/procucts/networking/mobile/agents/html.

[Cor01] Microsoft Corp. ASP.NET Mobile Web Development Overview, 2001. http://msdn.microsoft.com/en-us/library/ms178619.aspx, [Last accesed: December,2002].

[CTC04] J. Cao, D.C.K. Tse, and A.T.S. Chan. PDAgent: A Platform for Developing and Deploying Mobile Agent-Enabled Applications for Wireless Devices. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)*, pp. 510–517. IEEE Computer Society, 2004.

[CWW00] J.R. Chen, S.R. Wolfe, and S.D. Wragg. A Distributed Multi-Agent System for Collaborative Information Management and Sharing. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pp. 382–388, New York, NY, USA, 2000. ACM Press.

[DAR06] DARPA Agent Markup Language. http://www.daml.org, [Last accessed: April 2009], 2006.

[Dev93] P.T. Devanbu. Translating Description Logics to Information Server Queries. In *Proceedings of the ISMM International Conference on Information and Knowledge Management CIKM*, 1993.

[DH02] M. Degeratu and V. Hatzivassiloglou. Building Automatically a Business Registration Ontology. In *The Second National Conference on Digital Government (dg.o 2002), LA, CA*, May 2002.

[DIH00] C. Drummond, D. Ionescu, and R.C. Holte. A Learning Agent that Assists the Browsing of Software Libraries. *IEEE Transactions on Software Engineering*, 26(12):1179–1196, 2000.

[Dis09] Distributed Management of Data Laboratory, 2009. http://dmod.cs.uoi.gr/ [Last accessed: April 2009].

[DL01] M. D'Inverno and M. Luck. *Understanding Agent Systems*. Springer, 2001.

[DLNS96] F.M. Donini, M. Lenzerini, D.Ñardi, and A. Schaerf. Reasoning in Description Logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pp. 193–238. CSLI Publications, 1996.

[DMDH02] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to Map between Ontologies on the Semantic Web. In *In The Eleventh International WWW Conference, Hawaii, USA*, 2002.

[DMH+00] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The Semantic Web: The Roles of XML and RDF. *IEEE Internet Computing*, 04(5):63–74, 2000.

[DNMMS99] P. Dasgupta, N. Narasimhan, L. E. Moser, and P.M. Melliar-Smith. MAgNET: Mobile Agents for Networked Electronic Trading. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):509–525, 1999.

[DS01] M. Dikaiakos and G. Samaras. A Performance Analysis Framework for Mobile-Agent Systems. In *Revised Papers from the International Workshop on Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, volume 1887, pp. 180–187. Springer, 2001.

[dSCB+01] R.P. de Souza, M.N. Costa, R.M.M. Braga, M. Mattoso, and C.M.L. Werner. Software Components Reuse Through Web Search and Retrieval. In *International Workshop on Information Integration on the Web - Rio de Janeiro, RJ, Brasil*, 2001.

[Dun01] C.R. Dunne. Using Mobile Agents for Network Resource Discovery in Peer-to-Peer Networks. *ACM SIGecom Exchanges*, 2(3):1–9, 2001.

[Ear06] Earthweb & Sun Microsystems, 2006. http://www.gamelan.com, [Last accessed: April 2009].

[EFGK03] P.Th. Eugster, P.A. Felber, R. Guerraoui, and A.M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[eIML06] XIML (eXtensible Interface Markup Language), November 2006. http://www.ximl.org/, [Last accessed: April 2009].

[EOK04] P. Eaton, E. Ong, and J. Kubiatowicz. Improving Bandwidth Efficiency of Peer-to-Peer Storage. In *Proceedings of the 4th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'04)*, August 2004.

[ETSIE93] European Telecommunications Standards Institute (ETSI). European Digital Cellular Telecommunications System; Attachment Requirements for Global System for Mobile Communications (GSM) Mobile Stations; Access, november 1993.

[ETSIE98] European Telecommunications Standards Institute (ETSI). General Packet Radio Service (GPRS): Service Description, 1998.

[EVP00] J. Eisenstein, J. Vanderdonckt, and A. Puerta. Adapting to Mobile Contexts with User-Interface Modeling. *wmcsa*, 00:83, 2000.

[FDL07] GNU Free Documentation License. Handheld device, 2007. http://en.wikipedia.org/wiki/Mobile_device, [Last accessed: April 2009].

[FDL08a] GNU Free Documentation License. Boolean algebra (logic), 2008. http://en.wikipedia.org/wiki/Boolean_algebra_%28logic%29, [Last accessed: April 2009].

[FDL08b] GNU Free Documentation License. C (programming language), 2008. http://en.wikipedia.org/wiki/C_%28programming_language%29, [Last accessed: April 2009].

[FDL08c] GNU Free Documentation License. Digital AMPS, 2008. http://en.wikipedia.org/wiki/D-AMPS, [Last accessed: April 2009].

[FDL08d] GNU Free Documentation License. Integrated Digital Enhanced Network, 2008. http://en.wikipedia.org/wiki/IDEN, [Last accessed: April 2009].

[FDL08e] GNU Free Documentation License. Is-95, 2008. http://en.wikipedia.org/wiki/IS-95, [Last accessed: April 2009].

[FDL08f] GNU Free Documentation License. Notation3, 2008. http://en.wikipedia.org/wiki/Notation_3, [Last accessed: April 2009].

[FDL08g] GNU Free Documentation License. Personal Digital Cellular, 2008. http://en.wikipedia.org/wiki/Personal_Digital_Cellular, [Last accessed: April 2009].

[FDL08h] GNU Free Documentation License. Usenet, 2008. http://en.wikipedia.org/wiki/Usenet, [Last accessed: April 2009].

[FDL09] GNU Free Documentation License. JNLP: Java Networking Launching Protocol, 2009. http://es.wikipedia.org/wiki/JNLP, [Last accessed: April 2009].

[FFF+00] J. Frew, M. Freeston, N. Freitas, L. Hill, G. Janee, K. Lovette, R. Nideffer, T. Smith, and Q. Zheng. The Alexandria Digital Library Architecture. *International Journal on Digital Libraries, ISSN 1432-5012*, 2(3):259–268, 2000.

[FfIPA05] IEEE Foundation for Intelligent Physical Agents. The Foundation for Intelligent Physical Agents (FIPA), 2005. http://www.fipa.org/, [Last accessed: April 2009].

[FFMM94] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Third International Conference on Information and Knowledge Management (CIKM'94), Gaithersburg, Maryland, USA*, pp. 456–463. ACM Press, 1994.

[FG96] S. Franklin and A. Graesser. It is an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In *Third International Workshop on Agent Theories, Architectures and Languages, Budapest, Hungary*. Springer, 1996.

[FG99] Y. Fan and S. Gauch. Adaptive Agents for Information Gathering from Multiple, Distributed Information Sources. In *AAAI Symposyum on Intelligent Agents in Cyberspace, Palo Alto, California, USA*, 1999.

[FGBA96] A. Fox, S.D. Gribble, E.A. Brewer, and E. Amir. Adapting to Network and Client Variability via On-Demand Dynamic Distillation. *Proc. Seventh Intl. Conf. on Arch. Support for Prog. Lang. and Oper. Sys. (ASPLOS-VII), Cambridge, MA*, 1996.

[FGR02] S.P. Fonseca, M.L. Griss, and R.Letsinger. Agent Behavior Architectures a MAS Framework Comparison. In *First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Bologna, Italy*, pp. 86–87. ACM Press, 2002.

[FH03] E Friedman-Hill. *JESS in Action: rule-based systems in Java*. Manning Publications Co., 2003.

[FH07] E. Friedman-Hill. JESS, the Rule Engine for Java Platform, 2007. http://herzberg.ca.sandia.gov/jess/, [Last accessed: April 2009].

[Fie98] J. Fiedler. A Distributed Personalized Bews System Based on Mobile Agents. In *36th ACM Southeast Regional Conference, Marietta, Guam, USA*, pp. 130–135. ACM Press, 1998.

[fIPA98] FIPA Foundation for Intelligent Physical Agents. Agent Security Management. In *FIPA 98 Specification. Part 10, Version 1.0*, October 1998.

[FLM98] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World Wide Web: A Survey. In *SIGMOD Record*, 1998.

[FLS96] S. Falasconi, G. Lanzola, , and M. Stefanelli. Using Ontologies in Multi-Agent Systems. In *Tenth Knowledge Acquisition For Knowledge-Based Systems Workshop (KAW'96), Banff, Canada*, 1996.

[FM98] E.A. Fox and G. Marchionini. Toward a Worldwide Digital Library. *Commun. ACM*, 41(4):29–32, 1998.

[FPV98] A. Fuggetta, G. P. Picco, and G. Vigna. Understanding Code Mobility. *IEEE Transactions and Software Engineering*, 24(5):342–361, 1998.

[fRSSiTP98] European Institute for Research, Eurescom P712 Strategic Studies in Telecommunications (1998), and version A9 P815. Agent Based Computing - A Booklet for Executives, 1998. http://www.eurescom.de/~pub-deliverables/p800-series/P815/Booklet/!AGENT_B.pdf, [Last accessed: April 2009].

[FS04] J. Flinn and M. Satyanarayanan. Managing Battery Lifetime with Energy-Aware Adaptation. *ACM Transactions on Computer Systems (TOCS)*, 22(2):137–179, 2004.

[FTS+03] M. Fukuda, Y. Tanaka, N. Suzuki, L.F. Bic, and S. Kobayashi. A Mobile-Agent-Based PC Grid. In *Proceedings of the Autonomic Computing Workshop Fifth Annual International Workshop on Active Middleware Services (AMS'03), ISBN 0-7695-1983-0/01*, 2003.

[FWME00] R. S. Silva Filho, J. Wainer, E. Madeira, and C. Ellis. CORBA Based Architecture for Large Scale Workflow. *IEICE Transactions on Communication*, 83(5):988–998, 2000.

[GATTMed] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A Framework for Modeling and Evaluating Automatic Semantic Reconciliation. *Very Large Databases Journal*, to be published.

[GB03] F. Gustafsson and N. Bergman. *MATLAB® for Engineers Explained.* Springer-Verlag, 2003.

[GCK+01] R. Gray, G. Cybenko, D. Kotz, R. Peterson, and D. Rus. D'Agents: Applications and Performance of a Mobile-Agent System, 2001.

[GDCR98] R.S. Gray, D.Kotz, G. Cybenko, and D. Rus. D'Agents: Security in a Multiple-Language, Mobile-Agent System. *Lecture Notes in Computer Science*, 1419:154, 1998.

[GDP00] M. Goncalves, O.C.M.B. Duarte, and G. Pujolle. Evaluating the Network Performance Management based on Mobile Agents. In Eric Horlait, editor, *Second International Workshop on Mobile Agents for Telecommunication Applications (MATA'00), Paris, France*, pp. 95–102. Springer, 2000.

[Ges05] F. Gesellschaf. SeMoA - Secure Mobile Agents Project, 2005. http://www.semoa.org/, [Last accessed: April 2009].

[GF92] M. Genesereth and R. Fikes. Knowledge Interchange Format, version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[GG02] M. Gast and M.S. Gast. *802.11 Wireless Networks: The Definitive Guide (O'Reilly Networking).* O'Reilly; 1 edition, ISBN-10: 0596001835, ISBN-13: 978-0596001834, April 2002.

[GGMPW02] A. Graham, H. Garcia-Molina, A. Paepcke, and T. Winograd. Time as Essence for Photo Browsing Through Personal Digital Libraries. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pp. 326–335, New York, NY, USA, 2002. ACM Press.

[GHCN99] R. Ghanea-Hercock, J. C. Collis, and D. T. Ndumu. Co-operating Mobile Agents for Distributed Parallel Processing. In *Third Conference on Autonomous Agents (AGENTS'99), Seattle, Washington*, pp. 398–399. ACM Press, 1999.

[GIM+01] A. Goñi, A. Illarramendi, E. Mena, Y. Villate, and J. Rodriguez. ANTARCTICA: A Multiagent System for Internet Data Services in a Wireless Computing Framework. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems, Scottsdale, Arizona (USA)*, pp. 119–135. Lecture Notes in Computer Science (LNCS 2538) 2002, ISBN 3-540-00289-8, October 2001.

[GKCR00] R. S. Gray, D. Kotz, G. Cybenko, and D. Rus. Mobile Agents: Motivations and State-of-the-Art Systems. Technical report, Dartmouth College, 2000.

[GKN+96] R.S. Gray, D. Kotz, S.Ñog, D. Rus, and G. Cybenko. Mobile Agents for Mobile Computing. Technical Report TR96-285, Computer Science Department, Dartmouth College, Hanover, NH, 1996.

[GKP+01] R. S. Gray, D. Kotz, R. A. Peterson, J. Barton, D. Chacón, P. Gerken, M. Hofmann, J. Bradshaw, M. Breedy, R. Jeffers, and N. Suri. Mobile-Agent versus Client/Server Performance: Scalability in an Information-Retrieval Task. In *Fifth IEEE International Conference on Mobile Agents (MA'01), Atlanta, Georgia, USA*, volume 2240, pp. 229–243. Springer, 2001.

[Goo06] Google Inc., 2006. http://www.google.com/, [Last accessed: April 2009].

[Goñ95] A. Goñi. *Sistemas de Bases de Datos Federadas Basados en Sistemas Terminologicos: Tecnicas de Procesamiento Eficiente de Preguntas y Mantenimiento de la Consistencia*. PhD thesis, Basque Country University, December 1995.

[Gra96] R. S. Gray. Agent Tcl: A Flexible and Secure Mobile-Agent System. In M. Diekhans and M. Roseman, editors, *Fourth Annual Tcl/Tk Workshop (TCL 96)*, pp. 9–23, Monterey, CA, 1996.

[Gro99] Object Management Group, 1999. http://www.omg.org/technology/documents/formal/corba_iiop.htm, [Last accessed: April 2009].

[Gro06] Object Management Group, 2006. http://www.omg.com, [Last accessed: April 2009].

[Gru92] T. Gruber, 1992. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html, [Last accessed: April 2009].

[Gru93] T. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition, An International Journal of Knowledge Acquisition for Knowledge-Based Systems*, 5(2):199–220, June 1993.

[Gru07] T. Gruber, 2007. http://tomgruber.org/writing/ontology-definition-2007.htm, [Last accessed: April 2009].

[GS04] R. Gupta and A.K. Sonani. CompuP2P: An Architecture for Sharing of Computing Resources in Peer-to-Peer Networks with Selfish Nodes. In *Proceedings of Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[GSS06] R. Gupta, V. Sekhri, and A.K. Somani. CompuP2P: An Architecture for Internet Computing Using Peer-to-Peer Networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(11):1306–1320, 2006.

[Hal90] J.Y. Halpern. An Analysis of First-Order Logics of Probability. *Artificial Intelligence*, 46:311–350, 1990.

[Hal00] M. Hall. *Core Servlets and Java Server Pages(JSP)*. Prentice Hall PTR/Sun Microsystems Press, May 2000.

[HB00] Alex L. G. Hayzelden and Rachel A. Bourne. *Agent Technology for Communication Infrastructures*. John Wiley & Sons, 2000.

[HBGM+97] J. Hammer, M. Breunig, H. Garcia-Molina, S. Nestorov, V. Vassalos, and R. Yerneni. Template-Based Wrappers in the TSIMMIS System. In *Proceedings of the Twenty-Sixth SIGMOD International Conference on Management of Data, Tucson, Arizona*, May 1997.

[HCK97] C. Harrison, D. Chess, and A. Kershenbaum. Mobile agents: Are they a good idea? *In Mobile Object Systems: Towards the Programmable Internet*, pp. 46–48, 1997.

[HDB02] M.T. Hagan, H.B. Demuth, and M.H. Beale. *Neural Network Design*. Martin Hagan, ISBN: 0971732108, January 2002.

[Hen01] J. Hendler. Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2):30–37, 2001.

[Hes06] A. Hess. An Iterative Algorithm for Ontology Mapping Capable of Using Training Data. In *3rd European Semantic Web Conference (ESWC'06), Budva Montenegro*. ACM, June 2006.

[HKGA05] N. Houssos, K. Kafounis, V. Gazis, and N. Alonistioti. Application-Transparent Adaptation in Wireless Systems Beyond 3G. *International Journal of Management and Decision Making*, 6(1):81 – 100, 2005.

[HL96] B.C. Housel and D.B. Lindquist. WebExpress: a System for Optimizing Web Browsing in a Wireless Environment. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, pp. 108–116, New York, NY, USA, 1996. ACM Press.

[HL02] M. He and H. Leung. Agents in E-commerce: State of the Art. *Knowledge and Information Systems*, 4(3):257–282, 2002.

[HL05] S. Huang and F. Lin. Designing Intelligent Sales-Agent for Online Selling. In *Proceedings of the 7th international conference on Electronic commerce (ICEC'05). ACM International Conference Proceeding Series*, volume 113, pp. 279–286, New York, NY, USA, 2005. ACM Press.

[HM85] D. Heimbigner and D. McLeod. A Federated Architecture for Information Systems. *ACM Transactions on Office Information Systems*, 3,3, 1985.

[HM00] C. Hui-Min. Design and Implementation of the Agent-based EVMs System. Technical report, University of Berkley, 2000.

[HM01] V. Haarslev and R. Möller. Description of the RACER System and its Applications. In *Proceedings of the International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3*, pp. 132–141, August 2001.

[HM04] M.A. Haq and M. Matsumoto. MAMI: Mobile Agent based System for Mobile Internet. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, pp. 377–383. IEEE Computer Society, 2004.

[HMS+03] B. Horling, R. Mailler, J. Shen, R. Vincent, and V. Lesser. Using Autonomy, Organizational Design and Negotiation in a Distributed Sensor Network. In Victor Lesser, Charles Ortiz, and Milind Tambe, editors, *Distributed Sensor Networks: A multiagent perspective*, pp. 139–183. Kluwer Academic Publishers, 2003.

[Hor02a] Eric Horlait, editor. *Mobile Agents for Telecommunication Applications (Innovative Technology Series: Information Systems and Networks)*. Kogan Page Science, 2002.

[Hor02b] I. Horrocks. DAML+OIL: a Description Logic for the Semantic Web. *IEEE Bull. of the Technical Committee on Data Engineering*, 25(1):4–9, MAR 2002.

[Hor02c] I. Horrocks. Reasoning with Expressive Description Logics: Theory and Practice. In Andrei Voronkov, editor, *Proc. of the 18th Int. Conf. on Automated Deduction (CADE-18)*, number 2392 in Lecture Notes in Artificial Intelligence, pp. 1–15. Springer-Verlag, 2002.

[HR95] B. Hayes-Roth. An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence*, 72(1-2):329–365, 1995.

[HS95] M.A. Hernandez and S.J. Stolfo. The Merge/Purge Problem for Large Databases. In *SIGMOD Conference*, pp. 127–138, 1995.

[HSL99] B.C. Housel, G. Samaras, and D.B. Lindquist. WebExpress: a Client/Intercept Based System for Optimizing Web Browsing in a Wireless Environment. *Mob. Netw. Appl.*, 3(4):419–431, 1999.

[HT04] H. Holma and A. Toskala. *WCDMA for UMTS : Radio Access for Third Generation Mobile Communications*. John Wiley & Sons; 3rd edition, ISBN 0470870966, 2004.

[HT06] H. Holma and A. Toskala. *HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications*. John Wiley & Sons, July 2006.

[Hyl96] J. A. Hylton. Identifying and Merging Related Bibliographic Records. Technical Report MIT/LCS/TR-678, Massachusetts Institute of Technology, 1996.

[HZG03] Q. Huo, H. Zhu, and S. Greenwood. A Multi-Agent Software Environment for Testing Web-based Applications. In *27th Annual International Computer Software and Applications Conference. Dallas, Texas.*, November 2003.

[IAIC06] SRI International's Artificial Intelligence Center. The Open Agent Architecture, 2006. http://www.ai.sri.com/oaa/, [Last accessed: April 2009].

[IBM06a] IBM. Ibm websphere transcoding publisher, 2006. http://www-3.ibm.com/software/webservers/transcoding/, [Not available from April 2008].

[IBM06b] IBM Corporation. TME 10 Software Distribution - Mobile Agents SG24-4854-00, January 2006. http://www.redbooks.ibm.com/abstracts/sg244854.html.

[IHMT04] Z. G. Ives, A. Y. Halevy, P. Mork, and I. Tatarinov. Piazza: Mediation and Integration Infrastructure for Semantic Web Data. *Journal of Web Semantics*, 1(2):155–175, February 2004.

[IK96] T. Imielinski and H.F. Korth, editors. *Mobile Computing*. Kluwer Academic Publishers, 1996.

[IKT04] S. Idreos, M. Koubarakis, and C. Tryfonopoulos. P2P-DIET: an Extensible P2P Service that Unifies Ad-Hoc and Continuous Querying in Super-Peer Networks. In *ACM SIGMOD International Conference on Management of Data (SIGMOD'04), Paris, France*, pp. 933–934. ACM Press, 2004.

[Ila06] S. Ilarri. *LOQOMOTION: Processing of Continuous Location-Dependent Queries in Mobile Environments*. PhD thesis, University of Zaragoza, May 2006.

[IMI06] S. Ilarri, E. Mena, and A. Illarramendi. Location-Dependent Queries in Mobile Contexts: Distributed Processing Using Mobile Agents. *IEEE Transactions on Mobile Computing, ISSN 1536-1233*, 5(8):1029–1043, August 2006.

[Inc99] Advanstar Communications Inc. *GPS World's Big Book of GPS 2000*. Advanstar Communications Inc., 1999.

[Inf09] InfoLab, 2009. http://infolab.stanford.edu/ [Last accessed: April 2009].

[ITM05] S. Ilarri, R. Trillo, and E. Mena. Development of a Scalable Platform for Highly Mobile Agents. Technical report, University of Zaragoza, October 2005.

[ITM06] S. Ilarri, R. Trillo, and E. Mena. SPRINGS: A Scalable Platform for Highly Mobile Agents in Distributed Computing Environments. In *Fourth International WoWMoM 2006 Workshop on Mobile Distributed Computing (MDC'06), Niagara Falls/Buffalo, New York, USA*. IEEE Computer Society, 2006.

[JC02] T. Jewell and D.A. Chappell. *Java Web Services*. O'Reilly & Associates, March 2002.

[JPC00] H. Jeon, C. Petrie, and M.R. Cutkosky. JATLite: A Java Agent Infrastructure with Message Routing. *IEEE Internet Computing*, 4(2):87–96, 2000.

[jXU06] jXUL, 2006. http://jxul.sourceforge.net, [Last accessed: April 2009].

[KCG+02] D. Kotz, G. Cybenko, R.S. Gray, G. Jiang, R.A. Peterson, M.O. Hofmann, D.A. Chacón, K.R. Whitebread, and J.A. Hendler. Performance Analysis of Mobile Agents for Filtering Data Streams on Wireless Networks. *Mobile Networks and Applications*, 7(2):163–174, 2002.

[Ken00] R.P. Kennedy. Monitoring of Distributed Processes with Mobile Agents. In *Seventh IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'00), Edinburgh, Scotland, United Kingdom*. IEEE Computer Society, 2000.

[KL02] L. Khan and F. Luo. Ontology Construction for Information Selection. In *14th IEEE International Conference on Tools with Artificial Intelligence, Washington DC*, November 2002.

[KLK+04] J.L.Y. Koh, M.L. Lee, A.M. Khan, P.T.J. Tan, and V. Brusic. Duplicate Detection in Biological Data using Association Rule Mining. In *ECML/PKDD Workshop on Data Mining and Text Mining for Bioinformatics*, September 2004.

[KLP97] D. Koller, A.Y. Levy, and A. Pfeffer. P-CLASSIC: A Tractable Probabilistic Description Logics. In *Proceedings of the AAAI97 conference*, 1997.

[KMD02] A. Karmouch, T. Magedanz, and J. Delgado, editors. *Mobile Agents for Telecommunication Applications*, volume 2521. Springer, 2002.

[KMM99] K.H. Kramer, N. Minar, and P. Maes. Tutorial: Mobile Software Agents for Dynamic Routing. *Mobile Computing and Communications Review*, 3(2):12–16, 1999.

[KNL+01] H. Kaaranen, S. Naghian, L. Laitinen, A. Ahtiainen, and V. Niemi. *UMTS Networks: Architecture, Mobility and Services*. John Wiley & Sons; 1st edition, ISBN: 047148654X, August 2001.

[KR04] F. Koch and I. Rahwan. The Role of Agents in Intelligent Mobile Services. In *Seventh Pacific Rim International Workshop on Multi-Agents (PRIMA'04), Auckland, New Zealand*, pp. 115–127. Springer, 2004.

[Kru83] J.B. Kruskal. An Overview of Sequence Comparison: Time Warps, String Edits, and Macromolecules. *SIAM Review*, 25(2):201–237, April 1983.

[KSCP04] K. Karenos, G. Samaras, P.K. Chrysanthis, and E. Pitoura. Mobile Agent-Based Services for View Materialization. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(3):32–43, 2004.

[KWW00] S. Klemmer, S. Waterson, and K. Whitehouse. Towards a Location-Based Context-Aware Sensor Infrastructure. Technical report, CS Division, EECS Department, University of California at Berkeley, 2000.

[KX02] M. Kona and C. Xu. A Framework for Network Management Using Mobile Agents. In *16th International Parallel and Distributed Processing Symposium (IPDPS'02), Fort Lauderdale, Florida, USA*. IEEE Computer Society, 2002.

[Lab06] Telecom Italia Lab. Java Agent DEvelopment Framework, 2006. http://jade.tilab.com/, [Last accessed: April 2009].

[LAD04] M. Luck, R. Ashri, and M. D'Inverno. *Agent-Based Software Development*. Artech House Publishers, 2004.

[Lag00] C. Lagoze. The Cornell Digital Library Research Group: Architectures and Policies for Distributed Digital Libraries, February 2000.

[Lam94] L. Lamport. *LaTeX.A Document Preparation System. User's Guide and Reference Manual*. Addison-Wesley, 1994.

[Lar09] Large Scale Distributed Information Systems, 2009. http://lsdis.cs.uga.edu [Last accessed: April 2009].

[Lau05] R.Y.K. Lau. Adaptive Negotiation Agents for E-business. In *Proceedings of the 7th international conference on Electronic commerce (ICEC'05). ACM International Conference Proceeding Series*, volume 113, pp. 271–278. ACM Press, 2005.

[LBM+02] A.M. Ladd, K.E. Bekris, G. Marceau, A. Rudys, D.S. Wallach, and L.E. Kavraki. Using Wireless Ethernet for Localization. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, September 2002.

[LC05] J. Lu and J. Callan. Federated Search of Text-Based Digital Libriries in Hierarchical Peer-to-Peer Networks. In *Lecture Notes in Computer Science, ISSN 0302-9743, ISBN 978-3-540-25295-5*, volume 3408/2005. Springer Berlin/Heidelberg, 2005.

[LDKG04] B.T. Le, R. Dieng-Kuntz, and F. Gandon. On Ontology Matching Problems - for Building a Corporate Semantic Web in a Multi-Communities Organization. In *ICEIS (4)*, 2004.

[Les95] V.R. Lesser. Multiagent Systems: An Emerging Subdiscipline of AI. *ACM Computing Surveys*, 27(3):340–342, 1995.

[Les99] V.R. Lesser. Cooperative Multiagent Systems: A Personal View of the State of the Art. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):133–142, 1999.

[Lev66] V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Cybernetics and Control Theory*, 10(8):707–710, 1966. Original in *Doklady Akademii Nauk SSSR* 163(4): 845–848 (1965).

[LFP98] C. Lagoze, D. Fielding, and S. Payette. Making Global Digital Libraries Work: Collection Services, Connectivity Regions, and Collection Views. In *DL '98: Proceedings of the third ACM conference on Digital libraries*, pp. 134–143, New York, NY, USA, 1998. ACM Press.

[LG90] D. Lenat and R. V. Guha. *Building Large Knowledge Based Systems : Representation and Inference in the Cyc Project*. Addison-Wesley Publishing Company Inc, 1990.

[LG04] D. Lubke and J. M. Gomez. Applications for Mobile Agents in Peer-to-Peer-Networks. In *11th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'04)*. IEEE Computer Society, 2004.

[LJE+01] K. Ludwig, A. Josef, W. E. Edgar, S. Wolfgang, and G. Franz. Using Mobile Agents in Real world: A Survey and Evaluation of Agent Platforms. In *Second Workshop on Infrastructure for Agents, MAS, and Scalable MAS at Autonomous Agents, Montreal, Canada*. AAAI, 2001.

[LKR02] K. Lam, Alan Kwan, and Krithi Ramamritham. RTMonitor: Real-Time Data Monitoring Using Mobile Agent Technologies. In *28th International Conference on Very Large Data Bases (VLDB'02), Hong Kong, China*, pp. 1063–1066. Morgan Kaufmann, 2002.

[LMN02] M. L. Liu, A. Macias, and T. Ngo. E-delivery: A Shipment Tracking System Using Wireless Technology and Mobile Agents. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02), Las Vegas, Nevada, USA*, pp. 1838–1844. CSREA Press, 2002.

[LMNC04] F. Lopes, N.J. Mamede, A.Q. Novais, and H. Coelho. Negotiation Strategies for Autonomous Computational Agents. In *16th Eureopean Conference on Artificial Intelligence (ECAI'04), Valencia, Spain*, pp. 38–42. IOS Press, 2004.

[LO99a] D. Lange and M. Oshima. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1999.

[LO99b] D. Lange and M. Oshima. Seven Good Reasons for Mobile Agents. *Communications of the ACM*, 42:88–89, 1999.

[LS01] O. Lassila and R. Swick. Resource Description Framework (RDF) Model and Syntax Specification, February 2001. See http://www.w3.org/RDF, [Last accessed: April 2009] and http://www.w3.org/TR/1999/REC-rdf-syntax-19990222, [Last accessed: April 2009].

[LTE+06] L. Laera, V. Tamma, J. Euzenat, T. Bench-Capon, and T. R. Payne. Reaching Agreement over Ontology Alignments. In *In Proceedings of 5th International Semantic Web Conference (ISWC 2006), Athens, GA*, 2006.

[LTKZ03] D. Levine, R. Thomas, F. Kamangar, and G.V. Záruba. Mobile Agents for Pervasive Computing Using a Novel Method of Message Passing. In *Parallel and Distributed Processing Techniques and Applications (PDPTA'03), Las Vegas, Nevada, USA*, pp. 1380–1386. CSREA Press, 2003.

[LY99] T. Lindholm and F. Yellin. *The Java Virtual Machine Specification*. Addison-Wesley Professional, 1999.

[MÖ98] J.P. Müller. Architectures and Applications of Intelligent Agents: A Survey. *Knowledge Engineering Review*, 13(4):353–380, 1998.

[MA02] N. Mitrovic and U. Arronategui. Mobile Agent Security Using Proxy-agents and Trusted Domains. In *Second International Workshop on Security of Mobile Multiagent Systems (SEMAS'02), German AI Research Center (DFKI) Research Report, ISSN 0946-008X*, July 2002.

[Mac88] R.M. MacGregor. A deductive pattern matcher. In *AAAI-88, St Paul, Minnesota*, pp. 403–408, 1988.

[Mae95] P. Maes. Artificial Life meets Entertainment: Interacting with Lifelike Autonomous Agents. *Special Issue on New Horizons of Commercial and Industrial AI, Communications of the ACM*, 38(11):108–114, 1995.

[MBB+98] D. Milojicic, M. Breugst, I. Busse, J. Campbell, S. Covaci, B. Friedman, K. Kosaka, D. Lange, K. Ono, M. Oshima, C. Tham, S. Virdhagriswaran, and J. White. MASIF, The OMG Mobile Agent System Interoperability Facility. In *Proceedings of Mobile Agents '98*, September 1998.

[MBC+95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley Series in Parallel Computing - Chichester, 1995.

[MBR01] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. *The Very Large Databases (VLDB) Journal*, pp. 49–58, 2001.

[MCB84] M. Ajmone Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

[McL01] B. McLaughlin. *Java & XML, 2nd Edition: Solutions to Real-World Problems*. O'Reilly & Associates, September 2001.

[MCM03] J. Merseguer, J. Campos, and E. Mena. Analysing Internet Software Retrieval Systems: Modeling and Performance Comparison. *Wireless Networks: The Journal of Mobile Communication, Computation and Information (WINET)*, 9(3):223–238, May 2003.

[MCW00] S. McGrath, D. Chacon, and K. Whitebread. Intelligent Mobile Agents in the Military Domain. In *In Proc. of the Fourth International Conference on Autonomous Agents, Barcelona (Spain)*, 2000.

[MDF+02] N. Muscettola, G. Dorais, C. Fry, R. Levinson, and C. Plaunt. Idea: Planning at the Core of Autonomous Reactive Agents. In *Third International NASA Workshop on Planning and Scheduling for Space, Houston, Texas, USA*. The Institute for Advanced Interdisciplinary Research, 2002.

[MDW99] D. Milojicic, F. Douglis, and R. Wheeler. *Mobility: Processes, Computers, and Agents*. ACM Press/Addison-Wesley Publishing Co., 1999.

[ME96] A.E. Monge and C. Elkan. The Field Matching Problem: Algorithms and Applications. In *Knowledge Discovery and Data Mining*, pp. 267–270, 1996.

[ME97] A.E. Monge and C. Elkan. An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In *Research Issues on Data Mining and Knowledge Discovery*, pp. 0–, 1997.

[MFS06] T.S. Mitrovich, K.M. Ford, and N. Suri. Transparent Redirection of Network Sockets, 2006. http://www.ihmc.us/research/projects/nomads/mockets.pdf, [Last accessed: April 2009].

[MI01] E. Mena and A. Illarramendi. *Ontology-Based Query Processing for Global Information Systems*. Kluwer Academic Publishers, ISBN 0-7923-7375-8, 2001. June 2001.

[Mic02] Microsoft. Microsoft word, 2002. http://www.microsoft.com/office/word/default.asp, [Last accessed: April 2009].

[Mic06a] Sun Microsystems, 2006. http://java.sun.com, [Last accessed: April 2009].

[Mic06b] Sun Microsystems, 2006. http://java.sun.com, [Last accessed: April 2009].

[Mic07] Sun Microsystems. Java Naming and Directory Interface (JNDI), 2007. http://java.sun.com/products/jndi/, [Last accessed: April 2009].

[MIG00a] E. Mena, A. Illarramendi, and A. Goñi. A Software Retrieval Service based on Knowledge-driven Agents. In *Fith IFCIS International Conference on Cooperative Information Systems (CoopIS'2000), Springer series of Lecture Notes in Computer Science (LNCS), Eliat (Israel)*, September 2000.

[MIG00b] E. Mena, A. Illarramendi, and A. Goñi. Automatic Ontology Construction for a Multiagent-based Software Gathering Service. In *proceedings of the Fourth International ICMAS'2000 Workshop on Cooperative Information Agents (CIA'2000), Boston (USA)*, pp. 232–243. Springer series of Lecture Notes on Artificial Intelligence (LNAI), ISBN 3-540-67703-8, July 2000.

[MIG00c] E. Mena, A. Illarramendi, and A. Goñi. Customizable Software Retrieval Facility for Mobile Computers using Agents. In *proceedings of the Seventh International Conference on Parallel and Distributed Systems (ICPADS'2000), workshop International Flexible Networking and Cooperative Distributed Agents (FNCDA'2000), IEEE Computer Society, Iwate (Japan)*, July 2000.

[MIKS00] E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. *International journal on Distributed And Parallel Databases (DAPD)*, 8(2):223–272, April 2000.

[Mil95] G. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11), nov 1995.

[Mil99] D. Milojicic. Mobile agent applications. *IEEE Concurrency*, 7(3):80–90, 1999.

[Mil06] G. Miller. World Wide Web interface to WordNet 1.5, jun 2006. http://www.cogsci.princeton.edu/~wn/w3wn.html, [Not available from April 2008].

[MIRG06] E. Mena, A. Illarramendi, J.A. Royo, and A. Goñi. A Software Retrieval Service based on Adaptive Knowledge-Driven Agents for Wireless Environments. *Transactions on Autonomous and Adaptive Systems (TAAS)*, 1(1):67–90, September 2006.

[MLC98] D. Milojicic, W. LaForge, and D. Chauhan. Mobile Objects and Agents (MOA). In *Fourth USENIX Conference on Object-Oriented Technologies and Systems (COOTS'98), Santa Fe, New Mexico, USA*. USENIX, 1998.

[MM02] N. Mitrovic and E. Mena. Adaptive User Interface for Mobile Devices. In P. Forbrig, Q. Limbourg, B. Urban, and J. Vanderdonckt, editors, *Interactive Systems. Design, Specification, and Verification. 9th International Workshop DSV-IS 2002, Rostock (Germany)*, pp. 47–61. Springer Verlag Lecture Notes in Computer Science LNCS, ISBN 3-540-00266-9, June 2002.

[MM03] N. Mitrovic and E. Mena. Improving User Interface Usability Using Mobile Agents. In *Interactive Systems. Design, Specification, and Verification. 10th DSV-IS Workshop, Funchal, Madeira Island (Portugal)*, pp. 273–287. Springer Verlag Lecture Notes in Computer Science LNCS 2844, ISBN 3-540-20159-9, ISSN0302-9743, June 2003.

[MMR08] N. Mitrovic, E. Mena, and J.A. Royo. *Chapter XIX: Adaptive Interfaces in Mobile Environments - An Approach Based on Mobile Agents*, pp. 302–317. Information Science Reference (formerly Idea Group Reference), ISBN 978-1-59904-871-0, 2008.

[Moc87a] P. Mockapetris. Domain Names - Concepts and Facilities, November 1987. http://tools.ietf.org/html/rfc1034, [Last accessed: April 2009].

[Moc87b] P. Mockapetris. Domain Names - Implementation and Specification, November 1987. http://tools.ietf.org/html/rfc1035, [Last accessed: April 2009].

[Mor02a] L. Moreau. A Fault-Tolerant Directory Service for Mobile Agents based on Forwarding Pointers. In *17th ACM symposium on Applied computing (SAC'02), Madrid, Spain*, pp. 93–100. ACM Press, 2002.

[Mor02b] R. Morrow. *Bluetooth: Operation and Use*. McGraw-Hill Professional; 1st edition, ISBN: 007138779X, June 2002.

[MPH92] Michel Mouly, Marie-Bernadette Pautet, and Thomas Haug. *The GSM System for Mobile Communications*. Telecom Publishing, 1992.

[MR02] D.C. Montgomery and GcC. Runger. *Applied Statistics and Probability for Engineers*. Wiley Text Books; 3rd edition, ISBN: 0471204544, August 2002.

[MRIG02a] E. Mena, J.A. Royo, A. Illarramendi, and A. Goñi. Adaptable Software Retrieval Service for Wireless Environments Based on Mobile Agents. In *2002 International Conference on Wireless Networks (ICWN'02), Las Vegas, USA*, pp. 116–124. CSREA Press, ISBN 1-892512-30-0, June 2002.

[MRIG02b] E. Mena, J.A. Royo, A. Illarramendi, and A. Goñi. An Agent-based Approach for Helping Users of Hand-Held Devices to Browse Software Catalogs. In *Cooperative Information Agents VI, 6th International Workshop CIA 2002, Madrid (Spain)*, pp. 51–65. Lecture Notes on Artificial Intelligence (LNAI), ISBN 3-540-44173-5, September 2002.

[MRM04] N. Mitrovic, J.A. Royo, and E. Mena. ADUS: Indirect Generation of User interfaces on Wireless Devices. In *Fifteentth International Workshop on Database and Expert Systems Applications (DEXA'2004), Seventh International Workshop Mobility in Databases and Distributed Systems (MDDS'2004), Zaragoza (Spain)*. IEEE Computer Society, ISBN 0-7695-2195-9, ISSN 1529-4188, September 2004.

[MRM05] N. Mitrovic, J.A. Royo, and E. Mena. Adaptive User Interfaces Based on Mobile Agents: Monitoring the Behavior of Users in a Wireless Environment. In *I Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'2005), Granada, (Spain)*, pp. 371–378. Thomson, ISBN 84-9732-442-0, September 2005.

[MRM07] N. Mitrovic, J.A. Royo, and E. Mena. Performance Analysis of an Adaptive User Interface System Based on Mobile Agents. In *Engineering Interactive Systems. Conference on Design Specification and Verification of Interactive Systems. 14th DSV-IS Conference Salamanca (Spain)*. Springer Verlag Lecture Notes in Computer Science LNCS 4490, ISBN 978-3-540-92697-9, ISSN0302-9743, March 2007.

[NDA01] C. Nerguizian, C. Despins, and S. Affes. A Framework for Indoor Geolocation. *3rd WLAN Workshop 2001*, 2001.

[NR02] G. Navarro and M. Raffinot. *Flexible Patern Matching in Strings*. Cambridge University Press, 2002.

[NSS03] W. Nejdl, W. Siberski, and M. Sintek. Design Issues and Challenges for RDF- and Schema-Based Peer-to-Peer Systems. *SIGMOD Rec.*, 32(3):41–46, 2003.

[Nwa96] H.S. Nwana. Software Agents: An Overview. *Knowledge Engineering Review*, 11(3):205–244, 1996.

[OAS08] OASIS: Advancing Open Standards for the Information Society, 2008. http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm, [Last accessed: April 2009].

[Obj99] ObjectSpace, 1999. http://www.objectspace.com/, [Last accessed: April 2009].

[Ohr05] F. Ohrtman. *WiMAX Handbook*. McGraw-Hill Professional, ISBN: 0071454012, 2005.

[OMG93] OMG. The Common Object Request Broker : Architecture and Specification. Technical report, Object Management Group, Inc., December 1993.

[OS06] Inc. Overture Services. Altavista, 2006. http://www.altavista.digital.com.

[OSP05] Open Source Proyect. Aglets web site, 2005. http://aglets.sourceforge.net/, [Last accessed: April 2009].

[Pae06] A. Paepcke. InterBib: Bibliography-Related Services, 2006. http://interbib.stanford.edu/ [Last accessed: April, 2009].

[Pal07a] J.D. Palmer. Exploiting Bibliographic Web Services with CiTeX. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pp. 1673–1676, New York, NY, USA, 2007. ACM Press.

[Pal07b] Palowireless. HomeRF Resource Center, 2007, June 2007. http://www.palowireless.com/homerf/, [Last accessed: April 2009].

[PBH00] S. Poslad, P. Buckle, and R. Hadingham. The FIPA-OS Agent Platform: Open Source for Open Standards. In *Fifth International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM'00), Manchester, United Kingdom*, pp. 355–368, 2000.

[PBP05] T. Pedersen, S. Banerjee, and S. Patwardhan. Maximizing Semantic Relatedness to Perform Word Sense Disambiguation, 2005. University of Minnesota Supercomputing Institute Research Report UMSI 2005/25.

[PCWGM98] A. Paepcke, C.C.K. Chang, T. Winograd, and H. García-Molina. Interoperability for Digital Libraries Worldwide. *Communications of the ACM*, 41(4):33–42, April 1998.

[PGGMU95] Y. Papakonstantinou, A. Gupta, H. Garcia-Molina, and J. Ullman. A Query Translation Scheme for Rapid Implementation of Wrappers. In *Proceedings of the International Conference on Deductive and Object-Oriented Databases*, 1995.

[Pie94] S. Piepenburg. Easy MARC: A Simplified Guide to Creating Catalog Records for Library Automation Systems: Pre-format Integration, 1994.

[PMI03] J.M. Puyal, E. Mena, and A. Illarramendi. REMOTE: A Multiagent System to Select and Execute Remote Software Services in a Wireless Environment. In *WWW Workshop on E-Services and the Semantic Web (ESSW'03), Budapest, Hungary*, pp. 1–12, 2003.

[PMP03] J. Belenguer P.J. Molina and O. Pastor. Describing Just-UI Concepts Using a Task Notation. In *Interactive Systems. Design, Specification, and Verification. 10th International Workshop DSV-IS 2003, Funchal, Madeira Island (Portugal), J. Joaquim, N. Jardim Nunes, J. Falcao e Cunha (ed.), Springer Verlag Lecture Notes in Computer Science LNCS, I*, July 2003.

[PN03] S. J. Russell P. Norvig. *Artificial Intelligence: A Modern Approach*. Published Prentice Hall, ISBN 0137903952, 2003.

[PO02] E. Plaza and S. Ontañón. Cooperative Multiagent Learning. In *Adaptive Agents and Multi-Agents Systems (AAMAS) 2001 / 2002*, pp. 1–17. Springer, 2002.

[PR02] U. Pinsdorf and V. Roth. Mobile Agent Interoperability Patterns and Practice. In *Ninth IEEE International Conference on Engineering of Computer-Based Systems (ECBS'02), Lund, Sweden*, pp. 238–244. IEEE Computer Society, 2002.

[PRU06] L. Palopoli, D. Rosaci, and D. Ursino. Agents' roles in B2C e-commerce. *AI Communications*, 19(2):95–126, 2006.

[PS84] P. Patel-Schneider. Small can be Beautiful in Knowledge Representation. In *Proceedings IEEE Workshop on Principles of Knowledge-Based Systems*, pp. 11–16, 1984.

[PS98] E. Pitoura and G. Samaras. *Data Management for Mobile Computing*, volume 10. Kluwer Academic Publishers, 1998.

[PS04] C. Panayiotou and G. Samaras. mPERSONA, Personalized Portals for the Wireless User: An agent Approach. *Mobile Networks and Applications*, 9(6):663–677, 2004.

[PSP00] S. Papastavrou, G. Samaras, and E. Pitoura. Mobile Agents for WWW Distributed Database Access. *IEEE Transactions on Knowledge and Data Engineering*, 12(5):802–820, 2000.

[PVM+02] L. Popa, Y. Velegrakis, R. J. Miller, M. Hernandez, and R. Fagin. Translating Web Data. In *Very Large Data Bases Conference (VLDB '02)*, pp. 598–609, 2002.

[Pér98] L. C. Pérez. *Agentes Móviles en Bibliotecas Digitales*. PhD thesis, Universidad de las Américas-Puebla, Cholula, Pue. 72820 México, 1998.

[QHC06] Y. Qu, W. Hu, and G. Cheng. Constructing Virtual Documents for Ontology Matching. In *15th International World Wide Web Conference*, May 2006.

[QIC01a] H. Qi, S.S. Iyengar, and K. Chakrabarty. Distributed Multi-Resolution Data Integration Using Mobile Agents. *IEEE Transactions on Systems, Man and Cybernetics (Part C): Applications and Reviews*, 31(3):383–391, 2001.

[QIC01b] H. Qi, S.S. Iyengar, and K. Chakrabarty. Multisensor Data Fusion in Distributed Sensor Networks Using Mobile Agents. In *Fifth International Conference on Information Fusion, Annapolis, Massachusetts, USA*, pp. 11–16, 2001.

[Que08] Quealcom, 2008. http://www.qualcomm.com, [Last accessed: April 2009].

[Raa04] K. Raatikainen. Wireless Internet: Challenges and Solutions First Ten Years of Research in Mobile Computing at Helsinki University Computer Science Department and Challenges for the Next Ones. Series of Publications B, Report B-2004-3. Technical report, University of Helsinki, Department of Computer Science, September 2004.

[Rah93] M. Rahnema. Overview of the GSM System and Protocol Architecture. *IEEE Communications Magazine 31(4)*, pp. 92–100, April 1993.

[RAJF01] E. Roman, S. W. Ambler, T. Jewell, and F.Marinescu. *Mastering Enterprise JavaBeans (2nd Edition)*. John Wiley & Sons; ISBN: 0471417114; 2nd Bk&acc edition, December 2001.

[Ram04] H. Ramampiaro. Transactional Support for Cooperative Work: Using Mobile Agents in CAGISTrans. In *International Conference on Parallel and Distributed Computing Systems (PDCS'04), San Francisco, California, USA*, 2004.

[RASS99] M. Ranganathan, A. Acharya, S.D. Sharma, and J. Saltz. Network-aware Mobile Programs. *Mobility: Processes, Computers, and Agents*, pp. 567–581, 1999.

[RBM00] C.M.L. Werner R.M.M. Braga and M. Mattoso. Using Ontologies for Domain Information Retrieval. In *11th International Workshop on Database and Expert Systems Applications (DEXA'00). Greenwich, London, U.K.*, September 2000.

[Res02a] ISI Researchsoft. EndNote, 2002. http://www.endnote.com, [Last accessed: April 2009].

[Res02b] ISI Researchsoft. ProCite, 2002. http://www.procite.com, [Last accessed: April 2009].

[RG95] A.S. Rao and M.P. Georgeff. BDI Agents: From Theory to Practice. In *International Conference on Multiagent Systems (ICMAS'95), San Francisco, California, USA*, pp. 312–319. The MIT Press, 1995.

[RHC+02] M. Román, C. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt. Gaia: a Middleware Platform for Active Spaces. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):65–67, 2002.

[RHZ76] A. Rosenfeld, R. Hummel, and S. Zucker. Scene Labeling by Relaxation Operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6:420–433, June 1976.

[RJS01] V. Roth and M. Jalali-Sohi. Concepts and Architecture of a Security-Centric Mobile Agent Server. In *Fifth International Symposium on Autonomous Decentralized Systems (ISADS'01), Dallas, Texas, USA*, pp. 435. IEEE Computer Society, 2001.

[RM00] B. J. Rhodes and P. Maes. Just-in-time Information Retrieval Agents. *IBM Systems Journal*, 39(3-4):685–704, 2000.

[RM01a] J.A. Royo and E. Mena. Uso de Agentes Móviles para la Búsqueda y Recuperación de Información Bibliográfica. In A. Pérez P. De la Fuente, editor, *II Jornadas de Bibliotecas Digitales (JBIDI'2001), Almagro (Ciudad Real), Spain, ISBN 84-699-6276-0*, pp. 223–234. Universidad de Castilla-La Mancha, November 2001.

[RM01b] J.A. Royo and E. Mena. Uso de Agentes Móviles para la Búsqueda y Recuperación de Información Bibliográfica. *Revista Interamericana de Nuevas Tecnologías de la Información, ISSN 0122-3356*, 6(4):52–63, October–December 2001.

[RM02] J.A. Royo and E. Mena. Gestión de Bibliotecas Digitales de Publicaciones de Investigación. In Purificación García José H. Canós, editor, *III Jornadas de Bibliotecas Digitales (JBIDI'2002), El Escorial (Madrid), Spain, ISBN 84-688-0205-0*, pp. 77–86. Universidad Politécnica de Madrid, November 2002.

[RMA03] J.A. Royo, E. Mena, and R. Acero. Agentes Java para Dispositivos Móviles con Conexión Bluetooth. In *I Congreso Java Hispano, Madrid (Spain)*, pp. 44–55. Universidad Carlos III, ISBN 84-688-8080-0, October 2003.

[RMBI05] J.A. Royo, E. Mena, J. Bernad, and A. Illarramendi. Searching the web: From keywords to semantic queries. In *Third International Conference on Information Technology and Applications (ICITA'05), July 4-7, 2005, Sydney (Australia)*, pp. 244–249. IEEE Computer Society, ISBN 0-7695-2316-1, July 2005.

[RMCM03a] A. Ranganathan, R.E. McGrath, R.H. Campbell, and M.D. Mickunas. Ontologies in a Pervasive Computing Environment. In *Eighteenth International Joint Conference On Artificial Intelligence (IJCAI'03), Workshop on Information Integration on the Web (IIWeb'03), Acapulco (Mexico)*, August 2003.

[RMCM03b] A. Ranganathan, R.E. McGrath, R.H. Campbell, and M.D. Mickunas. Use of Ontologies in a Pervasive Computing Environment. *Knowl. Eng. Rev.*, 18(3):209–220, 2003.

[RMG05] J.A. Royo, E. Mena, and L.C. Gallego. Locating Users to Develop Location-Based Services in Wireless Local Area Networks. In *I Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI'2005), Granada (Spain)*, pp. 471–478. Thomson, ISBN 84-9732-442-0, September 2005.

[RN03] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2003.

[RRP04] D. Ratner, P. Reiher, and G.J. Popek. Roam: a Scalable Replication System for Mobility. *Mobile Networks and Applications*, 9(5):537–544, 2004.

[RTIM05] J.A. Royo, R. Trillo, S. Ilarri, and E. Mena. Gestión y Detección de Inconsistencias en Colecciones de Referencias Bibliográficas. In *V Jornadas de Bibliotecas Digitales (JBiDi'2005), Granada (Spain)*, pp. 29–36. Thomson, ISBN 84-9732-453-6, September 2005.

[RW98] J. Richvalsky and D. Watkins. Design and Implementation of a Digital Library. *Crossroads*, 5(2):15–19, 1998.

[RY98] E.S. Ristad and P.N. Yianilos. Learning String-Edit Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[Sam09] G. Samaras, 2009. http://www2.cs.ucy.ac.cy/ cssamara/samaras/index.htm [Last accessed: April 2009].

[Sat04] I. Satoh. Mobile Agents for Ambient Intelligence. In *Massively Multi-Agent Systems (MMAS'04), Kyoto, Japan*, pp. 187–201. Springer, 2004.

[SBB+00a] N. Suri, J.M. Bradshaw, M.R. Breedy, P.T. Groth, G.A. Hill, and R. Jeffers. Strong Mobility and Fine-Grained Resource Control in NOMADS. In *Second International Symposium on Agent Systems and Applications and Fourth International Symposium on Mobile Agents(ASA/MA'00), Zurich, Switzerland*, pp. 2–15. Springer, 2000.

[SBB+00b] N. Suri, J.M. Bradshaw, M.R. Breedy, P.T. Groth, G.A. Hill, R. Jeffers, T.S. Mitrovich, B.R. Pouliot, and D.S. Smith. NOMADS: Toward a Strong and Safe Mobile Agent System. In *Agents, Barcelona, Spain*, pp. 163–164. ACM Press, 2000.

[SBMS99] L.M. Silva, V. Batista, P. Martins, and G. Soares. Using Mobile Agents for Parallel Processing. In *International Symposium on Distributed Objects and Applications (DOA'99), Edinburgh, Scotland*, pp. 34. IEEE Computer Society, 1999.

[Sch97] B. R. Schatz. Information Retrieval in Digital Libraries: Bringing Search to the Net. *Science*, 275:327–334, 1997.

[SCS94] D.C. Smith, A. Cypher, and J. Spohrer. KidSim: Programming Agents without a Programming Language. *Commun. ACM*, 37(7):54–67, 1994.

[SDO00] V. S. Subrahmanian, Jurgen Dix, and Fatma Ozcan. *Heterogeneous Agent Systems*. MIT Press, Cambridge, MA, USA, 2000.

[SDSL99] G. Samaras, M.D. Dikaiakos, C. Spyrou, and A. Liverdos. Mobile Agent Platforms for Web Databases: A Qualitative and Quantitative Assessment. In *First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents (ASA/MA'99), Palm Springs, California, USA*, pp. 50. IEEE Computer Society, 1999.

[SG90] James W. Stamos and David K. Gifford. Implementing Remote Evaluation. *IEEE Transactions on Software Engineering*, 16(7):710–722, 1990.

[SGN93] A.P. Sheth, S.K. Gala, and S.B. Navathe. On Automatic Reasoning for Schema Integration. *International Journal on Intelligent and Cooperative Information Systems*, 2(1):23–50, march 1993.

[SH05] R. Srinivasan and J. Huang. Fluid Ontologies for Digital Museums. *International Journal on Digital Libraries*, pp. 193–204, May 2005.

[Sha05] E. Shakshuki. A Methodology for Evaluating Agent Toolkits. In *International Symposium on Information Technology: Coding and Computing (ITCC'05), Las Vegas, Nevada, USA*, volume 1, pp. 391–396. IEEE Computer Society, 2005.

[SHD+06] E. Sit, A. Haeberlen, F. Dabek, B.G. Chun, H. Weatherspoon, R. Morris, M.F. Kaashoek, and J. Kubiatowicz. Proactive Replication for Data Durability. In *Proceedings of the Fifth International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2006.

[She91] A.P. Sheth. Federated Database Systems for Managing Heterogeneous, Distributed and Autonomous Databases. In *Proc. of the VLDB 91. Tutorial Notes*, 1991.

[SHW98] R. C. Seacord, S. A. Hissam, and K. C. Wallnau. Agora: A search engine for software components. *IEEE Internet Computing*, 2(6):62–70, 1998.

[Sie04] Carles Sierra. Agent-mediated electronic commerce. *Autonomous Agents and Multi-Agent Systems*, 9(3):285–301, 2004.

[SL90] A. Sheth and J. Larson. Federated Database Systems for managing Distributed, Heterogeneous and Autonomous Databases. *ACM Computing Surveys*, 22(3), September 1990.

[SL97] A.J. Sanchez and J.J. Leggett. Agent Services for Users of Digital Libraries. *Journal of Network and Computer Applications*, 20(1):45–58, 1997.

[SM98] A. Sahai and C. Morin. Towards Distributed and Dynamic Network Management. In *IEEE/IFIP Network Operations and Management Symposium (NOMS'98), New Orleans, USA*. IEEE, 1998.

[SM06] Inc. Sun Microsystems. Java Applets, 2006. http://java.sun.com/applets/, [Last accessed: April 2009].

[Smi77] R. G. Smith. The Contract Net: A Formalism for the Control of Distributed Problem Solving. In *Fifth International Joint Conference on Artificial Intelligence (IJCAI'77), Cambridge, Michigan, USA*, pp. 472. William Kaufmann, 1977.

[SNBN01] W. Shen, D.H. Norrie, J.A. Barthes, and D.H. Norrie. *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. CRC Press, 2001.

[Sof05] Recursion Software. Voyager ORB developer's guide, 2005. http://www.recursionsw.com/Voyager/Voyager_ORB.pdf, [Last accessed March, 2006].

[SP97] G. Samaras and A. Pitsillides. Client/Intercept: a Computational Model for Wireless Environments. In *Proceedings of the 4th International Conference on Telecommunications (ICT'97)*, April 1997.

[SRDS01] A. R. Silva, A. Romäo, D. Deugo, and M. M. Da Silva. Towards a Reference Model for Surveying Mobile Agent Systems. *Autonomous Agents and Multi-Agent Systems*, 4(3):187–231, 2001.

[Sri95] R. Srinivasan. RPC: Remote Procedure Call Protocol Specification Version 2, August 1995. http://tools.ietf.org/html/rfc1831, [Last accessed: April 2009].

[SRP04] Rayan Stephan, Pradeep Ray, and N. Paramesh. Network Management Platform Based on Mobile Agents. *International Journal of Network Management*, 14(1):59–73, 2004.

[SS02] S Sathyanath and F Sahin. AISIMAM - An Artificial Immune System Based Intelligent Multi-Agent Model and its Application to a Mine Detection Problem. In *First International Conference on Artificial Immune Systems (ICARIS'02), Kent, Canterbury, United Kingdom*, volume 1, pp. 22–31. University of Kent at Canterbury Printing Unit, 2002.

[SS05a] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Computing Surveys (CSUR)*, 37(1):42–81, 2005.

[SS05b] Y. Sure and R. Studer. Semantic Web Technologies for Digital Libraries. *Library Management, ISSN: 0143-5124*, 26(4/5):190–195, 2005.

[SSEP99] C. Spyrou, G. Samaras, P. Evripidou, and E. Pitoura. Wireless Computational Models: Mobile Agents to the Rescue. In *Second International DEXA Workshop on Mobility in Databases and Distributed Systems (MDDS'99), Florence, Italy*, pp. 127–133. IEEE Computer Society, 1999.

[SSPE04a] C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou. Mobile Agents for Wireless Computing: the Convergence of Wireless Computational Models with Mobile-Agent Technologies. *Mobile Networks and Applications*, 9(5):517–528, 2004.

[SSPE04b] C. Spyrou, G. Samaras, E. Pitoura, and P. Evripidou. Mobile Agents for Wireless Computing: the Convergence of Wireless Computational Models with Mobile-Agent Technologies. *Mobile Networks and Applications*, 9(5):517–528, 2004.

[STK01] J. Snell, D. Tidwell, and P. Kulchenko. *Programming Web Services With Soap*. O'Reilly & Associates, December 2001.

[Sto01] H. Stottner. A Platform-Independent User Interface Description Language. Technical report, Technical Report 16, Institute for Practical Computer Science, Johannes Kepler University Linz,, 2001.

[SU06] Stanford University. Ontolingua home page, 2006. http://www.ksl.stanford.edu/software/ontolingua/, [Last accessed: April 2009].

[Sun06] Sun Microsystems, Inc. Java web start technology, 2006. http://java.sun.com/products/javawebstart/, [Last accessed: April 2009].

[Sur06] N. Suri. NOMADS web page, 2006. http://www.ihmc.us/research/projects/nomads/, [Last accessed: April 2009].

[SV00] P. Stone and M.M. Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[SVZP02] M. Scarpa, M. Villari, A. Zaia, and A. Puliafito. From Client/Server to Mobile Agents: An In-Depth Analysis of the Related Performance Aspects. *iscc*, 00:768, 2002.

[Syc89] K. P. Sycara. Multiagent Compromise Via Negotiation. In *Distributed Artificial Intelligence*, volume 2, pp. 119–138. Pitman Publishing: London, 1989.

[SZ96] K. Sycara and D. Zeng. Multi-Agent Integration of Information Gathering and Decision Support. In *Proceedings of the 12th European Conference on Artificial Intelligence*. John Wiley & Sons, Ltd., 1996.

[SZ97] A. Silberschatz and S. Zdonik. Database Systems - Breaking Out of the Box. *SIGMOD Record*, 26(3), September 1997.

[TA03] P. Tosic and G. Agha. Understanding and Modeling Agent Autonomy in Dynamic Multi-Agent, Multi-Task Environments. In *First European Workshop on Multi-Agent Systems (EUMAS'03), Oxford, United Kingdom*, 2003.

[TA04] P. Tosic and G. Agha. Towards a Hierarchical Taxonomy of Autonomous Agents. In *IEEE International Conference on Systems, Man and Cybernetics (IEEE-SMC'04), The Hague, The Netherlands*. IEEE, 2004.

[TC99] D. Thevenin and J. Coutaz. Plasticity of User Interfaces: Frame-work and Research Agenda. In *Proc of IFIP TC 13 Int. Conf. on Human-Computer Interaction INTERACT'99, Edinburgh*, August 1999.

[TCFW05] V. Tamma, S. Cranefield, T. Finin, and S. Willmott. *Ontologies for Agents: Theory and Experiences*. Whitestein Series in Software Agent Technologies. Springer, July 2005. (See http://www.birkhauser.ch/3-7643-7237-0, [Last accessed: April 2009]).

[TEDBT99] Y.L. Theng, N. Mohd-Nasir E. Duncker, G. Buchanan, and H. Thimbleby. Design Guidelines and User-Centred Digital Libraries. In *Lecture Notes in Computer Science, ISSN 0302-9743*, volume 1696/1999. Springer Berlin/Heidelberg, 1999.

[TGEM07] R. Trillo, J. Gracia, M. Espinoza, and E. Mena. Discovering the Semantics of User Keywords. *Journal on Universal Computer Science (JUCS). Special Issue: Ontologies and their Applications, ISSN 0948-695X*, 13(12):1908–1935, December 2007.

[TH06] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pp. 292–297. Springer, 2006.

[The96a] The Internet Engineering Task Force. MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text, 1996. http://tools.ietf.org/html/rfc2047, [Last accessed: April 2009].

[The96b] The Internet Engineering Task Force. Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures, 1996. http://tools.ietf.org/html/rfc2048, [Last accessed: April 2009].

[The96c] The Internet Engineering Task Force. Multipurpose Internet Mail Extensions (MIME) Part Two:Media Types, 1996. http://tools.ietf.org/html/rfc2046, [Last accessed: April 2009].

[The96d] The Internet Engineering Task Force. Multipurpose Internet Mail Extensions(MIME) Part One: Format of Internet Message Bodies, 1996. http://tools.ietf.org/html/rfc2045, [Last accessed: April 2009].

[The96e] The Internet Engineering Task Force. The Model Primary Content Type for Multipurpose Internet Mail Extensions, 1996. http://tools.ietf.org/html/rfc2077, [Last accessed: April 2009].

[Tho00] S. A. Thomas. *SSL & TLS Essentials: Securing the Web*. Wiley; Pap/Cdr edition, ISBN-10: 0471383546, ISBN-13: 978-0471383543, February 2000.

[Tho01] S. Thompson. Zeus Agent Platform, 2001. http://eprints.agentlink.org/5453/, [Last accessed: April 2009].

[Tho02] S. A. Thomas. *Network Security With Openssl*. O'Reilly & Associates, ISBN: 059600270X, June 2002.

[THT02] J.Y.L. Thong, W. Hong, and K.Y. Tam. Understanding User Acceptance of Digital Libraries: What Are the Roles of Interface Characteristics, Organizational Context, and Individual Differences? *Int. J. Hum.-Comput. Stud.*, 57(3):215–242, 2002.

[THT04] J.Y.L. Thong, W. Hong, and K. Yan Tam. What Leads to Acceptance of Digital Libraries? *Commun. ACM*, 47(11):78–83, 2004.

[Til05] Tilab. Jade - Java Agent Development Framework, 2005. http://jade.tilab.com/, [Last Accessed April 2009].

[TIM07] R. Trillo, S. Ilarri, and E. Mena. Comparison and Performance Evaluation of Mobile Agent Platforms. In *The Third International Conference on Autonomic and Autonomous Systems (ICAS'07), Athens, Greece*. IEEE Computer Society, ISBN 978-0-7695-2859-5, June 2007.

[TS00] C. Tsatsoulis and L. Soh. Intelligent Agents in Telecommunication Networks. In A.V. Vasilakos W. Pedrycz, editor, *Computational Intelligence in Telecommunications Networks*, pp. 479–504. CRC Press, 2000.

[TSB05] Tryllian Solutions B.V. Tryllian, 2005. http://www.tryllian.com, [Last accessed: April 2009].

[Tsc02] S. Tschumi. *Positioning in Mobile Ad-Hoc Networks.* PhD thesis, ETH, Zurich,Switzerland, July 2002.

[TSS+97] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden. A Survey of Active Network Research. *IEEE Communications Magazine*, 35(1):80–86, 1997.

[Tuc06] Tucows.Com Inc., 2006. http://www.tucows.com, [Last accessed: April 2009].

[Tut06] XUL Tutorial, 2006. http://www.xulplanet.com/tutorials/xultu/, [Last accessed: April 2009].

[TVP00] O. Tomarchio, L. Vita, and A. Puliafito. Active Monitoring in Grid Environments Using Mobile Agent Technology. In *Second Workshop on Active Middleware Services (AMS'00), Pittsburgh, USA*. Kluwer Academic Publishers, 2000.

[UGT02] TEX Users Group (TUG). TEX, 2002. http://www.tug.org, [Last accessed: April 2009].

[UK04] L.K. Ueda and F. Kon. Mobile Musical Agents: The Andante Project. In *Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA), Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, ISBN:1-58113-833-4, Vancouver, BC,Canada*, pp. 206–207, 2004. http://doi.acm.org/10.1145/1028752, [Last accessed: April 2009].

[VBH03] M.A. Viredaz, L.S. Brakmo, and W.R. Hamburgen. Energy Management on Handheld Devices. *Queue*, 1(7):44–52, 2003.

[vEHKB02] R. van Eijk, J. Hamers, T. Klos, and M.S. Bargh. Agent Technology for Designing Personalized Mobile Service Brokerage, 2002. Telematica Instituut, http://www.recursionsw.com/Mobile_Agent_Papers/Gigamobile.pdf, [Last Accessed 20 March, 2006].

[Ver98] S. Versteeg. Comparison of Mobile Agent Toolkits for Java. Technical Report June 1998, Departments of Computer Science and Software Engineering & Information Systems, Intelligent Agent Laboratory, University of Melbourne, 1998.

[VGGI98] Y. Villate, D. Gil, A. Goñi, and A. Illarramendi. Mobile Agents for Providing Mobile Computers with Data Services. In *Proceedings of the Ninth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 98)*, 1998.

[Vig99] G. Vigna, editor. *Mobile Agents and Security*, volume 1419. Springer, 1999.

[vLNPS87] K. von Luck, B. Nebel, C. Peltason, and A. Schmiedel. The Anatomy of the BACK System. Technical Report KIT Report 41, Technical University of Berlin, Berlin, F.R.G., 1987.

[VM04a] L. Vasiu and Q.H Mahmoud. Mobile agents in wireless devices. *IEEE Computer, ISSN 0018-9162*, 37(2):104–105, 2004.

[VM04b] L. Vasiu and Q.H. Mahmoud. Mobile Agents in Wireless Devices. *IEEE Computer*, 37(2):104–105, 2004.

[VMK98] H. Vogler, M. Moschgarth, and T. Kunkelmann. Enhancing Mobile Agents with Electronic Commerce Capabilities. In *Second International Workshop on Cooperative Information Agents II, Learning, Mobility and Electronic Commerce for Information Discovery on the Internet (CIA'98), Paris, France*, pp. 148–159. Springer, 1998.

[VV00] U. Varshney and R.J. Vetter. Emerging Mobile and Wireless Networks. *Communications of the ACM*, 43(6):73–81, 2000.

[WA97] P. Weinstein and G. Alloway. Seed Ontologies: Growing Digital Libraries as Distributed, Intelligent Systems. In *DL '97: Proceedings of the second ACM international conference on Digital libraries*, pp. 83–91, New York, NY, USA, 1997. ACM Press.

[Wal99] N. Wallace. *COM/DCOM Blue Book: The Essential Learning Guide for Component-Oriented Application Development for Windows*. Coriolis Group Books; Bk&CD Rom edition, ISBN-10: 1576104095, ISBN-13: 978-1576104095, February 1999.

[WBD99] P.C. Weinstein, W.P. Birmingham, and E.H. Durfee. Agent Based Digital Libraries: Decentralization and Coordination. *Communications Magazine*, 37(1):110–115, January 1999.

[Wei99] G. Weiss. *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[WJ95] M. Wooldridge and N.R. Jennings. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[WNRJ95] M. Wooldridge and editors N. R. Jennings. *Intelligent Agents — Theories, Architectures, and Languages*. Lecture Notes in Artificial Intelligence, Springer-Verlag,ISBN 3-540-58855-8, January 1995.

[Woo02] M. Wooldridge. *An Introduction To MultiAgent Systems*. John Wiley and Sons, Chichester, 2002.

[Wor01] World Wide Web Consortium (W3C). XML Schema Part 2: Datatypes, W3C Recommendation, World Wide Web Consortium, May 2001. This version is http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/. The latest version is available at http://www.w3.org/TR/xmlschema-2/, [Last accessed: April 2009].

[Wor06a] World Wide Web Consortium. HTML 4.01 Specification, W3C Recommendation 24, December 2006. http://www.w3.org/TR/html401/, [Last accessed: April 2009].

[Wor06b] World Wide Web Consortium. XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16, November 2006. http://www.w3.org/TR/1999/REC-xslt-19991116, [Last accessed: April 2009].

[Wor06c] World Wide Web Consortium (W3C), 2006. http://www.w3.org/, [Last accessed: April 2009].

[WTR05] Y. Wang, K.L. Tan, and J. Ren. Towards Autonomous and Automatic Evaluation and Negotiation in Agent-Mediated Internet Marketplaces. *Electronic Commerce Research*, 5(3-4):343–365, 2005.

[WWSV06] Wap Forum WAP-WML Specification Version 1.1, 16 Jun 1999, 2006. http://www.wapforum.org/, [Last accessed: April 2009].

[WWW00] G. Wu, Q. Wu, and H. Wang. A Novel Workflow Management Model Based on Mobile Agents for Internet Electronic Commerce. In *36th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-Asia'00)*. IEEE Computer Society, 2000.

[XW00] C. Xu and Brian Wims. A Mobile Agent Based Push Methodology for Global Parallel Computing. *Concurrency - Practice and Experience*, 12(8):705–726, 2000.

[Yah06] Yahoo! Inc., 2006. http://www.yahoo.com/, [Last accessed: April 2009].

[YF01] Y. Ye and G. Fischer. Context-Aware Browsing of Large Component Repositories. In *proceedings of the IEEE 16th International Conference on Automated Software Engineering. Coronado Island, CA.*, pp. 99–106, November 2001.

[Yia97] P. Yianilos. The LikeIt Intelligent String Comparison Facility. Technical Report 97-093, NEC Research Institute, 1997.

[YKP+01] I. Yen, L. Khan, B. Prabhakaran, F. B. Bastani, and J. Linn. An On-Line Repository for Embedded Software. In *13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01). Dallas, Texas*, November 2001.

[YPHM98] J. Yang, P. Pai, V. Honavar, and L. Miller. Mobile Intelligent Agents for Document Classification and Retrieval: A Machine Learning Approach. In *14th European Meeting on Cybernetics and Systems Research. Symposium on Agent Theory to Agent Implementation, Vienna, Austria*, 1998.

[YRG00] Y. Yang, P. F. Rana, and C. Georgousopoulos. Mobile Agents and the SARA Digital Library. *Proceedings of IEEE Advances in Digital Libraries 2000*, pp. 71–77, May 2000.

[YRW+02] Y. Yang, O.F. Rana, D.W. Walker, C. Georgousopoulos, G. Aloisio, and R. Williams. Agent Based Data Management in Digital Libraries. *Parallel Comput.*, 28(5):773–792, 2002.

[Zac03] J. Zachary. Protecting Mobile Code in the Wild. *IEEE Internet Computing*, 7(2):78–82, 2003.

[ZD97a] B. Zenel and D. Duchamp. A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment. In *Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking*, September 1997.

[ZD97b] B. Zenel and D. Duchamp. General Purpose Proxies: Solved and Unsolved Problems. In *HOTOS '97: Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS-VI)*, pp. 87, Washington, DC, USA, 1997. IEEE Computer Society.

[ZHS+04] B. Zhao, L. Huang, J. Stribling, S. Rhea, A. Joseph, and J. Kubiatowicz. Tapestry: A Resilient Global-Scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[Zhu04] H. Zhu. Cooperative Agent Approach to Quality Assurance and Testing Web Software. In *Proc. of COMPSAC'04, Workshop on Quality Assurance and Testing of Web-Based Applications, IEEE CS, Hong Kong*, pp. 110–113, Septeber 2004.

[ZSX+04] D. Zhang, Y. Shi, G. Xu, E. Chen, and H. Gu. Seamless Transfer of Pervasive Computing Task. In *International Conference of Computing, Communication and Control Technology (CCCT'04), Austin, Texas, USA*, 2004.